

**18th International Conference in Central Europe
on
Computer Graphics, Visualization and Computer Vision**

in co-operation with

EUROGRAPHICS

WSCG 2010

Communication Papers Proceedings

Edited by

Vaclav Skala, University of West Bohemia, Czech Republic

**18th International Conference in Central Europe
on
Computer Graphics, Visualization and Computer Vision**

in co-operation with

EUROGRAPHICS

WSCG 2010

Communication Papers Proceedings

Edited by

Vaclav Skala, University of West Bohemia, Czech Republic

Vaclav Skala – Union Agency

WSCG 2010 – Communication Papers Proceedings

Editor: Vaclav Skala
c/o University of West Bohemia, Univerzitni 8
CZ 306 14 Plzen
Czech Republic
skala@kiv.zcu.cz

Managing Editor: Vaclav Skala

Published and printed by:
Vaclav Skala – Union Agency
Na Mazinách 9
CZ 322 00 Plzen
Czech Republic

Hardcopy: *ISBN 978-80-86943-87-9*

WSCG 2010

Program Committee members

Adzhiev, V. (U.K.)
Balcisoy, S. (Turkey)
Benes, B. (USA)
Bengtsson, E. (Sweden)
Biri, V. (France)
Bittner, J. (Czech Republic)
Bouatouch, K. (France)
Brodlie, K. (U.K.)
Buehler, K. (Austria)
Csebfalvi, B. (Hungary)
Daniel, M. (France)
Davis, L. (USA)
de Geus, K. (Brazil)
Debelov, V. (Russia)
Ferguson, S. (U.K.)
Flaquer, J. (Spain)
Gavrilova, M. (Canada)
Gudukbay, U. (Turkey)
Gutierrez, D. (Spain)
Havran, V. (Czech Republic)
Chover, M. (Spain)
Jansen, F. (Netherlands)
Kruijff, E. (Austria)
Lee, B. (Korea)
Lee, T. (Taiwan)
Magnor, M. (Germany)
Mollá Vayá, R. (Spain)
Muller, H. (Germany)
Murtagh, F. (Ireland)
Pedrini, H. (Brazil)
Platis, N. (Greece)
Puppo, E. (Italy)
Purgathofer, W. (Austria)
Rojas-Sola, J. (Spain)
Rosenhahn, B. (Germany)
Rudomin, I. (Mexico)
Sakas, G. (Germany)
Sbert, M. (Spain)
Segura, R. (Spain)
Schumann, H. (Germany)
Sochor, J. (Czech Republic)
Stroud, I. (Switzerland)
Teschner, M. (Germany)
Theoharis, T. (Greece)
Tokuta, A. (USA)
Vergeest, J. (Netherlands)
Weiss, G. (Germany)
Zach, C. (Switzerland)
Zara, J. (Czech Republic)
Zemcik, P. (Czech Republic)
Zitova, B. (Czech Republic)

WSCG 2010

Board of Reviewers

Abas, M. (Malaysia)
Adzhiev, V. (United Kingdom)
Akleman, E. (United States)
Avenueau, L. (France)
Balcisoy, S. (Turkey)
Battiato, S. (Italy)
Benes, B. (United States)
Bengtsson, E. (Sweden)
Biri, V. (France)
Bittner, J. (Czech Republic)
Bouatouch, K. (France)
Bourdin, J. (France)
Bouville, C. (France)
Brodlie, K. (United Kingdom)
Bruni, V. (Italy)
Buehler, K. (Austria)
Buriol, T. (Brazil)
Camahort, E. (Spain)
CarmenJuan-Lizandra, M. (Spain)
Casciola, G. (Italy)
Csebfalvi, B. (Hungary)
Daniel, M. (France)
Davis, L. (United States)
de Geus, K. (Brazil)
Debelov, V. (Russia)
du Buf, H. (Portugal)
Durikovic, R. (Slovakia)
Erbacher, R. (United States)
Erleben, K. (Denmark)
Feng, J. (China)
Ferguson, S. (United Kingdom)
Ferko, A. (Slovakia)
Fernandes, A. (Portugal)
Flaquer, J. (Spain)
Galo, M. (Brazil)
Ganovelli, F. (Italy)
Garcia-Alonso, A. (Spain)
Gavrilova, M. (Canada)
Giannini, F. (Italy)
Gonzalez, P. (Spain)
Gudukbay, U. (Turkey)
Guérin, E. (France)
Gutierrez, D. (Spain)
Habel, R. (Austria)
Hanak, I. (Czech Republic)
Haro, A. (United States)
Hasler, N. (Germany)
Havran, V. (Czech Republic)
Hernández, B. (Mexico)
Herout, A. (Czech Republic)
Horain, P. (France)
House, D. (United States)
Chaudhuri, D. (India)
Chover, M. (Spain)
Jansen, F. (Netherlands)
Joan-Arinyo, R. (Spain)
Kohout, J. (Czech Republic)
Kruijff, E. (Austria)
Lanquetin, S. (France)
Lee, B. (Korea)
Lee, T. (Taiwan)
Liu, S. (China)
Liu, D. (Taiwan)
Maciel, A. (Brazil)
Magnor, M. (Germany)
Mandl, T. (Germany)
Matkovic, K. (Austria)
Mawussi, K. (France)
McMenemy, K. (Ireland)
Michoud, B. (France)
Mokhtari, M. (Canada)
Mollá Vayá, R. (Spain)
Montrucchio, B. (Italy)
Muller, H. (Germany)
Murtagh, F. (Ireland)
Pan, R. (China)

Papaioannou,G. (Greece)
Patane,G. (Italy)
Pedrini,H. (Brazil)
Pina,J. (Spain)
Platis,N. (Greece)
Plemenos,D. (France)
Post,F. (Netherlands)
Pratikakis,I. (Greece)
Puig,A. (Spain)
Puppo,E. (Italy)
Purgathofer,W. (Austria)
Renaud,c. (France)
Richardson,J. (United States)
Ripolles,O. (Spain)
Ritschel,T. (Germany)
Rojas-Sola,J. (Spain)
Rosenhahn,B. (Germany)
Rudomin,I. (Mexico)
Sakas,G. (Germany)
Sanna,A. (Italy)
Sbert,M. (Spain)
Segura,R. (Spain)
Sellent,A. (Germany)
Schneider,B. (United States)
Schumann,H. (Germany)
Sirakov,N. (United States)
Sochor,J. (Czech Republic)

Solis,A. (Mexico)
Sousa,A. (Portugal)
Steinicke,F. (Germany)
Stroud,I. (Switzerland)
Svoboda,T. (Czech Republic)
Teschner,M. (Germany)
Theoharis,T. (Greece)
Theußl,T. (Austria)
Tokuta,A. (United States)
Torrens,F. (Spain)
Tytkowski,K. (Poland)
Vanecek,P. (Czech Republic)
Vasa,L. (Czech Republic)
Veiga,L. (Portugal)
Vergeest,J. (Netherlands)
Vitulano,D. (Italy)
Weiss,G. (Germany)
Wu,S. (Brazil)
Yencharis,L. (United States)
Zach,C. (Switzerland)
Zachmann,G. (Germany)
Zalik,B. (Slovenia)
Zara,J. (Czech Republic)
Zemcik,P. (Czech Republic)
Zhu,Y. (United States)
Zitova,B. (Czech Republic)

Communication papers

Title	Page
Berger,K., Lipski,Ch., Magnor,M.: Target Space Interactivity - The end of 3D widgets	1
Havel,J.: Functional Programming of Geometry Shaders	9
Bhattacharya,J., Majumder,S.: Visual Odometric Navigation: the GFV Way.	15
Kaczmarczyk,J, Dohnalik,M., Cnudde,V., Zalewska,J.: The interpretation of X-ray Computed Microtomography images of rocks as an application of volume image processing and analysis	23
Yin,O.-S., Jin,A.T.B., Yan,H.B., Han,P.Y.: Offline Signature Verification through Probabilistic Neural Network	31
Kudelski,D., Mari,J.-L., Viseur,S.: Feature Line Detection on Triangulated Meshes: A Geological Application	39
Langs,A.,Bärz,J.: Confidence in Tone Mapping Applying a User-Driven Operator	47
Hast,A., Seipel,S., Ericsson,M.: Multiscale Texture Synthesis and Colorization of Greyscale Textures	55
Wacker,M., Wegner,M.: Making people move - walking techniques in a CAVE	63
Zlatuška,M., Havran,V.: Ray Tracing on a GPU with CUDA -- Comparative Study of Three Algorithms	69
Wang,S.,You,R., Chen,Y., Li,S., Wang,G.: Difference-Contribution Strategy for Seeding 2D Streamlines	77
Lipski,C., Bose,D., Eisemann,M., Berger,K., Magnor,M.: Sparse Bundle Adjustment Speedup Strategies	85
Danihelka,J., Kencl,L., Zara,J.: Reduction of Animated Models for Embedded Devices	89
Costa,V., Pereira,J., Jorge,J.: Multi-Level Hashed Grid Construction Methods	95
Dobrev,P., Rosenthal,P., Linsen,L.: Interactive Image-Space Point Cloud Rendering with Transparency and Shadows	101
Benes,P., Medek,P., Sochor,J.: Tracking single channel in protein dynamics	109
Portelli,D., Ganovelli,F., Tarini,M, Cignoni,P., Dellepiane,M., Scopigno,R.: A framework for Sketch Based User Assisted Fitting of Geometric Primitives	115
Mujika,A., Oyarzun,D., Arrieta,A., Carretero,M.P.: Real time accurate collision detection for virtual characters	123
Gaitatzes,A., Andreadis,A., Papaioannou,G., Chrysanthou,Y.: Fast Approximate Visibility on the GPU Using Precomputed 4D Visibility Fields	131
Rahajaniaina,A., Jessel,J.-P.: Parcel's information visualization on mobile Device	139
Giroud,A., Biri,V.: Modeling and rendering heterogeneous fog using wavelets	145

Bíscaró,H.H.: Efficient Reconstruction From Scattered Points	153
Liu,Y., Laycock,S.D.: Creating Continuous Force Feedback for Haptic Interaction of Volume Data Sets	161
Kolchin,K.: Surface Curvature Effects on Reflectance from Translucent Materials	169
Pintavirooj,Ch., Cohen,F.S., Iampa,W.: Fingerprint Alignment Based on Local Feature Combined with Affine Geometric Invariant	173
Wucharz, J., Loviscach,J.: Chrome, Gold and Silver on the Screen	179
Lazarevych,O., Szekely,G., Harders,M.: Decomposing the Linear Complementarity Problem into Separate Contact Regions	185
Awano,N., Nishio,K., Kobori,K.: Interactive Stipple Rendering for Point Clouds	193
Lee,H., Lavoué,G., Dupont,F.: New methods for progressive compression of colored 3D Mesh	199
Grund,N., Menzel,N., Guthe,M.: High-Quality Wavelet Compressed Textures for Real-time Rendering	207
Polceanu,M., Popovici,A., Popovici,D.M.: A system for panoramic navigation inside a 3D environment	213
Panning,A., Al-Hamadi,A., Michaelis,B.: Active Shape Models on adaptively refined mouth emphasizing color images	221
Ryu,D.-S., Jang,Ch.-J., Cho,H.G.: A User-Adaptive Image Browsing System with Summarization Layout for the Personal Photo Collections	229
Kobori,K., Hirose,K., Nishio,K.: Automatic Generation of Character Behavior by the Placement of Objects with Motion Data	237
Espinal,J., Allen,V., Amable,K., Bailey,R., Bischof,H.-P.: RenderMan's Power to Visualization's Rescue	243
Hassan,A.M., Al-Hamadi,A., Hasan,Y.M.Y., Wahab,M.A.A., Michaelis,B.: Image Authentication using Robust Image Hashing with Localization and Self-Recovery	251
Trávníček,Z., Berka,R.: Multi-Threaded Real-Time Video Grabber	259
Andrsen,V., Aanæs,H., Bærentzen,A., Nielsen,M.: Markov Random Fields on Triangle Meshes	265
Radziszewski,M., Alda,W., Boryczko,K.: Interactive Ray Tracing Client	271
Maslov,I.V., Detkova,Y.D., Gertner,I.: A Novel Image Transformation for Solving Complex Image Mapping Problems	279

Target Space Modeling - The End of 3D Widgets

Kai Berger
TU Braunschweig
Germany
berger@cg.tu-bs.de

Tobi Vaudrey
University of Auckland
New Zealand
t.vaudrey@auckland.ac.nz

Christian Linz
TU Braunschweig
Germany
linz@cg.tu-bs.de

Reinhard Klette
University of Auckland
New Zealand
r.klette@auckland.ac.nz

Christian Lipski
TU Braunschweig
Germany
lipski@cg.tu-bs.de

Marcus Magnor
TU Braunschweig
Germany
magnor@cg.tu-bs.de

ABSTRACT

In today's modeling tools, the graphical user interfaces are required to be accurate and intuitive to use. Most tools therefore rely on additional 3D-widgets (e.g., arrows or circles) that enable the user to operate towards a desired modeling result. In this paper we present, for the first time, a method that makes these widgets obsolete. We propose to use simple geometric primitives such as planes or spheres as low-dimensional subspaces, so called *target spaces* for the interaction. Instead of operating towards a modeling result, the user then directly steers the result. The target spaces suffice to be indicated to the user just as additional visual information. We verify by means of a user study that with our method it is now possible to develop accurate single-view GUIs without 3D-widgets that are highly intuitive to use.

Keywords: Graphical user interface, human computer interaction, interactivity, computer geometries, 3D interaction, graphics applications.

1 INTRODUCTION

Today's modeling applications are in general used for the task to position or deform complex objects or parts thereof. Common examples are the modeling of human characters or complex moving objects (e.g., parts of a car engine). In both examples objects are defined by a transformation hierarchy. For example, the pose of a human right hand depends on the pose of the right arm, whose pose is dependent of the current pose of the torso. The term *pose* defines the current position and rotation of an object. A pose is defined by six degrees of freedom (DOF).

The object's transformation hierarchy is determined by a kinematic chain (i.e., the assembly of several kinematic pairs connecting rigid body elements). Often this combination of kinematic pairs, with 6 DOF each, leaves the user even for simple models with a parameter space of high dimensionality. This high-dimensional parameter space poses a challenge for the user to interact with the model in an intuitive way.

Some applications enable the user to specify the translation or rotation of a rigid body numerically. They may be most accurate but modeling tasks become very time-consuming and unintuitive. Others present

an interface to manipulate the pose of each rigid body in a kinematic chain separately. Mostly authors show projections of an object on the three coordinate planes and a fourth viewport, where the user can freely choose the camera position. In general the free-viewpoint viewport contributes only little to the accuracy of the modeling results. Altogether, user interactions are commonly performed by using projections in some coordinate planes.

In this paper we present a new method that allows for intuitive modeling operations within a single viewport without any additional 3D widgets. By constraining the interaction space for a given operation in a 3D scene to subspaces defined by geometric primitives, the user succeeds in transforming the object or its subparts accurately and in a highly intuitive way.

The paper is organized as follows: After giving an overview of related work in the area of interactivity methods for object modeling in Section 2, we define the core idea of subspace interactivity in Section 3 and contrast it to state-of-the-art modeling software. Afterwards in Section 4, a set of geometric primitives used for subspace interactivity is introduced. The usability of our method is evaluated in a case study with a motion capturing software in Section 5, before we conclude in Section 6.

2 RELATED WORK

For a brief overview on research in graphical user interfaces for object manipulation, see Myers et al. [MHC⁺96]. The interaction process from a user's point of view is described by Wright [WFH00]. Our method is related to the following work, where each

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WSCG 2010 conference proceedings,
WSCG'2010, February 1 - 4, 2010
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

technique is focusing on certain aspects of interactivity:

Widgets and Input Mapping

Wu et al. [WATB03] present a toolkit which exhaustively uses *picking*, the method to determine the corresponding 3D-object for a selected 2D-pixel. A focus on 3D-widgets for transforming complex geometrical objects is provided by Conner et al. [CSH⁺92], who give an introduction to state definitions for typical widget operations (e.g., 3D-rotation). 3D-widgets became famous in the OpenInventor

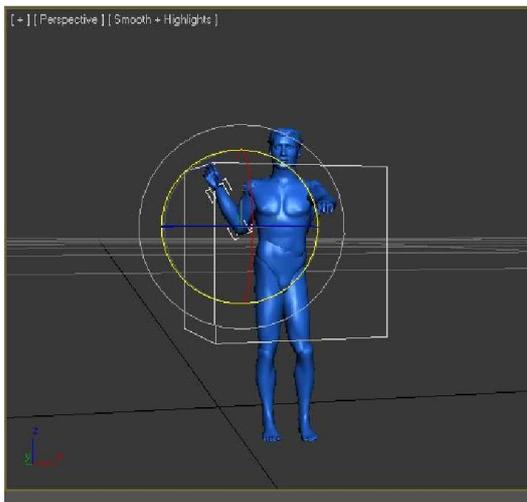
framework [SWH⁺05]. Another paper by Dollner et al. examines 3D-widgets as deformation handlers for geometrical objects [DH98]. 3D-sliders as a type of 3D-widgets and enhancement of 2D-sliders are described by Beckenridge et al. [BHMO01] and Stotts [Sto02].

Surface modeling

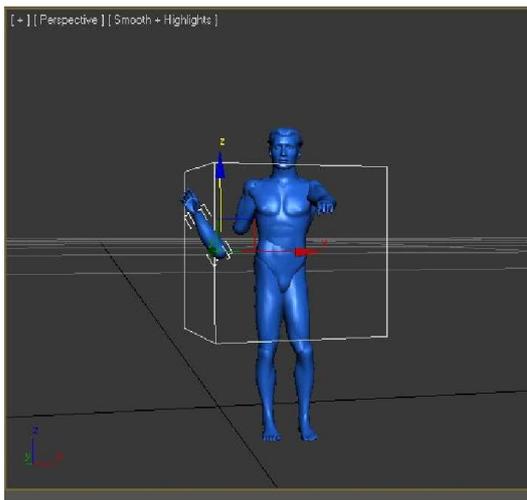
Schmidt et al. [SKKS09] present a surface modeling system based on enhanced 2D views. A complex surface is depicted by equidistant lines. The lines are colored in shades of gray according to their distance from the viewpoint. Thus the user can easily select and change surface parts within a single viewport by adjusting particular lines. Another system, ILoveSketch [BBS08], allows for 3D spline sketching in a single viewport by exploiting visual cues (e.g., vanishing points).

Human motion modeling

Buttussi et al. [BCN06] developed a tool to adjust the pose of a humanoid model. They use a single viewport to select joints and drag them fronto-parallelly, but they depend on a set of sliders in a second window to provide for accurate positioning. Popular modeling tools, such as Blender [Fou09] or 3D Studio Max [Aut09] employ 3D-widgets to transform nodes of the body's kinematic chain.



(a) Rotation



(b) Transposition

Figure 1: A typical object transformation based on 3D widgets. A rotation (a) around a certain axis is performed by dragging the corresponding circle; a transposition (b) is performed by dragging an axis-aligned arrow. Screenshots reproduced from [Aut09].

In 3D Studio Max, for example, an axis aligned transposition of a right hand is performed by dragging one of three displayed arrows towards the desired direction, Fig. 1 (b).

Our method, instead, introduces 1D- and 2D-subspaces for object transformation operations (e.g., rotation and transposition) and is thus independent of additional widgets, such as arrows, to be drawn in the scene. The deformation of body surfaces, however, is done by adjusting 3D sliders aligned to local coordinate axes. In the following we will mainly contrast the concepts of 3D Studio Max [Aut09], representing state-of-the-art modeling tools, to our method.

3 INPUT HANDLING WITH THE TARGET SPACE

Our main goal at this point is to provide a versatile user interface for modeling tasks which fulfills the following constraints:

1. The user should be able to perform any task in a single viewport.
2. The operation should be performed in a highly intuitive manner.
3. The result of the operation should be accurate compared to the desired task.

In state-of-the-art software, such as 3D Studio Max [Aut09], the user is provided with one or many 3D-widgets to solve the modeling task in one viewport. That means, additional objects, such as arrows or curved lines, have to be drawn around the object to be transformed. The user has to drag these objects in specific directions in order to come closer to the desired result for the object to be transformed. In general it can be stated that the screen is filled with additional objects to be handled. It is very likely that this may confuse a user.

Our approach, instead, is to consider the modeling task as a combination of lower-dimensional operations within the 3D space. An object can be transformed, for example, by a target translation $t \in T$ or rotation $r \in T$ within a lower-dimensional *target space* $T \subset \mathbb{R}^3$. This target space can be either 1D or 2D and is, thus, always embedded in the scene.

When the user triggers a requested operation for a rigid body [see Fig. 2 (a)], the target space T for this operation is optionally indicated to the user [see Fig. 2 (b)], and the backprojection Π of the mouse position p_{screen} to T [see Fig. 2 (c)] determines the desired target position p_T within the target space T .

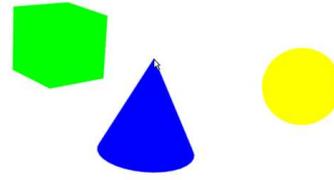
The backprojection Π can be written as

$$\Pi : p_{screen} \in \mathbb{R}^2 \mapsto p_T \in T \subset \mathbb{R}^3 \quad (1)$$

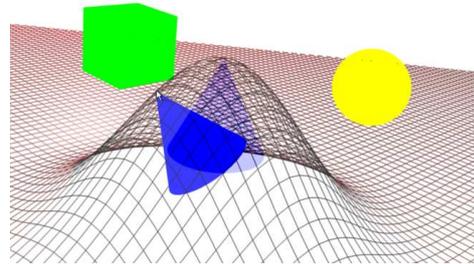
The key difference between the backprojection Π to p_T and a standard backprojection to the current scene is that only the mouse position p_{screen} is backprojected to the target space T . That is, the backprojection is partaken in a second scene which consists only of the geometric primitives which span T . The camera parameters remain constant in both scenes.

The advantage over state-of-the-art software, such as 3D Studio Max [Aut09], is now clearly that the user is independent of additional widgets to achieve desired object transformations. The user only sees the object and optionally the indicated target space. The target space may, for example, be shown as a grid to guide the user. Any mouse movement on the target space is directly interpreted as the new result state for the object. The screen becomes less confusing and more obvious, so that intuitivity increases, while accuracy stays steady.

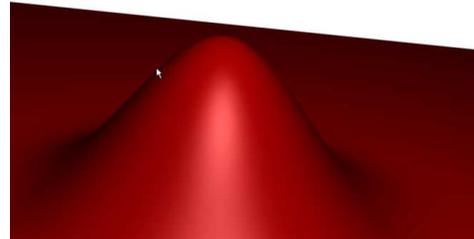
Any geometric primitive can serve as T , as long as it is a true subset of \mathbb{R}^3 . This is necessary for the backprojection, because otherwise the result would not be well-defined. In our further considerations we focus on thin lines, surfaces of spheres and planes for modeling tasks, but any Bézier spline or surface is also suitable. Once again we like to emphasize, that the indication of the target space to the user [see Fig. 2 (b)] is *optional*, and that the user can employ the operation on the target space also without any additional visual information.



(a) User selects object to be transformed



(b) Target space T is optionally indicated to user



(c) Mouse position is backprojected to T

Figure 2: Illustration of a transformation of a rigid body. The blue cone (a) is considered as an operation in the target space T . Its span is optionally indicated to the user, for example by a grid; see (b). The operation is done by backprojecting the mouse position into the target space T ; see the red surface in (c).

This is the *main difference* to the 3D widgets used in state-of-the-art modeling tools.

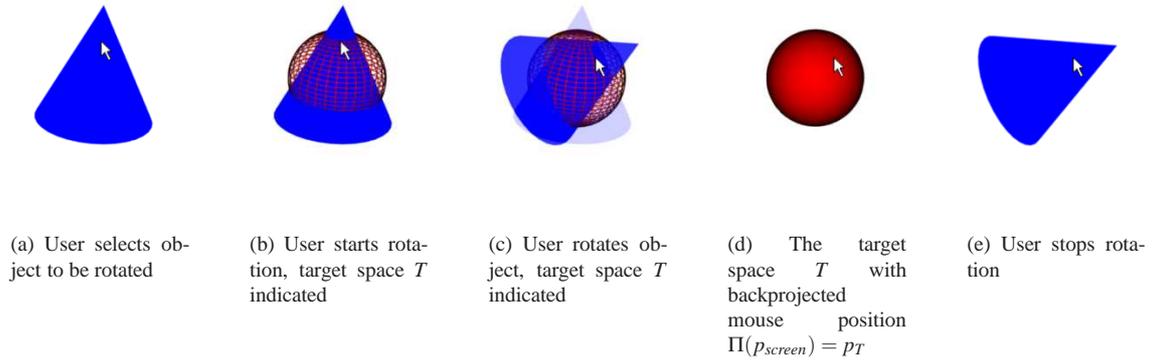


Figure 3: An object rotation with a sphere as geometric primitive for T . The object's center of gravity determines the sphere's center while its distance to the backprojected mouse position determines the radius. However, a larger radius is possible in order to increase usability.

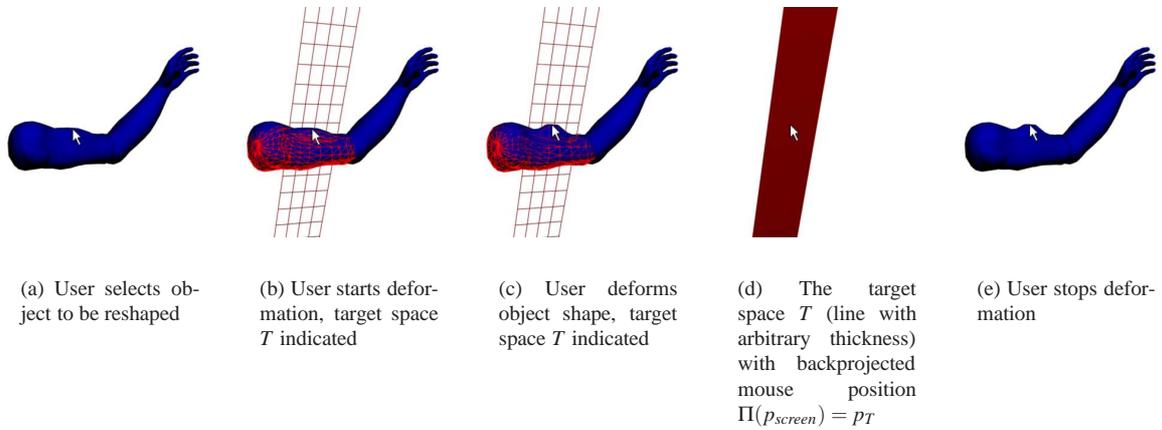


Figure 4: An object deformation with a line of arbitrary thickness as geometric primitive for T . The line is parallel to the normal of the selected patch and intersects the object at the backprojected mouse position.

4 GEOMETRIC PRIMITIVES IN THE TARGET SPACE

In most modeling applications we face the following basic transformation operations: transposition, rotation and surface deformation. The transposition and rotation operations have 3 DOF each; the DOF of a surface deformation depends on its tessellation. However, combining only few of those operations to a complex task will lead to a parameter space of high dimensionality. Thus we exploit the geometric primitives as target spaces for the elementary transformation tasks:

- A rotation of a rigid object is done on the surface of a sphere of fixed radius r with

$$p_T(\phi, \theta) = \begin{pmatrix} r \cdot \cos(\theta) \cdot \cos(\phi) \\ r \cdot \cos(\theta) \cdot \sin(\phi) \\ r \cdot \sin(\theta) \end{pmatrix} \quad (2)$$

where ϕ and θ determine the longitude and latitude angles. Thus, the transformation is performed in a 2D target space. The sphere center is mostly located

in the object's center of gravity. The radius can be determined by the distance to the selected point on the object; see Fig. 3. However, a larger radius is considerable in order to increase the usability

- A transposition of a rigid object is done on a plane (e.g., axis-aligned on the XY-plane) in the coordinate system of the object's parent in the transformation hierarchy with

$$p_T(u, v) = \begin{pmatrix} u \\ v \\ 0 \end{pmatrix} \quad (3)$$

where u and v determine the point on the plane. Thus the transformation is performed in a 2D target space, as well.

- A transposition of an object with a fixed distance to a local or global center is done on the surface of a sphere according to Eq. (2). Thus the transformation is also performed in a 2D target space.

- A parametrized surface deformation is done patch by patch. For each parametrized patch, spanned by tangent vectors \vec{r}_u, \vec{r}_v , the deformation is performed on a line parallel to the patch normal

$$\vec{p}_T(\alpha) = \alpha \cdot \frac{\vec{r}_u \times \vec{r}_v}{|\vec{r}_u \times \vec{r}_v|} \quad (4)$$

Here, α determines the point on the line. The target space in this case is only 1D, the user interacts with the surface by moving it upwards or downwards that line; see Fig. 4.

5 CASE STUDY: MOTION CAPTURING SYSTEMS

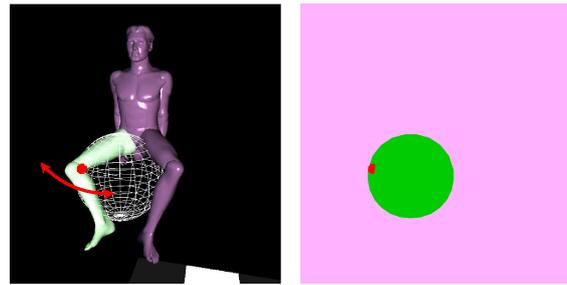
The main purpose of motion capturing applications is to fit humanoid 3D-models to the input data captured from one or many video cameras. Since in most cases the input data show only the actor in front of a well-distinguishable background, the application succeeds in recognizing and modeling the intended pose of the actor.

However, sometimes the system can produce an erroneous guess for the pose due to lighting conditions or ambiguous body constraints. Then it is very useful, if the system provides an intuitive interface for the user to correct the pose. Furthermore it is also useful for the user to define new motion sequences independently from input videos (e.g., to create new motions for an already captured character). Here, an intuitive user interface is inevitable.

In the following case study we apply the presented target space approach to implement a GUI for an existing motion capturing software. Although single joint positions can be adjusted numerically, motion capturing software has not provided an intuitive interface for a user so far. Main requirements for an interface are that the user could literally drag the body entities of the humanoid to desired poses and that he could deform the shape of the body in one single viewport. The user can rotate the camera around the captured scene like a trackball.

From these requirements we deduce the basic transformation operations of the application: While entities like hands or feet can be translated within a sphere around the shoulders or hips, other entities like the torso could be rotated in place. Furthermore the surface mesh of each entity could be deformed (i.e., inflated or deflated).

For the basic transformation operations we identify a suitable target space according to Section 4. Some entities can be transformed in multiple target spaces (e.g., the hand of the humanoid model can be dragged on a sphere around the shoulder or on a plane parallel to the shoulder's coordinate plane).



(a) Rotation of the knee, target space T is indicated to user

(b) Mouse position is back-projected to T

Figure 5: A rotation (a) of the model's knee is performed with the target space method (b): In the back-buffer the mouse position (small red sphere) is projected onto a simpler scene consisting of a sphere (green), that is tangential to the knee, the foot and the hip. A plane (pink) parallel to the viewing plane intersects the sphere at its center and thus prohibits marginal errors.

The user can pick an entity of the model by hovering over it with the mouse pointer. If multiple transformation operations are available, the user can choose the operation by pressing different keys.

A deformation of an entity's mesh can be done by moving the mouse parallel to the local surface normal. A mouse motion away from the surface indicates an inflation; a mouse motion closer to the surface indicates a deflation.

In the following, we briefly describe how the target space approach is implemented for the different operations mentioned above.

5.1 Rotation of humanoid body parts

The rotation of body parts is modeled as a transformation with a sphere surface as target space. An axis in the local coordinate system defines the rotation axis. The magnitude of a rotation angle is determined relatively from the difference of the backprojected mouse position when the mouse button is pressed and the backprojected mouse position when the mouse button is released again. Figure 5 shows the rotation operation and the according target space, the green sphere. The fronto-parallel pink plane intersects the sphere center to avoid backprojection problems, when the mouse pointer leaves the surface area of the sphere. The backprojected point p is then automatically evaluated as the intersection between a ray from the sphere center c to p and the surface of the sphere.

5.2 Translation of humanoid body parts

The translation of a body part is modeled both as a transformation with a plane as target space and with a

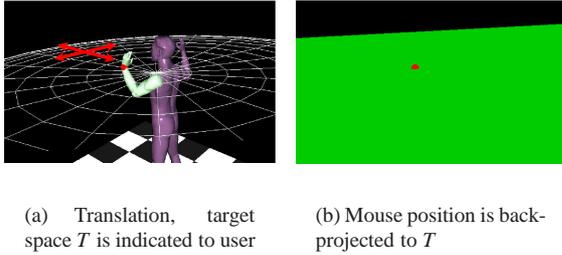


Figure 6: A translation (a) of the model's arm is performed with the same approach (b): In the backbuffer the mouse position (small red sphere) is projected onto a simpler scene consisting of a plane (green) intersecting the arm's origin.

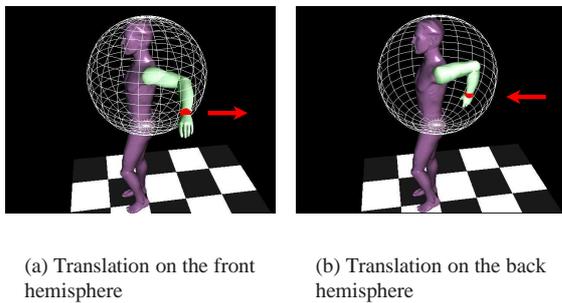
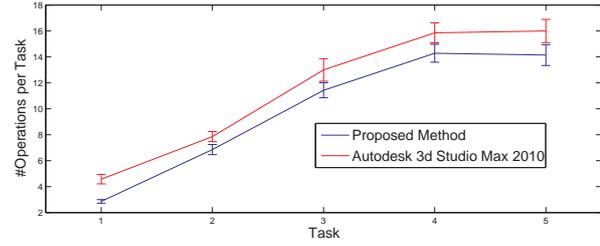


Figure 7: When leaving the sphere's surface with the mouse (a), the system automatically cuts off the half-sphere in front and enables the user to translate the joint to the backside (b), which would not be visible or reachable in normal view.

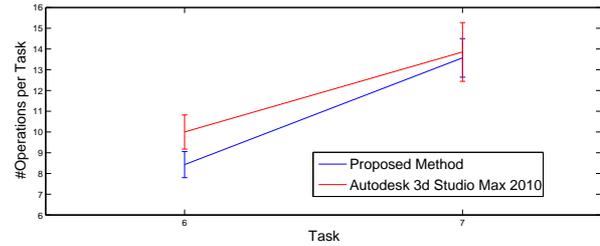
sphere surface as target space. The resulting position of the translated body part is evaluated absolutely by back-projecting the mouse position to the target space. An inverse kinematic chain [ZB94] is then computed for the parent nodes of the body part to account for body constraints (e.g. the flexion-extension range of a knee or an elbow).

Figure 6 shows the translation of the left wrist on a plane parallel to a coordinate plane of the shoulder's coordinate system and the corresponding target space, a green plane. The backprojected mouse position on the green plane determines the required position for the wrist.

Figure 7 shows the same translation of the left wrist with a sphere surface as target space. The sphere center in the target space is again intersected by a fronto-parallel plane. This time the plane is used to discriminate between the visible front side of the sphere and the invisible back side of the sphere. When the mouse pointer leaves the sphere surface and is backprojected on the plane, the system automatically cuts off the front half of the sphere. The user is then able to move along the inner surface of the sphere's back half.



(a) Results for the modeling tasks



(b) Results for the shaping tasks

Figure 8: The results of the user study. The users have been assigned five modeling (a) and two shaping tasks (b). For each task the mean number of necessary operations is plotted with variance. Note, that, on average, the proposed method needs less operations per task than the state-of-the-art tool.

5.3 Deformation of humanoid body parts

Each body part of the model, used in the system, consists of a coherent mesh. The deformation state of the mesh is defined by two fourth order polynomials of its longitudinal axis along for each of the remaining axes. The deformation parameters for each polynomial are 1D. Thus, we employ the target space approach by moving the selected patch in the mesh along a 1D-line, which is parallel to the patch's normal. Since the parameters are 1D, the thickness of the line is arbitrary.

5.4 Results - The User Study

The target space method has been implemented in an existing motion capturing software as graphical user interface. In a comparative user study with seven participants, the applicability of the method has been evaluated. Each user has been assigned five modeling tasks and two shaping tasks. In each task the users have been given an input model and a picture of a target pose (e.g., a man sitting, kicking, or lifting arms) or a target shape (e.g., fat, or muscular).

Further information about these tasks is shown in the Appendix. The users then had to change the pose or shape of the input model so that it resembled the picture of the target pose with as few modeling operations as possible.

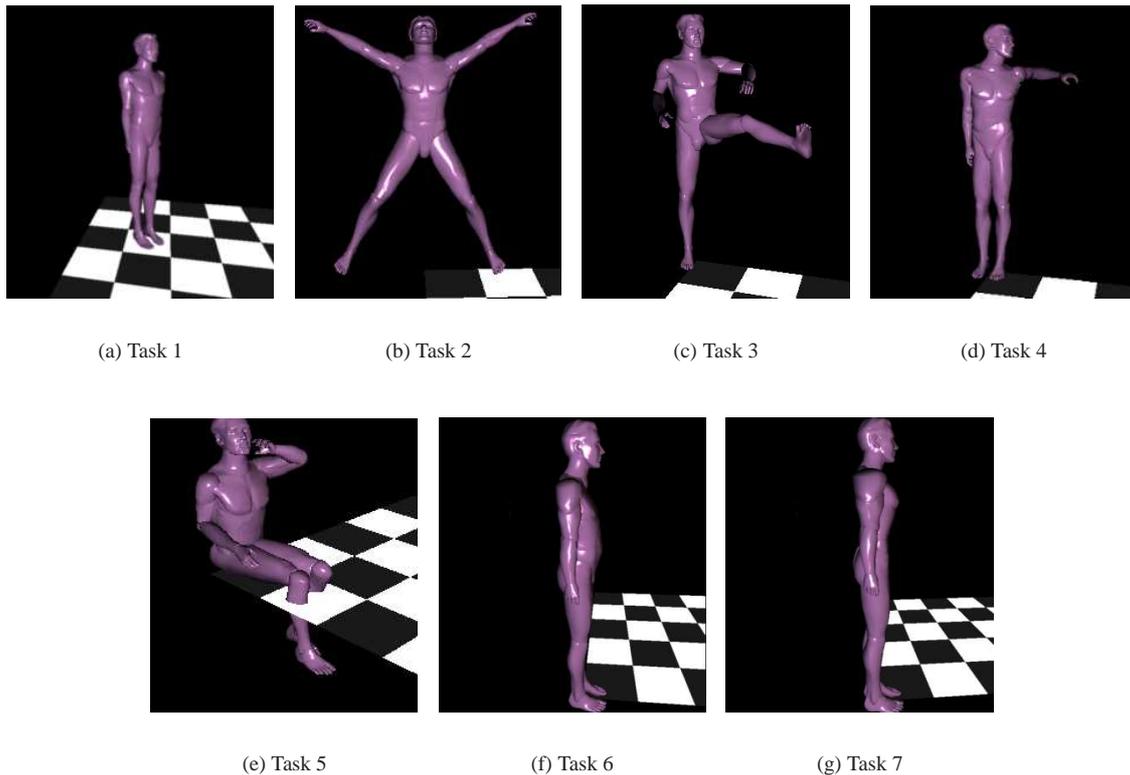


Figure 9: The seven modeling tasks in the conducted user study. Tasks 1-5 depict the poses the users had to model with both the proposed method and 3D Studio Max 2010 [Aut09]. Tasks 6 and 7 depict the different shapes the users had to give to the humanoid model.

In the first iteration the users modeled the pose with the proposed method. In the second iteration they used the commercial software Autodesk 3D Studio Max 2010. In both iterations, for each task the number of operations that were necessary to model the desired pose have been evaluated.

Figure 8 shows the mean number of necessary operations for each modeling (a) and shaping (b) task. The plots show that the proposed method enables the user to model desired poses and shapes with less necessary operations than with the state-of-the-art modeling software.

The users have also graded both modeling tools for intuitivity, learnability and usability. In terms of intuitivity and learnability, 71 % graded the proposed method as good as, or even better than the commercial tool. In terms of usability, 57 % graded the proposed method as good as, or even better than the commercial tool.

6 CONCLUSIONS

We presented a new approach for graphical user interfaces in modeling applications, which is based on lower-dimensional target spaces. The target spaces are embedded into the 3D scene and thus allow for single

viewport interactivity. While standard modeling applications rely on fronto-parallel operations or operation vectors that the user has to drag along the axes of a local coordinate system, our approach enables the user to transform the object in a very intuitive way, still achieving the desired accuracy.

We verified the applicability of the approach by a case study, where a graphical user interface had to be implemented for an existing motion capturing system. In a comparative user study we found that with the new approach users need less necessary operations for a modeling task than with state-of-the-art modeling software.

In the future we want to apply our approach to further modeling applications beyond motion capturing. We also want to exploit the constraints of kinematic chains of arbitrary objects for the target space modeling, because this will enhance the intuitivity of modeling tasks even more.

7 APPENDIX

In Fig. 9 we have listed the seven different modeling tasks of the conducted user study. In Task 1 [see Fig. 9 (a)], the users had to reposition the model and rotate it in an angle of 45° .

In Task 2 [see Fig. 9 (b)], the users had to lift the arms and legs to emulate a jumping pose.

In Task 3 [see Fig. 9 (c)], the users had to lift one leg and angle both arms to emulate a kicking pose.

In Task 4 [see Fig. 9 (d)], the users had to position the left arm and angle it and had to rotate the head towards the far left.

In Task 5 [see Fig. 9 (e)], the users had to model a person sitting and drinking a glass of water. Not only the arms, but also the hands had to be altered.

Tasks 6 and 7 regarded shaping operations. In Task 6 [see Fig. 9 (f)], the users had to give the humanoid model a potbelly. In Task 7 [see Fig. 9 (g)], the users had to shape the model like a body builder. In both tasks, several meshes had to be selected and reshaped in all three dimensions.

REFERENCES

- [Aut09] Autodesk. 3D Studio Max 2010. <http://www.autodesk.de/>, 2009.
- [BBS08] S.H. Bae, R. Balakrishnan, and K. Singh. ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 151–160. ACM New York, NY, USA, 2008.
- [BCN06] F. Buttussi, L. Chittaro, and D. Nadalutti. H-animator: a visual tool for modeling, reuse and sharing of X3D humanoid animations. In *Proceedings of the eleventh international conference on 3D web technology*, pages 109–117. ACM New York, NY, USA, 2006.
- [BHMO01] A. Breckenridge, B. Hamlet, D. Mehlhorn, and K. Oishi. A Dynamic Design Strategy for Visual and Haptic Development. *Proc. of the PHANTOM Users Group Workshop*, pages 31–37, 2001.
- [CSH⁺92] B. D. Conner, S. S. Snibbe, K. P. Herndon, D. C. Robbins, R. C. Zeleznik, and A. van Dam. Three-dimensional widgets. *Proc. of the symposium on Interactive 3D graphics*, pages 183–188, 1992.
- [DH98] J. Döllner and K. Hinrichs. Interactive, Animated 3D Widgets. *Computer Graphics International 1998*, pages 278–286, 1998.
- [Fou09] Blender Foundation. Blender. <http://www.blender.org/>, 2009.
- [MHC⁺96] B. Myers, J. Hollan, I. Cruz, et al. Strategic Directions in Human-Computer Interaction. *ACM Computing Surveys*, 28(4):794–809, 1996.
- [SKKS09] R. Schmidt, A. Khan, G. Kurtenbach, and K. Singh. On expert performance in 3D curve-drawing tasks. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pages 133–140. ACM, 2009.
- [Sto02] D. Stotts. 3D Sliders: Programming Uses for 3D Object Warping in Collaborative Virtual Environments. Technical report, University of North Carolina at Chapel Hill, 2002.
- [SWH⁺05] D. Stalling, M. Westerhoff, H.C. Hege, et al. Amira: A highly interactive system for visual data analysis. *The Visualization Handbook*, 38:749–67, 2005.
- [WATB03] S. T. Wu, M. Abrantes, D. Tost, and HC Batagelo. Picking and snapping for 3D input devices. *Brazilian Symposium on Computer Graphics and Image Processing*, pages 140–147, 2003.
- [WFH00] P. C. Wright, R. E. Fields, and M. D. Harrison. Analyzing Human-Computer Interaction as Distributed Cognition: The Resources Model. *Human-Computer Interaction*, 15(1):1–41, 2000.
- [ZB94] J. Zhao and N.I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics (TOG)*, 13(4):313–336, 1994.

Functional Programming of Geometry Shaders

Jiří Havel

Faculty of Information Technology

Brno University of Technology

ihavel@fit.vutbr.cz

ABSTRACT

This paper focuses on graphical shader programming, which is essential for real-time rendering. Opposite to classical low level, structured languages, functional approach is used in this work and existing work is extended to cover geometry shader programming. The compiler is able to transform the program in a way that is hard to achieve with classical languages. The program is written for all pipeline stages at once and the compiler does the partitioning. This allows the programmer to focus on program semantics and let the compiler take care of the efficient execution. First, this paper describes shader stages as functions in a mathematical manner. The process of program partitioning and transformation to one of the classical languages is described. Several examples show the differences between functional description and equivalent structured code.

Keywords: Rendering, Shaders, Functional Programming

1 INTRODUCTION

Graphical hardware has changed greatly since first graphic accelerators. Its architecture evolved from fixed function pipeline, which became more and more configurable to today's fully programmable SIMT processors. However the programming is still low-level. The graphical processors lack complex control structures in exchange for raw computation power. The three most used languages for shader programming (GLSL [7], HLSL [8] and Cg [9]) mimic very closely the structure of the rendering pipeline.

The number of programmable stages of the rendering pipeline has risen from two to five in the latest accelerators. This means that the programmer must maintain even higher number of programs, executed at once on a single primitive, and ensure their compatibility. The interfaces between pipeline stages must be compatible not only in types, which the compiler can check, but also in passed values, which cannot be checked automatically. Packing the shader programs into one effect file solves this problem only partially. Effect files are only multiple shader programs, packed into one file with some additional information. When the effect file contains a code for multiple generations of graphical cards, the dependencies are even harder to maintain. This paper focuses on splitting one program to multiple parts and automatic generation of interfaces between them. Vertex, geometry and fragment shaders are the point of interest. The next two - hull and domain shaders were added for performance reasons only and might be addressed in future work.

Functional approach seems suitable for shader programming. Shaders transform data without any side effects and run massively parallel. Functional programs tend to be more abstract and allow the compiler to reorganize the code more than imperative languages. Be-

cause functional programs are referentially transparent, the order, in which the program is executed, does not matter. Every program transformation that preserves the output value is allowed. As shader programming favors speed over code clarity, this can help readability and maintainability without sacrificing performance. Significant parts of shader programs could be generated automatically.

Functional programming languages undergo a rapid development in recent years. Functional languages leave academic ground and slowly become well known like Microsoft's F#. Elements from functional languages like closures and lambdas are used in current mainstream languages (Python, C#). Ideas from functional programming like map-reduce [2] are used for programming parallel algorithms. These successes suggest that functional programming loses its reputation of being slow and is used for computationally intensive tasks. Because rendering is a computationally intensive task of different type, this paper explores the usability of functional programming for it.

Section 2 describes languages that were used as inspiration for this work. Section 3 shows shaders as functions from a mathematical point of view. Section 4 describes the transformation from functional program to C-like representation that is compatible with common shader languages. Section 5 summarizes the advantages of this approach and discusses open issues for the following work.

2 RELATED WORK

One of the interesting functional languages for shader programming is Vertigo [3], which was developed by Conal Elliott at Microsoft Research. Vertigo is an embedded language, focused on geometry and texture generation. Complex shapes are built from simple primitives and transformations by function composition. A

significant part of the optimization is done by rewrite rules - common technique in functional programming, which is generally not applicable in imperative languages due to the lack of the referential transparency.

Another unfinished and interesting language for shader programming is Renaissance [1]. In this language, vertex and fragment shaders are specified as one program. The compiler splits the program and generates an interface between the vertex and fragment shader using simple rules that are based on expression frequencies and function linearity.

Expression frequencies correspond to pipeline stages, where the expression can be evaluated. Renaissance uses four frequencies - fragment, vertex, uniform and constant. Program is initially specified with fragment frequency and compiler determines lower frequencies for suitable expressions.

Function linearity is important for splitting vertex and fragment shader. For linear functions like addition is not important whether its input or output is interpolated over the rasterized primitive. This means, if input of linear function has vertex frequency, its output has vertex frequency too, so it can be safely moved to the vertex shader. Nonlinear functions like normalization can not be interpolated, so they must remain in the fragment shader. There exists another group of functions - partially linear - like multiplication. Its output can be interpolated if only one argument has vertex frequency and all other have frequency lower.

3 SHADERS AS FUNCTIONS

In this section and the following ones, a simplified Haskell [6] syntax will be used for program examples. Function types will be written in mathematical manner. For example $A \times B \rightarrow C$ means a function with a domains $A \times B$ (with two parameters of the type A and B) and a codomain C . Square brackets mean a list of values. It can be also an array, because the differences are not important here.

If we consider rendering as a function, the type of this function might be $U \times [A] \rightarrow [F]$. U denotes the uniform variables, textures and other rendering state, $[A]$ is the list of attributes of the rendered primitives and $[F]$ is the list of resulting fragments. This means the rendering takes the rendering state and the list of rendered vertices and transforms it to the list of fragments. These fragments are collected into the framebuffer. The rendering function can be split to three parts, equivalent to three pipeline stages.

The vertex shader does the transformation and lighting of all vertices. It has the type $U \times A \rightarrow V$. Because all vertices are processed identically, this function is simply mapped over input vertices. V is the vertex shader output.

The geometry shader follows the primitive assembly and takes one primitive consisting of one to six ver-

tices. It has type $U \times [V] \rightarrow [[G]]$. It takes one primitive, which can be viewed as a list of vertices and outputs several triangle (or line) strips. Each triangle strip is simply a list of vertices, so the complete output is a list of strips. $[[G]]$ denotes the interface between geometry and fragment shader.

The primitives from the geometry shader are assembled, rasterized, values are interpolated over them and used as input for the fragment shader. The fragment shader has type $U \times G \rightarrow F$.

Aside from the mentioned parts or frequencies of computation (vertex, geometry, fragment), another two frequencies exist. It is the constant and uniform frequency. The expressions with constant frequency are evaluated at compile time. The expressions with uniform frequency transform uniform variables before rendering. For example HLSL preshadings have uniform frequency.

These frequencies not only assign expressions to pipeline stages. They also express relative cost of the computation and their cost increases from constant to fragment. Calculating expression at constant or uniform frequency is beneficial always. The limit is only the amount of constant and uniform registers.

The benefit of moving possible calculations from geometry to vertex shader is caused by Post Transform Cache. This cache is located after vertex shader and stores its outputs. In ideal case, each vertex has to be transformed only once, but in reality, the capacity of the cache is up to several tens of vertices. When drawing single triangles, the vertex shader is executed three times per triangle. When drawing triangle strips, VS is executed once per triangle (plus two times per strip). With indexed rendering of optimized meshes, VS can be executed less than once per triangle [10]. This means, we can safely move to vertex shader even calculations that could be performed on only one vertex of the triangle.

Moving calculation from fragment to geometry shader is beneficial in all cases, when interpolation is less costly, than calculation.

3.1 Expression Splitting

As was mentioned in section 2, the program can be split into stages automatically by the compiler. This simplifies the programmer's work as he does not need to maintain the interfaces between stages manually. Aside from simple splitting, some expressions can be automatically moved into parts with lower frequencies. The programmer can write calculations that logically belong together at one place and let the compiler move them apart to achieve more efficient execution.

This section describes the process of determining the frequencies of program expressions. In the beginning, only frequencies of shader inputs are known. Constants

have constant frequency, uniform variables uniform frequency and vertex attributes have vertex frequency.

Selection of expressions with constant and uniform frequency is very similar. All function applications (function calls in structured languages) with constant frequency operands have constant frequency, too. Function applications with constant and uniform operands have uniform frequency. Listing 1 shows an example of vertex transformation and listing 2 equivalent code without declarations after frequency estimation and splitting.

```
uniform matrix4 model, view, projection
attribute vector3 position

— original code
position' = projection*view*model*position
```

Listing 1: Original code of Uniform and Vertex shader

```
— uniform part
tmp = projection*view*model

— vertex part
position' = tmp*position
```

Listing 2: Uniform and Vertex shader after splitting

Vertex and geometry shader can be split at the point, where vertices of the input primitive are indexed. The function **at** is used for this purpose. Before indexing, the calculations are done for the complete stream of vertices. The function **at** can be moved automatically further into the geometry part. When all inputs of a function use the same index, this function can be evaluated in the vertex shader and its output can be passed into the geometry shader. All unary functions fulfill this criterion trivially.

Example in listing 3 calculates the distance of one vertex of each triangle from the camera (this can be used for example for LOD selection). Because **length** is an unary function, it can be moved into the vertex shader safely. Multiplication with one uniform argument acts as an unary function, too. The transformed program is shown in listing 4.

```
uniform matrix4 modelView;
attribute vector3 position;

— original code
distance = length (modelView*(at position 1))
```

Listing 3: Original code of Vertex and Geometry shader

```
— vertex part
tmp = length (modelView*position)

— geometry part
distance = at tmp 1
```

Listing 4: Vertex and Geometry shader after splitting

When the geometry shader is not present, vertex and fragment shader can be partitioned fully automatically. This approach was used in Renaissance [1], but has some drawbacks. Because the program is practically

written as fragment shader, it is hard to express calculations such as Gouraud shading. Also new versions of shaders provide multiple modes of value interpolation. Because of these reasons, I propose another method.

The point of splitting is specified by one of three functions - **smooth**, **linear** and **flat**. These names come from three interpolation modes on graphical cards. Functions **smooth** and **linear** can be moved further into fragment part by the same manner as in Renaissance. Calculations with all arguments with **flat** interpolation mode can be always moved into geometry (or vertex) shader, because no interpolation is performed. The centroid option does not complicate the transformation, so it is omitted here for simplicity.

The example in listing 5 shows a simplified calculation of specular lighting with phong shading. The geometry shader is omitted for simplicity. The transformation of the light vector is completely uniform. Multiplication is partially linear, so transformation of normal vector can be done in the vertex shader. Normalization is a nonlinear operation, so it must be left in the fragment shader. The light vector can be normalized in the uniform part, because it is not interpolated. The transformed code is shown in listing 6.

```
uniform matrix4 modelView, normalMatrix;
uniform vector3 lightVec;
attribute vector3 normal;

— original code
norm = normalize (normalMatrix*(smooth normal))
lvec = normalize (modelView*lightVec)
color = norm 'dot' lvec
```

Listing 5: Original code of Vertex and Fragment shader

```
— uniform part
lvec = normalize (modelView*lightVec)

— vertex part
tmp = normalMatrix*normal

— fragment part
norm = normalize (smooth tmp)
color = norm 'dot' tmp1
```

Listing 6: Vertex and Fragment shader after splitting

4 PROGRAM TRANSFORMATION

Automatic partitioning of the shader program is not the only important difference between conventional and functional approach. Very useful feature of functional languages are closures, partial application and higher order functions. Closures are nested functions with some variables defined inside the outer function. Partial application means that for example binary function can take one argument and can be used as unary function afterwards. Higher order functions are functions that take another function as a parameter or return it.

All these features significantly improve code expressiveness. Especially higher order functions offer the possibility of sharing code structure, that is hard to

achieve or even not possible in structured languages. For complete implementation of these features, dynamic memory allocation is needed. Since the underlying hardware does not support it now, compiler must convert these features into equivalent structured code. The resulting code is often significantly less elegant, as will be shown in an example. The hardware also limits recursion, which must be limited to a form that can be automatically converted into loops. Sum-types, sometimes called discriminated unions, are also forbidden. Only product types - equivalent to C structures - are usable.

Enriched lambda calculus [5] can be used for program representation. This does not differ from other functional languages. The program is converted into a list of definitions which is topologically sorted. A definition is simply a named expression.

Because shaders do not have capabilities to support lazy evaluation, the program must be converted to an equivalent strict form. Both Vertigo and Renaissance solved this by complete substitution of all free variables in expressions. This approach is simple, but in the result, all common sub-expressions are lost.

In this paper a slightly more complicated approach is used. The program is lambda-lifted [4], so nested and anonymous functions are converted into C-like global functions. Substitution is done only to remove closures and partial applications, not for all variables. Lastly, all applications are merged into complete function calls.

Frequencies are estimated using rules from the previous section. For expressions without user-defined functions, the splitting is trivial. When a user-defined function is present, the frequencies inside it are estimated according to the parameter frequencies. Optionally, this function is also split into parts. Because of this splitting, library functions acting as one piece can be automatically split into multiple parts. This allows the use of library functions that silently cross the boundaries between shader stages and are both compact and effective.

Classical structured code can be now generated from the vertex and fragment part. The geometry part has one list of values for every output variable. To match the structure of the geometry shader, its output must be one list of structures containing every output variable. This conversion in functional languages done by the function **zip**. This function takes multiple lists and converts it to a single list of structures. The length of the resulting list is the length of the shortest input list.

4.1 Larger example

This example illustrates the compilation of a more complex shader program. The uniform variables are *modelView* and *normalMatrix*. The vertex attributes are *vertex* and *normal*. The required output variables are *position* with frequency geometry and *color* with frequency fragment. The source code without declaration of vari-

ables is shown in listing 7. This program transforms the input vertices and normals, splits the triangles into four parts as shown in figure 1 and calculates simple diffuse lighting. The splitting is described by function **gen**. This function is used for position, normal and light vector identically. A real program would add some modification after, but for this example, simple subdivision will suffice; any such complication would not affect the compilation process.

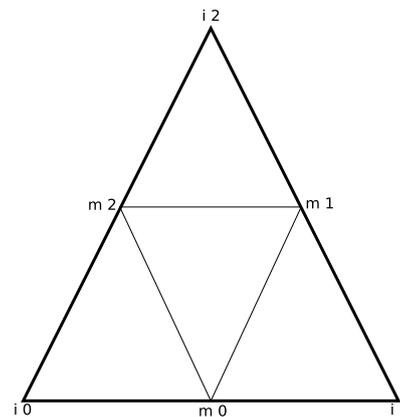


Figure 1: Subdivision of a triangle in the geometry shader in listing 7. The input vertices *i* and generated vertices *m* correspond to the list in the function **gen**.

```
tr_pos = modelView*vertex
tr_norm = normalMatrix*normal

gen i = [[i 0, m 2, m 0, m 1, i 1], [i 2, m 1, m 2]]
      where m x = (i x + i (x+1)%3)/2

position = gen (at ftransform)
lvec = lightPos - (smooth (gen (at tr_pos)))
norm = smooth (gen (at tr_norm))
color = (normalize lvec) `dot` (normalize norm)
```

Listing 7: Code for triangle transformation, subdivision and simple shading

The definitions are already sorted, so no reordering is needed. All expressions depend only on previous definitions. Lambda lifting splits the function **gen** and creates a new function **gen_m**. These two functions are now C-like global functions. The resulting code is shown in listing 8.

```
tr_pos = modelView*vertex
tr_norm = normalMatrix*normal

gen_m i x = (i x + i (x+1)%3)/2

gen i = let m x = gen_m i x in [[i 0, m 2, m 0, m
1, i 1], [i 2, m 1, m 2]]

position = gen (at ftransform)
lvec = lightPos - (smooth (gen (at tr_pos)))
norm = smooth (gen (at tr_norm))
color = (normalize lvec) `dot` (normalize norm)
```

Listing 8: Shader after lambda-lifting. Only the function **gen** differs from listing 7.

Partial applications of functions like *m* in the function **gen** or usages of the function **at** are substituted to places where the remaining arguments are applied. By this substitution, specialized lists for variables *position*, *lvec*, and *norm* are created. The function **gen** itself and the lifted function **gen_m** are removed as a dead code. The resulting code is shown in listing 9.

```
tr_pos = modelView*vertex
tr_norm = normalMatrix*normal

position = [[at ftransform 0, ((at ftransform 2) +
    (at ftransform (2+1)%3))/2 ...
lvec = lightPos - (smooth [[at tr_pos 0, ...
norm = smooth [[at tr_norm 0, ...
color = (normalize lvec) 'dot' (normalize norm)
```

Listing 9: Shader without partial applications and closures

Expression frequencies are estimated, expressions are split, constant expressions are evaluated and common subexpression elimination is done. Vertex and fragment parts are prepared for code generation. Geometry part needs zipping together, which is trivial. Listing 10 shows this situation.

```
— vertex frequency
tr_pos = modelView*position
tr_norm = normalMatrix*normal
tmp1 = ftransform
tmp2 = lightPos - tr_pos

— geometry frequency
position = [[at tmp1 0, ((at tmp1 2) + (at tmp1
0))/2 ...
lvec = [[at tmp2 0, ((at tmp2 2) ...
norm = [[at tr_norm 0, ...

— fragment frequency
color = (normalize lvec) 'dot' (normalize norm)
```

Listing 10: Code parts for each stage of the rendering pipeline

Listing 11 shows the generated code. The interface between the vertex and geometry shader are the variables *tr_norm*, *tmp1* and *tmp2*. The interface between the geometry and fragment shader are the variables *lvec* and *norm*.

```
// vertex shader
tr_pos = modelView*position;
tr_norm = normalMatrix*normal;
tmp1 = ftransform;
tmp2 = lightPos - tr_pos;

// geometry shader
position = tmp1[0];
lvec = tmp2[0];
norm = tr_norm[0];
emitVertex();
position = (tmp1[2] + tmp1[0])/2;
lvec = (tmp2[2] + tmp2[0])/2;
//... too long

// fragment shader
color = dot(normalize(lvec), normalize(norm));
```

Listing 11: Code equivalent to listing 7 in the target structured language

The final code does not contain the interfaces between shader stages, because they are straightforward. The code for the geometry shader was shortened, because all vertices are generated nearly identically. In classical languages, the structure of generated vertices cannot be shared, so the resulting code must be written by hand or generated by some preprocessing tool.

5 CONCLUSION AND FUTURE WORK

This paper presented a functional approach to the geometry shader programming. This approach has some interesting properties that are hard to achieve in conventional structured languages.

One program is written for all shader stages and the compiler does the necessary partitioning and interface generation. This simplifies the programmer's work, as he can write the code, where it logically belongs and let the compiler move it for an efficient execution.

Higher order functions allow the programmer to write the code more abstract. Abstract code often tends to be shorter and more readable. The code sharing is possible at a level that is hard to achieve by traditional languages.

Automatic partitioning of program also helps modularity. Library functions can be viewed as single blocks by the programmer, but parts of them can be executed in different stages of the pipeline.

These properties significantly improve the shader programming. However it is not likely that so massive shift of used paradigm could occur. Because of that, following work will focus on selecting useful parts that could be used to extend existing languages.

REFERENCES

- [1] Chad Austin and Dirk Reiners. Renaissance : A functional shading language. *Graphics Hardware*, 2005.
- [2] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. OSDI'04: Sixth Symposium on Operating System Design and Implementation, December 2004.
- [3] Conal Elliott. Programming graphics processors functionally. In *Proceedings of the 2004 Haskell Workshop*. ACM Press, 2004.
- [4] Thomas Johnsson. Lambda lifting: Transforming programs to recursive equations. pages 190–203. Springer-Verlag, 1985.
- [5] Simon Peyton Jones. *The Implementation of Functional Programming Languages*. Prentice Hall, 1987.
- [6] Simon Peyton Jones. Haskell 98 language and libraries: The revised report, 2003.
- [7] Khrohos Group. *OpenGL API and Shading Language Specification*, August 2009.
- [8] Microsoft Corporation. *DirectX Reference*, 2009.
- [9] NVIDIA Corporation. *NVIDIA GPU Programming Guide*, May 2009.
- [10] Pedro V. Sander, Diego Nehab, and Joshua Barczak. Fast triangle reordering for vertex locality and reduced overdraw. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 26(3), August 2007.

Visual Odometric Navigation: the Generalized Feature Vector way

J.Bhattacharya

CMERI

Mahatma Gandhi Avenue

India 713209,Durgapur,West Bengal

[bjhilik@cmeri.res.in](mailto:bjihilik@cmeri.res.in)

S.Majumder

CMERI

Mahatma Gandhi Avenue

India 713209,Durgapur,West Bengal

sjm@cmeri.res.in

ABSTRACT

One of the main challenges faced by object tracking and environment-modeling techniques is the frame-to-frame correspondence of the object of interest. False detections may lead to the tracking of wrong object thus misrepresenting information about the object location and its track. The tracking algorithm of the detected object should also be computationally inexpensive and suitable for real time applications. This paper discusses how GFV, a multidimensional entity encapsulating multiple feature parameters, can uniquely identify dominant features of an object, and increase the detection reliability due to its potential to function consistently in any kind of environment, uninfluenced by view point invariance or extrinsic factors, thus generating minimal false alarms. Further a method to determine the 3D position of the object is presented which works on uncalibrated camera images and can be successfully applied to online processes. Experimental analysis using a outdoor mobile robot have been carried out to establish the competence of the algorithm. A statistical approach to reject outlier data, if any, is applied while generating the trajectory of the mobile robot used for experiments

Keywords

Feature detection, trajectory identification, object tracking.

1. INTRODUCTION

The implementation of vision based automated systems in various fields like security, surveillance, robot navigation, remote environment sensing and medical diagnosis is in the continuous evolvement of research in tandem with the field of object tracking and environment modeling. The task of tracking encapsulates within it primary operations like image segmentation, object detection and extraction, depth estimation and finally, object trajectory estimation. The main challenge faced by the detection techniques lies in the frame-to-frame correspondence of the regions of interest; which becomes difficult for non-rigid objects exhibiting complex motion, or in frames where the object is occluded, or when the scene illumination is extremely influenced by environmental conditions. Mainly two approaches are taken in the vision based correspondence problem solving. These are area based and feature based techniques. Detection of feature from exteroceptive

sensors has remained an important area of research for several reasons. Firstly it provides the unique opportunity to abstract and encapsulate the dominant and distinguishable characteristics of the environment or scene from the sensory data. Secondly it is a process of reducing the resource requirement and the associated complexity of handling large data sets in real-time. Often features are defined as geometric primitives such as point, line, arc segments or some form of derived entities from the amplitude return history such as color and texture for example. In general, features segregate “objects of interest” from the raw sensory data. Various algorithms have been proposed by different researchers for object detection and depth recovery.

The progress of research in the field of feature detection using vision can be mainly categorized into four distinguishable classes. In the initial stage researchers mainly concentrated on detecting geometric features like edge and corners of the image to identify objects of interest. A large amount of work has been undergone in this area [1, 2]. The problems of most of these algorithms lie in the fact that they are not invariant to affine transformations and are also viewpoint dependent. The second class of algorithms uses primitive geometrical shapes as

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

features. General methods for shape recognition are moment based, structure based and Fourier descriptor based [3]. The next class of algorithms models the object as probabilistic distributions. These distributions represent features such as color [4], texture [5] etc. Another variation to color and texture detection of a region is background modeling which is beneficial in detecting only moving objects [6]. For the fourth class of algorithms, the object shape and appearance are generated simultaneously. These models also encode different views of an object, removing the shortcoming of viewpoint dependencies of the previous methods [7]. In spite of many uniqueness and advantages, most of these methods require large computational power and hence unsuitable for real-time navigation and tracking application. Another limitation is that many of these methods have been developed for a specific sensor suite and well-structured indoor environment for specific applications and consider camera calibration as a prerequisite rendering them unsuitable for outdoor and unstructured environment where extrinsic parameters are dominant rather than intrinsic. Present work extracts features by detecting and representing them in a generalized feature vector (GFV), which can be used to uniquely identify each of the dominant objects in an image. Once features are detected using GFV, the next important task lies in estimating the depth measurement of the object of interest. In the past few years several techniques for depth recovery and construction of depth maps have been developed. This area is still an active research area and development in this field is in continuous progress. The issue has been investigated by different researchers from different viewpoints but can be categorized mainly into six main classes. The first class includes all methods, which are based on depth measurement from two cameras. Finding corresponding points between the two images precedes depth calculation while using stereo [8]. The second class comprises of methods that use simple geometry to recover depth information [9]. The next class of algorithms are those that derive depth information of the targets from the velocity estimation of the targets [10]. The fourth class of algorithms consider calculation of depth from optical blur, defocusing techniques [11]. The next class uses interpolation functions for depth estimation [12]. The last class comprises of those methods which use auxiliary devices such as laser range finders or ultrasonic sensors to measure depth.

As a significant departure, the work reported here uses the image magnification to estimate the depth and thereby compute the trajectory. The main interesting issue of this algorithm is that it “does not require” explicit camera calibration “for depth

recovery”. The paper is organized in the following manner. Section 1 provides basic background of the problem. This section also includes an outline of various significant work carried out for consistent feature detection and depth estimation. Section 2 defines the GFV framework and its comparison with other conventional approaches briefly, whereas Section 3 includes the position determination of the detected object for trajectory development. Section 4 deals with results and performance analysis of experimental findings. Finally, discussion and conclusion of this work is presented in Section 5.

2. THE GFV FRAMEWORK

The basic idea of using GFV as a scene descriptor stems out of the fact that point features often require a secondary level of corroboration such as color and texture to make it invariant. The generalized feature vector (GFV) is considered to be a multidimensional entity, which can include multiple parameters like color, shape, energy, entropy, size ratios and many more. Some of these parameters may be orthogonal to the other. In principle GFV can include as many parameters as desired. Another uniqueness of GFV is that it can also accommodate “feature parameters obtained from other co-located sensors”. There is no limit on how many feature parameters can be included in GFV. Although inclusion of multiple parameters can improve the detection reliability it however may increase the computation cost. Therefore for optimal performance not more than three parameters should be used. However the actual number of parameters will depend on the application requirements and available computational resources. Figures 1 & 2 shown in the appendix at the end of the paper, further demonstrates the algorithmic flow of the GFV briefly using a sample image. The method mainly consists of two steps: - During first step a reference model of GFV is created which is then applied to the actual data in the second step. The details of the algorithm and its establishment have been discussed in the reference [13] and are beyond the scope of this paper.

The suitability of GFV lies in the fact that even when any information about the environment of the object to be detected or presence of other objects in its surrounding is not known, the method will provide reasonably accurate results instantly without false alarms. The user need not have to decide which features are to be matched or in which order they are to be matched in order to get the best matching. Thus GFV is self-deciding and can operate independently in any environment without any prior knowledge about it. Failures of many object detection algorithms, mainly due to view point invariance; occlusion and influence of other extrinsic factors can

be successfully resolved by the GFV. GFV also responds very well even in outdoor environment. Similar objects can be identified from a sequence of images taken at different time in different environmental conditions. Since GFV is essentially a multi parametric matching method, it is more robust compared to any other step-by-step matching algorithm.

3. POSITION CALCULATION

Any camera image is a 2D projection of the 3D world using perspective transformation. As a result, estimation or recovery of object distance from the camera requires elaborate mathematical procedure. Various depth detection algorithms using monocular camera and their relative merits have been already mentioned in section 1. In this section an alternative technique using the thin lens equation and the image magnification factor (shown in equation 1 below) is used to calculate the depth. This technique is suitable for online processes and doesn't require large computational overhead. For computing the object depth for every image frame, the magnification ratio is estimated for each image frame from its object and image dimension ratio. This method has only one limitation i.e. object shape; size and approximate dimensions should be predefined. The image dimensions like area, perimeter, shape and size ratios are already computed while detecting the object as seen in section 2. Any of the above mentioned dimensions may be used but the choice should be kept fixed for all the image frames. The depth estimation procedure is further illustrated below:

The thin lens equation gives

$$\frac{1}{u} + \frac{1}{v} = \frac{1}{f} \quad (1)$$

where u is the image distance, v is the object distance required to be calculated and f is the focal length of the lens. The magnification ratio m , is given by

$$m = \frac{u}{v} = \frac{I}{O} \quad (2)$$

Here, I and O give the image size and object size respectively. Substituting u as mv in equation 1, v can be written as

$$v = \frac{(1+m)f}{m} \quad (3)$$

While computing the depth d_n for each of the camera frames n using equation 3, there are two factors that should be resolved. Firstly obtaining the image size for computing the magnification factor should not consider the total surface of the extracted image. The part of the image to be considered for a particular frame is variable and depends on the viewpoint of the camera for that image frame. This fact is further

explained using figure 3 and 4 and subsequently elaborated in the discussion. Secondly, the depth dimension is obtained relative to the camera frame and need not be considered as the actual object distance relative to a fixed world coordinate system. Reason behind this approach is that the camera may be positioned and maneuvered using pan and tilt angle hence making the camera plane rotated with respect to the world frame. Further this depth cannot be associated with the depth dimensions of the other camera frames for trajectory identification as each frame may have a different orientation i.e. different pan and tilt angles of the camera and hence each of the depth dimensions refers relative distance measurement with respect to different camera planes. To obtain the actual depth in the world frame, the calculated depth in each frame needs to be transformed to the world coordinate system. In order to carry out this transformation, the knowledge of the extrinsic camera parameters is necessary for each image frame, which can be obtained through camera calibration. However, for real time applications the procedure becomes complex and time taking. The following paragraphs explain how the problems mentioned above are addressed.

Initially an example is used to illustrate how the appearance of an image of any particular object changes along with the viewpoint or rotation angles of the camera. This further helps to point out how the calculation for the image magnification depends on the viewpoint of the camera. Figure 3 below depicts a rectangular box viewed by the camera from three positions identified as 1, 2 and 3 respectively.

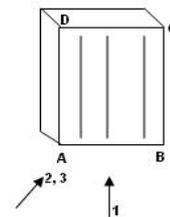


Fig 3: A rectangular object seen from three different camera positions 1, 2 and 3 are shown in the figure above

Position 1 assumes the camera to be perfectly aligned with the world frame therefore no rotation is considered. Positions (2) and (3) denote the same camera position however the camera angles are different. Position 2 considers a pan angle whereas position 3 assumes the camera frame to be rotated by both pan and tilt angles. The resulting image appearance for each of the camera positions is depicted in figure 4. For calculating the magnification ratio for figure 4a, the total surface of the image is to be considered; however for the

remaining two images (4b and 4c) of the figure or for any similar case where more than one of the side faces are visible, such a step would provide wrong results. Thus for correct magnification determination the surfaces needs to be separately identified in order to select the desired one among them. The GFV method discussed previously can easily separate out the region of interest of the object as it can detect all the outer corners of the image from which the boundary edges can be calculated. The selection of the corners to calculate the edge, which will denote the image size, depends on the shape of the object and will vary accordingly. In this particular case for different camera positions two bottom edges (bottom edges are used here just as an example, top or side edges can be used as well) may be detected. For example if figure 4c is considered, the two bottom edges detected are $E'A'$ and $A'B'$. As the object dimensions are known, one of the detected edges can now be selected depending on their length, i.e. if the matching is to be done with object side AB , then the longer among the two detected edges (considering side $AB > side EA$) will be chosen or vice versa. This part is conferred in details while discussing trajectory identification case studies later in the paper.

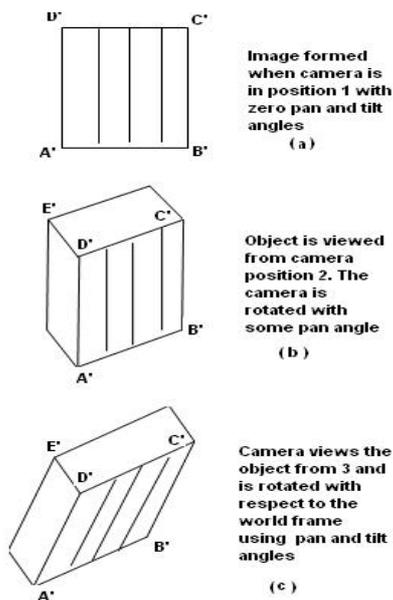


Figure 4: The image of the rectangular object formed for the three camera viewpoints depicted in figure 3 is seen above

The next task is to compute the camera rotation angles relative to the first frame so that the trajectory can be identified. Before getting into the details of the rotation angle computation process, Figure 5 depicts the rotation of the camera plane with respect to the world frame for camera position 3 of figure 3. This figure is used to establish the impact of the

camera rotations on its corresponding image frames. The relation between the rotation angles of the camera plane and its resultant image frame is discussed in subsequent paragraphs.

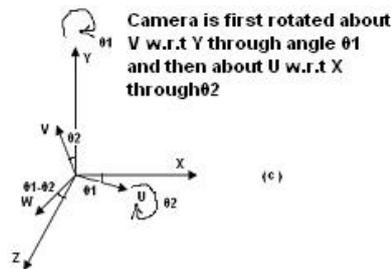


Figure 5: The camera plane orientation for the camera position 3.

In figure 5 the world frame is depicted by XYZ plane and UVW depicts the camera plane. The corresponding image frame for the camera orientation in figure 5 is depicted in Figure 6. $(PQRS)_1$ here depicts image frame 1 and $(PQRS)_n$ depicts the n^{th} frame. The first image frame is considered as the reference; hence it is assumed that the camera plane of the first frame is aligned with the world frame and all the other camera plane rotations are with respect to this reference frame. The consecutive camera plane rotations of figure 5, by angles θ_1 and θ_2 , effects the x -axis and y -axis of its image frame (n^{th} frame) to make θ_1 and θ_2 angles with the x and y axis of the first image plane respectively as shown in figure 6.

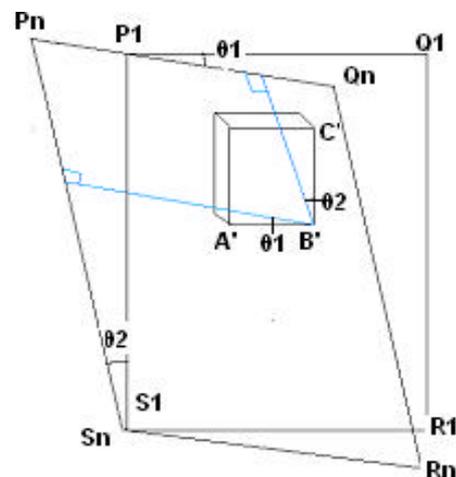


Figure 6: Camera frames $PQRS_1$ and $PQRS_n$ and their alignment is shown above in order to compute the rotation angles

Once these rotation angles are computed, the transformation from the n^{th} frame to the first frame can be undergone. As the camera is aligned with the world frame in the first image frame i.e. the pan and tilt angles of the camera is zero hence the actual depth can be obtained after this transformation. The

trajectory can still be generated even if the first frame is not aligned with the world plane as the relative rotations of all the frames with respect to the first one is computed and the transformation is carried out accordingly. But for such cases the actual depth cannot be determined.

From figure 6, using parallel line properties, it can be seen that angle between side $A'B'$ and the perpendicular from point B' on P_nS_n equals θ_1 and the angle between side $B'C'$ and the perpendicular from point B' on P_nQ_n equals θ_2 . If the coordinates of A' , B' and C' are given by (x_na, y_na) , (x_nb, y_nb) and (x_nc, y_nc) respectively the angles can be calculated from the figure using the following relations.

$$\theta_1 = \tan^{-1} \left(\frac{\text{abs}(y_na - y_nb)}{\text{abs}(x_na - x_nb)} \right) \dots (4)$$

$$\theta_2 = \tan^{-1} \left(\frac{\text{abs}(x_nc - x_nb)}{\text{abs}(y_nc - y_nb)} \right) \dots (5)$$

Thus the camera angles can be determined for every image plane from the above relations once the points a_n , b_n and c_n for every frame is determined. This concept is further used to identify the object trajectory. Two different cases of trajectory identification are discussed:

a) path generated by an object in a situation where the camera is fixed throughout,

b) path generated by a moving camera while tracking a fixed object.

A detailed discussion of the two cases is further presented below.

(a) Camera stationary, object moving

The following steps are executed while identifying the object trajectory:

1. The object of interest is extracted using the GFV algorithm discussed in section 2. The object is denoted in the image plane by its centroid position 1O in every frame n

$${}^1O = (x_n, y_n) \quad (6)$$

2. Five image corners (x_nlb, y_nlb) , (x_nrb, y_nrb) , (x_nrt, y_nrt) , (x_nbl, y_nbl) , (x_nbr, y_nbr) are determined. These points are the leftmost bottom, rightmost bottom, rightmost top, bottom leftmost and bottom rightmost pixels coordinates of the detected object and are used to determine image size. An algorithm below presents how corners can be selected for calculating image size of a three dimensional rectangular box when camera rotations are unknown.

```

if (xbl > xlb) && (xbl < xrb)
  corners[ ] = {xlb, xbl, xrb}
else if (xbr > xlb) && (xbr < xrb)
  corners[ ] = {xlb, xbr, xrb}
else corners[ ] = {xlb, xrb}
end
if size(corners) > 2

```

```

if(length(corners[1],corners[2]))>(length(corners[2],corners[3]))
  corners[3] = []
else
  corners[1] = []
end
end
end

```

From the algorithm, two corners (x_n1, y_n1) and (x_n2, y_n2) are selected based on the fact that the larger edge is used to calculate the magnification. The reverse can also be done if desired. The length of the edge formed by these corners can be used as the image size 1SZ_nx for determining the magnification ratio, 3D position of the object (discussed in step 4) and rotation angles of the camera. (Rotation angles will not be required for this case as the camera is fixed for all the frames). The equation 7 is used to calculate the image sizes 1SZ_nx and 1SZ_ny in pixels along the x and y axes respectively.

$${}^1SZ_nx = \sqrt{(y_n1 - y_n2)^2 + (x_n1 - x_n2)^2} \quad (7)$$

$${}^1SZ_ny = \sqrt{(y_nrt - y_nrb)^2 + (x_nrt - x_nrb)^2}$$

Magnification ratio can be calculated using the metric coordinates of the image size and the corresponding object side dimension.

3. Depth d_n is computed using equation 3.

4. The 3D coordinates of the image point 1O are given by the following equation:

$$[X(n) \ Y(n) \ Z(n)] = [x_n, SZ_nx / {}^1SZ_nx \quad y_n, SZ_ny / {}^1SZ_ny \quad d_n] \quad (8)$$

where SZ_x and SZ_y are the object sizes along the x and y dimension respectively. The 3D point calculated lies in the camera plane.

5. For graphical representation, the X and Z coordinates are used to denote the horizontal displacement and depth of the object respectively, the vertical displacement of the object i.e. the Y coordinate is not taken into consideration at present. Its utility will be later understood while discussing case (b). As the camera is fixed for all the image frames, all the n points $(X(n), Z(n))$ lie on the same XZ plane and thus can be plotted to identify the trajectory generated by the object.

(b) Camera moving, object stationary

When a moving camera captures a video of a fixed object then the displacements (change in centroid position) of the object observed in the image frames is due to the movement of the camera from frame to frame. This camera movement is calculated from these centroid displacements for identifying the trajectory generated by the camera. Initially the object is detected and the image sizes, depth and 3D Coordinates are calculated using equations 7, 3 and 8 respectively. The 3D points calculated for each frame lies on a different camera plane as the camera is in constant motion. The camera is considered to be positioned at the origin of a fixed reference frame for

the first image frame. The 3D coordinates of the first and nth frame can be related by rotation and translation matrices as shown in equation 10, where the rotation matrix denotes the camera rotation of the nth frame relative to the first frame and the translation matrix denotes the translation of the camera from the fixed origin. The experimental results given later in this paper use a set-up where the camera is fixed on a tripod mounted on a trolley. Thus only pan angle change is considered in the calculations. Using the selected corners (x_n1, y_n1) and (x_n2, y_n2) calculated in Step 2, the pan angle θ can be calculated using equation 9.

$$\theta = \tan^{-1} \left(\frac{ab(y_n2 - y_n1)}{ab(x_n2 - x_n1)} \right) \quad (9)$$

The affine transformation of the camera from the fixed frame to the nth frame is given by:

$$\begin{bmatrix} X(n) \\ Y(n) \\ Z(n) \end{bmatrix} = \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix} \begin{bmatrix} X(1) \\ Y(1) \\ Z(1) \end{bmatrix} - \begin{bmatrix} x'(n) \\ y'(n) \\ z'(n) \end{bmatrix} \quad (10)$$

If $y_n1b < y_nrb$ then θ is positive else it is negative. It is clear from equation 10 above that (x_n', y_n', z_n') is the translated origin of the camera plane in the nth frame and hence the required camera displacement. Equation 10 can be written as

$$\begin{aligned} x_n' &= X(1) \cos \theta - Z(1) \sin \theta - X(n) \\ z_n' &= X(1) \sin \theta + Z(1) \cos \theta - X(n) \end{aligned} \quad (11)$$

Once (x_n', z_n') is calculated using equation 11, plotting it for all the n image frames gives the calculated camera trajectory.

4. RESULTS AND PERFORMANCE ANALYSIS

The trajectory generation for moving objects or moving camera has been accomplished using the proposed method and some of the results are shown.

(i) Experiment 1:

Figure 7 represents a scene where the object is fixed and the camera is in motion.



Figure 7: White mark in the center shows the path followed by the camera mounted on a trolley. The track line was created using a white marker while pushing the trolley at an approximate constant speed.

The generated plot shown in figure 8 is a smoothed plot using the best polynomial fit. The best fits of the

plot are estimated using the norm of residuals of the fits and are again crosschecked by determining the R-Square values for each fit. The observed values are shown in Tables 1 and 2 respectively.

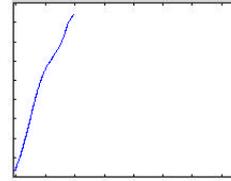


Figure 8: The calculated trajectory of the path shown in figure 7 using the present approach

TYPE	ORDER	NORM	StD
Poly	2	35.3552	2.2405
Poly	4	27.24	1.72
Poly	6	25.43	1.61
Poly	8	21.2	1.34
Poly	10	21.2	1.34

TABLE 1: Norm of Residuals.

It is seen that the norm of residuals converges after the eighth order fit. Coefficient calculation with 95% confidence bound and normalization by a mean of 5.833 and StD of 5.753 gives the corresponding R-square values.

TYPE	ORDER	SSE	R-SQUARE
Poly	2	1249.9	0.9921
Poly	4	741.81	0.9953
Poly	6	646.75	0.9959
Poly	8	452.557	0.9972
Poly	10	449.553	0.9972

TABLE 2: R-Squares values

Though the SSE values and the standard deviation decreases as the order of the fit increase, but the goodness of the fit (judged by the R-Square value) remains same after the 8th order fit. Hence for both the best-fit estimation techniques the eighth order fit is the optimal fit for the curve.

(ii) Experiment 2:

The next case shows generated object trajectories when the camera tracks a moving object from a fixed position.



Figure 9: White circular path depicts the path followed by the red object

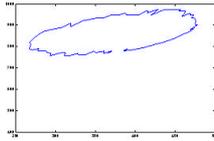


Figure 10: The generated trajectory of the path in fig 9

Similar to the previous case, a moving average filter is used to smooth the plot. It is seen that the three point averaging filter gives the best fit. The fact is further demonstrated in table 3 .

TYPE	NORM	StD	SSE	R-SQUARE
3 point	50.73	4.15	2574.5	0.9971
5 point	64.92	5.31	4215	0.9952

TABLE 3: Best-Fit Estimation

It is observed that the residuals diverge from 3-5 point averaging.

(iii) Experiment 3:

The following experiment was carried out with the All Terrain Robot developed at CMERI Durgapur during its testing on the grounds of the institute. The figures depict the identified trajectory (depicted by the blue colored plot) of the path traversed by the ATR.



Figure 11a: Trajectory identified after rejecting outlier data using averaging window of 4σ gate



Figure 11b: Trajectory identified after rejecting outlier data using 6σ gate



Figure 11c: Trajectory identified after rejecting outlier data using averaging window of 4σ gate

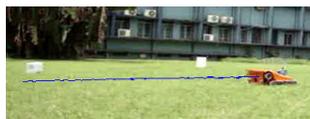


Figure 11d: No outlier detected

The statistical cut-off values were selected after estimating the rejection percentage for a gate of 3σ ,

4σ , 5σ and 6σ for figures 11a,b and c. Table 4 shows the rejection rates for the figures. A 7% rejection was considered to be the maximum allowable rejection rate and choice of the cut-off was made accordingly.

Figures/Gates	3σ	4σ	5σ	6σ
11a	8.25	6.5	6	5.5
11b	84.14	59.14	7.85	2.1
11c	23.625	1.125	1	1

TABLE 4: Rejection rates for different statistical cut-off gates

5. DISCUSSIONS AND CONCLUSION

This paper presents an odometric navigation using uncalibrated camera images. The proposed methodology relies on a simple but elegant approach for consistent feature detection using GFV method. These features are then used for generation of visual odometry of any mobile robot. The indoor and outdoor field experiments show that this is a more resilient and computationally efficient approach which can be used to resolve navigation problems. Work is in progress for online implementation of this methodology for autonomous navigation of an unmanned aerial robot project currently pursued by CMERI.

6. REFERENCES

1. H. Moravec , Visual Mapping By A Robot Rover, In Proceedings Of The International Joint Conference On Artificial Intelligence (Ijcai) 1979, 598–600.
2. Canny, A Computational Approach To Edge Detector, IEEE Transactions On Pami, 1986, Pp679-698,
3. D. Cremers, and C. Schnorr, Statistical Shape Knowledge In Variational Motion Segmentation,,Israel Nent. Cap. J, 2003, 21, 77–86.
4. Withagen, Klammer Schutte and Frans Groen, Likelihood Based Object Detection and Object Tracking Using Color Histograms, Proc. Icip2002, September 22-25, Rochester, New York
5. M. J. Nassiri, A. Vafaei, and A. Monadjemi, Pwaset , Texture Feature Extraction Using Slant-Hadamard Transform ,Volume 17 December 2006 Issn 1307-6884
6. C. Wren, A. Azarbajani, and A. Pentland, 1997. Real-Time Tracking Of The Human Body, Pfunder IEEE Trans. Patt. Analy. Mach. Intell. 19, 7, 780–785
7. S.Park, and J.K Aggarwal, A Hierarchical Bayesian Network For Event Recognition Of Human Actions And Interactions. Multimedia System,2004, Volume-10,Issue- 2,Pages 164–179
8. Stan Birchfield and Carlo Tomasi, Depth Discontinuities by Pixel-to-Pixel Stereo, International Journal of Computer Vision, Volume 35 , Issue 3, December 1999,Pages: 269 – 293,ISSN:0920-5691.
9. Y. L. Murphey, J. Chen, J. Crossman, J. Zhang, P. Richardson, and L. Sieh, .Depth nder, A Real-time Depth Detection System for Aided Driving, IEEE

Intelligent Vehicles Symposium Proceedings, October 2000.

10. Wietske I. Meyerind, Marco A. Gutierrez, Skrgio S. Furui, Marina S. Rebelo, Chdido P. Melo, Spatiotemporal-Frequency Analysis Applied to Motion Detection Proceedings of the 22nd Annual EMBS International Conference, July 23-28, 2000, 1720-1723.
11. Murali Subbarao, Tai Choi, Arman Nikzad, Focusing Techniques, OE/Technology SPIE Conference

12. Mirzabaki Mahdi, A New Method for Depth Detection Using Interpolation Functions, WSCG posters proceedings, February 2-6, 2004, Plzen, Czech Republic
13. J. Bhattacharya and S. Majumder, The Generalized Feature Vector (GFV): A New Approach for Vision Based Navigation of Outdoor Mobile Robot, 14th National conference on Machines and Mechanisms (NaCoMM-2009), NIT Durgapur, India, December 17-18, 2009

APPENDIX

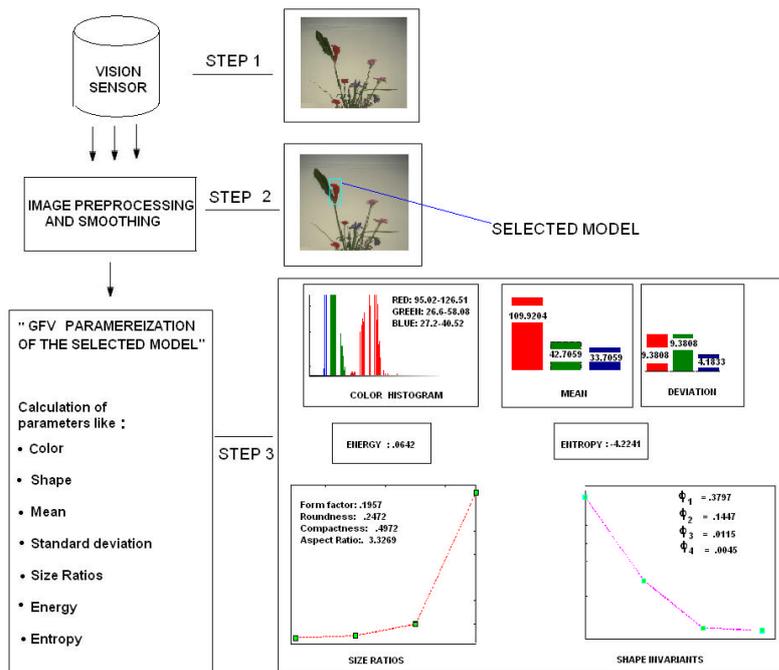


Figure 1: Reference model creation

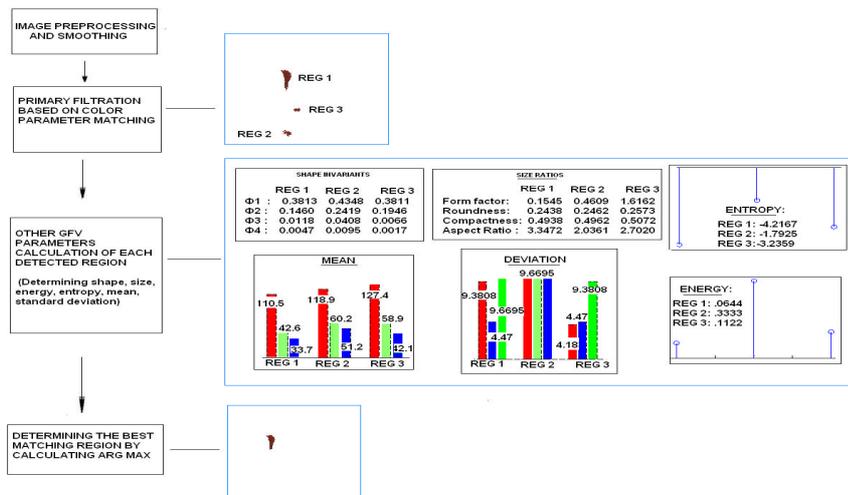


Figure 2: Application of GFV to experimental dataset.

The interpretation of X-ray Computed Microtomography images of rocks as an application of volume image processing and analysis

Kaczmarczyk J., Dohnalik M., Zalewska J.

Oil and Gas Institute
Well Logging Department
ul. Lubicz 25 A
Poland, 31-503 Kraków
jan.kaczmarczyk@inig.pl

Cnudde, V.

The Center for X-Ray Tomography
Department of Geology and Soil Science
Ghent University
Krigslaan 281, S8
Belgium, 9000 Ghent

ABSTRACT

X-ray computed microtomography (CMT) is a non-destructive method of investigating internal structure of examined objects. During the reconstruction of CMT measurement data, large volume images are generated. Therefore, the image processing and analysis are very important steps in CMT data interpretation.

The first step in analyzing the rocks is image segmentation. The differences in density are shown on the reconstructed image as the differences in gray level of voxel, so the proper threshold operation must be carried out. As a result, the different mineral phases and pores can be separated at the image.

Segmented and binarized image is the base for further operations which depend on the aim of research.

Numerical analysis gives information about the pore shapes and volumes as well as connections between pores in the pore network.

The image may also be used in numerical physics simulation (for example fluid flow simulation), but before that it has to be filtered and resampled. These operations are very important, because if performed poorly, they may lead to rupture the pore network.

The aim of this paper is to present authors' methodology of CMT image processing and analysis and to show problems occurring during these processes. The image processing of two rock samples CMT image will be presented.

Keywords

tomography, CMT, volume image, segmentation, image analysis

1. INTRODUCTION

1.1 X-ray Computed Microtomography

The foundations of microtomography were developed shortly after discovering the X-rays by Wilhelm Röntgen. In 1917 Johann Radon proposed the theory of computed tomography (CT)[Hsi03,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Rec08] - mathematical reconstruction of object's internal structure based on infinite number of its X-ray projections. On the basis of this theory EMI Scanner - the first medical CT scanner - was used to brain imaging in 1968. In 1970s medical scanners were used to rock cores imaging. Due to relatively low resolution (in order of mm) of these scanners, in 1990s the computed microtomography (CMT, micro-CT) systems were developed[Cnu06]. These systems, with resolution down to 0.4 μm , have a different geometry, with rotating examined object and stationary X-ray source-detector line. Additionally, the X-ray spot and detector pixel were reduced in order to increase resolution[Ket01, Kac08].

Scheme of CMT measurement and data processing was shown in figure 1[Fer07].

Every tomography measurement includes two steps: the data acquisition and the image reconstruction. During the data acquisition the set of 2D object's projections are collected. The gray level of every point of projection is determined by the Beer's law for complex materials[Ket01]:

$$I = I_0 e^{-\sum_i(\mu_i \cdot l_i)}, \quad (1.1)$$

where the I_0 is the initial X-ray intensity, I - beam intensity after passing the object, μ_i - linear attenuation coefficient and l_i - linear extent of i material. At the reconstruction step the internal structure of the examined object is calculated as a superposition of recorded projections.

After reconstruction process, the 3D gray-scale structure of object is obtained. The gray level of each point is proportional to linear attenuation coefficient μ_i of the material and it is (in case of X-rays) proportional to the material's density. The brighter voxel, the higher density of material in the volume element of object.

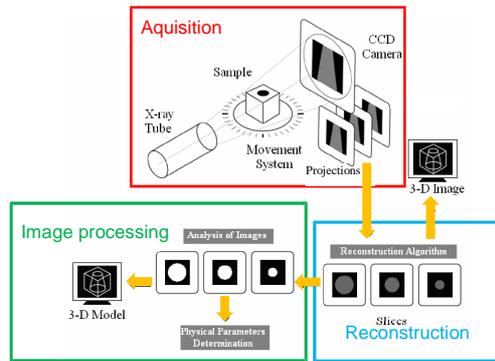


Figure 1. Principle of CMT measurement and data processing [Fer07]

1.2 Geophysical Aspect

Porosity and permeability are important properties of reservoir rocks[Sal03]. Porosity (ε) is the percentage of the sample's volume that is occupied by air (and may be occupied by some fluid):

$$\varepsilon = \frac{V_a}{V_s} = \frac{V_a}{V_r + V_a}, \quad (1.2)$$

where V_a is a volume occupied by air, V_s - sample's volume, and V_r - a volume occupied by a rock matrix. It is measured by gas absorption, mercury porosimetry or density measurements. Permeability (κ) is the ability of the material (e.g. rock) to transport fluids[Har00]. It is determined experimentally in the permeability test with use of Darcy's law:

$$u = -\frac{\kappa}{\eta} \nabla p, \quad (1.3)$$

where u is velocity field, η - dynamic viscosity of fluid, and ∇p - the gradient of pressure at the examined sample. The permeability may also be

calculated with the use of microscopic properties of the examined material as:

$$\kappa = C \cdot d^2, \quad (1.4)$$

where C is the dimensionless constant describing pores geometry and d is the average effective pore diameter.

Both properties may be calculated from CMT measurements[Nar09], but the acquired image has to be treated in specific way.

1.3 Image Processing

As a result of image reconstruction, the volume gray scale image is obtained. This is rather big data set (about 10 GB per every measurement), so every treatment which reduces its volume without loss of quality is desirable. In fact, the image processing depends on the aim of CMT imaging.

The first and the most important step of CMT image analysis is image segmentation. The initial gray-scale image must be divided into different phases - e.g. pores and different rock phases in case of rock's analysis. Three different segmentation techniques will be described later.

Analysis of CMT image requires an image containing a large amount of details. Fortunately, image analysis is not CPU and memory consuming process and image simplification is not required.

Dues to large volume of CMT data, for fluid flow simulation it is necessary to reduce the volume of reconstructed image. This may be done by pores' extraction and image resampling. The problem is the loss of information during image simplification. In the extreme cases image processing may lead to the rupture of the pores' connections, which results in producing false results of the simulation.

1.4 Image Analysis

One of CMT's advantages is the ability to show the real pore's shape and size (which is impossible with the use of conventional methods), so it is necessary to find the way of analyzing this features of the segmented image.

Pore's size may be easily described after its labeling. In this process every group of connected (in determined neighborhood) voxels (which is equivalent to pore) is labeled as another object and has a different gray level assigned. Then, properties of every object (as size or shape) may be easily described.

1.4 Scope of research

The aim of this article is to present CMT image processing and analysis. Analysis becomes a standard method of rock's characterization during mine survey. Image processing is an important step of

preparing CMT-based fluid flow simulation, which is the current topic of the authors' studies.

2. EXPERIMENTAL

2.1 CMT Equipment

The measurements were performed on X-Tek Benchtop CT-160Xi microtomograph. The current at Cu lamp was 60 μ A and voltage was 110 kV. The Varian PaxScan 2520V detector was used. The rock samples were in the form of core with a diameter of 10 mm. During every scan about 3000 projections were made with step about 0.12°.

2.2 Examined samples

To present the application of CMT measurements, two rock samples were chosen. The first one, sample 1, was a rock core excavated from oil-bearing area and its porosity (calculated from density measurements) was 29,45 %. The second one, sample 2, was a rock core with porosity 4,90 %.

Examined image size was 1000×1000×400 voxels in case of sample 1 and 556×951×552 voxels in case of sample 2.

2.3 Data Processing

The internal structure of the examined rocks was reconstructed with the use of Benchtop CT-Pro Client software with Feldkamp's algorithm[Fel84] for cone-beam experiment. The voxel size of reconstructed image was 5.8×5.8×5.8 μm^3 .

ImageJ[ImJ] was used for the histogram calculation.

VSG Avizo 6[Avi] software was used for the image segmentation and visualizations. Images were filtered by unsharp masking, segmented with the use of the threshold tool, and then islands (up to 2 voxels, 25 %) were removed. Before visualization, the images were resampled by factor 2. The surfaces were generated with constrained smoothing.

MAVI 1.3.1[MAV] (Modular Algorithms for Volume Images) software was used for pores' size analysis and pores extraction. The sample was binarized, labeled at neighborhood 26/8 and then the objects' features were calculated. The image was also divided into 6 pore classes according to their volume (table 1).

3. SEGMENTATION

3.1 Thresholding Techniques

Three different threshold techniques developed on the basis of [Mor00] were used.

First, threshold along boundaries, was used for segmenting CMT picture into pore network and rock matrix. On the histogram the minimum was found and this gray value was marked as Th_{min} . Next, the points with gray value $Th_{min}\pm 5$ were selected on the analyzed image (and the boundary between pore and

rock was marked). In the neighborhood-8 of 10 of these points, points with gray value of $Th_{min}+20$ were selected. The average of averages of all the selected points gray levels was adopted as threshold value (Th_B).

class	volume /voxels	colour
I	1-9	Yellow
II	10-99	Blue
III	100-999	Red
IV	1000-9999	Green
V	10000-99999	White
VI	> 100000	violet

Table 1. Pore's classes and colours of its visualization.

The phase-mean threshold was used for phase location analysis. The number of phases on the image was estimated visually. Then 10 points from every phase was randomly selected and the G_{phn} were calculated as the average of gray value of the points belonging to the n-phase. The threshold Th_{ij} (i and j are phases numbers) between the phases was calculated as the average of G_{phn} 's for the phases with similar gray values.

The histogram threshold was calculated (with Fityk[Fit] software) by fitting n Gaussian curves (where n is the number of rock's phases + 1) to the histogram of the CMT image. Threshold value (Th_h) was taken at the first curves intersection (figure 2).

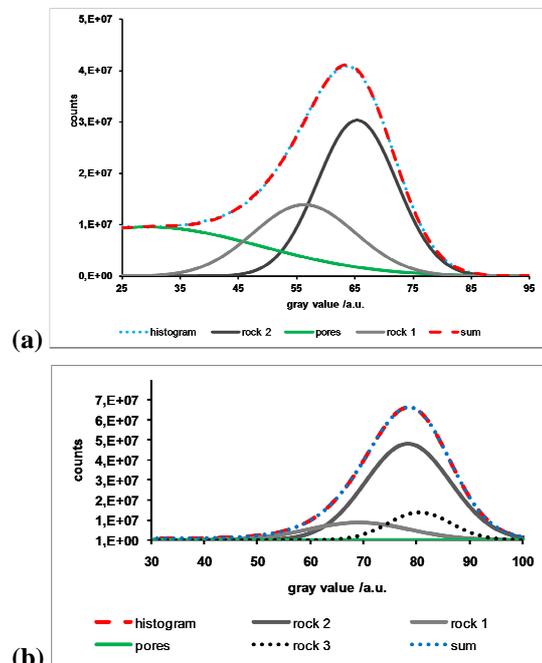


Figure 2. Histogram deconvolution (a) sample 1, 3 Gaussian curves, (b) sample 2, 4 Gaussian curves.

Threshold values estimated with different methods was shown in table 2.

sample	threshold method	threshold
1	boundary	40
	histogram	45
	phase-mean	35
2	boundary	40
	histogram	47
	phase-mean	39

Table 2. Values of threshold between pore and rock phase while using different threshold techniques.

The result of these three methods on image segmentation was shown in figures 3 and 4.

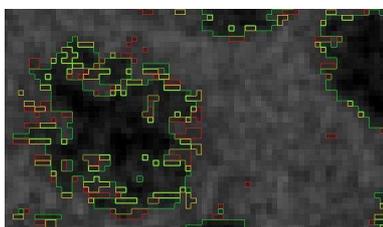


Figure 3. Sample 1 - the pore's border for three different threshold methods; yellow line - the threshold along boundaries, red - the histogram threshold, green - the phase-mean threshold.

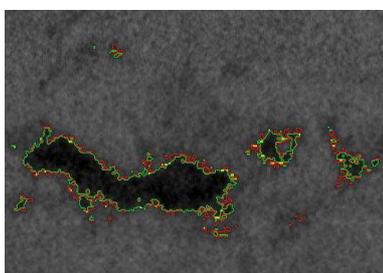


Figure 4. Sample 2 - the pore's border for three different threshold methods; yellow line - the threshold along boundaries, red - the histogram threshold, green - the phase-mean threshold.

As it was shown in figures 3 and 4, histogram threshold technique may result in shifting threshold (between pores and rock) gray value toward higher values.

3.2 The effect of Segmentation on Porosity

All of described thresholding techniques lead to calculate porosity of the examined sample as:

$$\epsilon = \frac{n_p}{\sum_i n_i} \cdot 100\%, \quad (3.1)$$

where n_p is number of voxels assigned to pore layer and n_i is number of voxels assigned to i -layer.

Table 3 shows the porosity values of samples 1 and 2 while using different threshold techniques.

sample	threshold method	porosity /%
1	boundary	27.2
	histogram	31.2
	phase-mean	23.7
2	boundary	2.0
	histogram	2.6
	phase-mean	1.9

Table 3. Porosity calculated with the use of equation (3.1) while using different threshold techniques.

Porosity values estimated by CMT measurements are generally lower than porosity values calculated with density measurements (sample 1 - 29,45 %, sample 2 - 4,90 %). This is due to the measurement resolution - while using CMT equipment it was impossible to notice pores with volume of less than $195 \mu\text{m}^3$.

It is worth noticing that histogram thresholding in case of sample 1 gave the porosity value higher than real porosity of examined sample. It proves that using described simple histogram segmentation technique is not accurate for rock's examining

Two other thresholding techniques gave reliable porosity values. Thresholding along boundaries leads to higher values, closer to the real porosity. Therefore, this technique was recognized as the best for rock's porosity evaluation.

3.3 Phase Location Analysis

The gray value of voxel on the reconstructed image is determined by attenuation coefficient of material and it is proportional to the material's density. Therefore the voxel's gray level may lead to phase-segmentation of reconstructed image. Every separated phase has a significantly different density. These phases may (but need not) correspond to mineral phases present in the examined sample.

The phase location analysis was performed only with the use of phase-mean thresholding. With the use of thresholding along boundaries it was impossible to determine more than two (pores and rock) phases. The histogram thresholding technique was rejected during porosity examining.

In sample 1 three phases (pores and two rock phases) were recognized (figure 5). In sample 2 four phases were selected (figure 6). Used threshold values were shown in table 4.

The volume fraction (f_i) of each phase was calculated as:

$$f_i = \frac{n_i}{\sum_j n_j}, \quad (3.2)$$

where n_i is number of voxels assigned to i -phase layer and n_j is number of voxels assigned to j -phase (i and j refer to rock layers only).

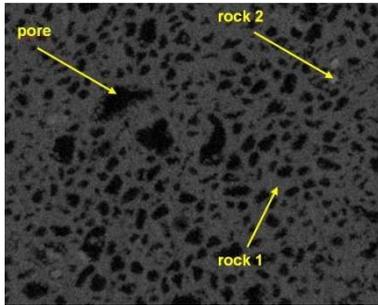


Figure 5. Pores and two different rock phases recognized in sample 1.

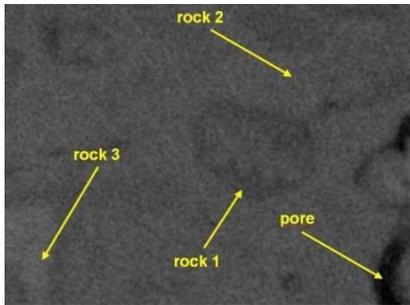


Figure 6. Pores and three different rock phases recognized in sample 2.

Sample	phase	gray value
1	pores	< 35
	rock 1	35-78
	rock 2	> 78
2	pores	< 39
	rock 1	40-72
	rock 2	72-84
	rock 3	> 84

Table 4. The gray values for each recognized phase in samples 1 and 2.

Figures 7 and 8 and table 5 present the results of phase location analysis of sample 1 and 2.

As it was shown in figure 5 and in table 5, the most volume in sample 1 is occupied by a phase with lower density. Phase rock 2, with higher density, is located in the clusters scattered through the bulk of the sample. The pore space in sample 1 is uniformly distributed through the sample's space.

In sample 2, as it was presented in figure 8 and in table 5, the main phase (55 %) is rock phase 2 (with medium density). It is uniformly distributed in sample's space. The densest phase, rock 3, is located mainly at the top of the sample. The pore space in

sample 2 is condensed in the crack in the middle of the sample.

Sample	phase	volume fraction /%
1	rock 1	95.3
	rock 2	4.7
2	rock 1	28.2
	rock 2	55.1
	rock 3	16.7

Table 5. Participation of rock phases of different density in sample's rock skeleton.

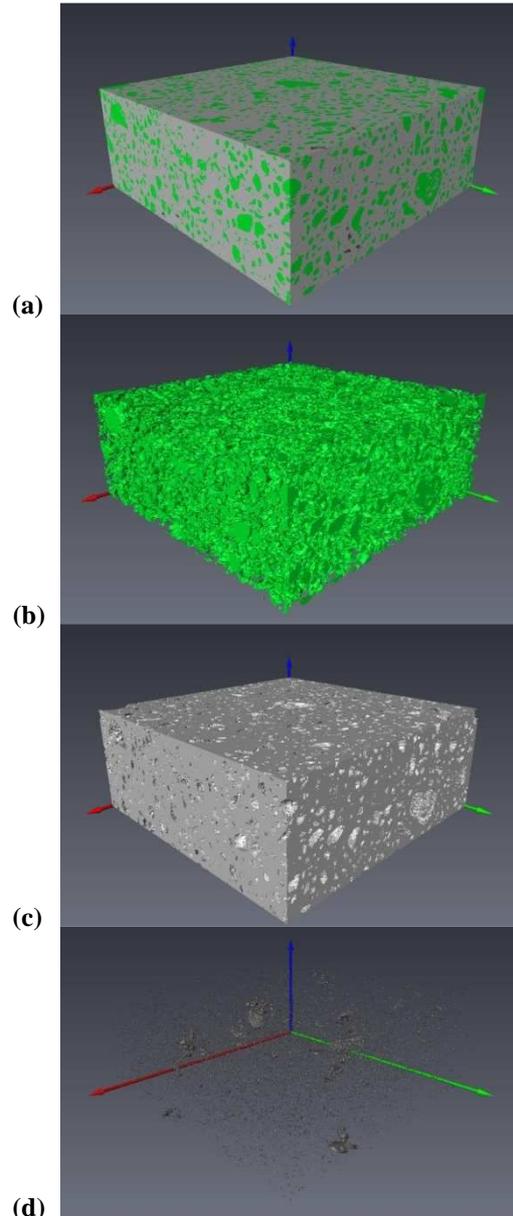


Figure 7. Phase location analysis of sample 1: (a) whole sample with 3 phases, (b) pore space, (c) rock 1 phase, (d) rock 2 phase. Phase's 1 density is higher than density of phase 2.

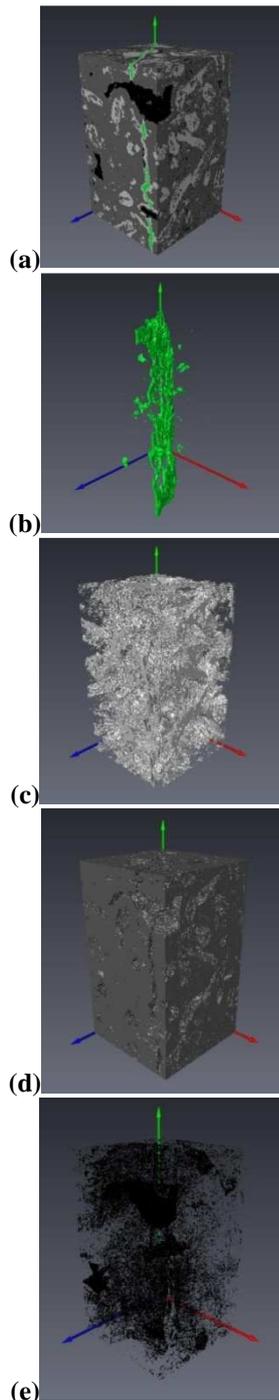


Figure 8. Phase localization in sample 2: (a) whole sample, (b) pore space, (c) rock 1, (d) rock 2, (e) rock 3; the darker color, the denser rock phase.

4. IMAGE ANALYSIS

4.1 Pore Size Distribution

The segmented (with the use of threshold along boundaries) image was saved as RAW data and the pores' layer was labeled (with use of neighborhood 26/8) in MAVI software. Then objects' features were calculated and the pores were divided into classes according to their volume (table 1). Every class was

saved in another RAW file and visualized. The objects' features were exported to CSV file and the pore size distribution graph was plotted for every sample.

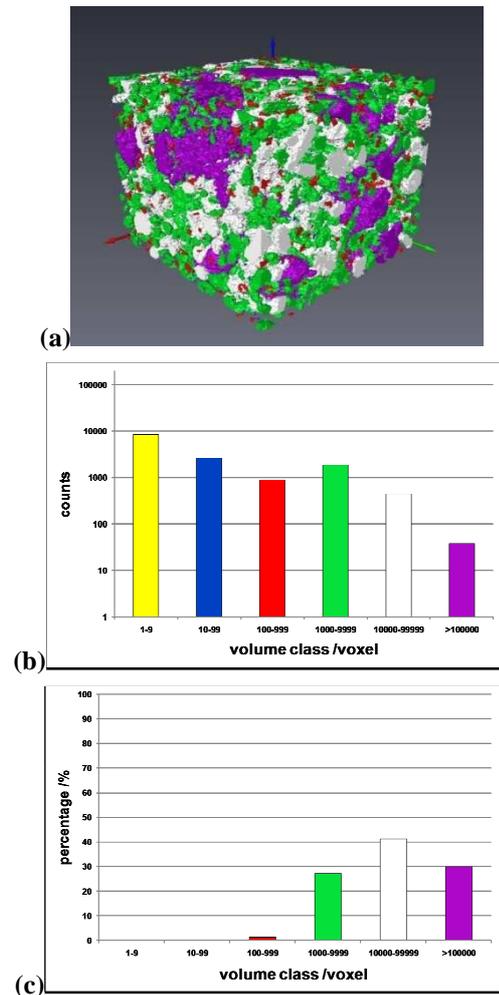


Figure 9. Pore size distribution of sample 1 (cropped to 500x500x400 voxels). (a) visualization of pores location, (b) pores quantitative distribution, (c) percentage (v/v) distribution.

Sample 1, because of its volume, was cropped to 500x500x400 voxels (selected region was located in the middle of sample). The pore's size distribution analysis of sample 1 was presented in figure 9. The cropped fragment of sample 1 contained 14198 objects (pores). As it was shown in figure 9a, they were uniformly distributed at whole sample volume. Figures 9b and 9c shown, that the cropped volume of sample 1 contains about 40 pores with volume above 100000 voxels, but the higher contribution of pore space volume belongs to pore with volume 10000-99999 voxels.

Pore size distribution analysis of sample 2 was presented in figure 10. The distribution of pores' size in sample 2 is quite different than in sample 1. Sample 2 contains only 1 pore with size beyond

100000 voxels, but it makes about 90 % of pore space volume. As it was written above, the pores are concentrated around the crack in the middle of the sample.

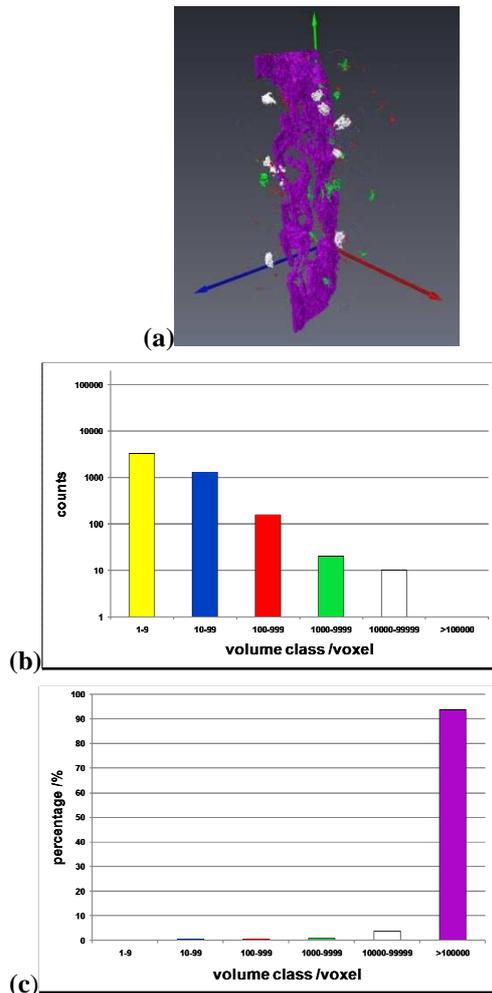


Figure 10. Pore size distribution of sample 2. (a) visualization of pores location, (b) pores quantitative distribution, (c) percentage (v/v) distribution.

5. PREPARING DATA FOR FLUID FLOW SIMULATION

Tortuosity analysis proves that sample 1 has no connections in pore space between opposite sides of the sample. Sample 2 has a channels with average tortuosity 1.1 in y direction (green axis in figures) and 1.4 in z direction (blue axis). Therefore only sample 2 was taken into consideration for fluid flow simulation.

In the fluid transport phenomena only pores with the highest volume participate, so the first step in image simplifying was pores extraction. A pore with volume of 5407370 voxels (the crack) was extracted from sample 2 and the image was saved in the RAW data file. Except this pore, sample 2 has no

connections between pores, so the other pores do not participate in fluid transport in this sample.

The extracted pore was labeled and visualized with the use of Avizo software. Next the labeled sample was linearly resampled by factor 2, 4, 6, 8, and 10. The results of resampling were shown in figure 11 and 12.

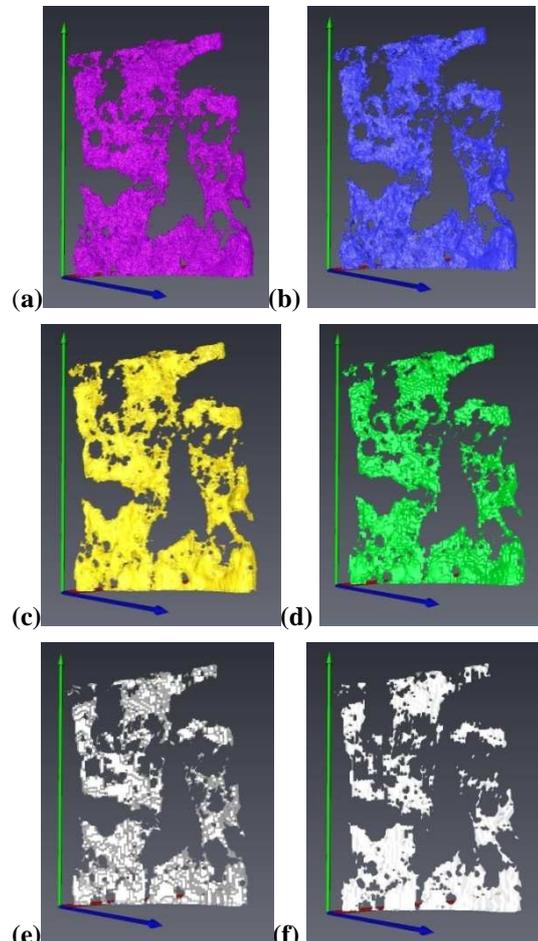


Figure 11. Crack seen in sample 2 (a) without resampling and resampled by factor (b) 2, (c) 4, (d) 6, (e) 8, (f) 10.

Resampling has noticeable effect on pores connections. The tortuosity analysis proves that resampling by factor 10 resulted in breaking all connections in y direction. In figure 12 the input crack and the crack resampled by factor 8 were compared and the visible ruptures in the pore network were marked.

It should be noticed that MAVI software takes account of neighborhood 26 of each voxel for tortuosity calculation. FEM calculation software COMSOL [Com] takes into account neighborhood 6. This means that the rupture of pore network during image resampling may have occurred earlier than it was detected in MAVI software.

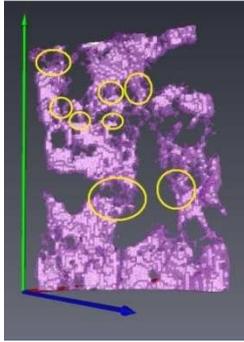


Figure 12. Comparison of figure 11(a) and 11(e). The visible interruptions of pore network were marked.

This example shows that preparing CMT image for fluid flow simulation is not a simple task. Connections between pores have a decisive impact on fluid flow in porous media, so they cannot be interrupted during the image processing.

6. Conclusions

CMT images of two rock samples with different porosity were processed and analyzed.

Three threshold methods were tested. Threshold based on the histogram deconvolution was rejected because porosity estimated with this method was higher than physical porosity of the sample. The best of examined segmentation methods for rock's porosity analysis was threshold along boundaries.

Phase analysis of the samples was executed with the use of phase-mean threshold. The sample was divided into phases with different density. The development of this method provides a basis for detection of different mineral phases in the sample with the use of CMT method.

While preparing sample to fluid flow simulation, the connections between pores were interrupted when the sample was resampled. Sample 2 after resampling by factor 10 has a dimensions $56 \times 95 \times 55$ voxels. It is acceptable, but the simulation takes a long time on computer with 8-cores processor. Thus some better way of image processing should be sought.

Optimization of image processing for simulation will be the aim of the further authors' research.

7. REFERENCES

- [Avi] VSG Avizo 6.1,
http://www.vsg3d.com/vsg_prod_avizo_overview.php
- [Cnu06] Cnuddle V., Masschaele B., Dierick M., Vlassenbroeck J., Van Hoorebeke L., Jacobs P.,

Recent progress in X-ray CT as a geosciences tool, *App. Geochem.* 21, pp. 826-832, 2006.

- [COM] COMSOL Multiphysics 3.5a,
<http://www.comsol.com/products/multiphysics/>
- [Fel84] Feldkamp L. A., Davic L. C., Kress J. W., Practical cone-beam algorithm, *Opt. Soc. Am. A* Vol. 1 No. 6, pp. 612-619, 1984.
- [Fer07] Fernandes J. S., Appoloni C. R., Moreira A. C., Fernandes C. P., Porosity and pore size distribution determination of tumblagooda formation sandstone by X-Ray Microtomography, 2007 International Nuclear Atlantic Conference INAC 2007.
- [Fit] Fityk 0.8.9, <http://www.unipress.waw.pl/fityk/>
- [Har00] Harrison J., Hudson J., *Engineering Rock Mechanism, Part 2*, chapter 9: Porosity, pp.141-159, Elsevier 2000.
- [Hsi03] Hsieh J., *Computed Tomography: Principles, Design, Artifacts and Recent Advances*, SPIE 2003, p. 7.
- [ImJ] ImageJ 1.42q, <http://rsbweb.nih.gov/ij/>
- [Kac08] Kachelrieß M., *Micro-CT, Molecular Imaging I, Handbook of Experimental Pharmacology 185/I*, pp.32-34, Springer, 2008.
- [Ket01] Ketcham R., Carlson W., *Acquisition, optimization and interpretation of X-ray computed tomographic imagery: applications to the geosciences*, *Comp. Geosc.* 27, pp. 381-400, 2001.
- [MAV] Fraunhofer ITWM MAVI 1.3.1,
http://www.itwm.fhg.de/bv/projects/MAVI/index_en.php
- [Mor00] Morse B., Brigham Young University lecture,
http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MORSE/threshold.pdf, 2000.
- [Nar09] Narsilio G., Buzzi O., Fityus S., Yun T. S., Smith D., *Upscaling of Navier-Stokes equations in porous media: Theoretical, numerical and experimental approach*, *Comp. Geotechnics* 39, pp. 1200-1206, Elsevier, 2009.
- [Rec08] Recur B., Desbarats P., Domenger J.-P., *Radon and Mojette Projections Equivalence for Tomographic Reconstruction Linear Systems*, WCSG's 2008.
- [Sal03] Salvato J., Nemerow N., Agardy F., *Environmental Engineering, chapter Water Quantity and Quality*, pp. 267-269, John Wiley & Sons, 2003.

It consists of interconnected processing elements (neurons) that work together to produce an output function. The output relies on the cooperation of the individual neurons within the network to operate. Neural networks often process the information parallel rather than in series (or sequentially). Since it relies on its member neurons collectively to perform its function, a unique property of a neural network is that it can still perform its overall function even if some of the neurons are not functioning. Thus, they are very robust to error or failure. It has been extensively used in offline signature verification over the last two decades. Few relevant researches are summarized below; however, due to the lack of standard database available, all results reported are based on the researcher groups' own independent database.

Mighell, Wilkinson, and Goodman [Mig89] proposed a backpropagation learning algorithm to detect random forgeries. By training 10 genuine signatures and 10 forgeries respectively, which latter tested on 70 genuine signatures and 56 forgeries, they reported a false rejection rate (FRR) of 1% with a false acceptance rate (FAR) of 4%.

Abbas [Abb94] investigated the suitability of using multilayered feedforward neural networks for the task of offline verification. The input to the network is a binary bitmap of size 160 X 35 pixels. The performance is evaluated against their private database of 480 signatures. They concluded that the method is the best for the casual forgeries where able to achieve 0% FAR but its ability to deal with skilled forgeries was still limited with FAR ranging from 0% to 60%.

Qi and Hunt [Qi95] proposed a multi-resolution approach to allocate the offline signature verification problem. The top-level representation of signatures is the global geometric features. A multi-resolution representation of signature is obtained using the wavelet transformation. By using a database of 450 signatures from 25 signatories, the classification is done through a vector quantization (VQ) classifier and an artificial neural network classifier respectively. VQ classifier allows the use of a consistent procedure in processing feature vectors of different length or resolution, and it is easy to implement because its training and classification procedures are relatively simple. However, it can only partition the feature space using hyperspheres, and is incapable of drawing complicated, nonlinear class boundaries. While, artificial neural network is capable of delineating arbitrarily complicated class boundaries, anyway, the performance is heavily depends on the network architecture and training method. The best VQ classification function is the accumulative, multi-resolution system which reported

on FRR of 6.7%, FAR of 13.3% for skilled forgery and FAR of 0% for simple forgery. On the other hand, the multi-resolution network yields the lowest verification error rate when independent features are used, FRR of 4.0%. FAR of 9.3% for skilled forgery and FAR of 1.3% for simple forgery are reported.

Kaewkongka, Chamnongthai, and Thipakorn [Kae99] proposed to use the Hough transform (general Radon transform) as the feature extractor. It extracts the parameterized Hough space from a signature skeleton as a unique characteristic feature of a signature. Evaluation is done through a backpropagation neural network. By using the dataset of 70 signatures, recognition rate of 95.24% is reported.

Quek and Zhou [Que02] proposed a system which is constructed on the basis of a novel fuzzy neural network called the POPFNN-TVR, which has a five-layer structure. Due to its characteristics, such as the learning ability, generalization ability, and high computational ability, it is very powerful to detect the skilled forgeries. After preprocessing, feature extraction is employed to reduce the image observation vector by measuring certain "properties" or "features" of the signature image. In this work, four kinds of features are extracted from the static image of the signature, which including reference pattern based features, global baseline, pressure features and slant features. All of them will be using as elements of the training vector. Two types of experiments are then conducted; first experiment is using the genuine signatures and forgeries as training data, while the second experiment is using only the genuine signatures as training data. Based on the signatures of 15 different signatories from 3 ethnic groups, the average of the individual EER, 22.4% is obtained for the first experiment. While for the second experiment, they claimed that comparable results are obtained.

Piyush Shanker et al. [Piy07] proposed an offline signature verification by using Dynamic Time Warping (DTW). They extract the vertical projection feature from the signature images, and comparing the reference and probe feature templates using elastic matching. The method is tested against the original DTW and modified DTW. The modified DTW achieved EER 2% which outperformed the original DTW at 29%.

Recently, Abdala Ali and Zhirkov [Abd09] proposed an offline signature verification comparing against Support Vector Machine (SVM) and K-Nearest Neighbor (KNN) classifiers. Their system achieves approximately 80% when using SVM, while approximately 70% for KNN.

Bansal et al. [Ban09] proposed an offline signature verification using critical region matching. This work

is mainly focus on the extraction of critical regions which are more prone to mistakes and matching through a modular graph matching approach. They reported 10.81% EER for skilled forgery.

3. OVERVIEW OF WORK

Generally, an offline handwritten signature verification system includes preprocessing, feature extraction and encoding as well as matching as depicted in Fig. 1. These processes will be further discussed in the following sections.

4. PREPROCESSING

Any ordinary scanner with enough resolution can be used as an image acquisition device. However, the scanning hardware may introduce certain noises to a signature image. Another source of noise may be speckled paper background on which the signature is signed on. These noises on signature image may thwart the feature extraction process. We do not figure the real noise distribution, but we use the median filter, which better preserves edges, lines, and corners.

After the smoothing, the images are converted into black-and-white images by using Adobe Photoshop. The threshold level is set to 100.

5. FEATURE EXTRACTION

Discrete Radon Transform (DRT)

DRT [Coe04] is chosen to transform the signature images into a feature space. It is able to transform two dimensional images with lines into a domain of possible line parameters, where each line in the image will give a peak positioned at the corresponding line parameters. DRT has several advantages. Each signature is a static image and contains no dynamic information, thus by calculating projections at different angles, simulated time evolution is created from one feature vector to the next, where the angle represent the dynamic variable [Coe04]. DRT

represents a projection (shadow) of the signature at different angle. A set of transform values is produced after the transformation. The DRT of an image can be calculated as follows. Assume that each signature image consists of N pixels in total, and that the intensity of the i th pixel is denoted by $I_i, i = 1, \dots, N$. The DRT is calculated using β non-overlapping beams per angle and Θ angles in total. The cumulative intensity of the pixels that lie within the j th beam is denoted by $R_j, j = 1, \dots, \beta\Theta$. This is called the j th beam sum. In its discrete form, the Radon transform can therefore be expressed as

$$R_j = \sum_{i=1}^N w_{ij} I_i, j = 1, 2, \dots, \beta\Theta, \text{ where } w_{ij} \text{ indicates the}$$

contribution of the i th pixel to the j th beam sum [Coe04]. The value of w_{ij} is determined by two-dimensional interpolation. Each projection therefore contains the beam sums that are calculated at a given angle.

Instead of Hough transform, we preferred DRT because it has a nice effect of attenuating the speckle noise in the images through summation, while the use of Hough transform is very delicate especially on noisy images.

Principle Component Analysis (PCA)

PCA has been widely used for dimensionality reduction in computer vision ([Lu03], [Tur91], and [Wan03]). It finds a set of orthogonal basis vectors which describe the major variations among the training images and with minimum reconstruction means square error. The successful implementation of PCA in various recognition tasks popularized the idea of matching images in the compressed subspaces.

Since the number of transformed values after DRT is too huge, PCA is utilized here for feature data compression. In the PCA method, the average of K DRT features with M dimension is defined as R_{avg} .

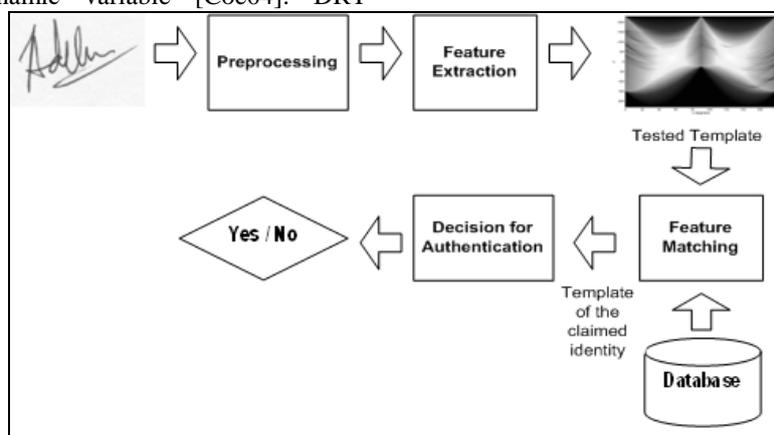


Figure 1. Block diagram of an offline handwritten signature verification.

Then, eigenvectors, v_k and eigenvalues, λ_k with symmetric matrix C are calculated. v_k determines the linear combination of K difference images with φ to form the EigenSignature, $U_l = \sum_{k=1}^K v_{lk} \varphi_k \quad \square l = 1, \dots, K$.

Then, $P(<K)$ EigenSignatures are chosen to correspond to the P highest eigenvalues, which imply that the P features are selected. An input DRT feature, R_k is transformed and projected into the EigenSignature space by the operation, $\rho_k = U_k(R_k - R_{avg})$, where $k = 1, \dots, P$.

Probabilistic Neural Network

Rather than ordinary matching approaches that are based on similarity matching concept, there is another popular method used for classification which the idea is to construct the decision boundaries directly by optimizing an error criterion. PNN which was first introduced by Specht ([Spe88], [Spe90]) is one such technique. It offers several advantages over backpropagation network. The rationale behind this is that, as a kernel-based approach to probability density function approximation, PNN posses the advantages to handle the complex, non-linear and imprecise problems such as signature verification.

In general, a PNN consists of three layers – a pattern, summation and output layers (apart from the input layer) as illustrated in Fig. 2. The pattern layer contains one neuron for each input vector in the training set, while the summation layer contains one neuron for each user class to be recognized. The output layer merely holds the maximum value of the summation neurons to yield the final outcome (probability score).

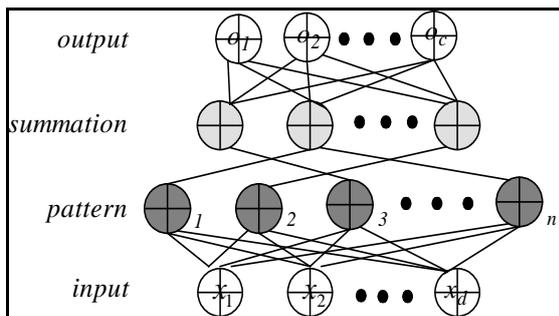


Figure 2. Basic configuration of a probabilistic neural network.

The network can simply be established by setting the weights of the network using the training set. The modifiable weights of the first layer are set by $\omega_{ij} = \rho_{ij}$ where ω_{ij} denoting the weight between i th neuron of the input layer and j th neuron in the pattern layer, and ρ_{ij} is the j element feature of ρ_i in the training set. The second layer weights are set by $\omega_{jk} = T_{jk}$, where ω_{jk} is the weight between neuron j in pattern layer and neuron k of the output layer, and 1 is assigned to T_{jk} if

pattern j of the training set belongs to user k and 0 otherwise. After the network is trained, it can be used for classification task. The outcome of the pattern layer is defined as $out_j = \exp\left(-\sum_{i=1}^m (\rho_i - \rho_{ij}) / \sigma\right)$. Note

that out_j is the output of neuron j in pattern layer and σ is the smoothing parameter of the Gaussian kernel which is the only independent parameter that can be decided by the user. The input of the summation layer

is calculated as $in_k = \sum_{j=1}^n out_j \times \omega_{jk}$ where in_k is the

input of neuron k in output layer. The outputs of the summation layer are binary neurons that produce the classification decision, i.e 1 is assigned to out_k if in_k is larger than the input of others neurons and 0 otherwise.

The smoothing parameters ($\sigma_1, \sigma_2, \dots,$ and σ_j) need to be carefully determined in order to obtain an optimal network. This factor needs to be selected to cause a reasonable amount of overlap; too small deviations will cause a very spiky approximation which cannot generalize, while too large deviations smooth out detail. An appropriate figure is easily chosen by experiment, by selecting a number which produces a low selection error, and fortunately PNNs are not too sensitive to the precise choice of smoothing factor. For convenience sake, we use a straightforward procedure to select the best value for σ . Firstly, an arbitrary value of σ is chosen to train the network, and then test it on a test set. This procedure is repeated for other σ 's values and the σ giving the least errors will be selected.

The motivation of using a PNN is driven by the generalization property and simple training scheme (only one epoch of training is required) of PNN. However, the speed of training is achieved at the cost of increase in complexity and computational/ memory requirements. The time complexity for training is $O(nP)$, where n denotes the number of training samples and P is the length of PCA feature data. In our context, the time complexity of PNN that depends on the P and n can be decreased notably due to the compressed feature data length. As such, the association of DRT and PNN is feasible in practical usage due to its high speed and accuracy performance.

6. EXPERIMENTS & DISCUSSIONS

Database and Setup

Our independent database comprised of 1000 genuine signatures, 500 casual forgeries, and 500 skilled forgeries which were collected from 100 writers and 10 forgers. Due to the non-repetitive nature of variation of the signatures, the signatures produced

will have certain variations among same writers. Thus, the data preparation was mainly divided into two stages. In the first stage, five sample signatures are registered per writer at a single contact session producing 500 samples. In the second stage, another set of five genuine signatures were supplied by the same writer during the contact sessions two weeks after the initial session, yielding another 500 samples. Thus, by recording the specific date, we can observe the variations among the same signature for a single session and different sessions. For the forgery part, the casual forgeries are obtained first; the forgers only allow viewing the writer's name but did not have the access to the signatory's signatures. The skilled forgeries are then obtained from the same group of forgers. We provided them with several samples of each signatory's genuine signature and they are allowed ample opportunity to practice on it.

The pen or pencil used by each writer is not prescribed but signatures are written within a pre-drawn 5 x 2 grid on A4 paper. These signatures were scanned into the computer using a 24-bit millions of colors, 600 dot-per-inch resolutions. The individual images are extracted and labeled with both the writer names and the signature class number.

We will evaluate the system based on false acceptance rate (FAR), false rejection rate (FRR), and equal error rate (EER).

Performance Evaluations

This method is evaluated by using random, casual and skilled forgeries from the mentioned independent database.

Four samples of each person are sequentially selected for Eigen basis construction and the remaining six samples are used for testing. To investigate the performance of PCA against the DRT-extracted signature images as the dimensionality reduction agent, we use different number of principle components (or feature length), varying from 10 – 200, as shown in Table 1.

It is interesting to discover that longer feature length leads to better result. The performance peaks when 100 principle components are used. However, this principle only holds to a certain point as the experimental results show that the result remains unchanged when the feature length is extended further. Thus, the PCA length is set to 100 for the following experiments.

Next, we investigate the performance of DRT by using three different distance metrics, which are cosine angle distance, L_1 (Manhattan) and L_2 (Euclidean) distance measure for random random (Fig. 3), casual (Fig. 4) and skilled (Fig. 5) forgeries respectively. DRT β is taken to be equal to the

highest dimension of the image (300 X 200 pixels after smoothing and converted into black-and-white image), which is 300, and works on $\Theta = 128$.

From the experiment, the cosine angle distance is outperforming towards L_1 (Manhattan) and L_2 (Euclidean) distances. This is because cosine angle distance usually gives a higher rank to vectors with larger variance (whereas applied to signature images) among its components.

Number of PCA Feature Length	Random Forgery (EER, %)	Casual Forgery (EER, %)	Skilled Forgery (EER, %)
10	8.75	12.00	23.00
30	8.33	11.65	22.45
50	7.45	11.00	21.00
80	7.11	10.20	20.22
100	6.95	9.87	19.56
120	6.95	9.87	19.56
150	6.95	9.87	19.56
180	6.95	9.87	19.56
200	6.95	9.87	19.56

Table 1. Equal error rates (EER, %) of using different number of principle components

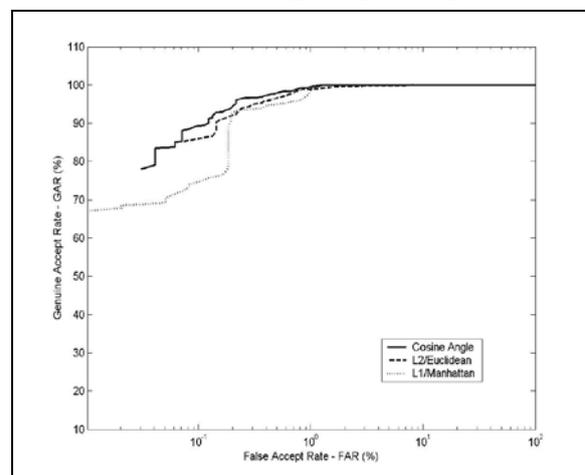


Figure 3. Receiving Operating Characteristic (ROC) curve of random forgery for three different distance metrics: cosine angle, L_1 (Manhattan) and L_2 (Euclidean) respectively.

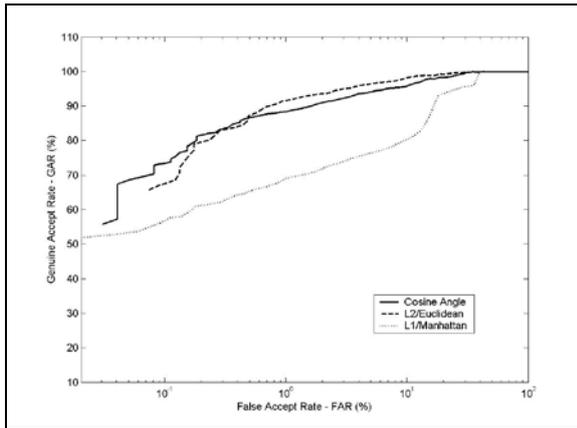


Figure 4. Receiving Operating Characteristic (ROC) curve of casual forgery for three different distance metrics: cosine angle, L_1 (Manhattan) and L_2 (Euclidean) respectively.

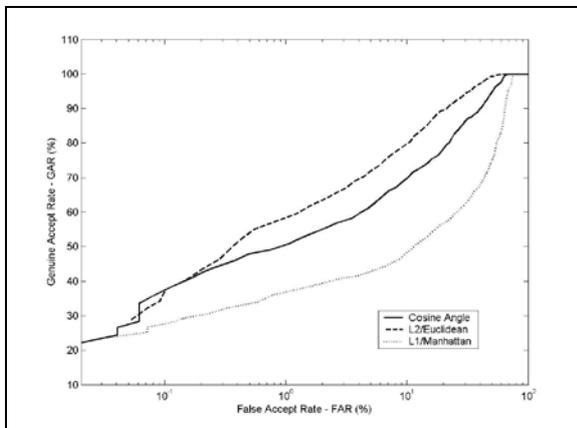


Figure 5. Receiving Operating Characteristic (ROC) curve of skilled forgery for three different distance metrics: cosine angle, L_1 (Manhattan) and L_2 (Euclidean) respectively.

However, it can be anticipated that the classification accuracy of the methods will improve when a more sophisticated classifier, PNN is used. In our system, $^{10}C_4 = 210$ runs are performed with different partitions between the training and testing sets by using a PNN smoothing parameter of $\sigma = 10$.

From the ROC curve showing in Fig. 6, the performance is greatly improved especially for casual and skilled forgeries. Table 2 summarizes the performance of PNN towards random, casual and skilled forgeries.

Besides, the experiment also shows that the computation time can be reduced significantly with just slight performance drop when only one template per user is used (as compared to the case of 4 training samples shown in Table 3 for skilled forgery). In this case, the time complexity of PNN that depends on the number of training samples, n and the length of PCA feature data, P can be decreased notably due to the

compressed feature data length through PCA and single training sample per user settings. As such, the association of DRT, PCA and PNN is feasible in practical usage due to its high speed and accuracy performance.

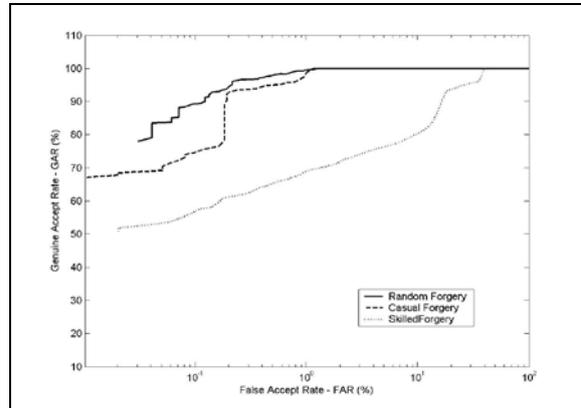


Figure 6. Receiving Operating Characteristic (ROC) curve for random, casual and skilled forgeries respectively when using: Eigen basis construction set = 4, principle component length = 100 when classified through PNN.

	FAR(%)	FRR(%)	EER(%)
Random Forgery	1.50	1.52	1.51
Casual Forgery	3.22	3.24	3.23
Skilled Forgery	12.98	13.16	13.07

Table 2. FAR, FRR and EER achievement (%) for random, casual and skilled forgeries respectively

Training Samples	Total time (minutes)	EER (%)
4	38.5	13.07
1	14	14.20

Table 3. Total time spent to run one course of experiment and the accuracy of PNN in skilled forgery context

Comparison with Other Research Groups' Techniques

It is very difficult to compare the performance of different signature verification systems due to the fact that different systems are using different signature data sets. The lack of a standard international signature database is a big problem for performance comparison.

However, few works that published in year 2009 including Piyush Shanker et al. [Piy07], Abdala Ali and Zhirkov [Abd09] (we implement only on SVM) and Bansal et al. [Ban09] algorithms have been

implemented and tested in our own independent database due to the close-similarity of our implementation details.

	Piyush et al.	Ali and Zhirkov	Bansal et al.	Our method
Random Forgery	1.45	1.13	1.23	1.51
Casual Forgery	3.21	2.43	3.15	3.23
Skilled Forgery	13.05	11.55	12.58	13.07

Table 4. Equal error rates (EER, %) of implementing different approaches towards our independent database

	Piyush et al.	Ali and Zhirkov	Bansal et al.	Our method
Random Forgery	125.0	120.0	80.0	38.5
Casual Forgery	125.0	120.0	80.0	38.5
Skilled Forgery	125.0	120.0	80.0	38.5

Table 5. Computation times (minutes) of different approaches towards our independent database

Referring to Table 4, it can be concluded that their algorithms are slightly outperform our method. However, by referring to Table 5, we can say that our system is more favorable in real world application context due to its shortest computation time. Piyush Shanker et al.'s modified DTW is stable, but somehow it is still not particularly fast. Abdala Ali and Zhirkov's SVM is powerful, but very time consuming to select the appropriate kernel functions and determining the belonging parameters during the development phase. Bansal et al.'s algorithm performs slightly better than ours, but required longer processing time.

7. CONCLUSIONS

This paper proposed an offline signature verification through DRT, PCA and PNN. The high accuracy is feasible to filter the forgery from the genuine signature, especially for skilled forgery; while the speed of the PNN is very favorable in real-world application. The results are encouraging and thus should motivating the research on skilled forgery detection especially for offline handwritten signature.

8. ACKNOWLEDGMENTS

Our thanks and appreciations to those referred work listed in literature.

9. REFERENCES

- [Abb94] Abbas, R. A prototype system for offline signature verification using multilayered feedforward neural networks. Minor Thesis, 1994.
- [Abd09] Abdala Ali, A.A., and Zhirkov, V.F. Offline signature verification using Radon transform and SVM/KNN classifiers. Transactions of Tambov State Technical University, no.1, pp.62-69, 2009.
- [Ban09] Bansal, A., Gupta, B., Khandelwal, G., and Chakraverty, S. Offline signature verification using critical region matching. International Journal of Signal Processing, Image Processing and Pattern, vol.2, no.1, 2009.
- [Coe04] Coetzer, J., Herbst, B.M., and du Preez, J.A. Offline signature verification using the discrete Radon transform and a hidden Markov model. EURASIP Journal on Applied Signal Processing, vol.4, pp.559-571, 2004.
- [Guo01] Guo, J.K., Doermann, D., and Rosefeld, A. Forgery detection by local correspondence. International Journal of Pattern Recognition and Artificial Intelligence, vol.15, no.4, pp.579-641, 2001.
- [Kae99] Kaewkongka, T., Chamnongthai, K., and Thipakorn, B. Offline signature recognition using parameterized Hough transform. Proceedings of the Fifth International Symposium on Signal Processing and Its Applications, 1999.
- [Lec94] Leclerc, F., and Plamondon, R. Automatic signature verification: the state of the art, 1989-1993. International Journal of Pattern Recognition and Artificial Intelligence, vol.8, no.3, pp.643-660, 1994.
- [Lu03] Lu, G., Zhang, D., and Wang, K. Palmprint recognition using Eigenpalms features. Pattern Recognition Letters, vol.24, no.9-10, pp.1473-1477, 2003.
- [Mig89] Mighell, D.A., Wilkinson, T.S., and Goodman, J.W. Backpropagation and its application to handwritten signature verification. Advances in Neural Information Processing Systems, vol.1, pp.340-347, 1989.
- [Piy07] Piyush Shanker, A., and Rajagopalan, A.N. Offline signature verification using dynamic time warping. Pattern Recognition Letters, vol.28, no.12, pp.1407-1414, 2007.
- [Pla89] Plamondon, R., and Lorette, G. Automatic signature verification and writer identification – the state of the art. Pattern Recognition, vol.22, no.2, pp.107-131, 1989.
- [Pla00] Plamondon, R., and Srihari, S.N. Online and offline handwritten recognition: a comprehensive survey. IEEE Transaction on Pattern Analysis and

- Machine Intelligence, vol.22, no.1, pp.63-84, 2000.
- [Qi95] Qi, Y.Y., and Hunt, B.R. A multiresolution approach to computer verification of handwritten signatures. IEEE Transactions on Image Processing, vol.4, no.6, 1995.
- [Que02] Quek, C., and Zhou, R.W. Antiforgery: a novel pseudo-outer product based fuzzy neural network driven signature verification system. Pattern Recognition Letters, vol.23, no.14, pp.1795-1816, 2002.
- [Spe88] Specht, D.F. Probabilistic neural networks for classification, mapping, or associative memory. Proceeding of the IEEE International Conference Neural Networks, vol.1, no.2, pp.525-532, 1988.
- [Spe90] Specht, D.F. Probabilistic neural networks (original contribution). Neural Networks, vol.3, no.1, pp.109-118, 1990.
- [Tur91] Turk, M.A., and Pentland, A.P. Eigenfaces for recognition. Journal of Cognitive NeuroScience, vol.3, no.1, pp.71-86, 1991.
- [Wan03] Wang, X., and Kuldip, K.P. Feature extraction and dimensionality reduction algorithms and their applications in vowel recognition. Pattern Recognition, vol.36, no.10, pp.2429-2439, 2003.

Feature Line Detection on Triangulated Meshes A Geological Application

Dimitri Kudelski
LSIS, UMR CNRS 6168
Campus de Luminy
Case 925
France, 13288 Marseille cedex 9
kudelski@univmed.fr

Jean-Luc Mari
LSIS, UMR CNRS 6168
Campus de Luminy
Case 925
France, 13288 Marseille cedex 9
mari@univmed.fr

Sophie Viseur
GSRC, EA 4234
Université de Provence
Case 67
France, 13331 Marseille cedex 3
sophie.viseur@univ-provence.fr

ABSTRACT

We present in this article an algorithm dedicated to the feature line detection on 3D triangulated outcrop meshes. These lines corresponding to geological elements can be extracted by geometrical properties. Our approach uses differential quantities and especially principal curvatures and their derivatives. The roots of these derivatives describe particular lines called *ridge lines* for convex parts and *ravine lines* for concave parts. Then it is possible to build a set of polylines matching with ridges and ravines. Finally we apply a directional filtering to keep geological structures oriented in a particular direction. The proposed algorithm fits in a basis of a tool devoted to assist geologists during the outcrop analysis and interpretation.

Keywords

geometric modeling, differential geometry, discrete curvatures, crest lines

1. INTRODUCTION

Many works dedicated to the crest line detection have been proposed these last years (e.g., [PKS⁺01, OBS04, YBS05]). Application fields of these methods are wide and various: non-photorealistic rendering [JDA07], mesh segmentation [SF04], medical imaging [MAM95], and geology [Nam08].

Since a few years, the *LIDAR*¹ scanning technology is used to capture cliffs or, more generally, *outcrops* (i.e., formations of rock strata that crop out). It generates a 3D point cloud which is afterwards triangulated to obtain a surface corresponding to the outcrop geometry. Combined with photo mapping techniques, it is possible to construct 3D models called *DOMs*² [BKJ05]. From this point, we propose a semi-automatic method devoted to the detection of geological objects (i.e., *fractures* and *stratigraphic limits*) from outcrop surfaces. This kind of elements is characterized by differential properties explained in the following. Therefore, the extraction is a problematic similar to the *crest line* detection. However before applying a method of crest line detection to outcrop surfaces, several particular constraints must be considered:

Outcrop rugosity

The intrinsic rugosity of observed outcrops makes

the generated surfaces highly complex. As the crest lines are characterized by curvature derivatives, this extraction is noise-sensitive. It is then necessary to use a noise-invariant and triangulation-invariant curvature estimator.

Results matching with observations

The presented method aims at detecting geological objects. Nevertheless, when applying traditional algorithms of crest line detection, the extracted features do not entirely correspond to elements with a geological meaning. An *a priori* knowledge is then necessary to realize a filtering to only extract targeted geological structures.

Interactivity

An additional constraint is the computational time due to the final application. The detection must be performed in a few seconds in order to keep interactivity with a real-time procedure. Moreover, this is particularly crucial as *LIDAR* scans often generate huge data sets which are difficult to manipulate. Because of this, we take great care to implement process with low computational time.

To understand the crest line detection problem, Section 2 describes the different criteria characterizing the geological objects. We review in Section 3 the related work established in the domains of curvature estimation and crest line detection. Then we detail each step of our approach in Section 4. Section 5

¹ *Light Detection And Ranging*

² *Digital Outcrop Model*

finally presents the results obtained with our algorithm applied on LIDAR data scans.

2. CHARACTERIZATION OF GEOLOGICAL OBJECTS

Fractures are like crevices more or less opened that affect a rock mass. Stratigraphic limits of geological bodies correspond to a change of rock type. Both of these geological features are displayed along the outcrop surface because of the erosion. It leads finally to step-like or a gutter-like shapes at their location. Figure 1 represents a diagram with the different patterns of targeted geological objects. Moreover, this pattern often varies along the same fracture or strata limit. Given Figure 1, it is indeed possible to see that the expected objects (depicted by the thick dashed lines) are located in the highest concave parts of the surface. These elements have a common geometrical criterion: they define lines located in areas with high curvature. Thus crest line algorithms can be applied to achieve the detection of such objects.

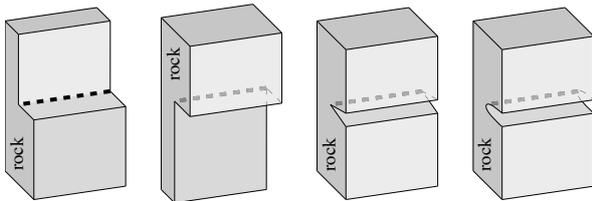


Figure 1: Diagram showing the different patterns of geological objects. The thick dashed lines illustrate the highest concave areas characterizing the expected features.

Feature lines are then defined by curvature extrema and then it corresponds to a zero-crossing of curvature derivatives. However, the rough set of crest lines extracted from a DOM does not represent the set of targeted geological objects. This is due to several factors, among which: (1) fractures often cut across strata limits. Depending on the erosion effect, an extracted crest line could then encompass features with different geological meaning; (2) the intrinsic rugosity of the rock or the variable direction of the outcrop can generate salient lines which do not represent any expected geological feature.

For these reasons, we suggest to add an *a priori* knowledge (*i.e.*, a global direction) to guide the extraction and filter feature lines. Our approach is dedicated to the detection of slightly sinuous structures. It is always the case for the fractures and very frequently for the strata limits.

The proposed method then relies on the crest line principle. It previously requires a per-vertex estimation

of differential quantities. Before describing each algorithm step, the following section gives an overview of existing crest line detection techniques.

3. RELATED WORK

3.1 Curvature Estimation

Differential properties characterize the local geometry of meshes. The notion of curvature describes precisely how the surface is locally bent. These geometrical descriptors are then used since a few years and several approaches have already been proposed in this domain. Some of them are presented in the following (for additional references see [MD02, GG06]).

Continuous methods

This type of methods tends to fit locally the surface with simple primitives (*e.g.*, plane, sphere or polynomial) or parametric functions or even implicit functions. These different techniques permit an analytical computation of curvatures. For example, in [Ham93], the authors proposed to approximate locally the surface with quadratic polynomials. Alternatively in [GI04], the fitting is performed via bi-cubic polynomials. Bi-quadratic Bézier patches can also be used to fit the surface such as in [RB05].

Discrete methods

To reduce the high computational time produced by local fitting, differential operators have been proposed. In [MDSB02], the authors suggested to use a curvature estimation based on cotangent weights and Voronoi areas. In another way, the dihedral angle (*i.e.*, angle between the normals of two adjacent faces) can be used as a discriminant property to compute curvatures [CSM03]. Additionally, the curvature tensor can be estimated by studying the per-vertex normal variation such as in [Rus04, BW07].

3.2 Crest Line Detection

The properties of crest lines are widely used for their efficiency as shape descriptors. This domain has become a field of intensive researches since the last decades and several approaches have been then proposed. The first family of techniques is based on extrema searching. It can be performed either by thresholding [RKS00, SF03], curvature derivatives [CP04, OBS04, YBS05], focal surfaces [LA98, WB01, YBYS07], or discretized operators [HPW05].

The second kind of methods relies on other differential properties. The dihedral angle can be used to detect sharp features such as in [HG01, PSK⁺02]. Then in [GPHW05], the authors proposed to apply active contour theory stemming from image processing

domain to detect characteristic lines. In addition in [LVJ05], Lee suggested to use a measure of a regional importance named *mesh saliency* based on contextual and visual criteria.

4. GEOLOGICAL FEATURE DETECTION

On the one hand, DOMs represent natural surfaces. These objects are characterized by an inherent noise due to the acquisition technology and a high intrinsic rugosity because of the surface alteration. On the other hand, due to their definition by high differential quantities, crest lines are very noise-sensitive. It is then necessary to select a robust curvature estimator. For these reasons, we chose to apply the method proposed in [GI04] considering its quality, accuracy and stable results (see [GG06]). Concerning the crest line detection method, we opted for the criteria expressed in [OBS04]. It relies on curvature derivatives and thus is scale-invariant. It is then possible to extract geological objects with different sizes.

4.1 Pre-processing Step

The inherent noise and rugosity of the data make the detection of smooth and continuous lines difficult. We then propose to use a pre-processing to increase these continuity and smoothness. Among all existing techniques, we chose to integrate a *Laplacian* smoothing (cf. Equation 1) on surface coordinates:

$$p' = p + \lambda \frac{1}{n} \sum_{i=1}^n (q_i - p), \quad (1)$$

where n is the number of adjacent vertices q_i to the vertex p and λ represents a step-size parameter.

Once the smoothing performed, the next step is to compute the differential quantities in order to detect the surface crest lines.

4.2 Estimation of Curvatures and their Derivatives

Several techniques of curvature estimation have been previously presented. The approach proposed in [GI04] fits locally the surface with a bi-cubic polynomial in the least-squares sense. Thus, the surface is expressed for each vertex thanks to the following equation:

$$f(x, y) = \frac{A}{2}x^2 + Bxy + \frac{C}{2}y^2 + Dx^3 + Ex^2y + Fxy^2 + Gy^3. \quad (2)$$

The *Weingarten matrix* (*i.e.*, the matrix of the second fundamental form) of the surface is therefore composed as:

$$W = \begin{bmatrix} A & B \\ B & C \end{bmatrix}. \quad (3)$$

The curvature values κ_{max} and κ_{min} (with $|\kappa_{max}| > |\kappa_{min}|$) are defined by the eigenvalues of W and the eigenvectors of W correspond to the principal curvature directions \vec{t}_{max} and \vec{t}_{min} . To obtain the curvature derivatives, it is possible to use the coefficients D, E, F and G of Equation 2 as suggested in [YBS05]:

$$e = \frac{\partial \kappa}{\partial \vec{t}} = \begin{bmatrix} u^2 \\ v^2 \end{bmatrix}^T \begin{bmatrix} D & E \\ F & G \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \quad (4)$$

where

$$\vec{t} = (u, v) \quad (5)$$

can correspond to either \vec{t}_{min} or \vec{t}_{max} . Consequently, two values called *extremality coefficients* (cf. [Thi96]) are then defined by:

$$e_{max} = \frac{\partial \kappa_{max}}{\partial \vec{t}_{max}} \quad e_{min} = \frac{\partial \kappa_{min}}{\partial \vec{t}_{min}}. \quad (6)$$

These coefficients are the support for the crest line detection, as described in the next section.

4.3 Crest Line Detection

The extremality coefficients describe curvature variations and crest lines are located where curvature extrema are reached. Thus, the crest lines are characterized by:

$$e_{max} = \frac{\partial \kappa_{max}}{\partial \vec{t}_{max}} = 0, \quad \frac{\partial e_{max}}{\partial \vec{t}_{max}} < 0, \quad \kappa_{max} > |\kappa_{min}| \quad (7)$$

for the ridge lines (convex areas) and:

$$e_{min} = \frac{\partial \kappa_{min}}{\partial \vec{t}_{min}} = 0, \quad \frac{\partial e_{min}}{\partial \vec{t}_{min}} > 0, \quad \kappa_{min} < -|\kappa_{max}| \quad (8)$$

for the ravine lines (concave areas).

The curvature sign gives information about the locally convexity or concavity of the surface. Ridges and ravines are dual notions according to the surface orientation: by flipping the surface orientation, convexity and concavity are swapped as for ridge and ravine lines.

As previously mentioned, extremality coefficients as derivatives, are highly sensitive to noise. For this reason, the pre-processing of smoothing the surface geometry is applied to compute the derivatives. However, original coordinates are restored before performing the detection. In this way, noise impact is reduced and even several artifacts due to the intrinsic surface rugosity are removed while maintaining the accuracy about the locations of the extracted feature lines.

Crest line detection is performed by searching crest vertices and curvature extrema (*i.e.*, roots of curvature derivatives). Let be ε an edge composed by

the vertices v_1 and v_2 . A vertex is considered as a crest vertex since a set of conditions described in [OBS04] is satisfied. For the sake of clarity and simplicity, only the case of ridge vertices is explained below. As ridges and ravines are dual notions, explained conditions can be easily transposed from a ridge to a ravine detection algorithm.

First, if the angle between principal directions $\vec{t}_{max}(v_1)$ and $\vec{t}_{max}(v_2)$ is obtuse, the vector $\vec{t}_{max}(v_2)$ is flipped as the sign of $e_{max}(v_2)$. The second step is to check if there is a zero-crossing of the curvature derivative on the edge. It appears when the signs of $e_{max}(v_1)$ and $e_{max}(v_2)$ are different:

$$e_{max}(v_1) \cdot e_{max}(v_2) < 0. \quad (9)$$

Curvature must also reach a local maxima which can be verified by a derivative test:

$$e_{max}(v_1) [(v_2 - v_1) \cdot \vec{t}_{max}(v_1)] > 0. \quad (10)$$

When Equations 9 and 10 are satisfied, the coordinates of the ridge vertex are found by a linear interpolation between v_1 and v_2 :

$$v_{ridge} = \frac{|e_{max}(v_2)|v_1 + |e_{max}(v_1)|v_2}{|e_{max}(v_1)| + |e_{max}(v_2)|}. \quad (11)$$

This process is applied on each edge of the mesh to obtain all the crest lines. These lines are defined by polylines built from crest vertices. Figures 2 and 3 summarize the method of crest line extraction and construction.

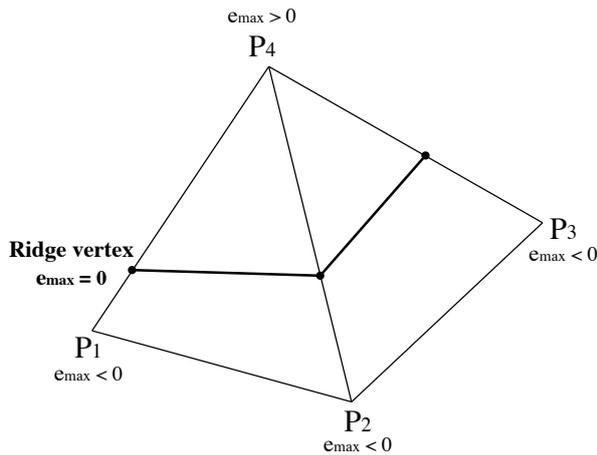


Figure 2: Process of ridge vertex extraction.

The proposed algorithm does not aim at extracting all crest lines but only geological feature lines. Thus, particular conditions must be honored during the feature extraction.

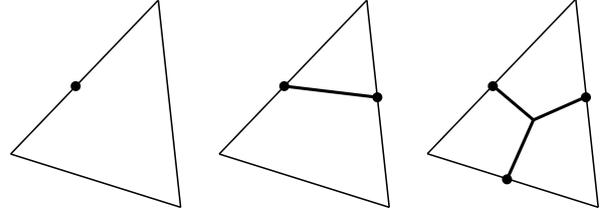


Figure 3: Construction of a feature line. On left, an isolated crest vertex can not define a line. In the middle, two crest vertices generate a straight line. Lastly, 3 crest vertices produce a T-junction between the three vertices and the triangle barycenter.

4.4 Directional Filtering

In order to keep only lines which have a geological meaning and are roughly oriented in a same user-defined direction \vec{D} , an *a priori* knowledge is integrated. It corresponds to a filtering process added to the detection algorithm previously described.

First, as mentioned in Section 2, only concave parts correspond to fractures or strata limits. Therefore only ravine lines characterize relevant objects. Secondly, geological structures are generally slightly sinuous. Their detection can be guided via an user-defined direction \vec{D} , corresponding to the rough direction of a family of targeted geological structures observed along the outcrop.

Let S be a surface of \mathcal{R}^3 and p a point of S . Principal directions of p are contained in a plane \mathcal{P} oriented according to \vec{N}_p (*i.e.*, the normal vector of p). As the shape of the geological objects can be locally described as parabolic surfaces, the curvature vector \vec{t}_{min} tends to follow this shape as shown by Figure 4.

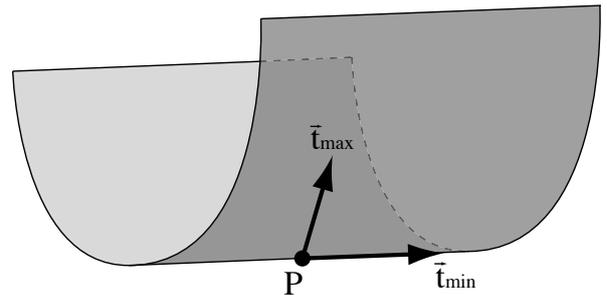


Figure 4: Principal curvature directions along a parabolic shape.

It is thus possible to use the direction of \vec{t}_{min} to filter lines oriented in the same direction of \vec{D} . However the direction \vec{D} is set globally by the user on the outcrop. The outcrop surface is not totally flat and its direction can vary locally. Thus it is not ensured that the vector \vec{D} will be contained in the plane \mathcal{P} . Therefore, a rotation is applied to transform \vec{D} into \vec{D}'

and to place this vector into the plane \mathcal{P} . This rotation has the following parameters:

$$\begin{aligned} \vec{axis} &= \vec{D} \times \vec{N}_p \\ angle &= \vec{D} \cdot \vec{N}_p. \end{aligned} \quad (12)$$

Once the rotation is applied, \vec{D}' is contained in the plane \mathcal{P} . A projection of \vec{D} onto \mathcal{P} could not have been considered as it may generate a null vector \vec{D}' as soon as \mathcal{P} is perpendicular to \vec{D} .

Finally, on the edges containing a root of curvature derivative, the absolute value s' of the dot product between \vec{D}' and \vec{t}_{min} is computed. Therefore when both \vec{t}_{min} vectors of an edge are collinear to \vec{D}' , the line is preserved otherwise it is removed. This step is illustrated by Figure 5.

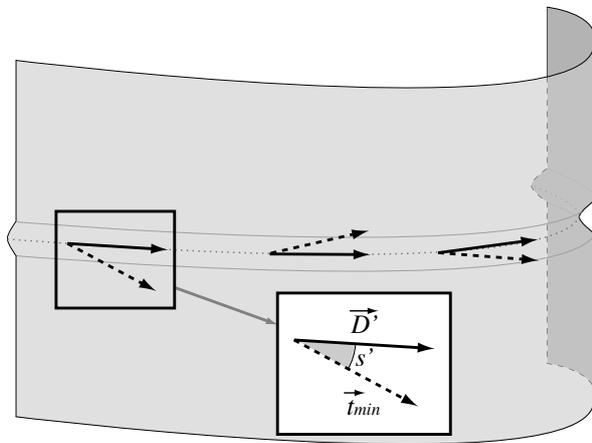


Figure 5: Directional filtering according to the vectors \vec{D}' and \vec{t}_{min} .

The direction \vec{D} is specified globally by the user and corresponds to the rough direction of the expected structures. However the direction of these objects may vary locally. Thus a threshold T , ranged from 0 to 1 is applied on s' as a tolerance factor: if T is equal to 1, the vectors \vec{t}_{min} and \vec{D}' must be strictly collinear to keep the line and inversely if T equals 0, all the ravine lines are kept.

5. RESULTS AND VALIDATION

The proposed approach devoted to the detection of geological objects onto numerical outcrop surfaces is composed of four main operations:

- a pre-processing smoothing;
- an estimation of curvatures and their derivatives;
- a crest line extraction;
- a directional filtering.

This algorithm is dedicated to the detection of geological features (*i.e.*, fractures and strata limits)

from 3D triangulated meshes built from LIDAR data points. Figure 6 shows the results obtained with different outcrop models. Figures 7 and 8 display the impact of the direction \vec{D} and the threshold T onto the detection of targeted geological features.

These parameters have to be set up manually by the user. They represent an *a priori* knowledge about the targeted geological structures to interpret. The direction \vec{D} can be determined by the geologists through the observation along the numerical outcrop. It may be noticed that this parameter could be also automatically deduced from a heuristic such as a *principal component analysis*. However the primary goal of the proposed approach is to assist the geologists in the outcrop interpretation. Moreover, due to the complexity of geological structure spatial organization, the full automatization of the algorithm could easily lead to several mismatch between geological reality and extracted lines which should be in fact removed *a posteriori* using manual or automated filtering. Then, the tolerance threshold T is used to constraint more or less the detection to the fixed orientation. It is set up according to the aspect of the observed limits (*i.e.*, straight or slightly sinuous).

The results obtained with the presented approach match with geological objects observed on outcrops and manually modeled by geologists. We notice however that some lines are incomplete or non-significative. This is due to umbilical points (*i.e.*, points locally spherical) without principal direction.

The computational time of our algorithm is low: it only requires less than 5 seconds (in part due to the computation of curvature values and their derivatives) to detect geological features of a surface composed by about 100k triangles (computed on an Intel Core 2 Duo 2.8 Ghz).

6. CONCLUSION

Several methods of crest line detection have been proposed in the litterature. However none was directly applicable to the context of geological feature extraction from 3D digital outcrop models. By relying on existing methods, we thus present an algorithm devoted to the feature line detection from LIDAR data scan satisfying new constraints.

The proposed approach is based on the estimation of curvature values and their derivatives. The extremality coefficients are computed from curvature derivatives to obtain ridge and ravine lines. Finally, a directional filtering is applied to preserve lines with geological meaning and oriented in a particular direction.

Feature lines corresponding to fractures and strata limits are extracted. The proposed tool enables the geologists to be assisted during the outcrop interpretation stage. As mentioned previously, the obtained results match with the elements manually modeled by geologists.

This approach is promising and can be improved. We plan to add post-processing to increase the quality of results concerning, for instance, the connectivity enhancement and the artifacts removal. In addition, extracted lines are slightly sinuous which concerns most of the targeted geological objects. Though, some strata limits are actually sinuous. This requires a pertinent relaxation of the proposed directional filtering.

The feature extraction corresponding to geological objects is a first step in the outcrop interpretation workflow. The next step would be the construction, from the extracted elements, of a graph to reproduce the layout of observed geological structures.

ACKNOWLEDGMENTS

The authors would like to thank ENI S.p.A. to support this research.

REFERENCES

- [BKJ05] Jerome A. Bellian, Charles Kerans, and David C. Jennette. Digital outcrop models; applications of terrestrial scanning lidar technology in stratigraphic modeling. *Journal of Sedimentary Research*, 75(2):166–176, 2005.
- [BW07] Harlen Costa Batagelo and Shin-Ting Wu. Estimating curvatures and their derivatives on meshes of arbitrary topology from sampling directions. *Vis. Comput.*, 23(9):803–812, 2007.
- [CP04] Frédéric Cazals and Marc Pouget. Ridges and umbilics of a sampled smooth surface: a complete picture gearing toward topological coherence. Research Report 5294, INRIA, 2004.
- [CSM03] David Cohen-Steiner and Jean-Marie Morvan. Restricted delaunay triangulations and normal cycle. In *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry*, pages 312–321, New York, NY, USA, 2003. ACM.
- [GG06] Timothy Gatzke and Cindy M. Grimm. Estimating curvature on triangular meshes. *International Journal of Shape Modeling*, 12(1):1–28, 2006.
- [GI04] Jack Goldfeather and Victoria Interrante. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Trans. Graph.*, 23(1):45–63, 2004.
- [GPHW05] Y.W. Guo, Q.S. Peng, G.F. Hu, and J. Wang. Smooth feature line detection for meshes. *Journal of Zhejiang University Science*, pages 460–468, 2005.
- [Ham93] B. Hamann. Curvature approximation for triangulated surfaces. *Springer Computing Supplementum*, pages 139–153, 1993.
- [HG01] Andreas Hubeli and Markus Gross. Multiresolution feature extraction for unstructured meshes. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 287–294, Washington, DC, USA, 2001. IEEE Computer Society.
- [HPW05] Klaus Hildebrandt, Konrad Polthier, and Max Wardetzky. Smooth feature lines on surface meshes. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing*, page 85, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.
- [JDA07] Tilke Judd, Frédo Durand, and Edward H. Adelson. Apparent ridges for line drawing. *ACM Trans. Graph.*, 26(3):19, 2007.
- [LA98] Gábor Lukács and László Andor. Computing natural division lines on free-form surfaces based on measured data. In *Proceedings of the international conference on Mathematical methods for curves and surfaces II Lillehammer, 1997*, pages 319–326, Nashville, TN, USA, 1998. Vanderbilt University.
- [LVJ05] Chang Ha Lee, Amitabh Varshney, and David W. Jacobs. Mesh saliency. *ACM Trans. Graph.*, 24(3):659–666, 2005.
- [MAM95] O. Monga, N. Armande, and P. Montesinos. Thin nets and crest lines: application to satellite data and medical images. In *ICIP '95: Proceedings of the 1995 International Conference on Image Processing (Vol.2)-Volume 2*, page 2468, Washington, DC, USA, 1995. IEEE Computer Society.
- [MD02] J.-L. Maltret and M. Daniel. Discrete curvatures and applications : a survey. Rapport de recherche LSIS.RR.2002.002, Laboratoire des Sciences de l'Information et des Systèmes, 2002.
- [MDSB02] M. Meyer, M. Desbrun, P. Schroder, and A.H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. *Vi-*

- sualization and mathematics*, 3:35–57, 2002.
- [Nam08] Van Tran Nam. *Traitement de surfaces triangulées pour la construction de modèles géologiques structuraux*. PhD thesis, Université de la Méditerranée, 2008.
- [OBS04] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. Ridge-valley lines on meshes via implicit surface fitting. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 609–612, New York, NY, USA, 2004. ACM.
- [PKS⁺01] D. L. Page, A. Koschan, Y. Sun, J. Paik, and M. A. Abidi. Robust crease detection and curvature estimation of piecewise smooth surfaces from triangle mesh approximations using normal voting. In *Conference on Computer Vision and Pattern Recognition*, volume 1, page 162, Los Alamitos, CA, USA, 2001. IEEE Computer Society.
- [PSK⁺02] DL Page, Y. Sun, AF Koschan, J. Paik, and MA Abidi. Normal vector voting: Crease detection and curvature estimation on large, noisy meshes. *Graphical Models*, 64:199–229, 2002.
- [RB05] Anshuman Razdan and MyungSoo Bae. Curvature estimation scheme for triangle meshes using biquadratic bézier patches. *Computer-Aided Design*, 37(14):1481–1491, 2005.
- [RKS00] Christian Rössl, Christian, Leif Kobbelt, and Hans-Peter Seidel. Extraction of feature lines on triangulated surfaces using morphological operators. In Andreas Butz, Antonio Krüger, and Patrick Olivier, editors, *Smart Graphics (AAAI Spring Symposium-00)*, volume 00-04 of *Technical Report / SS / American Association for Artificial Intelligence*, pages 71–75, Stanford, USA, 2000. American Association for Artificial Intelligence, AAAI Press.
- [Rus04] Szymon Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *Symposium on 3D Data Processing, Visualization, and Transmission*, Sept 2004.
- [SF03] G. Stylianou and G. Farin. Crest lines extraction from 3D triangulated meshes. *Hierarchical and geometrical methods in scientific visualization*, pages 269–281, 2003.
- [SF04] Georgios Stylianou and Gerald Farin. Crest lines for surface segmentation and flattening. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):536–544, 2004.
- [Thi96] Jean-Philippe Thirion. The extremal mesh and the understanding of 3d surfaces. *Int. J. Comput. Vision*, 19(2):115–128, 1996.
- [WB01] Kouki Watanabe and Alexander G. Belyaev. Detection of salient curvature features on polygonal surfaces. *Comput. Graph. Forum*, 20(3), 2001.
- [YBS05] Shin Yoshizawa, Alexander Belyaev, and Hans-Peter Seidel. Fast and robust detection of crest lines on meshes. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 227–232, New York, NY, USA, 2005. ACM.
- [YBYS07] Shin Yoshizawa, Alexander Belyaev, Hideo Yokota, and Hans-Peter Seidel. Fast and faithful geometric algorithm for detecting crest lines on meshes. In *PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, pages 231–237, Washington, DC, USA, 2007. IEEE Computer Society.

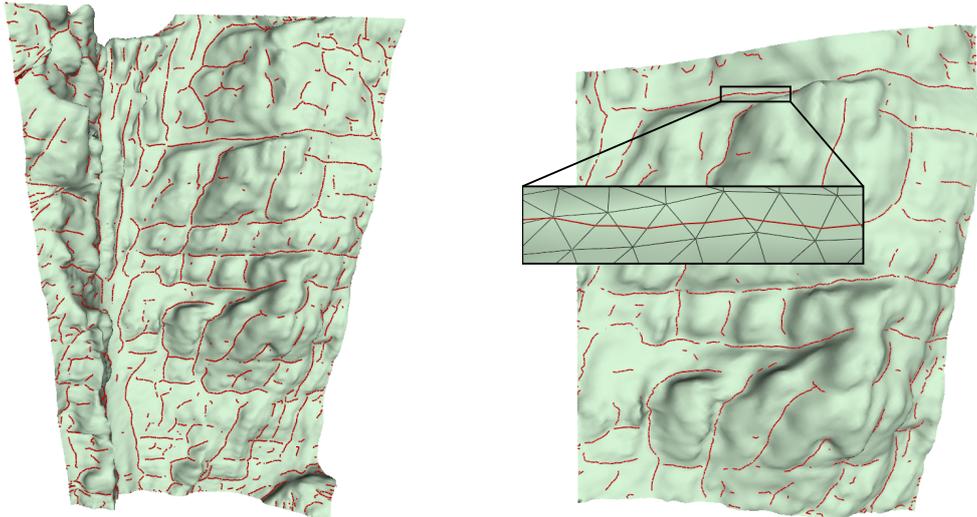


Figure 6: Application of our algorithm on LIDAR data scans without any filtering. On left, feature detection performed on the Malaval section (≈ 60000 triangles). On right, extraction of lines of the Pas-Morta section (≈ 20000 triangles).

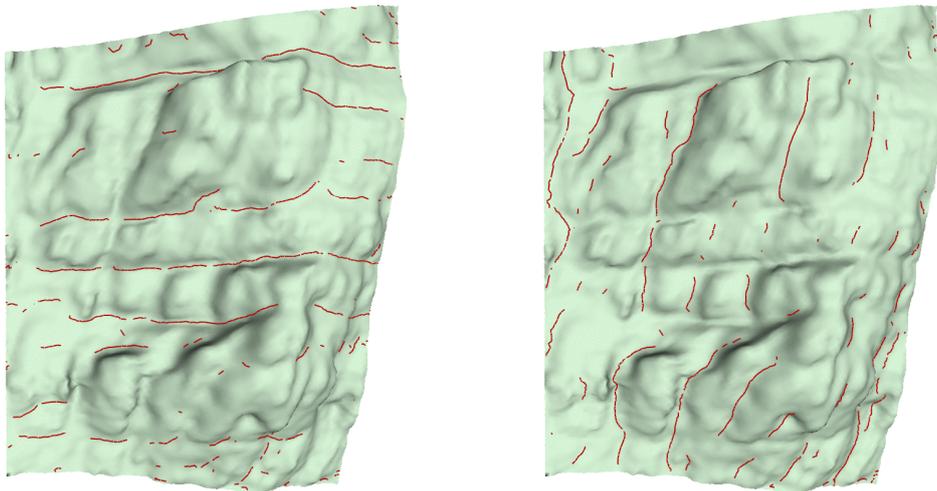


Figure 7: Application of two directional filters. The strata limits are extracted with a horizontal direction (left image) while the fractures are detected with a vertical direction (right image).

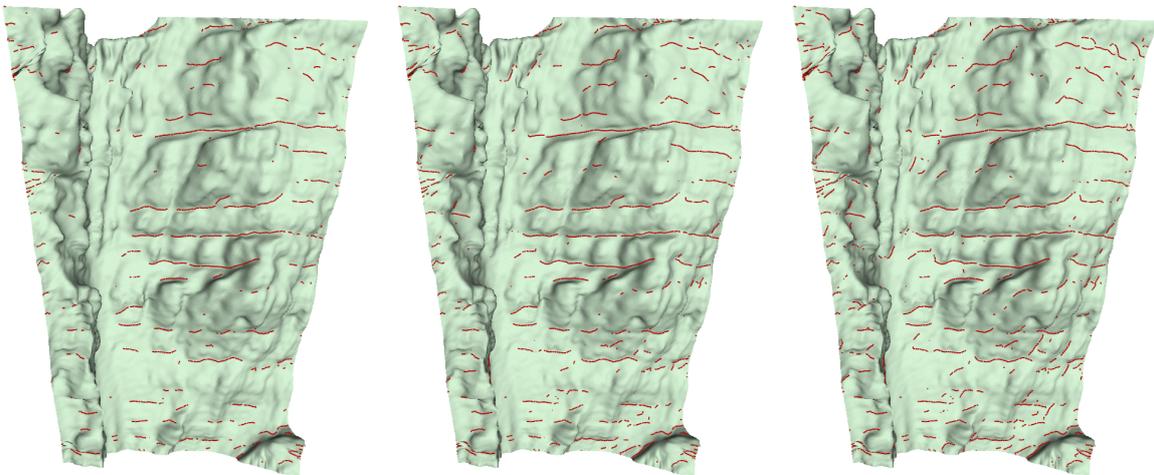


Figure 8: Influence of the tolerance threshold T with values of 0.85 (left image), 0.70 (middle image) and 0.55 (right image).

Confidence in Tone Mapping Applying a User-Driven Operator

Annabell Langs
University of Koblenz
Germany
allangs@uni-koblenz.de

Jakob Bärz
University of Koblenz
Germany
jbaerz@uni-koblenz.de

ABSTRACT

In photorealistic image synthesis, the natural appearance of a scene is predicted by simulating the illumination using radiometric values. Whenever the dynamic range of the simulated luminance values exceeds the capabilities of the display device, tone reproduction is necessary to reduce the contrast of the image. Although a significant number of tone mapping operators have been presented in the past, the reliability of the resulting low dynamic range images cannot be guaranteed. In the context of product design, decision-makers rely on a trustworthy colorimetric and photometric appearance. We believe that in a particular scenario a dedicated user-driven tone reproduction curve outperforms existing operators in terms of reliability, performance, and quality. In this paper, we propose a method to manually generate a tone mapping operator. The user is guided to select a set of simulated input luminance values and to map them to appropriate display luminance output quantities. These key mappings are interpolated to a tone mapping curve. A module was developed for Qt/Python to define and apply the operator. We evaluated the resulting low dynamic range images in a study with thirteen participants. The probands were asked to directly compare a real with a virtual scene displayed on a low dynamic range device as well as to rate the results in comparison to popular tone mapping operators. In addition to defining a dedicated curve for a specific scenario, another application of our approach is to generate a *standard observer* tone reproduction curve by interpolating a set of user-driven functions.

Keywords: tone mapping, image reproduction, high dynamic range (HDR), reliability, colorimetry, photometry.

1 INTRODUCTION

The aim of photorealistic computer graphics is to simulate virtual images by computing a set of radiometric measurements and to reproduce them exactly on the display device. The quality of both the simulation and the reproduction can be evaluated by comparing measurements of a real world scene with the respective measurements of the simulated and reproduced two-dimensional projection of the virtual scene on the display device. When the highest simulated luminance exceeds the maximum luminance of the output device, reproducibility is no longer possible. The same holds true for luminances below the black level of the display, respectively. One solution is to mark the luminance values, which are not reproducible, with false colors. In product design, a typical application field of photorealistic image synthesis, reliable rendering with natural appearance and colors is necessary. To display the simulated luminance values, the high dynamic range has to be reduced to fit the limited dynamic range of the output device.

In the past, a considerable number of popular tone mapping operators have been presented to address this issue. The key problem with all these approaches is to analyze to which extent the compressed image can be trusted. Especially in product design it is vital to create images decision-makers can rely on. In our opinion there is no single tone mapping operator to produce trustworthy results for each high dynamic range image and every presentation setup. In consequence, an expert in image processing needs to choose and validate a fitting operator for each setting and to adjust the individual parameters by hand. We strongly believe that it is more intuitive and less time-consuming to provide a highly reliable and interactive tool for the expert to create a dedicated tone reproduction curve for the given setting.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

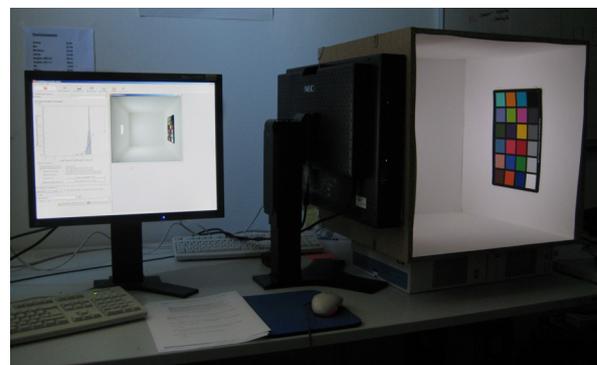


Figure 1: Setup of our direct comparison experiment

In this paper we present an approach of how to generate a user-driven tone reproduction curve. The user is guided to manually select a set of high dynamic input luminance values and map them to appropriate display luminances. These key mappings are interpolated to a tone mapping curve. An example module was developed for QtPfsGui [12] to define and to apply the operator. Another key contribution is the evaluation of both our tool and the results of the user-generated curves in a study with thirteen participants. The reliability of the compressed images was validated by directly comparing a real with a corresponding virtual scene as well as by performing a benchmark test with established tone mapping operators. Furthermore, we generated a *standard observer* tone reproduction curve by interpolating all thirteen user-defined curves. This operator was applied to our test scenario and compared to popular tone mapping operators.

The remainder of the paper is organized as follows: Section 2 outlines previous approaches to the tone reproduction problem. Our solution is presented in section 3, while the evaluation setup is described in section 4. In section 5 we show the results of our evaluation and section 6 concludes this paper.

2 RELATED WORK

The challenge to reproduce real world scenarios with high contrast on media having limited capabilities can be traced back to painting more than five centuries ago. As justly emphasized by MacCann [11], the Renaissance artists were the first to capture realistic perspective and illumination in paintings using a restricted color palette. MacCann also pointed out that tone mapping has always been a challenging problem of photography because of the low dynamic range of print media. The importance of color reproduction in both photography and television is clarified in Hunt [8].

In computer graphics, the problem first arose when physically based images were created using ray tracing or radiosity. As opposed to scanline rendering with arbitrary RGB color values, global illumination algorithms tried to faithfully predict nature by simulating real radiometric values exceeding the dynamic range of the display devices. Tumblin and Rushmeier [16] realized this issue and presented a brightness-preserving operator as a solution in 1993. Operators aiming to preserve brightness or contrast are also subsumed to perceptual-match reproduction, whereon this paper is focused. An early representative of contrast-preserving tone mapping was introduced by Ward [17] in 1994, which reduced computational costs while sustaining just noticeable differences in contrast.

During the last years, a number of popular tone reproduction operators with diverse foci, inspired by fields as photography or the human visual system have been proposed. Reinhard et al. [14] adapted the photographic *zone system* to manually map the subjective middle-grey of the scene to an appropriate display luminance. The contrast can be enhanced locally with a technique inspired by photographic *dodging and burning*. Drago et al. [5] exploited the logarithmic response of the human visual system to incoming luminance. They showed how to handle a wide dynamic range by applying logarithmic compression with individual bases of the logarithm to different picture elements. Another operator, inspired by the response of human photoreceptors, was introduced by Reinhard and Devlin [13]. Based on electrophysiological studies, they designed a sigmoidal function to closely resemble the properties of the receptors. Comprehensive reviews of the most important operators can be found in the state of the art report by Devlin et al. [4] or in the textbook *High Dynamic Range Imaging* [15].

The diversity of approaches to tone mapping necessitates a systematic evaluation. Despite the problem of how to analyze the quality and reliability of tone mapping, a number of attempts are summarized in [19]. Relevantly to our work, Yoshida et al. [19] conducted a psychophysical experiment with 14 human observers. The probands had to compare a real world setting with an HDR photograph of the same scene, compressed and displayed on an LDR display. We chose to model a virtual representation of our well-defined real world scene and simulated a photometrically and colorimetrically consistent ground-truth image. Opposed to the HDR photography approach, our method eliminates inaccuracies from the camera calibration.

Another innovative strategy has been introduced by Mantiuk et al. [10], fitting a generic tone reproduction operator to an HDR image and its LDR counterpart, generated by an unknown existing tone mapping algorithm. They showed that a very simple and computationally inexpensive generic tone mapping curve is often able to reproduce indistinguishable results compared to the original complex algorithm. Furthermore, Mantiuk et al. demonstrated how to use their model to combine several popular operators to a new tone mapping curve. Similarly, we believe that a single tone mapping curve can outperform previously proposed operators in a dedicated scenario. But in contrast to choosing and combining operators by hand, we provide the expert user with a flexible tool to manually create the curve. In addition, we intentionally support no local contrast adjustment and do not change the chromaticities using saturation correction.

3 APPROACH

3.1 Criteria for reliability

The primary purpose of our tone mapping operator is to generate reliable output images. A reliable image should be indistinguishable from the respective real world scene for the human observer. In order to evaluate this goal we need to define criteria for reliability. Tone mapping operators aiming to create realistic results assess those criteria differently than operators trying to generate aesthetic images. Cadik et al. [3] proposed some important criteria: brightness, contrast, reproduction of color, reproduction of details and special attributes like artifacts. They say that the overall image quality, often also called naturalness, depends on several of those criteria. Furthermore, criteria which affect the image globally, such as contrast, are more important. Criteria like the local reproduction of details are of less importance. Cadik et al. conclude that global tone mapping methods are better suited for realistic images. Local tone mapping algorithms can only compete if they have a strong global part.

Yoshida et al. [19] pointed out that global tone mapping can preserve contrast better than local methods, but, on the other hand, local operators are better at reproducing details. They also deduce that no single criterion is solely accountable for the naturalness of the image. In our evaluation we use the criteria brightness, contrast, reproduction of details in shadows and in highlights, and the overall image quality.

3.2 Selecting tone mapping benchmarks

Similarly to both studies mentioned before, we aimed at conducting an evaluation with different tone mapping approaches. Hence, we considered a number of popular operators as potential benchmark algorithms and evaluated their ability to return a reliable LDR image. Global tone mapping operators like *Histogram Adjustment* by Ward [17], Drago's *Adaptive Logarithmic Mapping* [5] or the *Photoreceptor Model* by Reinhard and Devlin [13] are based mainly on the human visual system. Local tone mapping operators like *Photographic Tone Reproduction* by Reinhard et al. [14] or Ashikhmin's *Spatially Variant Operator* [1] have been considered as well. Operators working in the frequency or gradient domain are also popular, for example Durand's *Bilateral Filtering* [6] or *Gradient Domain Compression* by Fattal [7].

As a result we decided to use the Adaptive Logarithmic Mapping, Photoreceptor Model, and Photographic Tone Reproduction in our evaluation. Both frequency and gradient domain methods returned images which are better suited for aesthetical purposes. Drago's

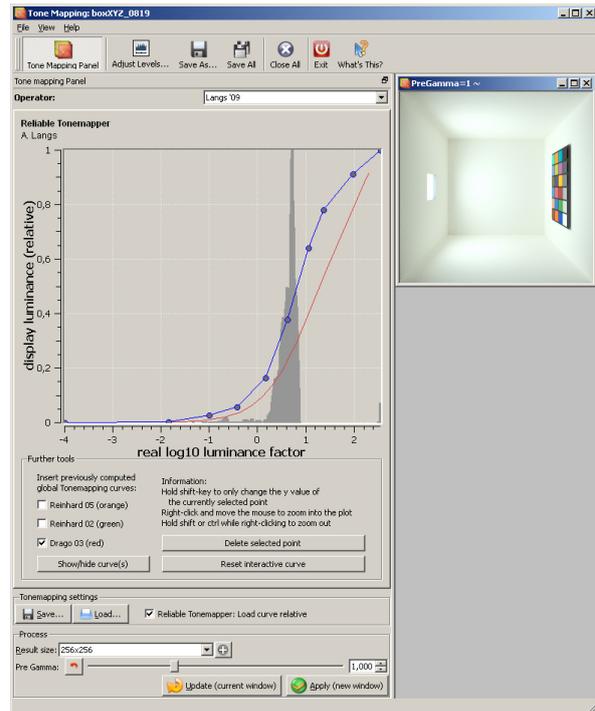


Figure 2: GUI of the interactive tone mapping tool

method has been chosen because it is very popular and a typical representative of the global operators. Reinhard's Photoreceptor Model benefits specially from the fundamentals of the human visual system and separately computes a sigmoidal function on the three RGB channels corresponding to the three cone types on the retina of the human eye. At last, the Photographic Tone Reproduction represents a local tone mapping operator with a different approach, namely combining tone mapping with traditional methods used in photography. To summarize, we chose three very different operators, two global ones and a local one as benchmarks.

3.3 Implementation and features

The idea behind our tone mapping operator is to give an expert a tool at his disposal to create a specific tone mapping curve. Common image editing software like Adobe® Photoshop® provide tools to edit a gradation curve exactly. This well-known graph metaphor is exploited in our implementation to easily, rapidly and interactively adjust a tone mapping curve.

We used the open source software Qtpfsgui and the Qt Widgets for Technical Applications¹ to implement our curve tool. The main window, as seen in figure 2, shows a plot with the x-axis representing the real \log_{10} luminance values and the y-axis representing

¹ <http://qwt.sourceforge.net>

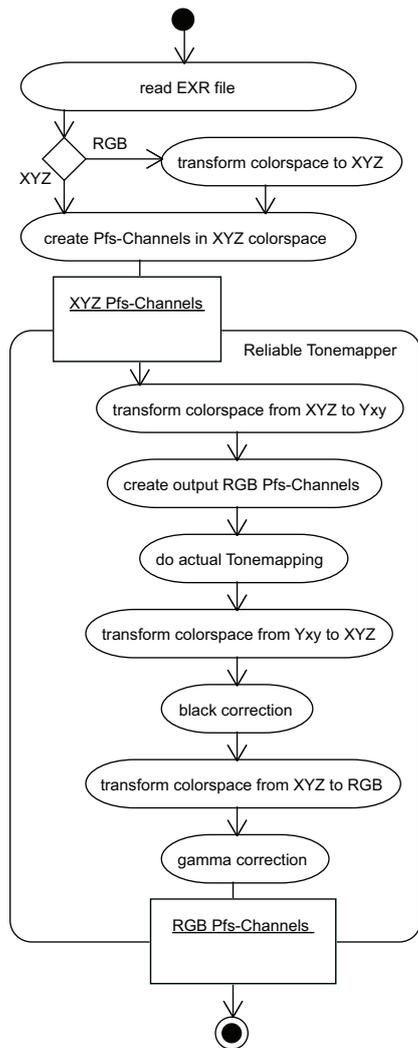


Figure 3: Activity diagram of the tone mapping pipeline

the relative display luminance. A display luminance of 1.0 corresponds to the maximum luminance of the display. The initial curve can be adjusted by setting or deleting control points. The curve is interpolated linearly between them. Holding the shift key restricts the movement of the point to the y-direction in order to avoid shifting the point and therefore changing a different input luminance value.

Additionally, luminance quantities in the image can be mapped precisely to a control point in the plot at the corresponding luminance value. A simple click inside the image creates the correct control point which can be further adjusted. For better comparison, it is possible to import tone mapping functions created by the operators from Drago and Reinhard. Furthermore, user-driven curves can be saved to or loaded from disc. Hence, those curves can be applied to arbitrary images by loading the control points absolutely or relatively. The former method loads the points at the absolute input luminance values regardless of their existence in

the new image. The latter option stretches or squeezes the points relatively to the new luminance range.

Another important aspect of our approach to tone mapping is the photometrically and colorimetrically consistent implementation in order to evaluate correctly later on. Figure 3 illustrates the pipeline a high dynamic range OpenEXR file has to pass before being displayed: Three channels in the CIE XYZ color space are created from the OpenEXR file. Those channels are transferred to the actual tone mapping algorithm which transforms the color space to Yxy. The plot curve is evaluated for every luminance value in the image, returning the value on the y-axis between 0.0 and 1.0 at the x-position of the requested input luminance. Only the luminance channel Y is changed by multiplying the looked-up y-value with the maximum display luminance. The display was measured with an X-Rite i1-pro spectroradiometer and the i1 Share software beforehand and the generated ICC profile was activated for gamma calibration. After the plot look-up process the color space is transformed back to XYZ and a black correction is executed. Then the color space is further transformed to RGB using the measured color primaries of the specific display. Finally, gamma correction is applied.

4 EVALUATION

4.1 Scenarios

To validate the reliability of our user-driven tone mapping operator, we conducted a relative comparison to the selected benchmark operators from section 3.2. Therefore, we generated an HDR photograph using exposure bracketing with a Canon EOS D30 digital reflex camera of a scene on our campus. The HDR image has been calibrated with a Kodak CS100A luminance measurement device and created by webHDR³. The campus scene is depicted on the left in figure 4.

Besides this relative comparison, we conducted a direct comparison between a real world scene and a simulation of the same scene, displayed on a low dynamic range device. We constructed a well-defined real world scene and modeled an exact virtual representation. A ground-truth simulation of the virtual scene was computed with path tracing using a spectral ray tracing system and measured radiometric input values for the light source and the materials. The simulated two-dimensional projection was displayed on a colorimetrically characterized and gamma-corrected device. In figure 1, a photograph of the setup is shown.

² <http://www.openexr.com/>

³ <http://luxal.dachary.org/webhdr/>

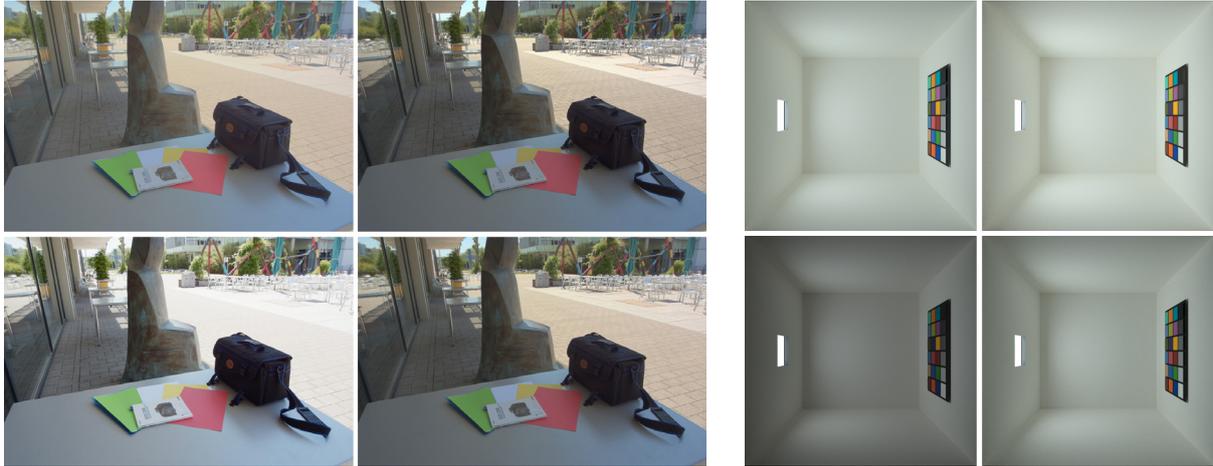


Figure 4: Campus scene (left hand side) and the well-defined box scene (right hand side), each reproduced with Drago's Adaptive Logarithmic Mapping (top left), Interactive tone mapping (top right), Reinhard and Devlin's Photoreceptor Model (bottom left), and Reinhard's Photographic Tone Reproduction (bottom right)

The box measures $0.5m$ in all three dimensions. The front side is left open for the observer. In the center of the left side a square hole with $0.1m$ edge length is spared for a calibrated NEC SpectraView 2690 as the light source. On the right, opposing the light, a Munsell ColorChecker chart was placed. The remainder of the interior of the box was wallpapered with a diffuse white Canson Mi-Teintes Paper. The spectral distributed radiance of the light source and the spectral distributed reflectance of the materials were measured with an X-Rite i1-pro spectroradiometer. An exact virtual representation of this box was modeled. We generated a ground-truth simulation using spectral path tracing. An EIZO FlexScan S2000 displayed the synthetic image using color calibration and black correction. The simulation was validated for low dynamic range images by comparing measurements of the display device with measurements in the box to grant trustworthy results. For our experiment, the light source was calibrated to the maximum luminance of $300cd/m^{-2}$ and the display device to $100cd/m^{-2}$. The box scenario is displayed on the right in figure 4.

4.2 User study

We believe that an individually created tone reproduction curve excels every other tone mapping operator in a specific scenario. We wanted to evaluate whether tone mapping curves created by our probands really deliver a good result measured by the former mentioned criteria and if there are concurrences between those curves and curves of popular tone mapping operators. We approached the question whether the users created similar curves or if there is even a kind of *standard observer* tone mapping curve. Finally, we checked how the users rated the usability of the curve tool.

Thirteen participants aged 21 to 30 years created curves and evaluated the resulting images. During the whole evaluation process we were present to help in case of operating problems or questions. The main functions and keyboard shortcuts of the interactive tool were explained. We pointed out key areas in the images to support the test persons when evaluating the reliability criteria. In a first step, all thirteen participants were asked to fill out a general questionnaire. Level of knowledge, experience with HDR software and existence of color deficiencies were some of the questions. One half of the test persons had some experience with other HDR software.

The first experiment was a relative comparison of the campus scene without direct reference. All participants had to rate four pictures displayed on an LDR device according to the criteria introduced in section 3.1. The setting of the Adaptive Logarithmic Mapping, Photoreceptor Model and Photographic Tone Reproduction were set to the default values proposed in their respective papers, with the following exceptions: the sharpening parameter of the Photographic Tone Reproduction was set to 1.6. The intensity parameter of the Photoreceptor Model was set to -3, the chromatic adaptation to 1.0, and the light adaptation to 0.0. We created our dedicated tone mapping curve for the interactive tone mapping without direct reference. The probands were asked to rate the criteria on a scale with five discrete steps, from *too few/low* to *too many/high* and a possibility to rate *accurate*.

In the second experiment the probands had to create their own tone mapping curve for the aforementioned box scene. The first request was to ensure that the grayscale of the colorchecker was reproduced correctly on the screen. Then they adjusted the color patches

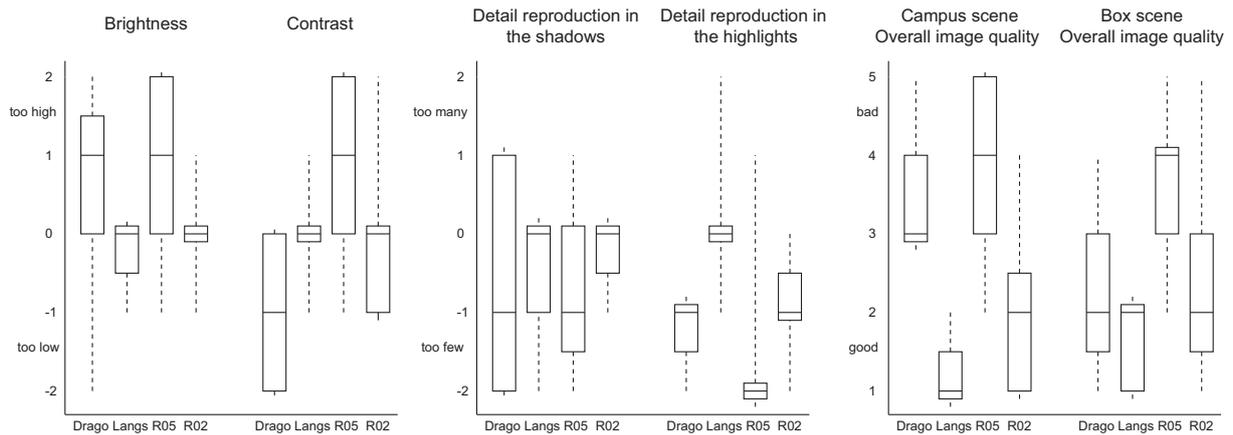


Figure 5: Tone mapping criteria evaluation. The operators are labeled as Drago2003 (Drago), Interactive Tonemapping (Langs), Reinhard2005 (R05), and Reinhard2002 (R02): the box shows the lower quartile, median, and upper quartile values. The dashed lines (whiskers) extend from the maximum to the minimum values of the datasets

and the box itself. In addition, the test persons were asked to rate the usability of the tone mapping interface and the time needed to create a curve. The last task was to compare the result of the own curve applied to the box scene with the results of the selected popular tone mapping methods. A grade was requested for the overall quality of each result.

5 RESULTS

5.1 Campus scene

The first criterion to be evaluated was the brightness of each image. Reinhard's Photographic Tone Reproduction (henceforth referred to as Reinhard02) was rated best, followed closely by the curve created with the interactive tone mapping approach. Both results from Drago's Adaptive Logarithmic Mapping (Drago03) and Reinhard and Devlin's Photoreceptor Model (Reinhard05) were rated too bright, probably because the bag and chair areas were not displayed bright or dark enough. The contrast was reproduced best by the interactive tone mapping operator. Reinhard02 created the second best contrast tending towards too little contrast. Again, the results of Drago03 and Reinhard05 were not satisfying. Reproduction of detail in the shadows was solved best by Reinhard02, followed by the interactive curve. The grades of the test persons for Drago03 were very scattered but tended to have too few details. Reinhard05 did not deliver sufficient details in the shadows. Details in highlights were preserved best by the interactive operator. All the other tone mapping algorithms retained too few details, according to our test persons. To summarize, Reinhard02 and the interactive operator performed equally well. Both the Drago03 and Reinhard05 tone mapping could not compete in our first scenario.

This observation is supported by the grades given for the overall image quality. If we can assume, that the grades are interval data, the interactive operator received the average grade 1.2, Reinhard02 2.0, Drago03 3.5, and Reinhard05 3.9 where 1 is very good and 5 is very bad. The Box-Whisker-Plots in figure 5 show all obtained data with median, upper and lower quartile and maximum plus minimum values. Anyhow, we can conclude that not a single criterion is important for the overall image quality because otherwise the grades for the interactive tone mapping operator and Reinhard02 grades would not differ so much. An expert curve seems to provide the best overall image quality, but needs to be created carefully beforehand for each scene.

5.2 Box scene

Fourteen curves have been created for the box scene over the course of the evaluation, one of which we created ourselves as a kind of expert curve. The created curves are very different because the emphasis has been set differently by the users: some only tried to map the greyscale correctly; others mapped all colors on the colorchecker precisely. Because of the latter outlier points have been inserted making the resulting curve bumpy and not monotonically increasing. This results in some artifacts and those points have therefore been erased from the dataset after the evaluation. In figure 6 those normalized curves are depicted. In this context, we thought of a standard observer curve averaged over all curves. This curve is shown in figure 6 on the right hand side. The standard observer curve matches the Drago03 curve for the box scene in a wide range of input luminance values. Reinhard02's curve for this scene has a similar shape. This might be caused by the box scene since the dynamic range ($\approx 300:1$) is close to the dynamic range of the display ($\approx 100:1$).

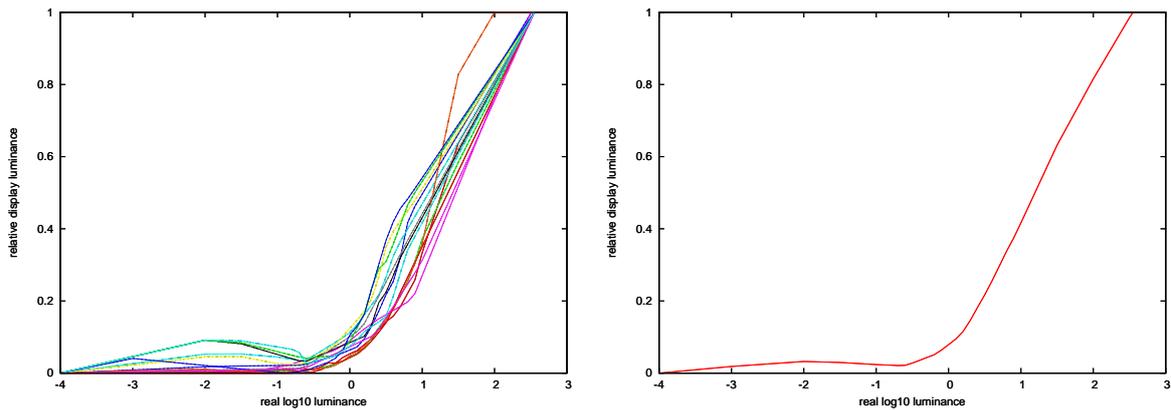


Figure 6: Tone mapping curves of the test persons (left hand side) and the standard observer curve (right hand side)

The participants needed between 5 and 24 control points with an average of 13 to create a satisfying curve. The linear interpolation seems to suffice for a reliable result. The probands required 15 minutes on average to generate their curves, which was overall rated as an acceptable time frame. Also, the users were pleased with the usability of the curve tool. After a short orientation time every participant was able to create a curve with an outcome they found acceptable.

At last, another grade for overall image quality has been given for the results of the individual curve and the other three benchmark operators. On average, the users were very content with their individual LDR image (average grade 1.7) followed by the results of Drago03 and Reinhard02 (2.3 and 2.4, respectively) and Reinhard05 (3.8). This is not surprising since we already concluded that the first three mentioned tone mapping curves are quite similar. Still, a specially created curve for a single selected scene yielded the best perceived overall quality.

5.3 Comparison to other evaluations

Kuang et al. [9] compared six tone mapping algorithms in their study, two global and four local ones. They used HDR images without reference and some invariant constructed scenes for a direct comparison. None of the tone mapping operators performed well for every image. Thus it is reasoned that there is a significant dependency between the tone reproduction method and the selected scenario. The Photographic Tone Reproduction and the *Bilateral Fast Filter* by Durand et al. [6], two of the four local operators, performed best on average. Both global operators, the *Sigmoid Transformation* [2] and Histogram Adjustment [18], did not achieve good results overall.

Another work from Yoshida et al. [19] examined seven tone mapping operators in direct comparison. Several criteria had to be rated by the participants.

Most importantly they found out that global operators achieved a higher contrast whereas local operators retain details better. No single criterion was deemed to exclusively contribute to the overall image quality. The results of the Photographic Tone Reproduction, Histogram Adjustment and Adaptive Logarithmic Mapping [5] were rated best in the overall image quality, the last two mentioned being global operators.

In fact, Cadik et al. suggest in their evaluation [3] with fourteen tone mapping methods that global operators perform better in terms of overall image quality. Local ones can only compete if they have a proper global part. Criteria, which apply to the whole image, like contrast, are most important for the overall quality. According to [3], this is why global operators exhibit better results. An example of a good local operator with a strong global part is the Photographic Tone Reproduction once more.

Comparing those observations with our results yields some conformity: Photographic Tone Reproduction produces good results in every evaluation including ours. Because our interactive tone mapping operator even exceeds the results of the Reinhard02 operator, it might be quite possible for our results to be reproduced under different conditions, although accompanied by a lot of work due to the need of a separate curve for every scene. The global approach of our tone mapping method can be reassured by the results of studies revealing that a global tone mapping operator delivers a better perceived overall image quality. A reliable result for every scene was the goal of our implementation and can be obtained by creating an individual curve.

6 CONCLUSION AND FUTURE WORK

In this paper, we proposed to apply an interactive and highly adjustable user-driven tone mapping function as

a dedicated tone reproduction operator for individual high dynamic range scenarios and specific presentation setups. This operator was designed and developed for industrial product design as main field of application, where decision-makers rely on trustworthy colorimetric and photometric lighting simulation and reproduction. The expert user is guided by our tool to define and to apply a set of key mappings between the scene luminance values and low dynamic range display quantities. These two-dimensional control points are interpolated to a tone reproduction curve that can be stored, loaded or compared to other tone mapping curves.

To validate our approach, we chose criteria for reliability and popular tone mapping operators as benchmarks. A user study with thirteen participants and two very different scenarios was conducted. The first experiment was based on a relative comparison of a user-defined expert curve and existing tone mapping operators from literature without direct reference. In the second experiment, the test persons were asked to generate their own tone reproduction curves in a scenario with direct reference to a well-defined real world setup and to compare the resulting display images with other published operators. Our findings include that the probands were able to create satisfying operators with a small number of thirteen key mappings in an acceptable time frame of fifteen minutes on average. The users rated the reliability of the individual curves higher than the results of existing tone mapping operators. Finally, we presented a standard observer tone mapping curve, generated by averaging the results of a number of test persons for an individual scene and presentation setup.

We expect interesting results from future work on evaluating similarities within groups of standard observer operators from different sets of scenes and presentation setups. Another future application of our approach is to generate the tone mapping curve using existing operators from literature and to use our tool for manual fine tuning. Lastly, we are looking forward to the results of evaluating our user-driven tone mapping operators in an industrial product design environment, where our tool was designed for.

REFERENCES

- [1] Michael Ashikhmin. A tone mapping algorithm for high contrast images. In P. Debevec and S. Gibson, editors, *13th Eurographics Workshop on Rendering*, pages 145–155. The Eurographics Association, 2002.
- [2] G.J. Braun and M.D. Fairchild. Image lightness rescaling using Sigmoidal contrast enhancement functions. *IS&T/SPIE Electronic Imaging '99, Color Imaging: Device Independent Color, Color Hardcopy, and Graphic Arts IV*:96–105, 1999.
- [3] Martin Cadik, Michael Wimmer, Laszlo Neumann, and Alessandro Artusi. Evaluation of HDR tone mapping methods using essential perceptual attributes. *Computers & Graphics*, 32:330–349, 2008.
- [4] K. Devlin, A. Chalmers, Alexander Wilkie, and Werner Purghofer. Star report on tone reproduction and physically based spectral rendering. In *Eurographics 2002*. Eurographics Association, 2002.
- [5] F. Drago, K. Myszkowski, T. Annen, and N. Chiba. Adaptive logarithmic mapping for displaying high contrast scenes. *Computer Graphics Forum*, 22:419–426, 2003.
- [6] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. In *Proceedings of ACM SIGGRAPH 2002, Annual Conference Series*, pages 257–266, 2002.
- [7] Raanan Fattal, Dani Lischinski, and Michael Werman. Gradient domain high dynamic range compression. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 249–256, New York, NY, USA, 2002. ACM.
- [8] R.W.G. Hunt. *The Reproduction of Colour in Photography, Printing and Television*. Fountain Press, Tolworth, 5th edition, 1995.
- [9] Jiangtao Kuang, Hiroshi Yamaguchi, Changmeng Liu, Garrett M. Johnson, and Mark D. Fairchild. Evaluating hdr rendering algorithms. <http://www.cis.rit.edu/fairchild/PDFs/PAP24.pdf>, 2006.
- [10] Rafal Mantiuk and Hans-Peter Seidel. Modeling a generic tone-mapping operator. *Computer Graphics Forum (Proc. EUROGRAPHICS)*, 27(2):699–708, 2008.
- [11] John J. McCann. Art, science, and appearance in hdr images. In *Journal of the Society for Information Display*, volume 15 (9), pages 709–719, September 2007.
- [12] QTPFSGUI. Open source graphical user interface application that aims to provide a workflow for hdr imaging. <http://qtpfsgui.sourceforge.net/>.
- [13] Erik Reinhard and Kate Devlin. Dynamic range reduction inspired by photoreceptor physiology. *IEEE Transactions on Visualization and Computer Graphics*, 11(1):13–24, 2005.
- [14] Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. Photographic tone reproduction for digital images. *ACM Trans. Graph.*, 21(3):267–276, 2002.
- [15] Erik Reinhard, Greg Ward, Sumanta Pattanaik, and Paul Debevec. *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.
- [16] Jack Tumblin and Holly Rushmeier. Tone reproduction for realistic images. *IEEE Comput. Graph. Appl.*, 13(6):42–48, 1993.
- [17] Greg Ward. A contrast-based scalefactor for luminance display. In Paul Heckbert, editor, *Graphics gems IV*, pages 415–421. Academic Press Professional, Inc., San Diego, CA, USA, 1994.
- [18] Gregory Ward Larson, Holly Rushmeier, and Christine Piatko. A visibility matching tone reproduction operator for high dynamic range scenes. In *SIGGRAPH '97: ACM SIGGRAPH 97 Visual Proceedings: The art and interdisciplinary programs of SIGGRAPH '97*, page 155, New York, NY, USA, 1997. ACM.
- [19] Akiko Yoshida, Volker Blanz, Karol Myszkowski, and Hans-Peter Seidel. Perceptual evaluation of tone mapping operators with real-world scenes. In Bernice E. Rogowitz, Thrasyvoulos N. Pappas, and Scott J. Daly, editors, *Human Vision and Electronic Imaging X, IS&T/SPIE's 17th Annual Symposium on Electronic Imaging (2005)*, volume 5666 of *SPIE Proceedings Series*, pages 192–203, San Jose, USA, January 2005. SPIE.

Multiscale Texture Synthesis and Colourization of Greyscale Textures

Anders Hast
Creative Media Lab
University of Gävle
Kungsbäcksvägen 47
SE-801 76, Gävle, Sweden
aht@hig.se

Martin Ericsson
UPPMAX
Uppsala University
Lägerhyddsvägen 2
SE-751 05, Uppsala, Sweden
martin.ericsson@it.uu.se

Stefan Seipel
GraphiX Center
University of Gävle
Kungsbäcksvägen 47
SE-801 76, Gävle, Sweden
ssl@hig.se

ABSTRACT

The main idea presented herein is to use a multiscale texture synthesis approach in order to both colourize and upscale greyscale textures. Such textures can be vintage photos to be used in archaeological or urban 3D visualizations and obviously the colour needs to be reconstructed some way. Due to limited quality, walls etc in such 3D visualizations will appear either pixelized or blurry when the viewer approaches them on a close distance. The latter if some kind of interpolation technique is being used to reduce the pixelization. The low resolution greyscale texture and a high resolution coloured texture is used for the colourization and upscaling, which will produce a colour version of the greyscale texture with 4 times higher resolution in each upscale step. The novel idea is to use multiscale texture synthesis in HSV space for the first upscale in order to create a RGB colour image for subsequent upscaling, using either ordinary RGB multiscale texture synthesis or continue using HSV multiscale texture synthesis. These two main approaches will be compared and discussed.

Keywords

Multiscale Texture Synthesis, Colorization, Colour transfer, Greyscale Photos.

1. INTRODUCTION

In the process of 3D virtual reconstruction and visualization of buildings it is necessary to acquire textures of walls etc and often photographs are being used for obtaining the textures. However such walls and parts of buildings, especially for archaeological visualizations, might not exist anymore and it is therefore necessary to use old photos, often greyscale ones. There are hence two problems that must be solved. First of all the colour must be reconstructed, which might be a hard task unless we know at least something about the colours that were used when the building was still standing. Furthermore the quality must be improved because aliasing problems will occur when these textures are used for the 3D models. Usually different interpolation techniques are being used in order to minimize the aliasing effect

when one moves close to the walls. One drawback with antialiasing [Fol97] is that it will make the texture look blurry, however this is preferred over having the pixels appear like big homogeneous square blocks, which makes the texture look pixelized.

We propose a novel approach for the colourization of a greyscale textures and a subsequent upscaling in order to increase the resolution so that the blurriness of interpolation can be avoided, using a modified variant of multiscale texture synthesis.

Multiscale Texture Synthesis

This paper does not deal with ordinary texture synthesis in general, but a short introduction will be given before we explain the basics of multiscale texture synthesis.

1.1.1 Texture Synthesis

Texture synthesis (TS) is the process of taking one smaller texture and then make it larger in size, not by tiling, but by synthesizing it [Efr99, Wei00]. Several approaches exist and the hierarchical TS method [Hee95] builds a tree of the texture with different sizes very much like in the mipmapping method [Wil83]. The smallest texture (on the lowest level) is then used in the first step and texels are randomly

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

taken from it and randomly inserted into the new synthesized texture of corresponding size. Then follows a process where a mask with a specific shape [Har01] is scanning the synthesized texture in a scanline order fashion while copying texels from the original texture, which has the best matching neighbourhood [We02]. Generally the matching is computed as the sum of the squared differences of the RGB values within the masks. In the same time as the texture is synthesized on this level, another texture is synthesized on a higher level by copying a 2x2 neighbourhood into that texture, which accordingly will be 4 times larger.

1.1.2 Multiscale Texture Synthesis

In multiscale texture synthesis [Lee08] (MSTS) there already exists a texture version available of the otherwise initially randomized and then synthesized texture, namely the target texture. Then a number of exemplar textures are taken so that they will contain similar details like the target texture but on higher levels, and they can subsequently be used to build a more detailed version of the target texture. An exemplar graph is built for this purpose where the target texture is placed in the root and textures with higher details are placed on the next levels depending on their resolution. One texture on one level can thus depend on several exemplar textures on other levels. Since the colours can differ on different levels it has been proposed to use a colour transfer function [Han08]. Another approach can be used when the colours in the target texture are substantially different from the exemplar texture [Has09], i.e. when the matching is bad.

1.1.3 HSV Multiscale Texture Synthesis

The HSV colour space MSTS method takes a different approach using only one exemplar image [Has09]. As an example: when 3D reconstruction of buildings is used by the proposed approach it is possible to use an image of the whole wall or large parts of a wall (see figure 1) using a high resolution camera.

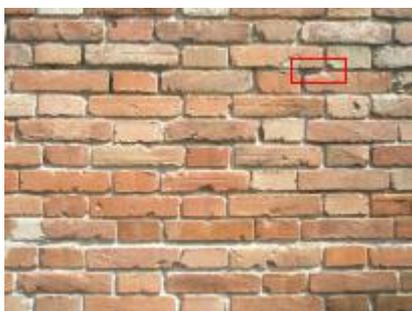


Figure 1. The target texture (204x153 pixels) with a red rectangle showing what part that will be zoomed.

The inserted details can be taken from an exemplar image taken from the same wall. This image will be taken on a close range and will therefore cover a small part of the wall as shown in figure 2. Note that the texture to the left has been down sampled to fit in the paper. To the right is shown a small part, inside the red rectangle, in its actual resolution. It is obvious that the exemplar texture has a high resolution compared to the target texture in figure 2, which is also shown in its actual resolution.

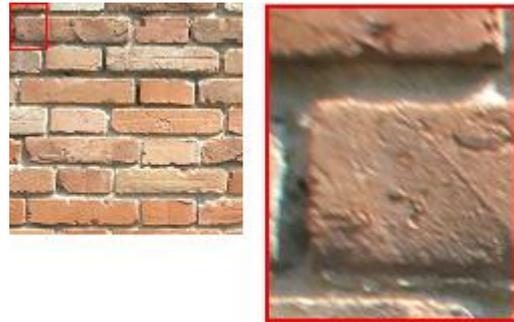


Figure 2. The exemplar texture to the left, covering a smaller part of the wall. As shown to the right it has a high resolution (768x768 pixels) that will be used for the upscaling.

The HSV method for MSTS (HSV-MSTS) can handle colour differences in the following way: Let us say that a single brick in a brick wall have a greenish tone in the otherwise red wall. Then this problem can be handled by converting the colours into the Hue, Saturation and Value (HSV) colour space [Son99]. The HSV colour model separates the colour into three channels, similar to the more common red, green and blue colour model (RGB) but instead it uses a measurement of hue, saturation and value also called brightness. This model of representing colours gives the ability to change the brightness independent of the other colour information in the picture. As the human perceptive system is more sensitive to brightness discrepancies, this potentially can give a perceptually better image.

In figure 3 the original image to the left is compared to the resulting texture from ordinary MSTS in the middle and the image to the right shows the HSV-MSTS. It is obvious that the colours are not represented correctly in the middle image. Converting to HSV space and synthesizing the V part while interpolating H and S, will give a much more accurate result. The problematic greenish brick (inside the red triangle) is synthesized keeping the greenish tone, using the HSV approach, while the ordinary MSTS makes it more red than green.

It should be noted that the target texture was originally larger in size and then down sampled by averaging 4 pixels into one. In this way it was possible to compare how close to the real thing the synthesis process was, i.e. we have a ground truth to compare with. Hence the texture to the right in figure 3 has a higher resolution than what was used in the process.

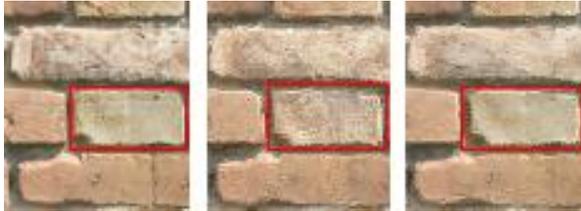


Figure 3. The high resolution target image to the left and a synthesised version in the middle using ordinary MSTs yielding colours, which are far from correct compared to the right where the HSV approach is used.

The synthesized textures on higher levels are constructed using the synthesized V elements and the H and S elements are taken from the target texture, which ensures that the original colours of the bricks are maintained. However it is important that the H and S elements are interpolated, e.g. bi-linearly (or using some other interpolation scheme [Gon93]), in the upscaling process, otherwise the colour will be visible as blocks.



Figure 4. Top: pixelization. Middle: Interpolation: Bottom: Multiscale Texture Synthesis (HSV).

In our approach a simple variant of bilinear interpolation was used taking into account only the 4-neighbours [Son99]. The result is shown in figure 4. In the top it can be seen how the texture (within the red rectangle in figure 1) is magnified without interpolation and in the middle interpolation has been used. Nonetheless the result is far from appealing. The result of the previously explained HSV-MSTS approach is shown in the bottom. It is obvious that this approach inserts details, making the image looking much better than just using interpolation in order to get rid of the pixelization.

Colourization of Greyscale Textures

Greyscale texture and image colourization is applied for an example in greyscale photo editing and scientific illustrations. The process of colourization increases the visual appeal of greyscale images and can perceptually enhance scientific illustrations [Che04]. It has also been used for colourization of classic movies, even though not all are that happy about that the visual experience is changed [Dan90].

Region based colourization can be performed by combining greyscale image matting algorithms [Smi96] with colour transferring techniques [Wel04]. First objects with that will have different colours are extracted from the greyscale image. Then each object is colorized using colour transferring and then these colorized objects are seamlessly composited [Mor95, Por84].

Colourization can be either user guided [Lev04] or automatic. And there are also techniques that use a combination of these two [Iro05]. The user guided method requires the user to scribble the desired colours in the interiors of the various regions. On the other hand, automatic techniques like the one proposed by Welsh et al [Wel02] colorizes an image by matching small pixel neighbourhoods in the image to those in the exemplar image, and transferring colours accordingly. Hence they propose to use a variant of texture synthesis since they are matching local pixel luminance statistics between colour example and target grey-scale image.

The procedure according to Welsh et al [Wel02] and later used by Karthikeyani et al [Kar07] is as follows: first each image is converted into the $\alpha\beta$ -colour space [Rud98]. (Similarly Pan et al [Pan04] used this space to add colour to video and animation clips). Then jittered sampling is used to select a small subset of pixels in the colour image as samples. Next each texel in the greyscale image is traversed in scan-line order and the best matching is selected using neighbourhood statistics within in a 5x5 mask. The best match is determined by using a weighted average of texel luminance and the neighbourhood statistics. The chromaticity values (α, β channels) of the best

match are then transferred to the greyscale. In order to obtain a better correspondence in the luminance range between the two images, luminance remapping [Her02] is performed.

2. COLOURIZING USING HSV-MSTS

This paper proposes how textures like wall textures can be colourized and upscaled using a novel idea that differs from the idea proposed mainly by Welsh et al [Wel02]. First of all we show that HSV-MSTS can be used for the colour transfer. Secondly we show that the upscaling process can be integrated in the process.

One of the reasons to use HSV-MSTS is that jittered samples would fail to find enough texels containing the mortar in the brick wall examples, unless the amount of samples is heavily increased. Furthermore we have found that the matching differences of the V value of each texel using a 3x3 mask is enough for a visually pleasing result, instead of a 5x5 mask matching neighbourhood statistics.

The novel idea is to use the previously explained fact that texture synthesis can be performed in HSV using the V channel for matching, in order to colourize a greyscale photo. In fact, for a greyscale photo only the V channel contains any information since there is no colour that can be saturated or be defined by its hue. Figure 5 shows three target textures in greyscale that will be colourized by the proposed approach. The exemplar texture will be the same as shown in figure 2, that is downscaled to a size that corresponds to the size of the target textures. However we will do the matching with one difference, we will use a greyscale version of that texture for matching. Then we can proceed in two ways. When a best match is found we can take the corresponding pixel from the colour version of the exemplar texture, either on the same level to construct a coloured version of the texture. As an alternative we can go a head and take the four corresponding pixels from the texture on a higher level (4 times larger), and hence make one step of upscaling on-the-fly.

Obtaining the Greyscale Exemplar

Anyhow, the greyscale exemplar image must be obtained from the colour version and this can be done in many ways. Since this paper presents a proof of concept rather than being applied on any certain vintage photo, we computed the greyscale exemplar in the same way as we computed the target image as it was originally a colour image too.

Generally the greyscale value can be computed as a weighted sum of the RGB value:

$$g = r * w_r + g * w_g + b * w_b \quad (1)$$

Here we have the opportunity to arrange the weights so that the proposed algorithm works in the best way, i.e. the grey level histograms of the two images should be made as similar as possible.

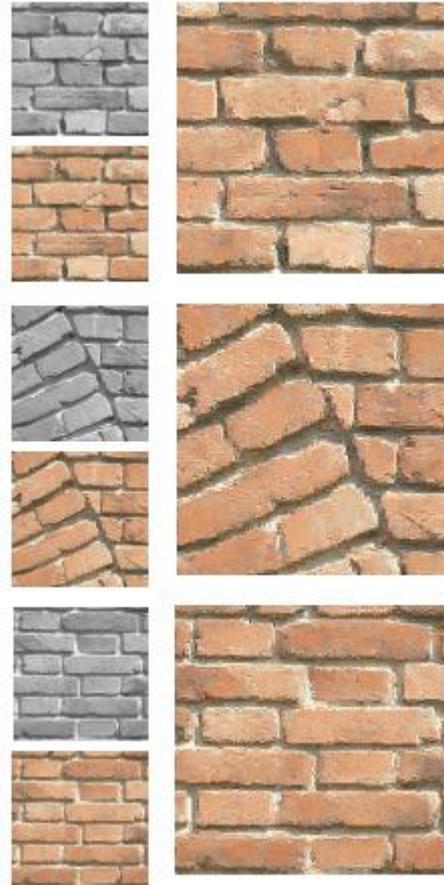


Figure 5. The three walls (top left) are colourized with no upscale (bottom left) and one step of ‘on-the-fly’ upscale to the right.

3. DISCUSSION

It should be noted that the target textures shown in figure 5 have a relatively low resolution, which is often the case for vintage photos. Nonetheless the algorithm works well despite the low resolution, but gives even better results if the resolution is higher. After the first colorization step we can proceed in another two ways, either we continue using the HSV variant or the RGB variant of MSTS.

In figure 6 the result from the subsequent upscalings are shown. Here we have been using the HSV approach all along. It appears like the bottom texture has a more spotty appearance compared to the more homogenously coloured texture in the top. Obviously the on-the-fly upscaling introduces a noisy behaviour and it is better to upscale a colourized texture twice. Besides that, the textures are quite equal in appearance.

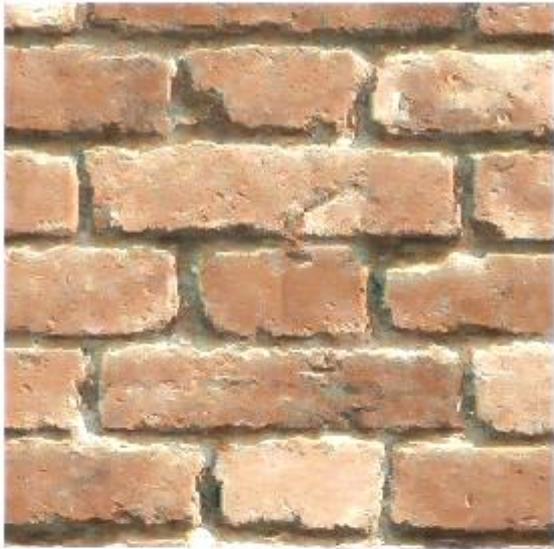


Figure 6. Top: the colourized wall HSV upscaled twice. Bottom: the colourized and on-the-fly HSV upscaled wall is then HSV upscaled once.

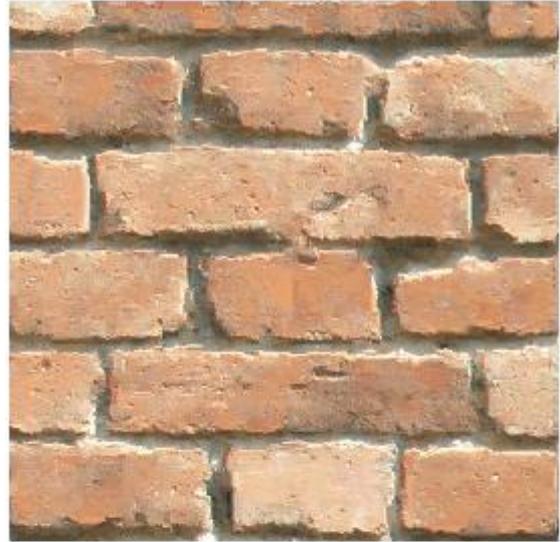


Figure 7. Top: the colourized wall upscaled twice. Bottom: the colourized and on the fly upscaled wall is upscaled once.

It should be noted that the V channel has been scaled with a small factor since the output was a bit darker compared to the original colour in the exemplar texture.

Next we go on to examine what happens if we would use the RGB variant for the subsequent steps and the result is shown in figure 7. It is quite hard to tell any difference in quality within the bricks, however the bottom images seem to have a less tendency to smear out details so that bricks become connected when there is just one or two pixels containing the mortar or when the border between bricks is quite fuzzy due to the low resolution. Therefore it seems like on-the-fly upscaling is to prefer when using the RGB-MSTS of the colorized textures but not for the HSV-MSTS.

Furthermore it can be noticed that the smearing out does not really occur at all when using the HSV-MSTS all along.

A close up of another target texture showing the edge of a brick arch, that has been upscaled so it is 16 times larger is shown in figure 8.



Figure 8. Left: the colourized wall upscaled twice. Middle: The original greyscale Right: The on-the-fly upscaled wall is upscaled once.

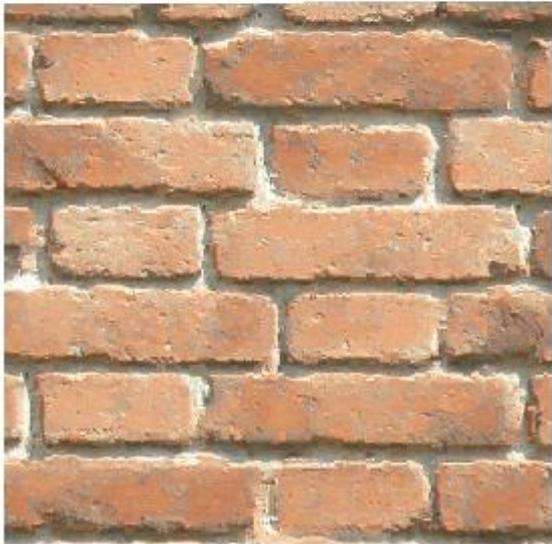
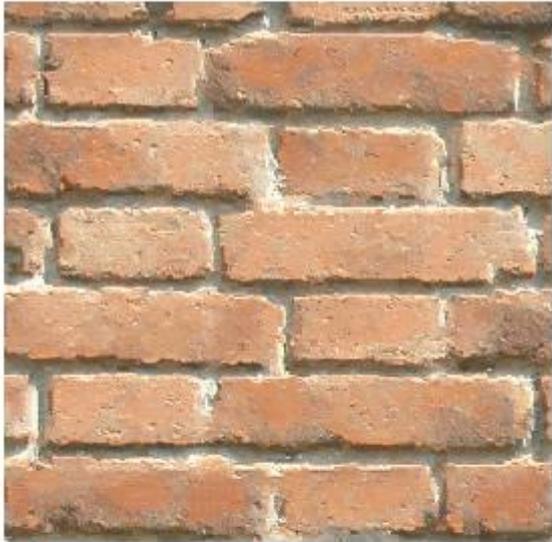


Figure 9. Top: the colourized wall upscaled twice. Bottom: the colourized and on the fly upscaled wall is upscaled once.

It is clear from the comparison that the on-the-fly upscaling generally gives a better result for greyscale textures with relatively low resolution as has been used in our tests. The same conclusion can be drawn from figure 9 and 10 where the other two test textures are shown (see also figure 5).

It should be remembered that even of the bottom textures are better they are not perfect and that is due to the extremely low resolution of the target textures used in our tests in order to test if the algorithm works for extreme cases. And therefore using higher spatial resolution will give even better results. It should also be noted that artefacts to great extent come from the fact that the matching was bad in these areas.

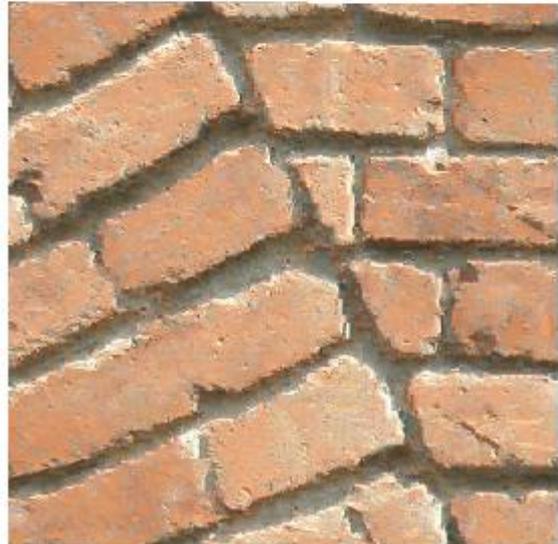


Figure 10. Top: the colourized wall upscaled twice. Bottom: the colourized and on the fly upscaled wall is upscaled once.

This depends on too large differences in the image content between the target texture and the exemplar. This could to great extent have been avoided using pre-processing of the images in order to normalize the intensities so that the impact of the flash etc is removed.

The tests were all performed in software and as we used rather small images time was not an issue. However, as shown in [Has09] we have implemented a fast version on the GPU and a parallel implementation on a HPC cluster is on the way. This will allow us to develop the idea further and make more thorough tests on larger images.

4. CONCLUSIONS

We have shown that colorization and upscaling of low resolution greyscale images can be performed using the recently published HSV-MSTS approach, using small masks like in our example a size of 3x3. Our tests also indicate that subsequent upscaling will become visually slightly better using RGB-MSTS and that on-the-fly upscaling is to prefer for this case.

Future Work

In the future we intend to develop the HSV-MSTS approach where one important task is to compare other colour spaces like the HSL and the $l\alpha\beta$ -colour spaces. Another important task is to work on real vintage photos for colorization and upscaling.

5. REFERENCES

- [Ale99] Alexei A. Efros and Thomas K. Leung, Texture Synthesis by Non-parametric Sampling, In Proceedings of ICCV 99, pp 1033-1038. 1999.
- [Che04] T. Chen, Y. Wang, V. Schillings, and C. Meinel, Greyscale Image Matting and Colourization, Proceedings of Asian Conference on Computer Vision, pp. 1164-1169, 2004.
- [Dan90] C. B. Daniels, Note on Colourization, The British Journal of Aesthetics, 30(1), pp68-70, 1990.
- [Fol97] Foley, J. D., van Dam, A., Feiner, S. K., Hughes, J. F. Computer Graphics - Principles and Practice, Addison-Wesley, pp. 617-646. 1997.
- [Gon93] Gonzales, R. C., Woods, R. E. Digital Image Processing, Addison-Wesley, pp. 300-302. 1993.
- [Han08] Han, C., Risser, E., Ramamoorthi, R. and Grinspun, E., Multiscale Texture Synthesis. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008), 27(3) pp. 51. 2008.
- [Har01] P. Harrison, A non-hierarchical procedure for re-synthesis of complex textures. In proceedings of the WSCG'01, pp 190-197, 2001.
- [Has09] A. Hast, M. Ericsson, T. Reiner, Improved Textures for 3D Virtual Reconstruction and Visualization by a Modified Multiscale Texture Synthesis Approach, ARCH-3D, pp. 1-6, 2009.
- [Hee95] Heeger, D. and Bergen, J. Pyramid-Based Texture Analysis / Synthesis. SIGGRAPH pp. 229-238. 1995.
- [Her02] A. Hertzmann, , C. Jacobs, N. Oliver, B. Curless, D. Salesin, 2001. Image Analogies, In Proceedings of ACM SIGGRAPH, pp. 341-346. 2002.
- [Kar07] V. Karthikeyani, K. Duraiswamy, P. Kamalakkannan, Conversion of Grey-scale image to Color Image with and without Texture Synthesis IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.4, April 2007.
- [Lee08] Lee, S-H., Park, H-W., Lee, J. and Kim, C-H. Multi-scale Texture Synthesis. Journal of KCGS (The Korea Computer Graphics Society), 14(2). 2008.
- [Lev04] A. Levin, D. Lischinski, and Y. Weiss, Colorization using Optimization, Proceedings of ACM SIGGRAPH, pp. 689-694, 2004.
- [Mor95] E. N. Mortensen and W. A. Barrett, Intelligent scissors for image composition, in Proceedings of ACM SIGGRAPH, pp. 191-198, 1995.
- [Pan04] Z. Pan, Z. Dong, M. Zhang, A New Algorithm for Adding Color to Video or Animation Clips, WSCG, pp.515-519, 2004.
- [Por84] T. Porter and T. Duff, Compositing digital images, in Proceedings of SIGGRAPH, pp. 253-259. 1984.
- [Rud98] D. L. Rudermann., T. W. Cronin and C. C. Chiao, Statistics of Cone Responses to Natural Images: Implications for Visual Coding, J. Optical Soc. Of America, vol 15, no. 8, pp. 2036-2045, 1998.
- [Wei00] Li-Yi Wei, Marc Levoy. Fast Texture Synthesis using Tree-structured Vector Quantization, Proceedings of Siggraph, pp 479-488. 2000.
- [Wei02] Li-Yi Wei. Texture synthesis by fixed neighborhood searching. PhD Thesis, Stanford University, Palo Alto, CA, USA, 2002.
- [Wel02] T. Welsh, M. Ashikhmin, and K. Mueller, Transferring colour to greyscale images, ACM TOG, vol. 20, no. 3, pp. 277-280, 2002.
- [Wil83] Lance Williams, Pyramidal Parametrics, Computer Graphics, 17(3) pp.1-11. 1983
- [Son99] Sonka, M., Hlavac, V. and Boyle, R. Image Processing, Analysis, and Machine Vision. Thomson Learning, USA. pp.28, 38. 1999.
- [Smi96] A. R. Smith and J. F. Blinn, Blue screen matting, in Proceedings of ACM SIGGRAPH, pp. 259-268. 1996.

Making People Move – Walking Techniques in a CAVE

Michael Wegner
University of Applied Sciences
Dresden, Germany
michael.wegner@informatik.htw-
dresden.de

Markus Wacker
University of Applied Sciences
Dresden, Germany
wacker@informatik.htw-
dresden.de

ABSTRACT

Among navigation techniques in Virtual Environments (VEs) physical walking is most natural and intuitive. If we look at the performance of users in a CAVE though, we notice that they almost never leave their starting point. In this paper we investigate how walking can be stimulated for navigation in a CAVE. However, our goal is not merely to mimic walking as such – as in most approaches – but rather to encourage users to take advantage of the entire tracking space at their disposal. We combine our proposed walking elements with other components to create new metaphors for navigation in VEs and compile the evaluation carried out during our thorough, informal test phase.

Keywords: interaction techniques, CAVE, navigation, travel task, walking

1 INTRODUCTION

Being one of the primary interactions in Virtual Environments (VEs), navigation has been an area of intensive research since the beginnings of virtual reality. Hence a vast literature and many approaches for navigation in VEs exist. It is clearly true that physical walking is one of the most natural and intuitive ways to navigate (cf. the study by Ruddle et al. [11]). If we look at the performance of users (especially novices to the system) in a CAVE however, we notice that they mostly seem to be pinpointed to the ground, almost never leaving their starting point even if the system offers an adequate space to move in. In this paper we investigate walking techniques that motivate the user to take advantage of the entire tracking space at their disposal. We combine these ideas with other navigation elements to create new metaphors for navigation in VEs. Here we concentrate on navigation in a CAVE with its easy and relatively unimpeded possibility for physical movement tracking (e.g. no cables or backpacks). The starting point for our research was to design navigation techniques for the exploration of machine models in a CAVE, that are moderately larger than the available physical volume. During our investigation we have developed a whole set of navigation techniques suitable for different travel tasks which we present here. Moreover we consider walking as a tool for building effective travel techniques and encouraging this natural movement. Our contribution consists in extending existing taxonomies by integrating the walking metaphor both in the theoretical con-

cept as well as providing concrete practical travel techniques.

It is important to have our technical setup in mind since it is vital for our modes of navigation: We use a five-sided CAVE (only back side is missing) with rectangular back projection and a resolution of 1600x1200 pixels on screens of 3.6 meter length and 2.4 meter height each. This means, that the side walls exceed the floor and the ceiling projection. The stereo effect is achieved using the INFITEC filter technique. Tracking of the user is done via an infrared camera setup from ART with retroreflective markers on trackable objects. Examples are the master INFITEC glasses for tracking the main user's head position and orientation (which we take for the position of the user) or a flightstick – a 6DOF device with additional buttons.

The remainder of this paper is organized as follows: first we give an overview of relevant existing approaches concerning navigation in a CAVE and classify our contribution. Next we recapitulate the basic concepts of a taxonomy for navigation. Afterwards we describe our navigation elements separately, combining them to whole navigation techniques in the following section. Finally we give an overview of user experience with the new navigation techniques, summarize our contribution, and give an outlook to future investigation.

2 RELATED WORK

Concerning navigation techniques, Anthes et al. [1] distinguish between movement vector and gaze orientation and give a host of models for each one. Tan et al. [13] propose a rather complex task-based taxonomy. Based on what the user is supposed to do in the VE (task selection) the designer develops an abstract solution to the problem (travel control) within the boundaries of available hardware and similar restrictions (user interface). In this paper we work the other way around, using the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

travel tasks described in [4] to evaluate our travel techniques in the end (see section 3).

There has been extensive research into walking interfaces, offering a multitude of different approaches in order to overcome the physical limits of the restricted space of interaction. However, all remain single standing solutions, none of them relating their technique to a high level taxonomy. Moreover their goal generally seems to be to mimic walking as closely as possible within the confines of a small tracking space.

A common approach for building walking interfaces is to introduce specialized hardware. Darken et al. [5] proposed an omni-directional treadmill, using two perpendicular treadmills to allow travel in any direction. Jiung-yao et al. [9] developed the gait sensing disc, an 'omni-directional ball-bearing disc locomotion device'. The CirculaFloor by Iwata et al. [8] uses movable tiles to achieve the same effect. None of these approaches however work well in a CAVE environment: Since here, in contrast to an HMD scenario, the user is still aware of his real world surroundings, additional visible hardware severely detracts from his sense of presence while also blocking at least part of the floor projection.

Another solution is to let the user mimic walking while actually staying in one place. Slater et al. [12] use a neural network to determine when the user is walking in place. In this approach the direction of travel is derived from the direction of the user's gaze while walking-in-place serves as the trigger for movement. The study concludes that Walking-in-Place yields a higher sense of presence than a pointing technique. Still, a later study (Usoh et al. [14]) pointed out '[...] that real walking is significantly better than both virtual walking and flying in ease (simplicity, straightforwardness, naturalness) as a mode of locomotion.' We do not think Walking-in-Place is particularly well suited for CAVEs since it tends to anchor the user in one place discouraging natural movement inside the projection space – a fact that we explicitly address in this paper.

Another answer to the problem of limited tracking space is the addition of translational or rotational gains to the user's movements in order to scale the virtual space or redirect the path the user is taking through the VE. Engel et al. [6] use a real-time controller to determine rotational gains on the fly and use these to redirect the user. However, since a CAVE is much smaller than the size of their tracking space (9x12 meters), this approach is not feasible for us. Interrante et al. [7] propose a metaphor of Seven League Boots similar to one of our ideas. Here, the covered walking distance is scaled in the VE but only in the direction the user is intending to walk. This is done by a weighted sum of gaze and walking direction. A study by Williams et al. [15] investigated how different translational gains affect performance for such techniques. They concluded

that even the highest tested gain (10:1) did not have a significant effect on errors or latency.

An approach especially designed for the CAVE is 'Redirected Walking in Place' by Razzaque et al. [10]. In order to avoid the user looking at the very often inexistent back wall and keep him turned towards the front, the rotation of the VE is constantly changed to compensate for the user's movements. This approach relies heavily on the user not making abrupt turns and not realizing that he is being made to turn by the simulation.

3 BASIC TOOLS

Bowman et al. [3] presented a task decomposition concept for the classification of travel techniques. Since it is relevant for our contribution and – in our eyes – a good starting point for designing interaction techniques and analyzing them, we here state the main ideas of this taxonomy. They divided the travel task into three subtasks:

- **Direction or target selection** specifies how or where the user moves,
- **Velocity/acceleration selection** specifies the speed control,
- **Condition of input** specifies how the travel is started, continued, and terminated.

In each subtask the developer can choose from a variety of possible components to form a complete travel technique (see figure 1). All in all Bowman et al. present

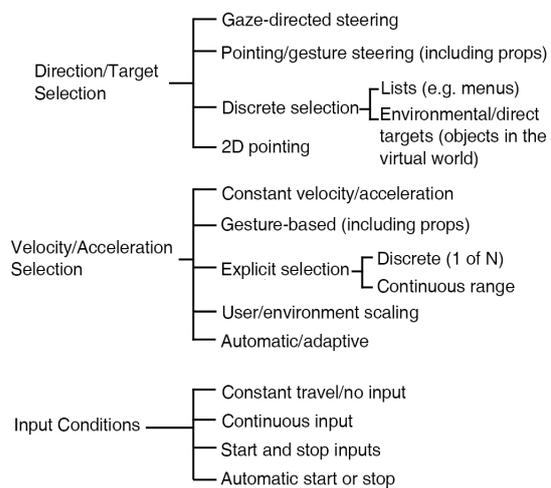


Figure 1: Taxonomy of travel techniques with travel subtasks (taken from [3]).

four different taxonomies to gain more complete understanding of the tasks and the techniques involved. However, none of these should be considered 'the correct one'. The one we have chosen presents the view on different components to form a whole technique.

4 EXTENDING THE TAXONOMY

When developing their taxonomy Bowman et al. explicitly do not take physical motion into account. Consequently we will describe in our first step how walking could be used in each component of the above taxonomy (for overview see figure 2).

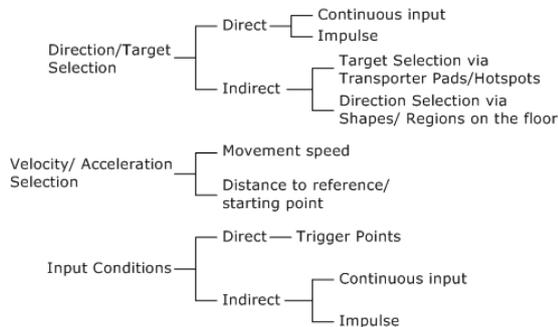


Figure 2: Extended Taxonomy with suggestions for walking interfaces

4.1 Direction/Target Selection

Generally speaking, there are two methods to determine the direction of travel via walking – direct and indirect.

In indirect approaches walking is only the means to reach certain key positions. The actual travel direction or target depends on the position of the user. A simple example are virtual transporter pads: if the user steps onto one he is automatically flown or teleported to his target. Of course this leaves him with little actual control. One might also assign certain directions of travel to certain positions or regions in the tracking space. In one of our techniques we assigned the direction of travel to each edge of the CAVE, respectively.

The direct approach records the actual movement direction of the user and applies it to the direction of travel in the VE. This works well as long as there is enough tracking space in the direction one wishes to travel. However, if the user is standing right beside a projection screen he will not be able to travel in this direction without first correcting his position. On the other hand, this approach allows for very natural relative travel (travel parallel to a reference point), because the user can look around freely while moving in another direction.

4.2 Velocity/Acceleration Selection

The first thought for a technique would be to use the speed of the user's movement to control the speed of travel. But since it is hard to estimate your own velocity, especially while the virtual world is being moved around you, we don't think this idea holds much potential, especially inside the cramped tracking space of a CAVE.

A more promising way to determine the speed of a travel technique through walking is to measure the distance from a specific point of interest (starting/reference point) to the actual position of the user. For example the velocity might increase if the user walks away from the center of the CAVE. Of course the starting point does not need to be fixed. One can also take the position of the user as a reference point when the technique is triggered. However, the longer the technique is active the higher the probability the user loses orientation and is unable to intuitively decelerate by walking back to where he came from.

4.3 Input Conditions

Walking can be used directly or indirectly to start or stop a travel technique. Indirectly by having the user move to or stand on special trigger points in the tracking space, and directly through movement itself. For example, a travel technique might be initiated or aborted by taking a step forward or backward respectively. However, since with this concept every movement would be considered a potential trigger, it effectively anchors the user in one spot – a problem we wanted to avoid. A travel technique might also be active as long as the user is actually walking around. For techniques that amplify movement this is a naturally occurring input condition (see our first proposed navigation technique in the next section).

5 NAVIGATION TECHNIQUES FOR A CAVE

Having expanded our toolbox with new walking elements, we now combine old and new elements of the taxonomy to give examples for new navigation techniques. These are intended to encourage users to move around inside the tracking space of a CAVE, opening the possibility for more intuitive navigation in a limited space of interaction.

5.1 Seven-league boots

Our first navigation technique is an exaggerated movement technique similar to the *Seven League Boots* proposed by Interrante et al. [7]. The user travels by walking around but his tracked movement in the real world is scaled to allow him to cover greater distances in the VE. To this end, we simply multiply the difference between the user's position in two successive frames by a (variable) gain.

Concerning taxonomy in this case we use movement direction to determine the direction of travel. Two input conditions have to be met for the travel technique to be active. First the user has to hold down a button while he is using the technique. Secondly he has to move around in order to get the desired effect. Of course one might consider the user's movement the only input condition

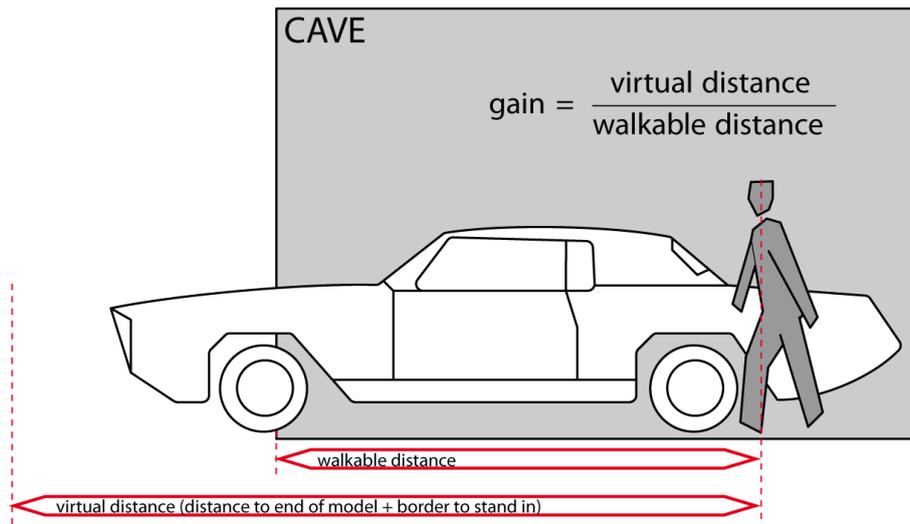


Figure 3: Automatic calculation of gain using virtual and walkable distance in our *Seven-league boots technique*

but this would make normal or even downscaled precision movement (without gain / not pressing the button) impossible (cf. exploration, search vs. maneuvering in section 6).

In this technique the velocity of travel is directly dependent on the user's velocity and the gain used to scale the movement. The easiest way to determine speed is to just assign a fixed value to the gain. Of course with a gain that is too small one constantly has to double back inside the tracking space in order to travel a significant distance in a specific direction. If the gain is too high one risks reaching points far beyond the actual content, making exact travel to a specific target nearly impossible. One could try to give dynamic speed control to the user via hand input (e.g. velocity scales with the distance from the hand to the body) or even by taking walking speed into account. But since it is hard for users to get a good feeling for distance and speed in VEs this could easily overburden them.

Instead we propose to automatically calculate and set the gain every time the technique is triggered. The goal is to automatically allow the user to walk to every point in the VE at every time no matter where his starting position might be. To achieve this two variables have to be taken into account (see figure 3). The first is the distance from the user's position to the 'edges' of the VE. Secondly one has to account for the user's position inside the CAVE itself i.e. the distance the user can freely walk before hitting a projection screen. We now take the ratio of virtual and walkable distance for every direction as gain whenever the technique is triggered. If the user is standing very close to a wall, the corresponding direction is not considered because of the potentially unnaturally high gain owing to a very small trackable distance. Of course we also assumed that the user does not intend to walk into a wall.

In particular for rooms or objects that are slightly larger than the tracking space of the CAVE we think this approach is very promising because the gains tend to remain near the value one. For example, previewing the design of a new car or construction machine can easily be accomplished with this technique even though its dimensions are generally larger than the 3.6 x 2.4 meters of our CAVE. We will report on users' experience in the next section.

5.2 Other techniques

While we think the *Seven-league boots* are the most highly developed of our techniques, there are other ideas that warrant further investigation.

In our *scrolling technique* we used walking solely as the trigger for movement. Whenever the user steps near a projection wall (e.g. less than one meter distance – see figure 4) he starts to travel with constant speed in the direction of the wall. Much like scrolling with the

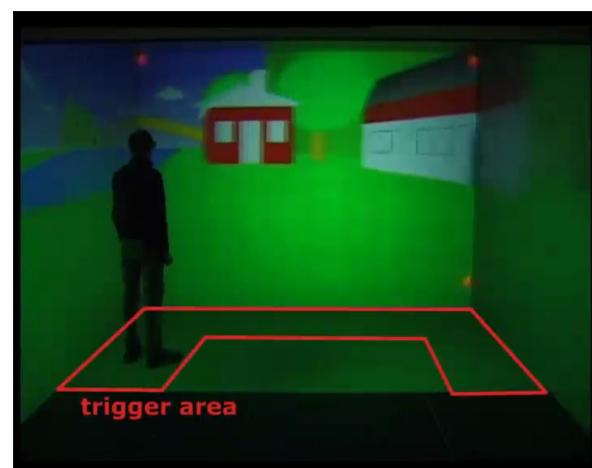


Figure 4: Area the user has to stand in to initiate travel in our *scrolling technique*

mouse on a desktop (often applied in computer games) the user can 'scroll' the VE by stepping to the appropriate side. Unfortunately this can become tedious very fast, especially if the speed of the travel technique is high. If you overshoot your target you have to move to the opposite side to reverse the direction. In case that happens more than once the user gets tired or irritated very quickly.

Hence we modified the above technique by using the direction of his gaze as the travel direction. This approach has two advantages. Areas of the tracking space that are otherwise rarely visited are assigned a practical use. Furthermore, the only input condition for this technique is the position of the user, leaving hands and possible input devices free for other actions. However, one should somehow visualize the area (eg. on the floor or as a virtual transparent curtain in space) that triggers the technique to avoid having it triggered by accident.

The *directed stepping technique* also uses movement to determine direction. However, it does not depend on continuous input. By taking a step in any direction while holding down a button the user can trigger travelling in that direction with a constant speed (see figure 5). Whenever he steps in another direction while the technique is active the direction of travel is changed accordingly. This allows for very fast, albeit potentially disconcerting course corrections. The user can instantaneously reverse direction by simply stepping in the opposite direction.

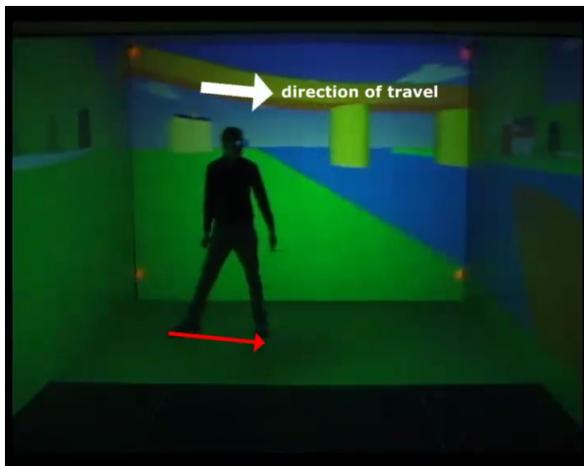


Figure 5: *directed stepping* – a step in a direction triggers movement in that direction

6 RESULTS

In order to analyze our results we again refer to Bowman et al. [4] who distinguish three main travel tasks:

- **Exploration:** Browsing the environment without a goal, obtaining information about the objects and locations within the world. This task is typical for (but

not limited to) the beginning of an interaction with an environment.

- **Search:** Travel to a specific target location.
- **Maneuvering:** Subtle positioning (e.g. of the viewpoint) in a local area with precise movements involved.

We here analyze our techniques in relation to these three task types, while also taking into account factors like size or structure of the VE.

Exploration of small to medium sized VEs was our main goal when designing *Seven-league boots*. The idea was to give a designer or mechanical engineer an easy to use tool to examine models of cars or machines in the CAVE that do not fit in the captured volume. To that end the technique proved to be uniquely effective. Generally it could be argued that this technique is well suited for all three task types as long as the VE is relatively small. Manoeuvring to exact locations is as easy as literally walking there (with or without pressed button, i.e. scaling gain). Since the technique is very similar to a pure walking interface (like the one Ruddle and Lessels [11] use) it stands to reason that searching should be equally as effective. Moreover we have not met problems with the scaling of the lateral movements to result in excessive swaying, discomfiting feeling in open space, and disastrous effects in case of closed spaces reported in [7]. Some questions (in parts related with the results in [7]) remain to be answered. One is how well the performance of *Seven-league boots* scales with the size of the VE. How high can the gain get before the technique becomes unusable? How does the structure of the VE (e.g. object clutter) factor into this?

Our *scrolling technique* on the other hand is more suited for travel in large VEs, especially if accuracy and relative travel is not a concern. We observed that most often the user walks to a trigger region in the direction he wishes to travel even though he does not have to since the determination of direction is based on gaze. Since when starting travel he is generally looking at his target this is still quite intuitive. However, we found that actions like correcting the course after overshooting do not come naturally to most users. Furthermore in our case, the user might have to stare at a projection wall he is standing directly in front of, resulting in a view of only a sea of pixels rather than a clear image.

Directed stepping is still work in progress and needs further investigation. Especially with this technique, the step length to change direction has to be chosen carefully. If it is too short, small (unintentional) movements of the body might render the technique unstable. If it is too long starting travel at all might be virtually impossible. Because this technique is much less dependent on available tracking space it seems especially suited for travel in larger VEs. Similar to pointing techniques it allows for relative movement meaning the user

can freely look around while travelling in any direction. However it still has to be determined how this technique performs for small course corrections during travel.

Finally we did not address the case of critical information being located in the direction of the missing back wall of the CAVE. We deliberately did not allow for rotation of the VE in any of our techniques since we wanted to keep them simple and not disorient the user.

In general we believe that navigation techniques should be customized to the requirements of the particular VE. We want to stress that walking should not be overlooked in this design process, especially when looking for a way to determine the direction of travel (be it directly or indirectly). There is also some potential for walking as an input condition. For now, we did not find a satisfying way to use walking for determining velocity and leave this open to further research.

7 CONCLUSION AND FUTURE WORK

We considered walking to be combined to new navigation techniques according to the taxonomy of Bowman et al. [3]. We believe our extension of the taxonomy can be a valid starting point for designing walking interfaces. Especially in a CAVE such techniques allow us to utilize the available tracking space as an additional input device. Of course the examples we gave for the different components of the taxonomy are far from complete. We still carry on with our research looking at different ways walking can be used for navigation.

Moreover, the navigation techniques we proposed are issue of further investigation. For the *Seven-league boots* we plan to experiment with ways for segmenting the VE into compartments to keep gains small while using this technique. This might be done beforehand or dynamically by the user. The *scrolling technique* might work better with an alternative method for determining direction. We also want to test different ways for rotating the VE (starting from the results of [10]) to make up for the missing back wall of the CAVE.

After an informal testing phase in order to optimize the techniques more generalized user tests for our different approaches have to be carried out to obtain more objective results about their performance. By using testbed evaluation as suggested by Bowman et al. [2] we hope to gain important knowledge of how well our navigation techniques work in relation to other options.

REFERENCES

[1] C. Anthes, P. Heinzlreiter, G. Kurka, and J. Volkert. Navigation models for a flexible, multi-mode vr navigation framework. In *VRCAI '04*, pages 476–479, 2004.

[2] D. A. Bowman, D. B. Johnson, and L. F. Hodges. Testbed evaluation of virtual environment interaction techniques. *Presence: Teleoper. Virtual Environ.*, 10(1):75–95, 2001.

[3] D. A. Bowman, D. Koller, and L. Hodges. Travel in immersive virtual environments: An evaluation of viewpoint motion control techniques. *VRAIS '97*, pages 45–52, 1997.

[4] D. A. Bowman, E. Kruijff, J.J LaViola, and I. Poupyrev. *3D User Interfaces Theory and Practice*. Addison-Wesley, 2005.

[5] R. P. Darken, W. R. Cockayne, and D. Carmein. The omnidirectional treadmill: a locomotion device for virtual worlds. In *UIST '97*, pages 213–221, 1997.

[6] D. Engel, C. Curio, L. Tcheang, B. Mohler, and H. H. Bülthoff. A psychophysically calibrated controller for navigating through large environments in a limited free-walking space. In *VRST '08*, pages 157–164, 2008.

[7] V. Interrante, B. Ries, and L. Anderson. Seven league boots: A new metaphor for augmented locomotion through moderately large scale immersive virtual environments. *3D User Interfaces*, 2007.

[8] H. Iwata, H. Yano, H. Fukushima, and H. Noma. Circulafloor. In *ACM SIGGRAPH 2004 Emerging technologies*, page 3, 2004.

[9] H. Jiung-yao, C. Wen-hsin, L. Yung-ting, B. Hua-hseng, T. Chi-fu, G. Chung-Yun, and L. Hwa-teng. The gait sensing disc - a compact locomotion device for the virtual environment. In *WSCG*, 2000.

[10] S. Razzaque, D. Swapp, M. Slater, M. C. Whitton, and A. Steed. Redirected walking in place. In *EGVE '02*, pages 123–130, 2002.

[11] R. A. Ruddle and S. Lessels. The benefits of using a walking interface to navigate virtual environments. *ACM Trans. Comput.-Hum. Interact.*, 16(1):1–18, 2009.

[12] M. Slater, M. Usoh, and A. Steed. Taking steps: the influence of a walking technique on presence in virtual reality. *ACM Trans. Comput.-Hum. Interact.*, 2(3):201–219, 1995.

[13] D. S. Tan, G. G. Robertson, and M. Czerwinski. Exploring 3d navigation: combining speed-coupled flying with orbiting. In *SIGCHI '01*, pages 418–425, 2001.

[14] M. Usoh, K. Arthur, M. C. Whitton, R. Bastos, A. Steed, M. Slater, and F. P. Brooks, Jr. Walking > walking-in-place > flying, in virtual environments. In *SIGGRAPH '99*, pages 359–364. ACM Press/Addison-Wesley Publishing Co., 1999.

[15] B. Williams, G. Narasimham, T. P. McNamara, T. H. Carr, J. J. Rieser, and B. Bodenheimer. Updating orientation in large virtual environments using scaled translational gain. In *APGV '06*, pages 21–28, 2006.

Ray Tracing on a GPU with CUDA – Comparative Study of Three Algorithms

Martin Zlatuška
Czech Technical University in Prague
Faculty of Electrical Engineering
Czech Republic
zlatum1{@}fel.cvut.cz

Vlastimil Havran
Czech Technical University in Prague
Faculty of Electrical Engineering
Czech Republic
havran{@}fel.cvut.cz

ABSTRACT

We present a comparative study of ray tracing algorithms implemented on a GPU for three published papers using different spatial data structures evaluated for performance on nine static scenes in walk-through animation. We compare the performance for uniform grids, bounding volume hierarchies (BVHs), and kd-trees evaluated on a GPU for ray casting and Whitted-style ray tracing. We show that performance of ray tracing with BVHs exceeds the performance of ray tracing with kd-trees for coherent rays. Contrary, the ray tracing with kd-trees is faster than that with BVHs for incoherent rays. The performance of ray tracing with uniform grids is slower than both ray tracing with BVHs and kd-trees except for uniformly populated scenes. We show that the performance is highly sensitive to details of implementation on kd-trees.

Keywords: GPU programming, CUDA, performance study, ray tracing, uniform grids, kd-trees, bounding volume hierarchies.

1 INTRODUCTION

While modern graphics cards (GPUs) allow for general computation in a parallel manner, one of the most prominent applications for a GPU is image synthesis. This is thanks to the inherent parallel nature of ray tracing and other global illumination algorithms – the decomposition of images into pixels provides a natural way of creating individual tasks for many parallel processors. Unlike the GPUs a few years ago, modern ones allow us full programmability similar to general CPUs, while the streaming computation model has its own specific issues. This has to be taken into account when adopting the data structures and traversal algorithms for ray tracing on a GPU architecture.

In this paper we compare three formerly published papers that implement ray tracing with spatial data structures on a GPU. These are uniform grids [Pur02], kd-trees [Hor07], and bounding volume hierarchies [Gün07]. While the algorithms were successfully mapped to a GPU, their performance have not been carefully compared on a current programmable GPU architecture as a common implementation framework was not available. In this paper we first present such a comparison study dealing with efficiency of three different data structures for ray tracing on a GPU. We restrict ourselves to a static setting irrespective of the

construction time as the data structures are built offline on a CPU for our tests. We show on a kd-tree that even small changes to the implementation of traversal code can lead to the significant change of performance.

This paper is further structured as follows. Section 2 summarizes the previous work of ray tracing on a GPU and performance comparison of data structures for ray tracing. Section 3 describes our choices for implementation. Section 4 shows the results from measurements on two GPUs for a set of scenes. Further it discusses the bottlenecks of a contemporary GPU architecture for ray tracing algorithms. Section 5 concludes the paper with possible perspectives for future work.

2 PREVIOUS WORK

In this section we review chronologically the most significant papers that address mapping of spatial data structures for ray tracing to a GPU. We discuss briefly all three data structures of our interest: uniform grids, kd-trees, and BVHs, while we avoid the discussion of results on other computer architectures except for a GPU as such surveys for CPU implementation have been provided for example in [Wal07].

Uniform grids. The first ray tracing algorithm mapped fully on a GPU has been published by Purcell et al. [Pur02] and uses a uniform grid. Their implementation mapped the computation by means of shaders while their data resided in a texture. In a concurrent work Carr et al. [Car02] present the architecture of a software ray tracer on a GPU with a focus on ray-triangle intersection with predefined BVH hierarchy. The mapping of both mentioned approaches had been influenced by architectural limitations. Recently, Kalojanov and Slusallek [Kal09] presented

the algorithm for parallel construction of uniform grids on a GPU.

Kd-trees. A stack on a GPU with a low level of programmability was studied by Ernst et al. [Ern04] and used for stack-based kd-tree traversal algorithm. Foley and Sugerman [Fol05] presented two algorithms for kd-tree traversal without a stack. Their first algorithm called *kd-restart* is in fact the algorithm published by Kaplan [Kap85]. The second stack-less algorithm called *kd-backtrack* requires the storage of the bounding box and link nodes to its parent for every node of a tree, which significantly increases the memory footprint and hence it decreases performance. Both presented algorithms increase the number of nodes traversed compared to stack-based traversal algorithms. Another paper by Horn et al. [Hor07] addresses the lack of local memory to implement the stack much more efficiently. They propose the use of a push-down and short stack which can avoid most of the restarts of a traversal from the root node. This is possible as ray tracing with the kd-tree traverses only a few leaves on average. In concurrent work Popov et al. [Pop07] suggest to use the augmentation of a data structure by neighbor links among the nodes of a kd-tree. They even exceed the performance of CPU-based ray tracers while they achieve comparable performance as in [Hor07]. Further, Zhou et al. [Zho08] proposed the algorithm for kd-tree construction on a GPU. This method yields the performance of kd-tree construction comparable to CPU-based algorithms for kd-tree construction [She07]. This can be used for dynamic scenes up to 200,000 triangles to yield interactive performance.

Bounding Volume Hierarchies. Bounding volume hierarchies (BVHs) were also successfully implemented on a GPU. Thrane and Simonsen [Thr05] in fact compare kd-trees, uniform grids, and bounding volume hierarchies implemented on a GPU (hardware of year 2005). They conclude the performance of BVHs is low, however higher than the performance of other two data structures when no ray packets are used. Carr et al. [Car06] implemented a variant of BVHs in combination with geometry images. Günther et al. [Gün07] use ray packets and yield interactive performance comparable or exceeding CPU-based implementation, but only for primary and shadow rays. Recently, Lauterbach et al. [Lau09] present an algorithm for fast BVH construction on a GPU, where they report performance comparable to kd-trees [Zho08] only for one scene. Recently, Torres et al. [Tor09] published an algorithm for stack-less BVH traversal, where the use of stack is replaced by ropes connecting the nodes of a BVH in a sibling order. Very recently, Aila and Laine [Ail09] analyze the efficiency of various CUDA kernels for ray tracing with BVH (This paper is not included in our study as our research was completed in January 2009 in [Zla09]).

Comparison. For algorithms on a CPU it is believed that the hierarchical spatial data structures (both kd-trees and BVHs) built up in a top-down fashion yield similar performance. A decade old study by Havran et al. [Hav00] provides thorough performance comparison of twelve data structures implemented on a CPU. More recently Havran [Hav07] discusses the similarities and differences of top-down constructed spatial hierarchies (kd-trees and BVHs) and uniform grids. He argues that while kd-trees and BVHs have very similar properties as they can be mutually emulated in a constant time and space, uniform grids can outperform hierarchical data structures only for uniform distribution of objects in the scene.

To our best knowledge a proper recent experimental comparison of different ray tracers on a modern programmable GPU (year 2008 and 2009) has not been available. We would like fill the gap by our paper for a current GPU architecture (CUDA) of NVidia for a static scene setting (walk-through).

3 ALGORITHM IMPLEMENTATION

We have implemented a standalone compact program that does not need the support by other 3rd party libraries. The program implements a parser for scene format *PLY*, format *BART* [Lex01], and subset of Open Inventor format. While the data structures are built offline on a CPU, the created data structures are transferred to a GPU and used for ray tracing algorithm entirely on the GPU. To study the efficiency of shooting rays using different data structures this methodology is sufficient. The traversal algorithms and shading on the GPU were implemented using NVidia CUDA [PRG08].

The geometry of a scene consisting solely of triangles is represented by a list L_v of vertices and list of materials L_m , where each triangle has a list of three indices to L_v plus an index to the L_m . We tested also the variant where each triangle is represented directly by three vertices, however the memory consumption was increased with the negative impact to the performance. Shading is implemented via simple Phong model and is included in timing. The program can run in two modes - for measurement purposes and with GUI. Since we released the source code to public, we do not discuss many tiny but often relevant implementation details in this paper. Our paper serves as the summary of the Master Thesis [Zla09], where many details are stated, decision choices for that particular solution are discussed, and several unsuccessful attempts to improve the efficiency of algorithm implementations are described.

Below we describe the selected details of our implementation for uniform grids, kd-trees, and bounding-volume hierarchies.

3.1 Uniform Grids

The implementation of uniform grids loosely follows the paper [Pur02], with the traversal algorithm described in [Ama87]. The implementation is easier as CUDA is used instead of shaders. To decrease the number of registers we used a constant cache to store the values that do not change such as the direction and origin of the ray and precomputed values for 3D DDA traversal. This allows us to save five registers and get better occupancy [PRG08]. The threads have to be synchronized to compute the intersection with the triangles in the cells. The threads for the rays that do not intersect any cell with triangles are idle.

We tried to optimize the traversal algorithm by sharing the load of rays with many ray-triangle intersections with rays that do not need to compute many ray-triangle intersections. This required the rescheduling of the computation during the visit to the cell. However the resulting algorithm was several times slower than a simple algorithm, where some threads become inactive either when the computation is finished or a ray intersects an empty cell. Further, we also tried to implement packet tracing [Wal06] on a GPU. Although the pilot implementation has a uniform access to the memory and common branching, it resulted in an increased number of cells that were traversed. As a result, for packet of size 8×8 and for packets of size 4×4 the performance was substantially decreased compared to the simple implementation.

3.2 Kd-Trees

Kd-trees were built with surface area heuristics according to the sampling approach described in [She07]. Internally, each node of a kd-tree is represented by 8 Bytes, using the compact representation described in [Wal01].

We have been experimenting with several traversal algorithms and finally we decided to use a short stack traversal [Hor07] with four entries to compromise between number of traversal steps and occupancy. The stack is stored in shared memory. We aim at minimizing the conflicts in the shared memory as the threads for the rays are computed rather independently. We show the performance of two versions of kd-tree traversal code which illustrates the performance of very similar solutions. Initially, we stored three values to the short stack - “mint, maxt, and node address”. However, we can decrease the size of stack entry to only two values, as for the farther node traversed the *mint* is equal to *maxt* for the node we just traversed. This changes the occupancy and performance as we show in Section 4. The traversal algorithm referred to as *kdt-3* stores three values to the stack, while the algorithm *kdt-2* stores only two values to the stack.

3.3 Bounding Volume Hierarchies

The BVHs were built in top-down fashion with surface area heuristics using the centroids of bounding boxes for scene triangles, following the paper by Günther et al. [Gün07]. As a BVH does not need to store the *mint* and *maxt* values along the ray, only the node address is saved to the stack. For packet traversal, the stack can be shared by all the threads in a packet, which increases the utilization of the resources. The stack does not need to be shortened to only several entries, which minimizes the number of traversal steps. The stack is similarly to kd-trees stored in shared memory.

The order of traversal among several threads is resolved by a concurrent write to the shared memory, where four memory entries are first initialized to zero. Each thread then writes the preference to one of four entries, value one for one of the four cases: traverse left, traverse right, traverse both, traverse none. The serialization of write operation may occur as threads record their information.

When rays diverge, the traversal continues to the node where most of the rays need to traverse. This is implemented by parallel reduction using auxiliary memory with one entry for each thread. Each thread writes either -1 when a left child should be visited as first, 0 for no preference, and 1 for the right child. The decision which first node should be traversed is then resolved by parallel reduction – the most node wanting to be traversed by most of the rays is visited as first while the other node is stored to the stack. When a thread does not need to visit any node, the node stores simply 0 as a preference. This is different from the algorithm described in [Gün07] and this change increases the performance for secondary rays by up to 20%. The disadvantage of BVH compared to kd-tree is the increased memory space required by the BVH node representation, it is 32 Bytes, which is 4 times higher than for a kd-tree node. However, it is compensated as the number of nodes and object references is strictly limited by the number of objects, so the storage of the whole BVH is typically smaller than the one for a kd-tree.

4 RESULTS

In this section we describe the results for measurement on nine scenes. To provide more variability to testing, we used three scenes of individual objects courtesy of Stanford scene repository, three scenes from BART [Lex01] (camera animated, objects not animated), and three other general interior architectural scenes. The rendered images of all scenes are shown in Figures 3, 4, and 5. These scenes are frequently used to test the performance of ray tracing and global illumination algorithm, the BART scenes [Lex01] scenes were designed for benchmarking of ray tracing.

To decrease the view dependence of results, we created a static walk-through animation for each scene of

length 400 frames. All the performance results in this paper were measured on a GPU NVidia GeForce GTX 280 (June 2008), which has compute capability 1.3, 240 multi-threaded processor cores on 600 MHz, and 1 GByte of memory with a bandwidth of 141.7GB/sec. We also measured the results on an older, low-level GPU, an NVidia GeForce 8600GT (April 2007), where we got between 1/10 and 1/6 of the performance for the NVidia GeForce GTX 280.

The static properties of data structures for all nine scenes are shown in Table 1. The average computation time for the animation for a frame is shown in Table 2 for three settings: (1) shooting only primary rays, (2) primary and shadow rays, and (3) Whitted-style recursive ray tracing with two bounces for secondary rays. The occupancy for three scenes is shown in Table 3 for different settings of compilation in dependence on the number of registers where the maximum rendering times are reported. The results demonstrate that both the setting and the use of either three or two values stored to the traversal stack for a kd-tree have remarkable impact on performance. The dependence on the resolution is shown in Figures 1 and 2 for scene *Dragon* and *Robots*. More detailed results and evaluation can be found in [Zla09].

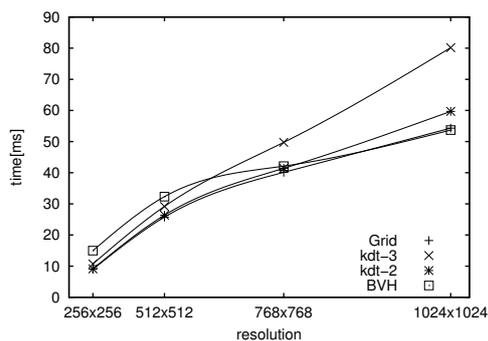


Figure 1: The dependence of computation time[ms] on resolution for scene *Dragon* for different resolutions.

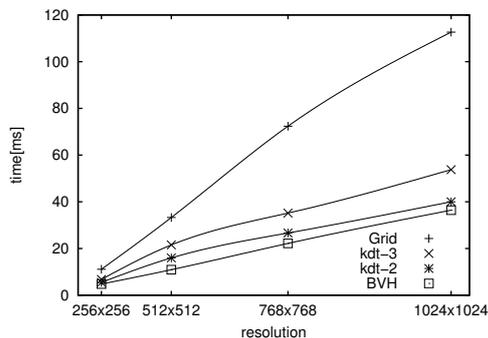


Figure 2: The dependence of computation time[ms] on resolution for scene *Robots* for different resolutions.

Discussion

While an interested reader can draw his/her own conclusion from the numbers in tables, let us provide our

interpretation of the measured data. As we tested the performance on two different architectures (G80 and GT200), we can report: the progress of hardware was the most beneficial for the performance increase of a ray tracing with uniform grids. Similarly to the implementation on a CPU, the performance of uniform grids is superior only for uniformly populated scenes (*Bunny*, *Dragon*, and *Buddha*).

The (packet) ray tracing with BVH of incoherent rays is memory bound but is relatively well masked by switching threads. However, BVH has higher performance for (coherent) primary and shadow rays. This is in concordance with the results of Günther et al. [Gün07]. For traversing individual diverging (incoherent) rays such as secondary reflected rays in path tracing, the performance of BVH significantly deteriorates.

For diverging rays the kd-tree with its own short stack for each ray (thread) is a more efficient solution. The small size of each kd-tree node decreases the data traffic between memory and the processor cores. The bottleneck for the kd-tree traversal is a lack of larger local and fast memory for the stack implementation. The increase of local memory should lead to higher performance for upcoming GPU architecture(s).

As the performance of GPU ray tracing is dependent on many details in the implementation, this paper is accompanied by the source code available at: <http://dcgi.felk.cvut.cz/members/havran/rtgpu2009/>. We hope that the released source code can be further utilized in rendering applications and performance studies in future.

5 CONCLUSION AND FUTURE WORK

In this paper that serves as a summary of [Zla09] we have described a performance study comparing ray tracing implemented with CUDA on modern GPU from NVidia. We optimized the implementation for three data structures and traversal algorithms for ray tracing and compared the performance obtained from measurements for nine scenes for shooting primary rays, ray casting with shadow rays, and recursive Whitted-style ray tracing. The performance differed for coherent rays, where the bounding volume hierarchy is the winner, and for incoherent rays, where kd-trees seem to be more efficient on average when implemented using the short-stack as suggested by Horn et al. [Hor07]. However, the performance of ray tracing algorithm on a GPU is sensitive to many implementation details, likely due to the relatively small local cache on GPU architectures.

As future work, the implementation could be extended by several other data structures that can be efficiently mapped to a GPU architecture. The measure-

scene	# triangles	# lights	grid					kd-tree					BVH			
			Grid size	#refs	size [MB]	#trav. steps	#int. tests	#leaves [$\times 10^3$]	#refs	size [MB]	#trav. steps	#int. tests	#leaves [$\times 10^3$]	size [MB]	#trav. steps	#int. tests
			Bunny	69451	1	83×82×64	3.7	8.3	47.4	37.2	154	5.5	9.2	54.0	12.3	23.0
Dragon	871414	1	273×193×122	3.1	103.1	117.3	45.5	978	2.3	58.5	68.8	10.4	295.0	35.8	114.1	28.0
Buddha	1087716	1	128×312×129	2.8	102.1	100.8	43.1	1265	2.5	76.5	64.3	8.8	389.0	44.7	130.9	30.9
Robots	71708	1	128×209×268	19.2	79.9	40.4	66.5	82	6.6	12.7	30.5	8.8	25.0	6.1	30.6	3.5
Museum	14380	2	71×43×106	9.5	5.1	60.2	33.6	26	4.5	2.0	19.2	6.7	4.6	0.9	40.1	3.9
Kitchen	110559	4	254×128×256	14.1	89.3	154.8	12.7	164	3.9	11.2	6.3	1.0	36.4	4.9	40.0	5.2
Theatre	53832	2	172×135×60	23.0	31.4	56.6	37.9	124	8.6	10.9	17.4	5.4	17.7	3.3	33.1	3.7
Office	36310	3	93×55×93	6.5	7.9	73.9	71.6	55	6.4	5.1	11.2	4.4	11.1	11.5	30.5	4.9
Conference R.	298866	2	387×246×93	10.3	121.3	165.7	31.5	338	8.7	51.6	14.3	4.8	97.9	14.5	34.6	5.9

Table 1: The properties of the test scenes and the spatial data structures built up for them. The general properties include number of triangles and light sources. For each data structure we report the number of leaves/cells. #refs corresponds the average number of references to objects in leaves. The storage for the data structure is given in MBytes. The number of intersection tests and traversal steps are reported for primary and secondary rays, the other results are in [Zla09].

	primary rays				primary and shadow rays				primary, shadow, secondary rays			
	time[ms]				time[ms]				time[ms]			
	grid	kdt-3	kdt-2	BVH	grid	kdt-3	kdt-2	BVH	grid	kdt-3	kdt-2	BVH
Bunny	16.0	41.6	31.5	13.8	27.4	61.7	52.1	27.0	—	—	—	—
	116%	301%	228%	100%	53%	118%	100%	52%	—	—	—	—
Dragon	40.2	55.9	42.3	39.0	73.3	86.0	73.4	80.0	—	—	—	—
	103%	143%	108%	100%	100%	117%	100%	109%	—	—	—	—
Buddha	34.6	45.6	34.2	36.8	69.1	73.5	62.9	81.8	—	—	—	—
	94%	124%	93%	100%	110%	117%	100%	130%	—	—	—	—
Robots	27.3	20.5	16.2	25.9	53.8	35.6	30.1	50.0	89.0	43.8	38.7	64.3
	105%	79%	63%	100%	179%	118%	100%	166%	230%	113%	100%	166%
Museum	25.0	46.2	35.7	20.0	68.3	86.2	73.4	53.2	168.5	184.1	162.4	163.7
	125%	231%	179%	100%	93%	117%	100%	72%	104%	113%	100%	101%
Kitchen	41.6	40.5	31.9	29.3	209.6	130.0	110.8	138.8	442.8	244.4	214.3	403.9
	142%	138%	109%	100%	189%	117%	100%	125%	207%	114%	100%	188%
Theatre	43.1	42.3	33.1	34.3	119.7	87.3	74.3	93.6	379.7	201.6	177.5	292.1
	126%	123%	97%	100%	161%	117%	100%	126%	214%	114%	100%	165%
Office	52.9	44.1	34.2	22.7	218.6	116.1	101.5	87.9	224.0	120.1	107.7	94.2
	233%	194%	151%	100%	215%	114%	100%	87%	208%	112%	100%	87%
Conference Room	83.2	83.7	66.2	28.9	228.2	153.0	132.7	85.2	292.8	—	—	114.1
	288%	290%	229%	100%	172%	115%	100%	64%	(257%)	—	—	(100%)
Average[%]	148%	181%	140%	100%	141%	117%	100%	104%	193%	113%	100%	142%

Table 2: Average computation time for a frame [ms] for three settings rendered in resolution 1024×1024 for rendering 400 frames animations: (1) primary rays only (2) primary and shadow rays (ray casting) (3) primary, shadow, and secondary rays for recursion depth two (one primary ray per pixel). For individual objects (Bunny, Dragon, and Buddha) the setting (3) is meaningless. There was not enough memory for scene Conference Room to compute the recursive ray tracing with kd-trees. Timing includes also shading by Phong model. kdt-3/kdt-2 stands for storing 3 or 2 values to the stack during traversal.

ments and observations can provide interesting feedback to architects of graphics hardware in future.

ACKNOWLEDGMENTS

This work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research program MSM 6840770014 and LC-06008 (Center for Computer Graphics) and the Aktion Kontakt OE/CZ grant no. 2009/6.

REFERENCES

- [Ail09] Aila, T., and Laine, S.. Understanding the Efficiency of Ray Traversal on GPUs. In *Proceedings of High-Performance Graphics 2009*, pages 145–150, New York, NY, USA, 2009. ACM.
- [Ama87] Amanatides, J., and Woo, A. A fast voxel traversal algorithm for ray tracing. In G. Marechal, editor, *Eurographics '87*, pages 3–10. North-Holland, August 1987.

	# reg.	occupancy [%]	Robots		Kitchen		Museum	
			time[ms]	speedup [%]	time[ms]	speedup [%]	time[ms]	speedup [%]
grid	59	25	430.7		907.7		266.2	
	32	50	350.2	19	704.8	23	220.5	18
kdt-3	56	25	110.7		429.5		240.5	
	32	25	129.0	-18	496.2	-15	278.2	-15
kdt-2	56	25	110.7		429.5		240.5	
	40	37.5	96.0	13	372.2	13	211.4	12
BVH	53	25	151.0		1064.8		274.7	
	32	50	129.9	25	762.4	29	214.4	22

Table 3: GPU occupancy and timing for NVidia GeForce GTX 280 for three BART scenes for ray tracing with primary, secondary, and shadow rays in resolution 1024×1024 .

- [Car02] Carr, N.A., Hall, J.D., and Hart, J.C. The ray engine. In *HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 37–46, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.
- [Car06] Carr, N.A., Hoberock, J., Crane, K., and Hart, J.C. Fast GPU ray tracing of dynamic meshes using geometry images. In *GI '06: Proceedings of Graphics Interface 2006*, pages 203–209, Toronto, Ont., Canada, Canada, 2006. Canadian Information Processing Society.
- [Ern04] Ernst, M., Vogelsgang, C., and Greiner, G. Stack implementation on programmable graphics hardware. In *Vision Modeling and Visualization 2004*, pages 255–262, 2004.
- [Fol05] Foley, T., and Sugerman, J. KD-tree acceleration structures for a GPU raytracer. In *HWWS '05: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 15–22, New York, NY, USA, 2005. ACM.
- [Gün07] Günther, J., Popov, S., Seidel, H.-P., and Slusallek, P. Realtime Ray Tracing on GPU with BVH-based Packet Traversal. In *Proceedings of the IEEE/Eurographics Symposium on Interactive Ray Tracing 2007*, pages 113–118, September 2007.
- [Hav00] Havran, V., Pšikryl, J., and Purgathofer, W. Statistical Comparison of Ray-Shooting Efficiency Schemes. Technical Report TR-186-2-00-14, Institute of Computer Graphics, Vienna University of Technology, Favoritenstrasse 9/186, A-1040 Vienna, Austria, May 2000.
- [Hav07] Havran, V. About the Relation between Spatial Subdivisions and Object Hierarchies Used in Ray Tracing. In *23rd Spring Conference on Computer Graphics (SCCG 2007)*, pages 55–60, Budmerice, Slovakia, May 2007.
- [Hor07] Horn, D.R., Sugerman, J., Houston, M., and Hanrahan, P. Interactive k-D Tree GPU Raytracing. In *13D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 167–174, New York, NY, USA, 2007. ACM.
- [Kal09] Kalojanov, J. and Slusallek, P. A parallel algorithm for construction of uniform grids. In *HPG '09: Proceedings of the Conference on High Performance Graphics 2009*, pages 23–28, New York, NY, USA, 2009. ACM.
- [Kap85] Kaplan, M.R. The uses of spatial coherence in ray tracing. In *ACM SIGGRAPH '85 Course Notes 11*, July 1985.
- [Lau09] Lauterbach, C., Garland, M., Sengupta, S., Luebke, D., and Manocha, D. Fast BVH Construction on GPUs. *Computer Graphics Forum*, 28(2):375–384, April 2009. (Proceedings of Eurographics 2007).
- [Lex01] Lext, J., Assarsson, U., and Möller, T. A Benchmark for Animated Ray Tracing. *IEEE Comput. Graph. Appl.*, 21(2):22–31, 2001.
- [Pop07] Popov, S., Günther, J., Seidel, H.-P., and Slusallek, P. Stackless KD-Tree Traversal for High Performance GPU Ray Tracing. *Computer Graphics Forum*, 26(3):415–424, September 2007. (Proceedings of Eurographics).
- [PRG08] NVIDIA CUDA Compute Unified Device Architecture - Programming Guide, 2008. Version 2.1.
- [Pur02] Purcell, T.J., Buck, I., Mark, W.-R., and Hanrahan, P. Ray tracing on programmable graphics hardware. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 703–712, New York, NY, USA, 2002. ACM.
- [She07] M. Shevtsov, A. Soupikov, and A. Kapustin. Highly parallel fast kd-tree construction for interactive ray tracing of dynamic scenes. *Computer Graphics Forum*, 26(3):395–404, September 2007. (Proceedings of Eurographics).
- [Thr05] Thrane, N., and Simonsen, L.O. A comparison of acceleration structures for GPU assisted ray trac-



Figure 3: Stanford scenes: Bunny, Buddha, Dragon.



Figure 4: BART scenes: Robots, Museum, Kitchen.



Figure 5: MGF scenes: Theatre, Office, Conference Room.

- ing. M.Sc. Thesis, University of Aarhus, Denmark, 2005.
- [Tor09] Torres, R., Martin, P.J., and Gavilanes, A. Ray Casting using a Roped BVH with CUDA. In *25th Spring Conference on Computer Graphics (SCCG 2009)*, pages 107–114, Budmerice, Slovakia, April 2009.
- [Wal01] Wald, I., Slusallek, P., Benthin, C., and Wagner, M. Interactive Rendering with Coherent Ray Tracing. *Computer Graphics Forum*, 20(3):153–164, 2001. (Proceedings of Eurographics).
- [Wal06] Wald, I., Ize, T., Kensler, A., Knoll, A., and Parker, S.G. Ray Tracing Animated Scenes using Coherent Grid Traversal. *ACM Transactions on Graphics*, pages 485–493, 2006. (Proceedings of ACM SIGGRAPH 2006).
- [Wal07] Wald, I., Mark, W.R., Günther, J., Boulos, S., and Ize, T. Warren Hunt, Steven G Parker, and Peter Shirley. State of the Art in Ray Tracing Animated Scenes. In *Eurographics 2007 State of the Art Reports*, 2007.
- [Zho08] Zhou, K., Hou, Q., Wang, R., and Guo, B. Real-time KD-tree construction on graphics hardware. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, pages 1–11, New York, NY, USA, 2008. ACM.
- [Zla09] Zlatuška, M. Ray Tracing Algorithms on Modern GPUs. M.Sc. Thesis, Czech Technical University in Prague, Jan 2009. <http://dcgi.felk.cvut.cz/members/havran/rtgpu2009/>.

Difference-Contribution Strategy for Seeding 2D Streamlines

Shaorong Wang, Rui You, Yisong Chen, Sheng Li, Guoping Wang

The Key Lab of Machine perception and intelligent, MOE, Beijing, China, 100871

School of Electronics Engineering and Computer Science, Peking University, Beijing, China, 100871

{wangsr@graphics.pku.edu.cn, yourui@graphics.pku.edu.cn, chenys@graphics.pku.edu.cn, lisheng@graphics.pku.edu.cn, gwang@graphics.pku.edu.cn}

ABSTRACT

This paper presents a novel seeding strategy for streamline visualization of 2D vector field. The main idea of our approach is to capture the spatial-varying features in a vector field. Generally speaking, we measure the difference between the inflow and the outflow to evaluate the local spatial-varying feature at a specified field point. A Difference-Contribution Matrix (DCM) is then calculated to describe the global appearance of the field. We draw streamlines by choosing the local extreme points in DCM as seeds. DCM is physics-related thus reflects intrinsic characteristics of the vector field. The strategy performs well in revealing features of the vector field even with relatively few streamlines.

Keywords

Seeding strategy, Streamline, Difference-Contribution Matrix

1. INTRODUCTION

Vector fields are commonly used in many scientific and engineering domains, such as astronomy, aeronautics, and meteorology. Visualization of vector fields is important for properties analysis. The most common approaches include geometry-based, texture-based, feature-based, and streamline-based approaches.

Geometry-based approaches, such as arrow and hedgehog plots, give a visual perception of local flow feature.

Texture-based methods give a dense representation of the vector field. However, they can't provide visual focuses on significant information of vector field and obtain visually pleasing images requires an intrinsically huge computational expense.

Feature-based visualization approaches seek to compute a more abstract representation that already contains the important properties in a condensed form and suppresses superfluous information. Anyway, the feature is always not easy to be extracted.

The most popular flow visualization method in use today is still streamlines and those derived from streamlines because they provide sparse visualization that focus on significant structures and can be

combined with other visualization techniques. Furthermore, they are faster to compute and can be rendered at any resolution at interactive rates.

The quality of visualization of the streamlines highly relies on the seeding strategy, which includes seed location and a length of each streamline. In other words, it's very important to select a set of streamlines to represent the vector fields comprehensibly and completely. On the one hand, placing too many streamlines can make the final images cluttered, and hence make it more difficult to understand the data. On the other hand, we may miss important flow features if too few streamlines placed. An ideal streamline seed placement algorithm should be able to generate visually pleasing and technically illustrative images.

There are several seeding strategies developed in the past years, such as evenly-spaced streamlines algorithm [Liu06], and feature-guided algorithm. A criteria of seeding strategy is proposed by Verma et al. [Ver00]. Coverage, no important features of the vector field should be missed and the streamlines should cover the whole domain; Uniformity, the distribution of streamlines should be more or less uniform across the domain; Continuity, long continuous streamlines are preferred over short ones.

In this paper, we define a Difference-Contribution Matrix (DCM) as a metric for flow features. We propose a novel 2D streamline seeding strategy according to the DCM. Suppose a region including inflow and outflow shown in Figure 1, there is cross interface between the flow and the region. If the area of inflow interface is not the same as that of outflow interface, changes happen in the region. The greater

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

the difference between the inflow and the outflow, the greater the vector fields change.

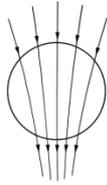


Figure 1. Inflow and outflow

Compared to the past approaches, the strategy proposed in this paper give higher priority to the variation of the streamline than to the density of the streamlines. This is because the former represents more flow feature. In other words, if there is little variation in a region, the streamline is nearly evenly distributed in the region and they can be represented by fewer streamlines. If the variation is great in a region, more streamlines are needed to provide the detail.

The seeding strategy in this paper is based on the DCM. Streamline starting points are seeded depending on the maxima of the matrix. Because DCM is defined by the physical meaning of the vector fields, our seeding strategy is able to qualitatively capture more important flow features with less streamlines, hence less clutter and occlusion.

The advection of streamlines in the previous streamline placement algorithms can be terminated by explicit inter-streamline distance control. This may cause visual discontinuity of the flow pattern, especially when it is near the vicinity of critical points. Our seeding algorithm only determines complete streamlines which are integrated as long as possible until they leave the domain, reach a critical point, or generate a loop. Without abruptly stopping the streamlines, the flow patterns shown in the visualization are much easier to understand.

2. RELATED WORK

Overview of vector field visualization techniques can be found in [Lar04] and [Pos03]. We consider here the most relevant work in streamline visualization. A number of techniques with different objectives have been developed. We group the present seeding strategies into four categories: image based, direct, feature based and vector field property-based.

Image-based method searches for an energy function's minimal value to place seeds, in which the energy function is defined in image space according to streamlines. In [Tur96], techniques for automated placing of seed points were developed to achieve a nearly uniform, dense distribution of streamlines for 2D flow fields. Mao et al. [Mao98] extend this approach to 3D curved surfaces. For 3D flow fields, seeding strategies typically involve analysis of the

underlying flow field to visualize certain features using sparse distributions.

Direct methods place new streamlines with a certain heuristic rule without computing any global energy function. A seeding strategy for automated placing of seed points was developed to achieve a nearly uniform, dense distribution of streamlines for 2D flow field [Job97]. The technique is extended to unsteady flows in [Job00], and multi resolution flow visualization in [Job01]. By defining a 3D Euclidean distance metric, the strategy is directly extended to 3D field [Mat03]. The seeding strategy presented by Mebarki et al. [Meb05] starts new streamlines in the center of the biggest remaining voids, and achieve good continuity and uniformity of the streamlines by a greedy algorithm. Liu et al. [Liu06] improves continuity by prioritizing streamline elongation over new streamline insertion.

Feature-based flow visualization is concerned with the extraction of specific patterns of interest, or features. Verma et al. [Ver00] first proposed a feature-based strategy for 2D vector field visualization. The seeding strategy is extended to 3D vector fields by Ye et al. [Ye05].

Streamline similarity and streamlines density are both properties of vector field. They can be regarded as the criteria of adding new streamlines. Li et al. [Li07] proposed a 3D image-space streamline placement method. They control the seeding and generation of streamlines in image space to avoid visual cluttering. Schlemmer et al. [Sch07] defined the streamline density of a region as the ratio between the number of occupied pixels by streamlines and the total number of pixels in the region.

3. DISTRIBUTION-BASED SEEDING STRATEGY

3.1. In-out Contribution Matrix

We first give some definitions about our idea. For a non-zero vector at any position in a vector field, there is a streamline passing through the position. A streamline is a Complete Streamline if either of the following conditions is satisfied:

The ending point overlaps the starting point. In other words, the streamline is a closed curve.

The endpoint is on the border of the vector field, or the vector at the endpoint is zero.

First a set of Complete Streamlines are generated to cover the vector field domain, which is called as the Complete Streamline Set. The Complete Streamline Set can be generated uniformly or randomly. The former method is chosen in this paper: The vector field domain is evenly divided into $m \times n$ squares, and then streamlines are seeded at each square's center. If all the streamlines are regarded to be

different, we get a Complete Streamline Set with $m \times n$ Complete Streamlines.

For a given point p in the vector field, c_p is a circle of radius r centered at p . We partition the circle c_p into congruent curve segment units uniformly. Each unit u_i has an outward-weight $w_{out}(u_i)$ and an inward-weight $w_{in}(u_i)$, both of which are initialized with 0 and $0 < w_{in}, w_{out} < 1$. Given a Complete Streamline Set S_{line} , subset S_{sub} contains all streamlines in S_{line} which have intersection with c_p . For each streamline l in S_{sub} , cp is the intersection point of l and c_p , $N(p)$ is the number of all intersection points of subset S_{sub} and c_p . Let \mathbf{V} be the vector at the intersection point cp , if \mathbf{V} is outward to the circle c_p , cp is called as an outward intersection point, otherwise it is an inward intersection point. For every inward intersection point cp_i , we calculate its inward contribution $Con_{in}(cp_i, u_j)$ to every unit u_j :

$$Con_{in}(cp_i, u_j) = F(Dis(cp_i, u_j))$$

$Con_{in}(cp_i, u_j) = F(Dis(cp_i, u_j))$ Where $Dis(cp_i, u_j)$ is the distance between cp_i and u_j , and $F()$ is a decreasing function.

The weight of every unit u_j is updated by every inward intersection point cp_i :

$$w_{in}(u_j) = w_{in}(u_j) + Con_{in}(cp_i, u_j)$$

$$w_{in}(u_j) = 1, \text{ if } (w_{in}(u_j) > 1)$$

The inward-contribution of point is defined as

$$Con_{in}(p) = \sum_j w(u_j)$$

And the outward-contribution is calculated the same as that of inward-contribution.

Support points have been sampled uniformly in the vector field, for each sampling point $p(i, j)$ we calculate its inward and outward contribution $Con_{in/out}(p)$. Then the Density Matrix $\mathbf{Mat}_{density}$, Out-Contribution Matrix \mathbf{Mat}_{out} , In-Contribution Matrix \mathbf{Mat}_{in} , Signed-Difference-Contribution Matrix \mathbf{Mat}_{sdelta} and Difference-Contribution Matrix(DCM) \mathbf{Mat}_{del} can be defined as:

$$\mathbf{Mat}_{density} = (N(p(i, j)))$$

$$\mathbf{Mat}_{out} = (Con_{out}(p(i, j)))$$

$$\mathbf{Mat}_{in} = (Con_{in}(p(i, j)))$$

$$\mathbf{Mat}_{sdelta} = \mathbf{Mat}_{in} - \mathbf{Mat}_{out}$$

$$\mathbf{Mat}_{del} = abs(\mathbf{Mat}_{sdelta})$$

The following statements of DCM are obvious:

1. For any element a in \mathbf{Mat}_{in} , \mathbf{Mat}_{out} , $a > 0$
2. If $\mathbf{Mat}_{in}(i, j) > 0$ and $\mathbf{Mat}_{out}(i, j) = 0$, there exists convergent points around $p(i, j)$.
3. If $\mathbf{Mat}_{in}(i, j) = 0, \mathbf{Mat}_{out}(i, j) > 0$, there exists divergent points around $p(i, j)$.
4. If $\mathbf{Mat}_{sdelta}(i, j) > 0$, $\mathbf{Mat}_{in}(i, j) > \mathbf{Mat}_{out}(i, j)$, a flow will be "squeezed" when the flow p passes through the region around $P(i, j)$.
5. If $\mathbf{Mat}_{sdelta}(i, j) < 0$, $\mathbf{Mat}_{in}(i, j) < \mathbf{Mat}_{out}(i, j)$, a flow will be "expanded" when the flow passes through the region around $p(i, j)$.

From above definition, DCM is somewhat like divergence. The divergence represents the volume density of the outward flux of a vector field from an infinitesimal volume around a given point. The divergence of the velocity field in that region would have a nonzero value only when the region is a source or sink. As shown in Figure 1, if there is no sink or source in the region, divergence is 0. On the contrary, the length variation between inflow interface and outflow interface is nonzero, which is

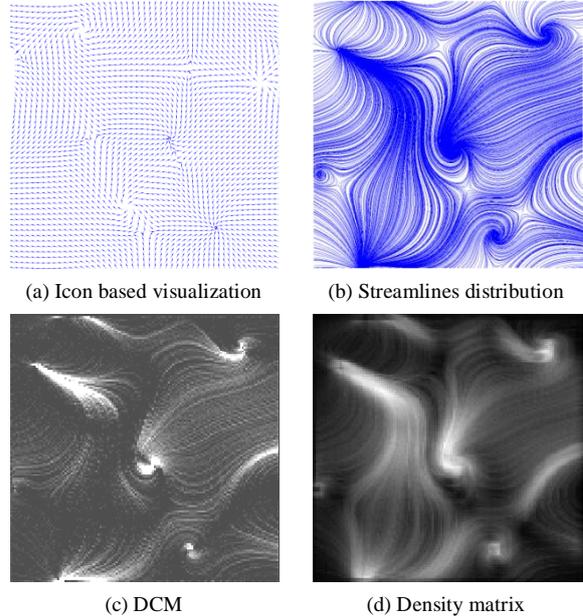


Figure 2. Vector field and its statistics matrix described by our DCM.

Figure 2(a) shows icon based visualization result. Figure 2(b) shows the sample streamlines. Figure 2(c) shows DCM and Figure 2(d) shows the density matrix. Vector field variation is more enhanced in DCM than that in Density Matrix. The density of the consistent region may be very higher, while the value of DCM may be very little.

3.2. DCM seeding strategy

We try to sort the seeds according to the variation of the vector field. A seed with greater variation has higher priority.

In this section, DCM defined in the past section is used to represent the variation of the vector field. According to this DCM streamline start points are seeded mainly depending on the maxima of the matrix. The generation of each streamline lowers the matrix locally until the given condition is satisfied.

3.2.1. Initialization

To start our iterative seeding strategy, we need an initialization set of streamlines. The maxima of DCM can be regarded as the initial seed. As the streamlines vary greatly around the elements of big values in the DCM, and the feature are more evident. If there are several candidate seeds with the same value, we randomly get one from the candidates. Thus if we assume a constant DCM, start points are generated randomly and would not be picked in a row.

If there are some critical points in the vector field, the topology structure is an import property of the vector field. To discover the vector field's detail, seeds around the critical points are preferred. DCM captures sources or sinks nodes easily. On the other hand, streamline around a saddle are much less than around other positions. So seeds around saddle are placed firstly. The location and classification methods of critical points can be found in [Gre92] and [Hel89] [Hel91].

3.2.2 Iteration

Each of the iteration consists of two major parts:

1. Trace a new complete streamline in forward and backward direction and test for intersections.
2. Update the DCM according to the new streamline.

In step 1, new seed is picked by get the maxima of DCM. As described in the initial step, if there are several candidate seeds with the same value, we randomly get one from the candidates.

The element priority of DCM around the new streamline is lowered after the streamline is added. If the DCM is not updated, the next candidate seed may be very close to the previous one and the generated streamlines are also very close to each other. So an update process is taken after a new complete streamline is added.

Obviously the influence from the new streamline on the vector field's feature of a given region is related to the distance between the streamline and the region. For a given new streamline, we first get all streamlines' positions in DCM, which is denoted as a position set S_p . All the elements of these positions are set to 0, which means that no streamline will be added more than once. The other elements in DCM are updated by their distances to the set S_p . For a given position p , the value $DCM(p)$ is updated by a function $F_{update}()$ as follows:

$$DCM(p) = F_{update}(Dis, DCM(p))$$

Where Dis is the distance between p and set S_p .

For a given $DCM(p)$, Dis is non-negative. The longer Dis is, the smaller $DCM(p)$ is. In other words, the farther away from the region, the less influence the new streamline has on the region.

If the distances between all position and the set SP are calculated during DCM update process, too many CPU resources will be consumed. Given a maximal distance d_{max} , if we have $d > d_{max}$, then $DCM(p)$ is the same as the previous value. So we only update those values whose distances to set SP are no more than d_{max} . Inspired by [Set99], a fast marching method is adopted in this paper.

If seeds around the saddles are placed firstly, we update the DCM when all the streamlines from the saddle seeds are generated.

3.2.3. Termination

The algorithm terminates if either of the following happens.

- The number of streamlines is greater than a given value. If the number is too small, some important detail may be missed.
- DCM satisfies some conditions, such as the minimum of DCM is smaller than the given value, which means the most important feather is captured.

4. RESULTS AND DISCUSSION

We tested our approach for some analytical and computational data sets. The data sets are used to compare random seeding against DCM seeding. The quality of streamlines relays on the coverage, uniformity and continuity. For the continuity, all the streamlines generated by our method are complete streamline, which means the streamlines are the longest of all the streamlines passing through the same seeds. Because there are no standards to compare uniformity and continuity quantitatively, we compare the results with other methods visually.

Our results have been generated on a Windows Vista ThinkPad T61p notebook equipped with an Intel

Core2 Duo T7500 2.2GHZ CPU, 3GB Ram, Nvidia Qurdro FX 570M 128M GPU. All the three tests cost no more than 10 seconds including the DCM calculation process which costs most of the time.

Figure 3 shows the comparison with other methods. The vector field consists of 50*50 vectors. All method have almost the same results with more

streamlines. The compared algorithm tends to produce short separated streamline and is much more obvious when using less streamlines. Our method does not require as much uniformity as others do, by which it can capture more features with less streamlines, which is shown in center and right of Figure 3(d).

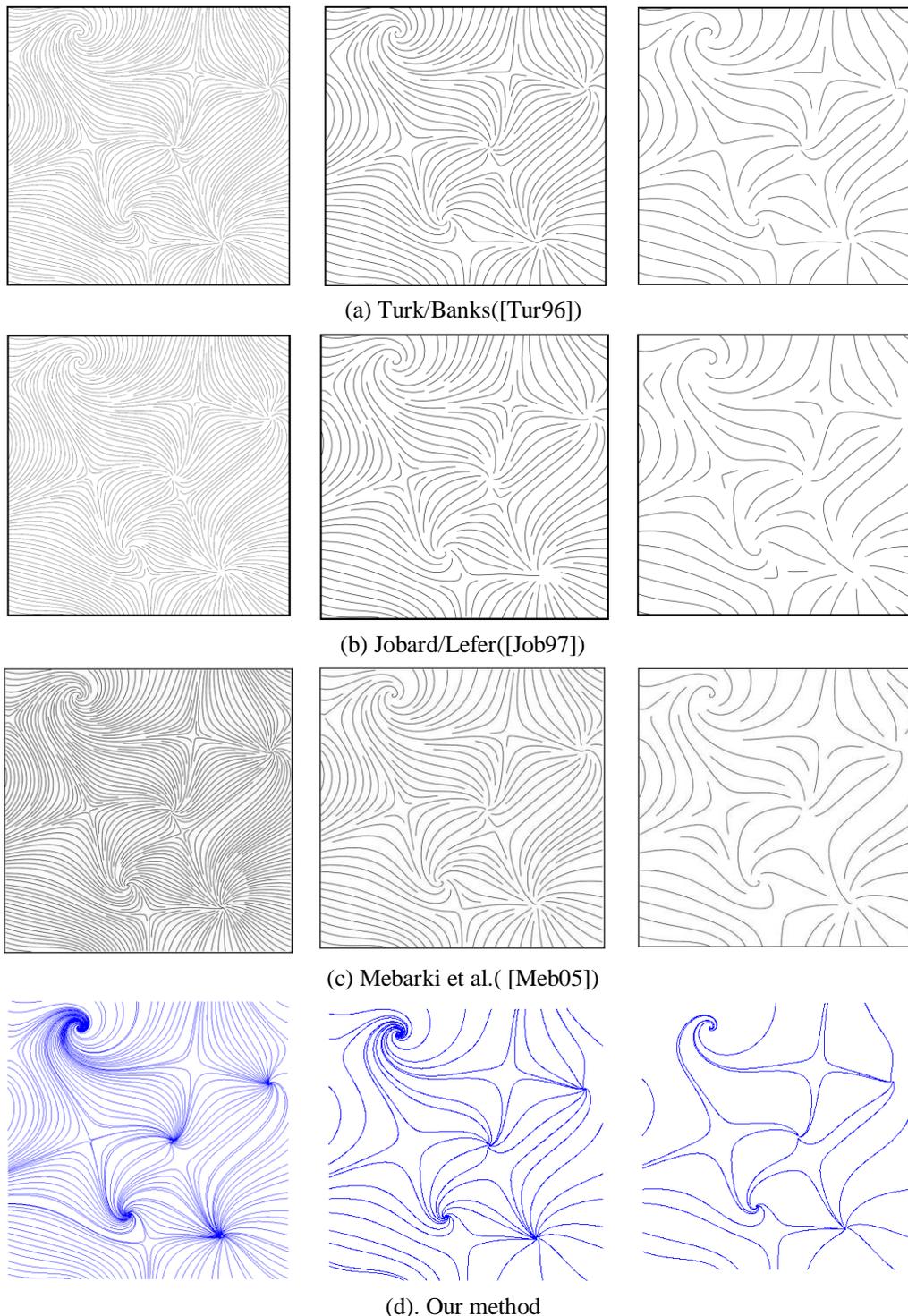


Figure 3 Comparison of streamline placement techniques

Figure 4 shows a slice of a 3D vector field. The vector field consists of 128×128 vectors, which comes from simulation of swirling jet entering fluid. Figure 4(a) and 4(b) show results of our method. The swirl of the vector field is well captured. On the other hand, Figure 4 (c) and (d) show the results of algorithm of Jobard/Lefer. The swirl is not so distinct, for the streamlines are not long enough to reveal the features.

Figure 5 shows comparison with algorithm of Vermal et al. The vector field consists of 70×70 vectors. Figure 5(a) and 5(b) show results of algorithm of Verma. The algorithm does perform well in the critical regions. In other words, the critical regions can not be well represented, especially when fewer streamlines are used. Figure 5(c) and 5(d) show results of our method. Very few streamlines are produced in Figure 5(d), but the critical regions are very clear.

Our algorithm only uses complete streamlines. The long streamlines are preferred in this paper, while discontinuities in the layout with shorter streamlines may impair the impression of a flow field.

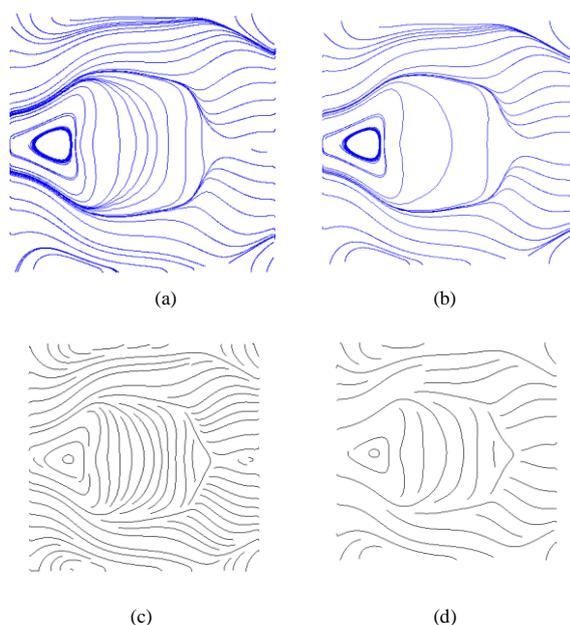


Figure 4 Swirling jet entering fluid at rest.

Our seeding strategy picks position with the maxima of DCM. The greater difference-contribution the position has, the greater the variation is. The position with great variation is picked firstly, such as convergent point. And there are less streamlines in the region with lower difference-contribution, such as in Figure 4(b) while the streamlines in Figure 4(d) are still even almost everywhere.

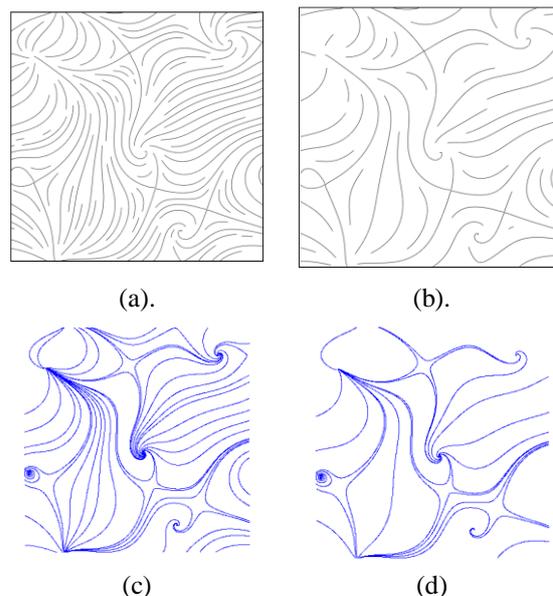


Figure 5. Comparison to feature-based technique

5. CONCLUSION

A DCM seeding strategy is proposed in this paper. We introduced inward and outward contribution of a position as variation measure of the vector field. Then DCM is defined. The streamline starting points are seeded mainly depending on the maxima of the DCM matrix. The generation of each streamline lowers the matrix locally until the given condition is satisfied.

The new approach catches regions with great variation and the vector field can be represented by less streamlines.

6. ACKNOWLEDGEMENTS

We would like to thank Roger Crawfis for providing the tornado dataset, and also University of California Davis for the provision of the swirling jet dataset.

The work described in this paper was supported by Chinese National High-Tech R&D Program Grant (2007AA01Z318, 2007AA01Z159, 2009AA01Z324), National Natural Science Foundation of China (90915010, 60925007, 60973052, 60703062, 60833007, U0735004), National Basic Research Program of China(2010CB328002).

7. REFERENCES

- [Liu06] Liu, Z.: 'An Advanced Evenly-Spaced Streamline Placement Algorithm', IEEE Transactions on Visualization and Computer Graphics, 2006, 12, (5), pp. 965-972
- [Ver00] Verma, V., D. Kao, and A. Pang. A flow-guided streamline seeding strategy. in IEEE Visualization 2000, pp. 163-170

- [Lar04] Laramee, R.S., Hauser, H., Doleisch, H., Vrolijk, B., Post, F.H., and Weiskopf, D.: 'The state of the art in flow visualization: dense and texture-based techniques', *Computer Graphics Forum*, 2004, 23, (2), pp. 203-221
- [Pos03] Post, F.H., Vrolijk, B., Hauser, H., Laramee, R.S., and Doleisch, H.: 'The state of the art in flow visualisation: Feature extraction and tracking', *Computer Graphics Forum*, 2003, 22, (4), pp. 775-792
- [Tur96] Turk, G. and D. Banks. Image-guided streamline placement. in *SIGGRAPH 1996*. pp. 453-460. New York, USA.
- [Mao98] Mao, X.Y., et al. Image-guided streamline placement on curvilinear grid surfaces. in *IEEE Visualization '98*. 1998. pp. 135-142.
- [Job97] Jobard, B. and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. in *Visualization in scientific computing '1997*, pp. 43-56.
- [Job00] Jobard, B., and Lefer, W.: 'Unsteady flow visualization by animating evenly-spaced streamlines', *Computer Graphics Forum*, 2000, 19, (3), pp. C31-C39.
- [Job01] Jobard, B., and Lefer, W.: 'Multiresolution flow visualization', *WSCG '2001: Short Communications and Posters*, 2001, pp. P34-P37.
- [Mat03] Mattausch, O., et al. Strategies for interactive exploration of 3D flow using evenly-spaced illuminated streamlines. in *Spring Conference on Computer Graphics*. 2003: ACM New York, NY, USA, pp. 213-222.
- [Meb05] Mebarki, A., P. Alliez, and O. Devillers. Farthest point seeding for efficient placement of streamlines. in *IEEE Visualization 2005*, pp. 479-486.
- [Ye05] Ye, X.H., Kao, D., and Pang, A.: 'Strategy for seeding 3D streamlines', *IEEE Visualization 2005, Proceedings*, 2005, pp. 471-478.
- [Li07] Li, L.Y., and Shen, H.W.: 'Image-based streamline generation and rendering', *IEEE Transactions on Visualization and Computer Graphics*, 2007, 13, (3), pp. 630-640.
- [Sch07] Schlemmer, M., et al. Priority Streamlines: A context-based Visualization of Flow Fields. in *EuroVis07: Joint Eurographics - IEEE VGTC Symposium on Visualization*. 2007, pp. 227-234.
- [Gre92] Greene, J.M.: 'Locating three-dimensional roots by a bisection method', *J. Comput. Phys.*, 1992, 98, (2), pp. 194-198.
- [Hel89] Helman, J., and Hesselink, L.: 'Representation and Display of Vector Field Topology in Fluid-Flow Data Sets', *Computer*, 1989, 22, (8), pp. 27-36.
- [Hel91] Helman, J.L., and Hesselink, L.: 'Visualizing Vector Field Topology in Fluid-Flows', *Ieee Computer Graphics and Applications*, 1991, 11, (3), pp. 36-46.
- [Set99] Sethian, J.A.: 'Fast marching methods', *Siam Rev*, 1999, 41, (2), pp. 199-235.

Sparse Bundle Adjustment Speedup Strategies

Christian Lipski
TU Braunschweig,
Germany
lipski@cg.tu-bs.de

Denis Bose
TU Braunschweig,
Germany
bose@tu-bs.de

Martin Eisemann
TU Braunschweig,
Germany
eisemann@cg.tu-bs.de

Kai Berger
TU Braunschweig,
Germany
berger@cg.tu-bs.de

Marcus Magnor
TU Braunschweig,
Germany
magnor@cg.tu-bs.de

ABSTRACT

Over the past years, Structure-from-Motion calibration algorithms have become widely popular for many applications in computer graphics. From an unordered set of photographs, they manage to robustly estimate intrinsic and extrinsic camera parameters for each image. One major drawback is the quadratic computation time of existing algorithms. This paper presents different strategies to overcome this problem by only working on subsets of images and merging the results. A quantitative comparison of these strategies reveals the trade-off between accuracy and computation time.

Keywords: Camera Calibration, Sparse Bundle Adjustment, Structure-from-Motion.

1 INTRODUCTION

Many of today's vision and graphics applications are based on well-calibrated cameras. The camera calibration process has been widely explored in the past years and many methods have been proposed - ranging from classical checkerboard recordings to calibration without a priori known patterns [PGV⁺04, SSS08]. These recent methods require the recorded images only to obtain a multitude of feature points (e.g. SIFT-features) for a proper *self-calibration*. Especially image-based modeling and rendering applications benefit from the development: The camera setup can be freely chosen and a calibration recording session has become obsolete. Furthermore, the camera setup does not need to be fixed during the recording anymore. Scenes recorded with multiple handheld cameras can nowadays be reconstructed by employing the self-calibration methods. The method most widely used in the research community is the Sparse Bundle Adjustment, or *Bundler* for short, introduced by Snavely et al. [SSS08]. The recorded images are searched for feature points, e.g. SIFT-features. Feature points, that are shared between any two images are considered as correspondence points. After an initial estimate of camera parameters, these points are triangulated and reprojected to the images. The reprojection error, i.e. the euclidean distance between the original feature locations and their reprojections on the image plane is minimized during the so-called bundle adjustment. Being considered as a milestone in the community,

this tool, however, has serious issues regarding the computation time.

In this paper we examine the reasons for these issues and propose new methods to significantly reduce the computation time whilst keeping the reprojection error minimal. The paper is outlined as follows. We give a brief overview to recent advances in calibration methods in Section 2, also focussing on Bundler's runtime issues. Afterwards, we introduce two strategies to tackle these problems in Section 3. We justify our methods with a quantitative analysis in Section 4 and conclude in Section 5.

2 RELATED WORK

While our work mainly improves Bundler by Snavely et al. [SSS08], a renowned tool for 3D object reconstruction from uncalibrated multicamera footage used by many other scientists [WMC04, Sna08, JB09], we also relate to the following previous work in the field of multicamera calibration.

A good overview of calibration algorithms can be found in the paper by Triggs et al. [TMHF99]. The commercial tool *Boujou* [vic09] reconstructs 3D models from moving uncalibrated cameras. Hasler et al. [HRT⁺09, THWS08] calibrate multiple moving unsynchronized cameras by first finding each camera's trajectory (using KLT-tracking and RANSAC-fitting). An approach based on geometric dissimilarity measurement is described by Denzler et al. [BBD09]. They rely on a less restrictive matching method compared to [SSS08].

However, most calibration approaches, including the Sparse Bundle Adjustment [SSS08], suffer from long computation times. Schwartz et al. [SK09] investigate the preconditions of multicamera calibration and suggest to merge connected components for an initial estimate to achieve computation speedup. Byrod et al. suggest an iterative adjusting approach by solving the problem with a conjugate gradient method. They pre-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

condition the matrix with a multiscale Gauss-Seidel approach. He et al. [HQH08] try to improve the computation time by propagating matches between camera pairs.

Our approaches, instead, address the computation time problem by applying Bundler to a limited selection of images, and incorporating the other images at a later stage.

2.1 Bundler: Sparse Bundle Adjustment

As our work is based on the work of Snavely et al. [SSS08], we will give a brief introduction into the Bundler Calibration pipeline. Bundler accepts an unordered set of photographs as input, along with an initial estimate of the focal lengths of the cameras that took these images. A calibration of the images is the output of the algorithm which provides the relative rotations R and translations t of all cameras along with the intrinsic parameters (focal length and radial lens distortion). The first part of the Pipeline is an image feature extraction. Snavely et al. proposed to use SIFT features [Low04] for this task. This step runs in linear time. A pairwise feature matching phase matches the key features of all images pairs. This step runs in quadratic time. The two most promising images are chosen for an initial calibration. After calibration, an initial set of 3D points is obtained via triangulation of the corresponding points. The bundle adjustment step refines the calibration by minimizing the reprojection errors of the obtained points. The remaining cameras are added one by one: If at least six correspondences to the already reconstructed 3D points are known, an initial estimate of its parameters is calculated via Direct Linear Transformation. A bundle adjustment step refines the initial parameters of the camera, new reconstructed 3D points may be added and a global bundle adjustment step is performed. This final phase runs in quadratic time. We can see that both the key feature matching and the bundle adjustment run in quadratic time with respect to the amount m of input images. The overall computational complexity of Bundler is therefore $O(m^2)$.

3 SPEEDUP STRATEGIES

Data sets containing just a few hundred images may lead to run-times of several days on a single CPU. Instead of focussing on algorithmic techniques to tackle this problem, our approaches reduce the number of images used as an input to the sparse bundle adjustment. We developed two different strategies that let Bundler only run on subsets of images, thus decreasing the overall run-time.

3.1 Merge Images Approach

We partition the set of images into n subsets of equal size. Given an (arbitrarily chosen) order of images, the

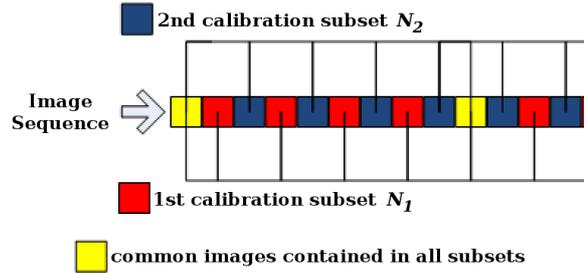


Figure 1: Merge Images Approach for $n = 2$ and $k = 9$. Two subsets are created and separated independently (blue and red boxes). All subsets contain a set of common images (yellow boxes). Both subsets are merged via a Procrustes transformation.

first, the $n + 1$ st, the $2n + 1$ st, etc... image are put in subset N_1 . The second, $n + 2$ nd, $2n + 2$ nd, etc... image are placed in subset N_2 and so forth, see Fig. 1. Afterwards, we make sure that the image subsets also contain some common images. We select each k th image from the original image set and add it to each subset if it is not already present in that set. Each subset is calibrated with Bundler independently. We are now faced with the problem that we obtained n calibrations of the same scene. We arbitrarily pick the first subset to be our reference set and merge the other calibration results into this reference system. The subset's reference systems differ in their location z_n , their rotation R_n and their scale b_n . So, a Procrustes transformation Φ has to be obtained for each subset to align it with the reference set. When this transformation is known, new rotation matrices R_{new} and translation vectors $t_{w_{new}}$ are obtained. We recall that the position p of a camera can be derived from its rotation matrix R and its translation vector t .

$$p = -R^T t. \quad (1)$$

We can obtain a set of common points for all subsets of images when we compute the camera positions for the common images in each set. For each image subset, we obtain the transformation ϕ that maps the set of common camera locations to the one of the reference calibration. We make use of the matlab implementation of the Procrustes Analysis. The same transformation can be used to obtain the camera locations p_{new} , the rotation matrices R_{new} and the translation vectors t_{new} : The new camera locations and rotation matrices can be derived by directly applying ϕ . The translation vectors are computed as follows:

$$t_{new} = -R_{new}^T p_{new} \quad (2)$$

The speedup caused by this strategy can be formalized by a reduction of the complexity from $O(m^2)$, where m is the total number of images, to $O(n \cdot (m/n + m/k)^2)$. As we will show in Section 4, an adequate se-

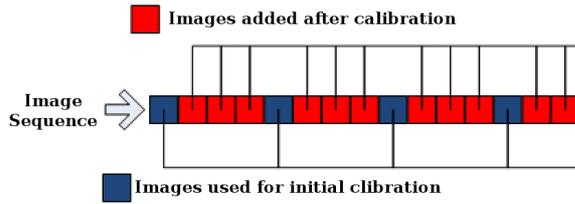


Figure 2: Add Images approach with $n = 4$. Only each n th image is used for the initial calibration (red boxes). The other images are added using via Direct Linear Transformation.



Figure 3: A representative frame of the test sequence.

lection of k and m can cause a dramatic speedup, while preserving a high accuracy, i.e. a low reprojection error.

3.2 Add Images Approach

The original implementation of bundler provides the opportunity to add images to an already calibrated set of images. We exploit this feature and determine a subset of images that is calibrated instead of the complete set of images. We add every n th image into the subset, calibrate the subset and add all remaining images via Bundler's *Add Images* feature, Fig 2. When adding images to the calibrated set of images, no new bundle adjustment iteration is performed. I.e., only the optimal rotation matrix and translation vector for the new image is determined, no new 3D points are inserted and no optimization of the camera parameters is performed. Therefore, adding images runs in linear time. Instead of the original computational complexity of $O(m^2)$, the Add Images Approach has a complexity of $O((m/n)^2 + (m - m/n))$, which is even faster than the Merge Image Approach.

4 RESULTS

Our speedup strategies are tested on the graffiti image sequence, Fig. 3. This test sequence contains the recordings of 5 non-stationary camcorders, all pointed towards a juggler in front of a highly textured wall. Each camera recorded 40 video frames, resulting in a total size of 200 images. The image size is 480px \times 270px. We calibrate the set of 200 images with the original bundler algorithm, the Merge Images Approach and the Add Images Approach. Several calibration runs

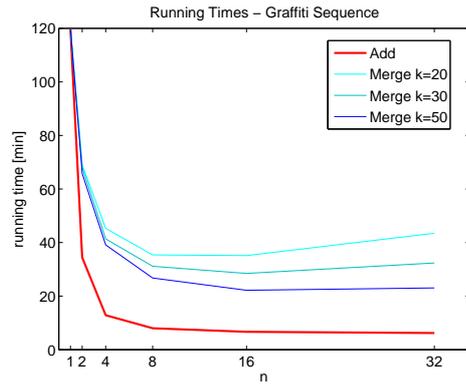


Figure 4: Runtimes for Bundler using both speedup strategies with different parameters n and k . We obtained results for $n = 1, 2, 4, 8, 16, 32$ and $k = 20, 30, 50$. Please note that $n = 1$ is identical to a calibration without speedup. Compared to the original Bundler calibration ($n = 1$), a significant speedup can be achieved in all cases.

with different parameters quantitatively determine the tradeoff between computation time and accuracy.

As an error measure, we use the reprojection error of the reconstructed 3D points. In order to make all speedup scenarios comparable, we have to make a slight alteration to the Add Image approach. When using this approach, the reconstructed point sets tend to be much smaller with increasing n . Because not all images are used for Bundle Adjustment, less reconstructed points are added. It is also quite likely that only these points will be incorporated into that set that have a low reprojection error: Bundler either optimizes or discards points. Therefore, we store a list of reconstructed 3D points and their image locations when running Bundler without a speedup strategy. When evaluating the reprojection error with the Add Images method, we reconstruct the full set of 3D points by triangulation of the previously stored image locations. We then measure the reprojection error of the full set of 3D points. For both speedup methods, we calibrate with $n = 2, 4, 8, 16, 32$. In the case of the Merge Images method, we did individual test runs for each different n with $k = 20, 30, 50$.

The computation times, Fig. 4, reveal that the Add Images Approach outperforms the Merge Images Approach in terms of speed. For $n = 32$ it takes just 6 instead of 120 minutes to perform the calibration. This is not surprising, as the Merge Images method does run n separate calibrations instead of only a single one. With computation times as low as 22 minutes, the Merge Images method still achieves a remarkable result. When choosing $k > n$, the runtimes start to increase again, as a lot of redundant frames are incorporated into the calibrations. All calibration runs are performed on a 2.66 Ghz Intel CPU using a single core. In defense of the Merge Images method one must admit that the Merge

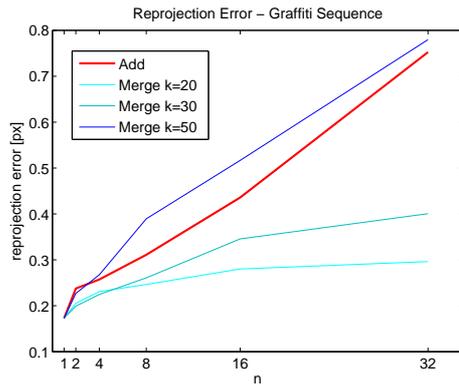


Figure 5: Average reprojection error for both the Add Images and the Merge Images approach. Please note that $n = 1$ is identical to a calibration without speedup.

Images method can be easily parallelized. In contrast, the Add Images approach runs a consecutive algorithm.

When we look at the reprojection error, one can see that for low k ($k = 20, 30$) values, the Merge Images Method achieves much better results, Fig. 5. With higher k ($k = 50$) the merging of data sets seems to become unstable. The Add Images method's reprojection error increases linear with n . Although, for $n = 32$ the mean error still stays below 0.8 px.

When we look at the mean deviation of the error, we see that it keeps low in all scenarios where the Merge Approach is used, Fig. 6. On the other hand, the deviation of the error climbs up to a value of 1.6 px when using the Add Images Approach. This can be explained by the fact that many of these points were not considered for bundle adjustment and that a few large outliers exist. The shown quantitative results lead to the interpretation that both approaches succeed in their task to speed up the computation while maintaining a low reprojection error. When a very high speedup is required, the Add Images approach is the first choice, especially for high values of n , drastic speedups are achieved. When accuracy is crucial, the Merge Images approach is the more advisable choice. One should pick $n < k$ when using the Merge Images method, otherwise the speedup will significantly diminish.

5 CONCLUSION

We introduced two methods, i.e., the Merge Images and the Add Images approach, to speed up the computation in the camera calibration tool Bundler. We found that both methods achieve comparably fair results, i.e. minimal reprojection error.

In the future we want to examine, if clustering of images will lead to further speedup. I.e., if instead of picking images arbitrarily for our calibration subsets, a more considerate preselection of images can be used to further improve the accuracy of the calibration.

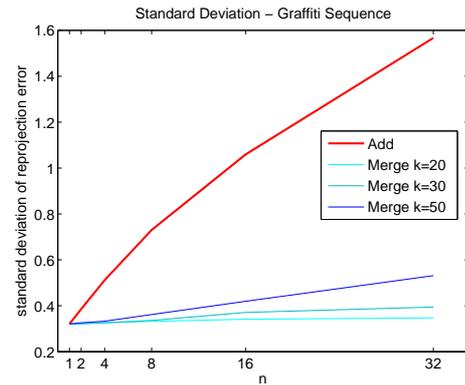


Figure 6: Standard deviation of the reprojection error for both speedup strategies.

REFERENCES

- [BBD09] M. Brückner, F. Bajramovic, and J. Denzler. Geometric and probabilistic image dissimilarity measures for common field of view detection. *Proc. of CVPR*, pages 2052–2057, 2009.
- [HQH08] S. He, Y. Qi, and F. Hou. Photo Traveler: A System for Exploring Photos in 3D. *Proc. of the JCSIS*, 2008.
- [HRT⁺09] N. Hasler, B. Rosenhahn, T. Thormählen, M. Wand, J. Gall, and H.-P. Seidel. Markerless motion capture with unsynchronized moving cameras. In *Proc. of CVPR*, pages –, Miami, USA, June 2009. IEEE Computer Society. (to appear).
- [JB09] K. Josephson and M. Byrod. Pose estimation with radial distortion and unknown focal length. *Proc. of CVPR*, pages 2419–2426, 2009.
- [Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Intl. Journal of Computer Vision*, 60:91–110, 2004.
- [PGV⁺04] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, and R. Koch J. Tops. Visual modeling with a hand-held camera. *Intl. Journal of Computer Vision*, pages 207–232, 2004.
- [SK09] C. Schwartz and R. Klein. Improving initial estimations for structure from motion methods. In *Proc. of the CESC*, April 2009.
- [Sna08] N. Snavely. *Scene Reconstruction and Visualization from Community Photo Collections*. PhD in computer science, University of Washington, Computer Science and Engineering, University of Washington, Box 352350, Seattle, WA 98195-2350, USA, 2008.
- [SSS08] N. Snavely, S.M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *Intl. Journal of Computer Vision*, 80(2):189–210, 2008.
- [THWS08] T. Thormählen, N. Hasler, M. Wand, and H.-P. Seidel. Merging of feature tracks for camera motion estimation from video. In *Proc. of the CVMP*, London, UK, November 2008.
- [TMHF99] B. Triggs, P.F. McLauchlan, R.I. Hartley, and A.W. Fitzgibbon. Bundle adjustment—a modern synthesis. *Lecture Notes in Computer Science*, pages 298–372, 1999.
- [vic09] vicon. boujou. <http://www.vicon.com/boujou/>, 2009.
- [WMC04] K. H. Wong, M. Ming, and Y. Chang. 3d model reconstruction by constrained bundle adjustment. In *Proc. of ICPR*, pages 902–905, Washington, DC, USA, 2004. IEEE Computer Society.

Reduction of Animated Models for Embedded Devices

Jiri Danihelka, Lukas Kencl, Jiri Zara

Czech Technical University in Prague, Faculty of Electrical Engineering

{danihjir, kencl, zara}@fel.cvut.cz

ABSTRACT

We present a new supplementary method for reduction of animated 3D polygonal models. The method is applicable mainly in animation of human faces and it is based on intelligent merging of visemes represented by key polygonal meshes. It is useful for devices with limited CPU and memory resources like mobile phones or other embedded devices. Using this approach we can reduce operation memory needs and time to load the model from storage. We describe the algorithm for viseme merging and we prove that our method is optimal for selected metrics. Finally we validate method performance on an example and compare with the case when only traditional methods for 3D models reduction are used.

Keywords: animation, model, reduction, viseme

1 INTRODUCTION

Modern technology devices like personal computers and mobile phones are becoming more and more powerful and complicated. Many people have difficulties controlling miscellaneous computer systems and applications [17]. Computer graphics and designers of computer programs look for new kinds of interfaces to control still more complex computer programs. Talking-head interface seems to be a promising alternative to traditional menu/windows/icons interface for sophisticated applications. Such interface has proven to be useful as a virtual news reader [1], blog enhancement [11] and in many other cases.

So far talking-head interface has been applied mostly on desktop PCs. However, recent small electronic equipment, such as mobile phones, pocket computers and embedded devices possess enough CPU power to offer the talking-head interface as well.

Current smartphones and pocket computers usually have 128MB or 256 MB of RAM. Most of this memory is occupied by the operating system(OS) itself or by OS extensions like HTC TouchFLO or Samsung TouchWiz (formerly pocket computers had only 16 or 32 MB of operation memory, but the OS was stored in read-only memory rather than in RAM). The lack of memory is a bottleneck for animations computed by interpolation of polygonal meshes, because it requires a lot of possibly large polygonal meshes loaded in memory.

To achieve the lowest memory requirements, we have decided to reduce both the amount of polygons in the

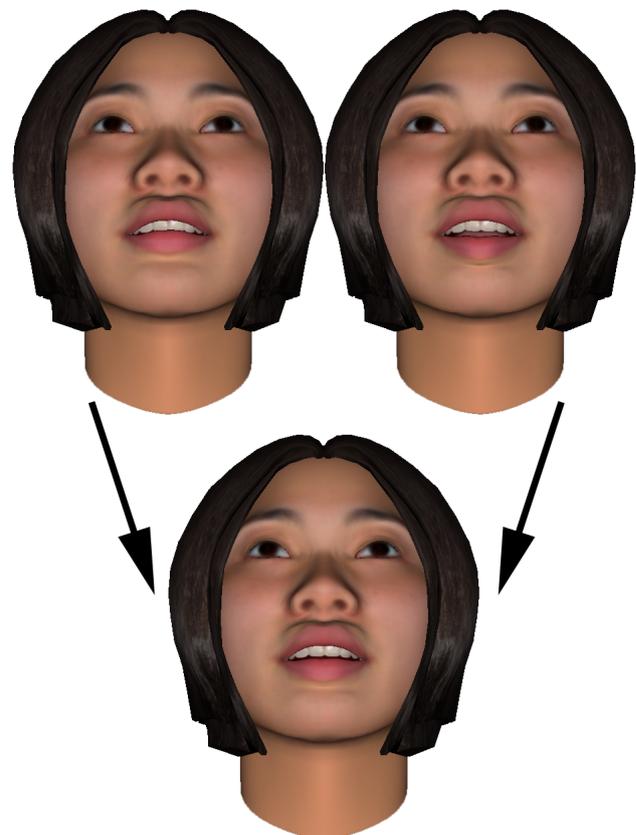


Figure 1: A talking head keyframe model articulating the phoneme "f" (left) is similar to a keyframe model articulating the phoneme "th" (right). Our algorithm detects such similarity and replaces both models with one merged model (down).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

mesh and the number of key meshes (see figure 1). We propose a dissimilarity metric to detect similar models and a technique to merge them. We prove that our merging technique is optimal for the given dissimilarity metric.

2 RELATED WORK

Traditional methods for polygonal reduction are sufficiently covered in [10] and [15]. Specific aspects about geometric rendering and model reduction on mobile phones and embedded devices were presented by Pulli et al. [16].

An interesting way for speeding up morphing animation on embedded devices was proposed by Berner [5]. It is based on optimization strategies by omitting less important polygonal meshes during the animation.

In our research we aim to develop software compatible with the Xface animation framework [2, 3] that is open-source and widely used in academia. There are also more advanced animation frameworks that use skeleton-muscle [18] animation model instead of MPEG-4 standard. The best known of them is Greta [13]. A method of anatomical musculature modeling to achieve realistic and real-time figure animation was proposed by Zuo Li et al. [12].

However none of the works above focuses on reducing the number of visemes (as our work does).

3 FACE ANIMATION PRINCIPLES

3.1 Phonemes and visemes

When using face animation in talking-head applications, we have to consider both visual and audio effects. They are described by visemes and phonemes. A phoneme is an element of spoken language similarly like a letter is an element of written language. A viseme is an element of facial animation. It describes the particular facial position when pronouncing a phoneme. Usually one phoneme corresponds to one viseme, but sometimes multiple phonemes share the same viseme. This happens when facial position of two or more phonemes differs only by position of non-displayed body parts like vocal cords or a tongue.

The frequencies of occurrence of phonemes and visemes depend on spoken language, there are also differences e.g. between frequencies in British and American English. English has 40 different phonemes.

For our algorithm we need to know the frequencies of phonemes and visemes. The frequencies of phonemes can be determined by converting a long text (at least several pages) using a phonetic transcription software and then by counting the phoneme frequencies in the transcribed text. Such process is usually part of text-to-speech-engine pre-processing of text input for voice synthesis. There is also a free transcription engine available together with typical frequencies of American English phonemes [6]. Having the frequencies of phonemes one can determine the frequencies of visemes using phoneme-to-viseme mapping function.

For our experiments we use the FaceGen facial editor [19] to generate human head visemes. This editor generates 16 different visemes.

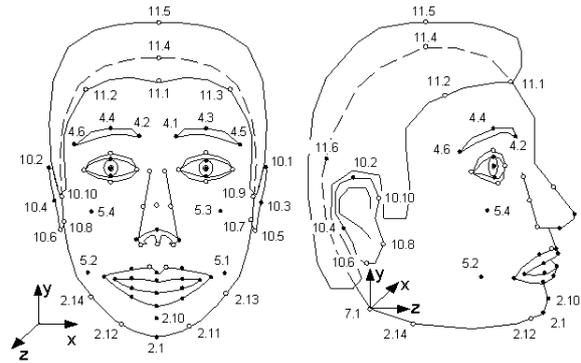


Figure 2: A subset of feature points (FP) defined in MPEG-4 facial animation standard [8]

3.2 MPEG-4 animation

The most widely accepted standard for human face animation is the ISO standard MPEG-4 released by the Moving Pictures Experts Group in 1999 [7, 8].

In this standard 84 feature points (FPs) are specified on human face (see figure 2). The facial animation is controlled by 68 parameters called Facial Animation Parameters (FAPs).

The MPEG-4 standard allows two ways of facial animation. The first one manipulates the feature points individually and can achieve various range of facial expressions. The second one is based on interpolating between two keyframe models. This interpolation can be done either linearly or with cubic interpolation function.

In this paper we focus on the keyframe facial animation. This approach is less CPU intensive and the visual results of this animation are sufficient for mobile phones and embedded devices.

4 DEFINITIONS

4.1 Polygonal model

For purposes of this paper, the polygonal model is a triplet (V, E, P) of vertices V, edges E, and polygons P. To avoid rendering problems with general polygons after geometric transformations, we triangulate all polygons in advance.

Fully triangulated models allow us use a specific metric for model comparison (see section 4.3). They also fit very well into commonly used graphics libraries for mobile phones and embedded devices like OpenGL ES (OpenGL for Embedded Systems) [9] which are optimized for processing triangles only.

4.2 Interpolable set of models

We call polygon models interpolable if they differ only in coordinates of their vertices. Interpolable models have the same topology and the same number of vertices, edges and polygons. There must also be given a

bijection function that matches the corresponding vertices/edges/polygons.

4.3 Polygonal model dissimilarity

We define the polygonal model dissimilarity as a metric (distance function) ρ for two interpolable models.

$$\rho(A, B) := \sum_{k=1}^{\|V\|} w(v_k) \|v_{A,k} - v_{B,k}\|^2 \quad (1)$$

where

A and B are the polygonal models.

$w(v)$ is the weight of the vertex v . It represent an importance of the vertex in the model. The author of the model can set higher weights for vertices important for human perception.

For models with unspecified weights, we have considered two general metrics:

$$\rho_1(A, B) := \sum_{k=1}^{\|V\|} \|v_{A,k} - v_{B,k}\|^2 \quad (2)$$

$$\rho_2(A, B) := \sum_{k=1}^{\|V\|} S(v_N) \|v_{A,k} - v_{B,k}\|^2 \quad (3)$$

where

$S(v_{N,k})$ is a sum of surfaces of triangles incident with vertex $v_{N,k}$. Since the triangle surface may differ for individual visemes, we work with polygon surfaces in the neutral expression of the model $N = (V_N, E_N, P_N)$.

The first metric assumes that more important areas are tessellated more densely. The weight of a face part is given by a number of its vertices.

The second metric can be used if each part of the model surface is equally important for the animation. If we use this metric it is necessary to split all polygons to triangles first as mentioned in section 4.1. We have proven that both metrics give the same results if applied in our reduction algorithm. Thus the real implementation can utilize the first and more simple metric only.

4.4 Dissimilarity for sets of polygonal models

Let $\mathbb{A} = \{A_1, A_2, \dots, A_n\}$, $\mathbb{B} = \{B_1, B_2, \dots, B_m\}$ are two sets of polygonal models that represents visemes. Let $f(A_1), f(A_2), \dots, f(A_n)$ are frequencies of visemes in \mathbb{A} . If we have a dissimilarity metric for polygonal models $\rho(A, B)$, we can define dissimilarity for two sets of polygonal models $\rho_f(\mathbb{A}, \mathbb{B})$ as:

$$\rho_f(\mathbb{A}, \mathbb{B}) = \sum_{i=1}^n f(A_i) \min_{j=1 \dots m} \rho(A_i, B_j) \quad (4)$$

It is the sum of distances from each model from \mathbb{A} to its most similar models in \mathbb{B} . Note that dissimilarity function for sets of polygonal models is not a metric because it is not symmetrical.

4.5 Problem definition

We describe an algorithm for the following problem:

Input:

Set of polygonal models $\mathbb{A} = \{A_1, A_2, \dots, A_n\}$. These models represent visemes of a human face that have frequencies $f(A_1), f(A_2), \dots, f(A_n)$. An integer number m ; $m < n$

Task:

Find a set of new polygonal models with m elements $\mathbb{B} = \{B_1, B_2, \dots, B_m\}$ that is the most similar to \mathbb{A} . ($\rho_f(\mathbb{A}, \mathbb{B})$ is minimal for all such sets of polygonal models)

5 FINDING OPTIMAL SOLUTION

The solution for the problem is described in two steps: Firstly, we describe how to solve the extreme case when $m = \|\mathbb{B}\| = 1$. Then we describe the solution for arbitrary value of $\|\mathbb{B}\|$.

5.1 Case $\|\mathbb{B}\| = m = 1$

We have to find such a set of polygonal models $\mathbb{B} = (B)$ with one element for which the expression in equation (4) is minimal.

$$\mathbb{B} = \arg \min_{\mathbb{B}; \|\mathbb{B}\|=1} (\rho_f(\mathbb{A}, \mathbb{B})) \quad (5)$$

We the definition of the dissimilarity for sets (see equation (4)):

$$\mathbb{B} = \arg \min_{\mathbb{B}; \|\mathbb{B}\|=1} \left(\sum_{i=1}^n f(A_i) \min_{j=1 \dots m} \rho(A_i, B_j) \right) \quad (6)$$

Because $m = 1$ we can leave out the second minimum.

$$B = \arg \min_B \left(\sum_{i=1}^n f(A_i) \rho(A_i, B) \right) \quad (7)$$

Now we use the definition of model dissimilarity metric (see equation (1)).

$$B = \arg \min_B \left(\sum_{i=1}^n f(A_i) \sum_{k=1}^{\|V\|} w(v_k) \|v_{A_i,k} - v_{B,k}\|^2 \right) \quad (8)$$

We swap the summations.

$$B = \arg \min_B \left(\sum_{k=1}^{\|V\|} \sum_{i=1}^n f(A_i) w(v_k) \|v_{A_i,k} - v_{B,k}\|^2 \right) \quad (9)$$

Since the vertices of model B are mutually independent, we can calculate each of them individually.

$$v_{B,k} = \arg \min_{v_{B,k}} \left(\sum_{i=1}^n f(A_i) w(v_k) \|v_{A_i,k} - v_{B,k}\|^2 \right) \quad (10)$$

The vertex weight $w(v_k)$ remains constant for individual vertex. Thus it does not affect the argmin expression. We can leave it out.

$$V_{B,k} = \arg \min_{V_{B,k}} \left(\sum_{i=1}^n f(A_i) \|v_{A_i,k} - v_{B,k}\|^2 \right) \quad (11)$$

We use the definition of the Euclidian distance. $v_{A_i,k} = [x_{A_i,k}, y_{A_i,k}, z_{A_i,k}]$, $v_{B,k} = [x_{B,k}, y_{B,k}, z_{B,k}]$

$$V_{B,k} = \arg \min_{[x_{B,k}, y_{B,k}, z_{B,k}]} \sum_{i=1}^n f(A_i) (x_{A_i,k} - x_{B,k})^2 + f(A_i) (y_{A_i,k} - y_{B,k})^2 + f(A_i) (z_{A_i,k} - z_{B,k})^2 \quad (12)$$

We can determine individual coordinates separately, because they are independent on each other. Let us consider the x-coordinate only:

$$x_{B,k} = \arg \min_{x_{B,k}} \sum_{i=1}^n f(A_i) (x_{A_i,k} - x_{B,k})^2 \quad (13)$$

We expand the expression.

$$x_{B,k} = \arg \min_{x_{B,k}} \sum_{i=1}^n f(A_i) (x_{A_i,k}^2 - 2x_{A_i,k}x_{B,k} + x_{B,k}^2) \quad (14)$$

In order to find the minimum, we find where the derivation is equal to 0.

$$0 = \frac{\partial}{\partial x_{B,k}} \sum_{i=1}^n f(A_i) (x_{A_i,k}^2 - 2x_{A_i,k}x_{B,k} + x_{B,k}^2) \quad (15)$$

After the derivation we get:

$$0 = \sum_{i=1}^n f(A_i) (-2x_{A_i,k} + 2x_{B,k}) \quad (16)$$

The second derivation is equal to $2\sum_{i=1}^n f(A_i)$. This is greater than 0 because all of the frequencies are positive. Thus this is a minimum. We express the $x_{B,k}$.

$$x_{B,k} = \frac{\sum_{i=1}^n f(A_i) x_{A_i,k}}{\sum_{i=1}^n f(A_i)} \quad (17)$$

We express the vertex $v_{B,k}$:

$$v_{B,k} = \frac{\sum_{i=1}^n f(A_i) v_{A_i,k}}{\sum_{i=1}^n f(A_i)} \quad (18)$$

We finally express the model B :

$$B = \frac{\sum_{i=1}^n f(A_i) A_i}{\sum_{i=1}^n f(A_i)} \quad (19)$$

5.2 Case $\|\mathbb{B}\| = m > 1$

We have to find such a set of polygonal models $\mathbb{B} = (B_1, B_2, \dots, B_m)$ with m elements for which the expression in formula 4 is minimal.

$$\mathbb{B} = \arg \min_{\mathbb{B}; \|\mathbb{B}\|=m} (\rho_f(\mathbb{A}, \mathbb{B})) \quad (20)$$

We use a dynamic programming approach:

Let $minDis[\mathbb{T}, p]$ is an array of real numbers indexed by a subset $\mathbb{T} \subset \mathbb{A}$ and an integer $p \in \{1 \dots m\}$ defined as:

$$minDis[\mathbb{T}, p] := \min_{\mathbb{U}; \|\mathbb{U}\|=p} (\rho_f(\mathbb{T}, \mathbb{U})) \quad (21)$$

This array represents the distance for all subsets of \mathbb{A} to its optimal reductions of size p . If we are able to fill the array, we can find the answer to our problem in the field $minDis[\mathbb{A}, m]$. We describe an algorithm to fill the array $minDis[\mathbb{T}, p]$ with values. For $p = 1$ we can use the equation (19).

$$minDis[\mathbb{T}, 1] = \rho_f(\mathbb{T}, \left\{ \frac{\sum_{i=1}^n f(T_i) T_i}{\sum_{i=1}^n f(T_i)} \right\}) \quad (22)$$

Now we can increase the value of p step-by-step and compute the values of remaining fields of the array $minDis$. We try to find a subset $\mathbb{V} \subset \mathbb{T}$ that is reduced to a single mesh during the optimal reduction. The reduction is optimal if the sum of reduction of \mathbb{V} to one mesh and reduction of $\mathbb{T} \setminus \mathbb{V}$ to $p - 1$ meshes is minimal.

$$minDis[\mathbb{T}, p] = \min_{\mathbb{V} \subset \mathbb{T}} (minDis[\mathbb{V}, 1] + minDis[\mathbb{T} \setminus \mathbb{V}, p - 1]) \quad (23)$$

Using the algorithm above we can compute the dissimilarity during the optimal reduction. We can find the set \mathbb{B} itself easily by making notes about the performed reductions (found sets \mathbb{V}) during the algorithm.

The time complexity of the algorithm is $O(n2^n \|V\| + 4^n m)$. The spacial complexity of the algorithm is $O(n \|V\| + 2^n m)$. The algorithm is exponential to n . It is not a principal drawback because the values of n and m are small (e.g. $n = 16$, $m = 10$) and we use this reduction only once for each set of models.

6 IMPLEMENTATION

We have implemented the algorithm in Java. For our measurement we used a computer with Intel Core Duo processor T8300 2.4GHz with 2 GB of RAM. (Our implementation is single thread only.) We measured the time needed to reduce 16 visemes to 10 visemes. Each of these visemes was represented by a polygonal model with 3000 triangles. Initial reductions for the case $p = 1$ took 2 minutes and 43 seconds. Dynamic programming reductions for the case $p > 1$ took 2 minutes and 23 seconds. Input/output operations took 12 seconds. The total time was 5 minutes and 18 seconds.

```

input A
input  $f(A_1), f(A_2) \dots f(A_n)$ 
input  $m$ 
for  $\mathbb{T} \subset \mathbb{A}$  do
   $\text{minDis}[\mathbb{T}, 1] := \rho_f(\mathbb{T}, \frac{\sum_{i=1}^n f(T_i)T_i}{\sum_{i=1}^n f(T_i)})$ 
for  $p := 2$  to  $m$  do
  for  $\mathbb{T} \subset \mathbb{A}$  do
     $\text{currentMinDistance} := \infty$ 
    for  $\mathbb{V} \subset \mathbb{T}$  do
       $\text{distance} := \text{minDis}[\mathbb{V}, 1] +$ 
         $\text{minDis}[\mathbb{T} \setminus \mathbb{V}, p - 1]$ 
      if  $\text{distance} < \text{currentMinDistance}$  then
         $\text{currentMinDistance} := \text{distance}$ 
       $\text{minDis}[\mathbb{T}, p] := \text{currentMinDistance}$ 
output  $\text{minDis}[\mathbb{A}, m]$ 

```

Algorithm 1: Algorithm for optimal mesh reduction

We use VRML (Virtual Reality Markup Language) as our input and output format for polygonal meshes. The output from our application is compatible with XFaceEd face editor proposed by Balci in [3].

7 PERFORMANCE VALIDATION

We have compared animation of a head with unreduced set of 16 visemes and the same head with reduced set of 10 visemes. We used a textured head model with 3000 triangles exported from FaceGen [19] for our measurements and Windows Mobile phone HTC Touch Pro with OpenGL ES[9] support. An application with unreduced model required 18 seconds for startup, an application with the reduced model required only 8 seconds for startup. The speed of the model animation was 5.4 FPS for the unreduced and 12.2 FPS for the reduced version. The unreduced version was likely slowed down by memory swapping. The animation of the reduced version appeared much more smooth.

8 CONCLUSION AND FUTURE WORK

The presented method primary focusses on the head animation but it is general enough for use in other animation techniques using polygonal mesh interpolation (e.g. body, animals). In our work, we intend to investigate further reduction techniques as part of our ongoing effort of designing an open platform for development of talking-head applications on mobile phones (using the XFace framework developed by Balci [2, 4]).

ACKNOWLEDGEMENTS

This research has been partially supported by the MSMT under the research program MSM 6840770014, the research program LC-06008 (Center for Computer Graphics) and by Vodafone Foundation Czech Republic.

REFERENCES

- [1] Marc Alexa, Uwe Berner, Michael Hellenschmidt, and Thomas Rieger. An animation system for user interface agents. In *Proceedings of WSCG 2001*, 2001.
- [2] Koray Balci. Xface: Mpeg-4 based open source toolkit for 3d facial animation. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 399–402, New York, NY, USA, 2004. ACM.
- [3] Koray Balci. Xfaceed: authoring tool for embodied conversational agents. In *ICMI '05: Proceedings of the 7th international conference on Multimodal interfaces*, pages 208–213, New York, NY, USA, 2005. ACM.
- [4] Koray Balci, Elena Not, Massimo Zancanaro, and Fabio Pianesi. Xface open source project and smil-agent scripting language for creating and animating embodied conversational agents. In *ACM Multimedia*, September 2007.
- [5] Uwe Berner. Optimized face animation with morph-targets. *Journal of WSCG 2004*, 12, 2004.
- [6] Foreignword. English-Truespel (USA Accent) Text Conversion Tool. <http://www.foreignword.com/dictionary/truespel/transpel.htm>.
- [7] ISO/IEC 14496-1:1999. *Information technology – Coding of audio-visual objects – Part 1: Systems*. ISO, Geneva, Switzerland.
- [8] ISO/IEC 14496-2:1999. *Information technology – Coding of audio-visual objects – Part 2: Visual*. ISO, Geneva, Switzerland.
- [9] Khronos Groups. OpenGL ES - The Standard for Embedded Accelerated 3D Graphics. <http://www.khronos.org/opengles/>.
- [10] Mike Krus, Patrick Bourdot, Françoise Guisnel, and Guillaume Thibault. Levels of detail & polygonal simplification. *Crossroads*, 3(4):13–19, 1997.
- [11] Ladislav Kunc, Pavel Slavik, and Jan Kleindienst. Talking head as life blog. In *Text, Speech and Dialogue*, Lecture Notes in Computer Science, pages 365–372, 2008.
- [12] Zuo Li, LI Jin-tao, and Wang Zhao-qi. Anatomical human musculature modeling for real-time deformation. *Journal of WSCG 2003*, 11, 2003.
- [13] Radoslaw Niewiadomski, Elisabetta Bevacqua, Maurizio Mancini, and Catherine Pelachaud. Greta: an interactive expressive eca system. In *AAMAS '09: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 1399–1400, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.
- [14] Igor S. Pandzic and Robert Forchheimer, editors. *MPEG-4 Facial Animation: The Standard, Implementation and Applications*. John Wiley & Sons, Inc., New York, NY, USA, 2003.
- [15] W. Pasman and F. W. Jansen. Scheduling level of detail with guaranteed quality and cost. In *Web3D '02: Proceedings of the seventh international conference on 3D Web technology*, pages 43–51, New York, NY, USA, 2002. ACM.
- [16] Kari Pulli, Jani Vaarala, Ville Miettinen, Robert Simpson, Tomi Aarnio, and Mark Callow. The mobile 3d ecosystem. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, page 1, New York, NY, USA, 2007. ACM.
- [17] Thomas Rieger. Avatar gestures. *Journal of WSCG 2003*, 11:379–386, 2003.
- [18] Eftychios Sifakis, Andrew Selle, Avram Robinson-Mosher, and Ronald Fedkiw. Simulating speech with a physics-based facial muscle model. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 261–270, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.
- [19] Singular Inversion. FaceGen. www.facegen.com.

Multi-Level Hashed Grid Construction Methods

Vasco Costa
INESC-ID / IST
Lisboa, Portugal
vasc@vimmi.inesc-id.pt

João Pereira
INESC-ID / IST
Lisboa, Portugal
jap@vimmi.inesc-id.pt

Joaquim Jorge
INESC-ID / IST
Lisboa, Portugal
jaj@vimmi.inesc-id.pt

ABSTRACT

Ray tracing is an inherently parallel visualization algorithm. However to achieve good performance, at interactive frame rates, an acceleration structure to decrease the number of per ray primitive intersections is required. Grid acceleration structures have some of the fastest build times, with $O(N)$ complexity, but traditionally achieved this at a high memory cost. Recent research has reduced the memory footprint by employing compression for one-level grids. Render time performance can be improved using multi-level grids. We describe two methods for building such multi-level grids. In the first method we employ a recursive compressed grid in which grid cells are adaptively subdivided in a variable fashion. The second method uses a finely divided compressed grid, with a lower resolution macrocell overlay to speed up traversal. We analyze the performance of these new algorithms, which enable improved render times, versus existing solutions.

Keywords

Ray tracing, spatial subdivision, grid.

1. INTRODUCTION

Realtime ray tracing is an active area of research [Wal07]. Even traditionally skeptical hardware vendors have recently demonstrated, or made available, realtime ray tracing solutions [Sei08]. Ray tracing is desirable for several reasons, namely per pixel accurate shadows, reflections and refractions. It can also be used as a base for other global illumination algorithms such as path tracing, and photon mapping, to add more effects such as caustics and diffuse interreflections.

In the naive ray tracing algorithm, it is necessary to search the nearest intersected primitive for each ray. Without an acceleration structure, the complexity for such an algorithm is $O(N)$, where N is the number of primitives in the scene. Hence to enable realtime ray tracing for complex scenes, with many primitives, acceleration structures are used. These acceleration structures can theoretically reduce per ray complexity to $O(\log N)$.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Ideally an acceleration structure should be fast to build and use as little memory space as possible, while still delivering good render time performance.

This work describes our efforts to combine the desirable traits of multi-level grid [Jev89,Wal06] render time performance, with the low build time and memory consumption characteristics of row displacement compression [Lag08].

Existing related work in this area is surveyed in Section 2. Section 3 describes the proposed multi-level grid construction methods. The performance results of these methods are analyzed in Section 4. Finally conclusions are presented in Section 5.

2. RELATED WORK

Grid acceleration structures for ray tracing were first described by Fujimoto et al. [Fuj89]. These acceleration structures subdivide 3D space in near cubical cells. It was found that grids, by eliminating vertical traversal time costs present in other acceleration structures popular at the time, had increased overall render time performance. 3DDA, a 3D extension of the raster line drawing algorithm, was employed for ray grid traversal.

An improved grid traversal algorithm was later near simultaneously devised by several researchers [Woo87,Cle88]. This algorithm is still employed today. The historical grid ray tracing acceleration structures around this period are described by Havran et al. [Hav99]. Grid dimensions ($M_x \times M_y \times M_z$) are determined based on heuristics related to the number

of scene primitives, scene bounding box, and certain constant factors.

Recently Lagae and Dutré [Lag08] employed grid row displacement compression (i.e. hashing) to reduce the memory footprint of this kind of acceleration structure. It does this by compressing empty cells. By allocating all memory, before inserting primitives into the data structure, build time performance was also improved. The render time performance of this one-level grid algorithm is however inferior to non-compressed multi-level algorithms, such as the rgrid used by the Manta ray tracer [Big06], as shall be seen in Section 4.

Kim et al. [Kim09] have created compressed versions of the bounding volume hierarchy (BVH) acceleration structure, one of the acceleration structures first used in ray tracing. Kim et al. also compress the triangle mesh and page data to the disk providing increased memory savings.

BVH acceleration structures have higher construction time complexity than grids. BVH construction complexity is $O(N \log N)$ versus a grid construction complexity of $O(N)$.

More recent, faster to build, grid acceleration structures have many advantages. However further work is necessary to improve their render time performance. This work aims at filling this gap.

3. METHODS

The classification of multi-level grid construction methods employed here is based on that of Jevans and Wyvill [Jev89].

Variable construction methods recursively subdivide the grid, by employing subgrids in each cell. Subgrid dimensions are chosen using a similar heuristic to that employed for the first cell division level. Memory consumption is hard to predict, usually leading to the use of dynamic memory allocation along the construction method.

Fixed construction methods use a fixed ratio, finer subdivision than a regular one-level grid would employ. Since the total size of a grid acceleration structure can be known in advance, all memory allocation can be done before the method is employed. A fixed construction grid can be build using macrocells for the lower resolution levels.

Fixed construction methods have good performance for uniformly distributed scenes, such as laser scanned models. Variable construction methods adapt more easily to varying scene primitive distribution but at increased memory consumption and build time costs.

The following heuristic, attributed to Woo, is employed to determine grid dimensions:

$$M_i = \frac{S_i}{\max\{S_i\}} \sqrt[3]{\rho N} \quad (i \in \{x, y, z\})$$

Equation 1. Woo's heuristic. S_i is the scene bounding box size in dimension i , ρ is 4.

Via profiling we noticed some characteristics in the existing algorithms [Lag08, Big06] described at Section 2. Grid traversal dominates render time, and one-level grids spend a lot more time doing ray/triangle intersections than multi-level grids. In attempting to improve render-time performance we posed the following hypothesis: we can reduce the number of ray/triangle intersections by using smaller cells, with fewer triangles per cell. To reduce traversal time we can employ a multi-level structure to skip empty cells in larger steps.

3.1. Multi-Level Variable Hashed Grid

This subsection describes the multi-level variable hashed grid implementation. It is a recursive grid, with the top level grid and subgrids using the hashed grid [Lag08] algorithm. This grid has a maximum grid depth size of 2.

First the top level hashed grid is built using the algorithm described by Lagae et al. [Lag08] but using the heuristic from Equation 1. We selected a grid density ρ of 4 since it empirically provided good render time performance. Each cell of this top level grid is then subdivided using the same algorithm, creating a new subgrid, for each cell containing more than a certain number of primitives.

3.2. Multi-Level Fixed Hashed Grid

In this subsection a multi-level fixed hashed grid is described. It is a high resolution hashed grid [Lag08] with multi-level macrocells [Wal06] to speedup traversal.

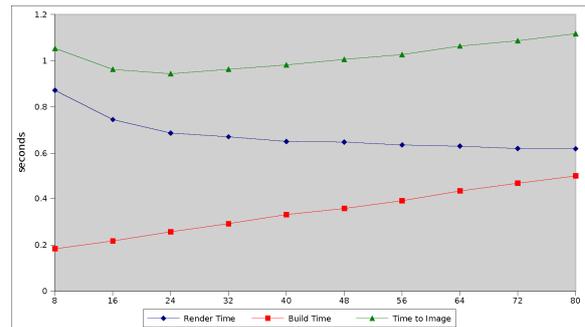


Figure 1. Timings for the Buddha scene according to grid density.

First a finely divided one-level hashed grid is built in a similar fashion to that of Lagae et al. [Lag08], but using the grid heuristic described in Equation 1 with a high grid density parameter to reduce cell size.

					
	Bunny	Dragon	Buddha	Asian Dragon	Thai Statue
Scene statistics					
# triangles	69.45K	871.41K	1.09 M	7.22 M	10 M
memory	1.2MB	15.0MB	18.7MB	123.9MB	171.7MB
Manta recursive grid [Big06]					
Primitive intersections/ray	1.58	1.58	1.56	0.91	1.17
Cell traversals/ray	4.73	5.80	4.95	6.44	7.00
Grid traversals/ray	1.38	1.28	1.17	0.72	0.82
Build Time (s)	0.47	3.46	4.50	20.44	29.59
Render Time (s)	0.30	0.52	0.34	0.36	0.58
Time to Image (s)	0.78	3.98	4.84	20.80	30.17
One-level hashed grid [Lag08]					
Primitive intersections/ray	8.35	9.87	9.53	13.15	12.67
Cell traversals/ray	14.53	35.23	26.93	93.14	100.76
Grid traversals/ray	0.00	0.00	0.00	0.00	0.00
Build Time (s)	0.02	0.22	0.26	1.48	2.07
Render Time (s)	0.58	0.89	0.78	1.60	1.80
Time to Image (s)	0.60	1.11	1.04	3.09	3.88
Multi-level variable hashed grid					
Primitive intersections/ray	3.99	3.83	3.92	1.93	2.63
Cell traversals/ray	15.12	26.05	17.21	68.31	69.38
Grid traversals/ray	0.54	0.53	0.53	0.27	0.36
Build Time (s)	0.09	0.75	0.81	4.09	6.29
Render Time (s)	0.51	0.64	0.55	1.00	1.11
Time to Image (s)	0.60	1.39	1.36	5.10	7.39
Multi-level fixed hashed grid					
Primitive intersections/ray	6.14	8.26	10.06	8.74	9.06
Cell traversals/ray	14.04	17.86	13.10	29.97	27.31
Grid traversals/ray	0.57	0.47	0.45	0.24	0.25
Build Time (s)	0.04	0.39	0.29	3.09	3.45
Render Time (s)	0.57	0.68	0.67	0.79	0.82
Time to Image (s)	0.61	1.07	0.97	3.88	4.27

Table 1. Scene triangle mesh statistics, render time profile results, timings for the studied grid acceleration structures.

We empirically chose the grid density parameter by analyzing the behavior for the Buddha scene as can be seen in Figure 1. We selected a grid density ρ of 32 since it features adequate render time without having a severe impact on time to image.

Next multi-level macrocells [Wal06], are built to skip empty cells in larger steps during traversal. Macrocells overlay a coarser grid over the finely divided grid. The macrocells for each level consist of a 3D bit array with information if a region of space is empty or not. To speed up this construction step macrocells are downscaled by a factor S of 6 on each extent. We arrived at this value by empirically analyzing algorithm behavior for the tested scenes. Wald et al. [Wal06] reached the same value with a

different heuristic and test scenes. Macrocell downscaling can be done with a quick 3D bitmap scaling operation.

4. PERFORMANCE AND RESULTS

This section evaluates the performance of the grid construction methods.

All tests were performed on a single Intel Core 2 Duo processor at 3 GHz. The machine has 4GB of RAM running the Linux operating system. The algorithms were implemented in C++ using STL and Boost without use of assembly or intrinsics.

Only a single thread was used, with one ray per pixel and diffuse shading, at 1024×1024 resolution. A

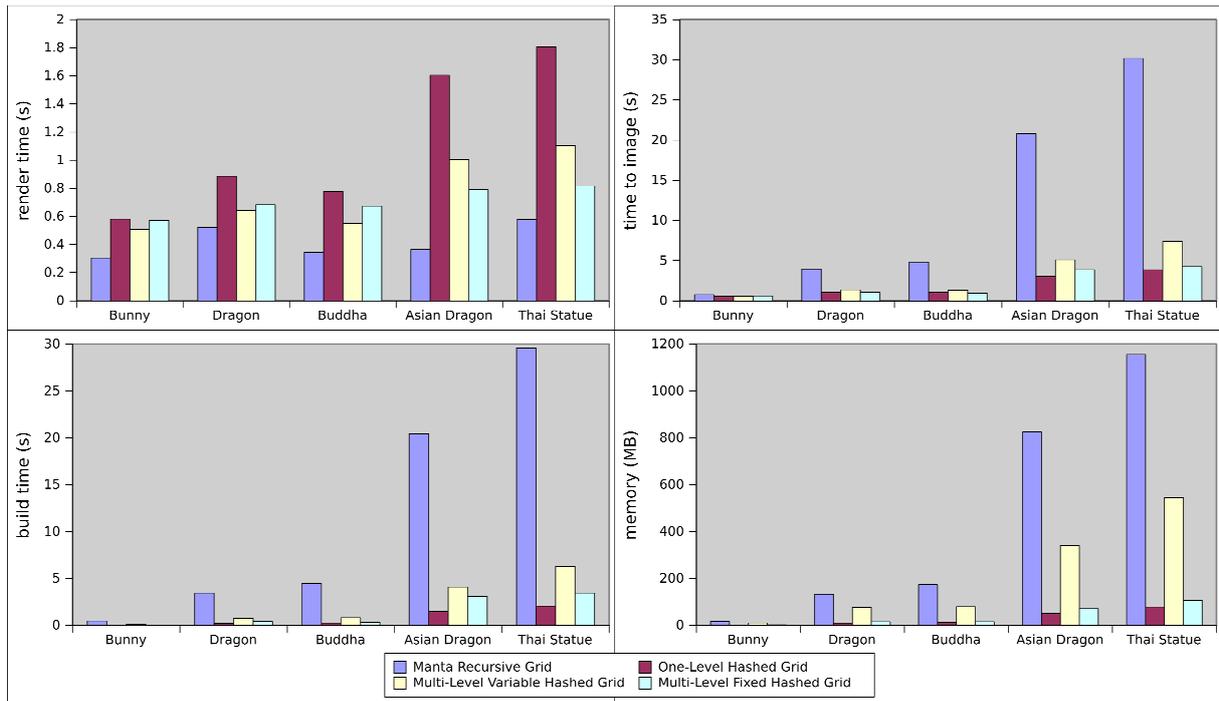


Figure 2. From bottom right clockwise: memory consumption; build time; render time; time to image acceleration structure statistics for the tested scenes.

variety of models from the Stanford 3D Scanning Repository were used for the evaluation.

The top of Table 1 shows scene statistics such as number of triangles, memory used by the triangles. These scenes were chosen because the system is expected to support visualization of laser scanned architectural models. Scene memory usage is computed by using 12 bytes per triangle to store vertex index information (three machine words for each vertex index), plus 12 bytes per vertex (three floating point numbers for each coordinate). This provides reduced memory usage in an expedient fashion. Ray/triangle intersection was done using the Möller-Trumbore [Mol05] intersection algorithm because of its low memory requirements.

For performance comparison purposes with existing published algorithms the recursive grid from the Manta interactive ray tracer [Big06] was tested. An implementation of the hashed grid algorithm by Lagae and Dutré [Lag08] was added to the system to serve as the one-level compressed grid baseline.

The multi-level hashed grid structures feature improved render time performance compared to the one-level hashed grid. This is markedly so for the larger scenes where over twice the render time performance is achieved. Of the multi-level hashed grid methods, the fixed hashed grid is better for the larger scenes, as can be seen at top left in Figure 2. Fixed grid features improved render times, versus the variable grid, due to several factors: the fixed grid has a smaller memory footprint (and increased

memory coherence); the cells of the top hierarchical level of the fixed grid have a larger volume, skipping empty regions of space faster, this is reflected in the cell traversals/ray.

The recursive grid from Manta has even better render time performance, although the performance difference varies according to the tested scene.

These performance results required a more in depth examination by profiling the acceleration structures in terms of number of primitive intersections, horizontal cell traversals and vertical grid traversals.

Profiling, seen in Table 1, shows improved Manta render time performance is due to the lower number of ray/primitive intersections and horizontal cell traversals used by the recursive grid to display the same scene.

Manta employs a deeper variable grid structure with maximum depth of 3 and has a modified heuristic. This enables improved render time performance but comes at a big build time penalty. It takes six times longer to build the acceleration structure for the Thai Statue scene for example as can be seen at the bottom left of Figure 2.

Memory usage paints a similar picture to the build time statistics. The Thai Statue scene uses around ten times more memory in the non-compressed Manta multi-level acceleration structure versus the fastest compressed multi-level acceleration structure we implemented.

The compressed multi-level grid acceleration methods of note feature much improved performance on the figures of merit. Time to first image in particular is much improved versus the times achieved by Manta using algorithms of the same class. The multi-level fixed hashed grid has a similarly low time to image compared to the one-level hashed grid. This makes it the best option among the multi-level grids for the tested scenes.

5. CONCLUSION

Multi-level compressed grid methods achieve best of class performance by combining the desirable traits from existing algorithms: low memory requirements, fast build and render times. The algorithms presented here could still use some work in the heuristics, as the multi-level heuristic from Manta has quicker render times. There is also room for expansion in improving the number of cell traversals and primitive intersections per ray. Alternative methods for speeding up traversal time by skipping empty voxels, not studied in this work, include proximity clouds [Coh94], macro-regions [Dev89], and similar directional techniques [Sem97].

We would also like to implement these algorithms on GPUs to investigate the performance characteristics of compressed structures on that hardware class.

6. ACKNOWLEDGEMENTS

It would not have been possible to make the tests in this work without the models from the Stanford 3D Scanning Repository.

This work was supported by the Portuguese Foundation for Science and Technology project VIZIR (PTDC/EIA/66655/2006).

7. REFERENCES

- [Big06] J. Bigler, A. Stephens and S. G. Parker. Design for Parallel Interactive Ray Tracing Systems Proceedings of the IEEE Symposium on Interactive Ray Tracing, 2006.
- [Coh94] D. Cohen, and Z. Sheffer. Proximity clouds - an acceleration technique for 3D grid traversal. *The Visual Computer*, 11(1): 27–38, 1994.
- [Cle88] J. Cleary and G. Wyvill. Analysis of an algorithm for fast ray tracing using uniform space subdivision. *The Visual Computer*, 4(2):65–83, 1988.
- [Dev89] O. Devillers. The macro-regions: an efficient space subdivision structure for ray tracing. In *Eurographics '89*, pages 27–38, 1989.
- [Fuj89] A. Fujimoto, T. Tanaka, and K. Iwata. Arts: Accelerated ray-tracing system. *Computer Graphics and Applications*, IEEE, 6(4):16–26, 1986.
- [Jev89] D. Jevans and B. Wyvill. Adaptive voxel subdivision for ray tracing. In *Graphics Interface '89*, pages 164–172, June 1989.
- [Hav99] V. Havran, F. Sixta, and S. Databases. Comparison of hierarchical grids. *Ray Tracing News*, 12(1):1–4, 1999.
- [Lag08] A. Lagae and P. Dutré. Compact, fast and robust grids for ray tracing. *Computer Graphics Forum (Proceedings of the 19th Eurographics Symposium on Rendering)*, 27(8), 2008.
- [Kim09] Tae-Joon Kim, Bochang Moon, Duksu Kim, Sung-Eui Yoon. RACBVHs: Random-Accessible Compressed Bounding Volume Hierarchies. *IEEE Transactions on Visualization and Computer Graphics*, 17 Jun. 2009.
- [Mol05] T. Möller and B. Trumbore. Fast, minimum storage ray/triangle intersection. In *International Conference on Computer Graphics and Interactive Techniques*. ACM Press New York, NY, USA, 2005.
- [Sei08] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugeran, R. Cavin, R. Espasa, E. Grochowski, T. Juan, and P. Hanrahan. Larrabee: a many-core x86 architecture for visual computing. *ACM SIGGRAPH*, 2008.
- [Sem97] S.K. Semwal, and H. Kvanstrom. Directed Safe Zones and the Dual Extent Algorithms for Efficient Grid Traversal during Ray Tracing. In *Graphics Interface '97*, pages 76–87, May 1997.
- [Wal06] I. Wald, T. Ize, A. Kensler, A. Knoll, and S. Parker. Ray tracing animated scenes using coherent grid traversal. In *International Conference on Computer Graphics and Interactive Techniques*, pages 485–493. ACM Press New York, NY, USA, 2006.
- [Wal07] I. Wald, W. Mark, J. Gunther, S. Boulos, T. Ize, W. Hunt, S. Parker, P. Shirley. State of the art in ray tracing animated scenes *Eurographics 2007 State of the Art Reports*, 2007.
- [Woo87] J. Amanatides and A. Woo. A fast voxel traversal algorithm for ray tracing. In *Eurographics '87*, pages 3–10, 1987.

Interactive Image-space Point Cloud Rendering with Transparency and Shadows

Petar Dobrev

Paul Rosenthal

Lars Linsen

Jacobs University, Bremen, Germany

{p.dobrev, p.rosenthal, l.linsen}@jacobs-university.de

ABSTRACT

Point-based rendering methods have proven to be effective for the display of large point cloud surface models. For a realistic visualization of the models, transparency and shadows are essential features. We propose a method for point cloud rendering with transparency and shadows at interactive rates. Our approach does not require any global or local surface reconstruction method, but operates directly on the point cloud. All passes are executed in image space and no pre-computation steps are required. The underlying technique for our approach is a depth peeling method for point cloud surface representations. Having detected a sorted sequence of surface layers, they can be blended front to back with given opacity values to obtain renderings with transparency. These computation steps achieve interactive frame rates. For renderings with shadows, we determine a point cloud shadow texture that stores for each point of a point cloud whether it is lit by a given light source. The extraction of the layer of lit points is obtained using the depth peeling technique, again. For the shadow texture computation, we also apply a Monte-Carlo integration method to approximate light from an area light source, leading to soft shadows. Shadow computations for point light sources are executed at interactive frame rates. Shadow computations for area light sources are performed at interactive or near-interactive frame rates depending on the approximation quality.

Keywords: point-based rendering, shadows, transparency

1 INTRODUCTION

Ever since the emergence of 3D scanning devices, surface representation and rendering of the scanned objects has been an active area of research. Acquiring consistent renderings of the surfaces is not trivial as the output of the scanning processes are point clouds with no information about the connectivity between the points. Several techniques have been developed to remedy this problem, ranging from global and local surface reconstruction to methods entirely operating in image space. Traditional approaches involve the generation of a triangular mesh from the point cloud, e.g. [3], which represents a (typically closed) manifold, and the subsequent application of standard mesh rendering techniques for display. Such global surface reconstruction approaches, however, scale superlinearly in the number of points and are slow when applied to the large datasets that can be obtained by modern scanning devices.

This observation led to the idea of using local surface reconstruction methods instead. Local surface reconstruction methods compute for each point a subset of neighboring points and extend the point to a local surface representation based on plane or surface fitting to its neighborhood [1]. The point cloud rendering is, then, obtained by displaying the (blended) extensions.

The local surface reconstruction itself is linear in the number of points, but it relies on a fast and appropriate computation of a neighborhood for each point in a pre-computation step. The speed and quality of the approach depends heavily on the choice of the neighborhood.

As the number of points increases, the surface elements tend to shrink and when projected to the image plane have nearly pixel size. This observation was already made by Grossman and Dally [6], who presented an approach just using points as rendering primitives and some image-space considerations to obtain surface renderings without holes. Recently, this image-space technique has been re-considered and improved [8, 11, 13]. This method has the advantage that no surface reconstruction is required and that all image-space operations can efficiently be implemented on the GPU, utilizing its speed and parallelism. It only assumes points (and a surface normal for appropriate illumination). Our approach builds upon the ideas of Rosenthal and Linsen [11]. The image-space operations for transforming a projected point cloud to a surface rendering include image filters to fill holes in the projected surface, which originate from pixels that exhibit background information or occluded/hidden surface parts, and smoothing filters. The contribution of this paper is to provide transparency and shadow capabilities for such point cloud renderings at high frame rates using a *depth peeling* technique.

Depth peeling is a multi-pass technique used to extract (or “peel”) layers of surfaces with respect to a

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

given viewpoint from a scene with multiple surface layers. While standard depth testing in image space provides the nearest fragments of the scene (i.e., the closest layer), depth peeling with n passes extracts n such layers. We describe our depth peeling approach for point cloud surface representations in Section 3.

The information extracted by the depth peeling approach can be put to different applications. We exploit this information for enhancing the capabilities of interactive point cloud renderings with transparency and (soft) shadows. To achieve the first goal, we developed a method for order-independent transparency computation described in Section 4. Once the depth peeling approach has acquired the surface layers, they are blended with object-specific opacity values in the order of their acquisition. This approach allows for rendering of multiple surfaces in one scene using different opacity values for each.

Our second goal was the shadow computation in scenes with point cloud surface representations and the interactive rendering of such scenes. To determine lit and unlit regions of the scene, one has to determine, which points are visible from the light source and which are not. This can be done by rendering the scene with the viewpoint being the position of the light source. In this setting, all those points that are visible can be marked as lit. This approach assumes that we apply the image-space rendering approach with the filters that remove occluded surface parts. The result can be stored in form of a point cloud shadow texture. However, since the scene is typically composed of a large number of points, it is more than likely that multiple visible points project to the same pixel such that marking only one of those points as lit would result in an inconsistent shadow texture. To extract and mark multiple lit points that project to the same pixel, we apply the depth peeling technique, again. Once all lit points have been marked, the scene is rendered from the viewpoint of the observer, where the unlit points are rendered without diffuse or specular lighting, i.e., only using ambient light. To create soft shadows and alleviate aliasing artifacts, we use a Monte-Carlo integration method to approximate light intensity from an area light source. Details are given in Section 5.

The GPU implementation of the algorithms allows us to achieve interactive rates for layer extraction, transparent renderings, and renderings of scenes with (soft) shadows. Results of all steps are presented in Section 6.

2 RELATED WORK

An effective way to incorporate transparency and/or shadows to point-based rendering is the use of ray tracing methods as introduced by Schaufler and Jensen [12]. However, such approaches are typically far from achieving interactive frame rates. The only

interactive ray tracing algorithm of point-based models was introduced by Wald and Seidel [14], but they restricted themselves to scenes with shadows, i.e., transparency is not supported. The original EWA splatting paper [16] presents a method for transparency utilizing a software multi-layered framebuffer with fixed number of layers per pixel. Zhang and Pajarola [15] introduced the *deferred blending* approach, which requires only one geometry pass for both visibility culling and blending. They also propose an extension how to use this approach to achieve order-independent transparency with one geometry pass.

An approach to incorporate shadows into interactive point-based rendering can be obtained in a straight-forward manner when first reconstructing the surface from the point cloud (globally or locally) and subsequently apply standard shadow mapping techniques [4]. Botsch et al. [2] applied shadow maps to EWA splatting using GPU implementation to achieve interactive rates. Guennebaud and Gross [7] presented another local surface reconstruction technique, employing moving least squares fitting of algebraic spheres, and also applied shadow mapping to it.

The shadow computation in our approach is similar to irradiance textures (also known as “pre-baked” lighting) in mesh-based rendering [10, 9]. Lit surfaces are determined and stored in a texture by rendering the scene with the viewpoint being the position of the light source. In the rendering pass this information is used to determine which surfaces should be drawn in shadow, and which not.

3 DEPTH PEELING

Depth peeling was introduced by Everitt [5] and is a technique to partition a static 3D scene into sorted layers of geometry. As the name suggests, the layers are extracted in an iterative fashion by “peeling” off one layer after another. The sorting is induced by the given viewpoint. Hence, in each iteration the fragments of the projected visible scene are determined, stored as a representation of the current layer, and removed to compute the subsequent layers. Figure 1 illustrates the depth peeling idea. The depth peeling technique is im-

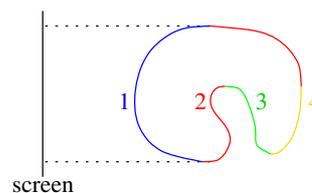


Figure 1: 2D illustration of depth peeling: visible layers of geometry are extracted from front to back. First layer is shown in blue, second in red, third in green, and fourth in yellow.

plemented in a multi-pass algorithm, i.e., to extract n layers the whole scene has to be rendered n times. Each rendering pass is performed with enabled depth testing such that the points closest to the viewpoint and their distances to the viewer are recorded. For the second up to the n th pass, only those points are rendered, whose distance to the viewer is greater than the distance recorded in the preceding pass.

As we want to avoid any (global or local) object-space surface reconstruction, we apply the depth peeling technique to scenes consisting of points only. Consequently, each layer is represented as a set of projected points. Depending on the sampling rate that has been used to acquire the surface, the screen resolution, and the distance to the viewer, it may happen that the points projected to the image plane do not cover all the screen pixels that a reconstructed surface would. Hence, the surface layer may exhibit holes where the background or points of hidden surface layers become visible. Figure 2 illustrates this effect for a 2D scene that is projected to a 1D screen consisting of five pixels. The projection of the first surface layer (blue points) should cover the entire screen. However, there are pixels to which no blue point is mapped. Instead, the second surface layer (red color) or even the background of the scene (grey color) is visible. These gaps in the surface representation of the first layer need to be filled appropriately. Of course, the same issue may arise for all other extracted layers. Hence, in each rendering pass, we apply image-space operations to the extracted layer to fill the gaps in the surface. The image-space opera-

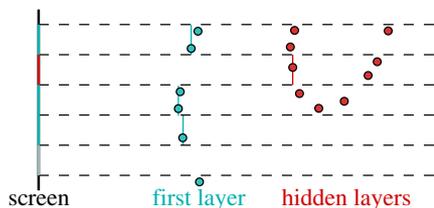


Figure 2: When projecting first layer (blue) in point cloud representation to the screen, the layer exhibits holes such that hidden layers (red) or the background (grey) become visible.

tions are executed on the rendering texture using depth information stored in the depth buffer. The operations are executed in four steps: filling surface gaps in form of background pixels (grey pixel in Figure 2), filling surface gaps in form of occluded pixels (red pixel in Figure 2), smoothing the image for an improved rendering quality of the extracted layer, and anti-aliasing applied to the silhouettes and feature lines in the resulting image.

To fill holes caused by pixels exposing background information, one has to identify which background pixels represent holes in the surface layer and which do not. To determine reliably which pixels are to be filled

and which not, we apply a filter that checks the 3×3 neighborhood of each background pixel against the set of masks shown in Figure 3. In Figure 3, the framed pixel is the candidate to be filled and the bright ones are neighboring background pixels. The dark pixels may be background or non-background pixels. If the neighborhood matches any of the configurations, the pixel is not filled. Otherwise, its color and depth information is replaced by the color and depth information of the pixel with smallest depth within the stencil of the mask, i.e., within the 3×3 neighborhood. The filters in Figure 3 have been proposed by Rosenthal and Linsen for image-space point cloud rendering. For a detailed discussion of the filters and their application, we refer to the literature [11]. The application of the gap filling step may have to be iterated to fill larger gaps. The operations are always executed on both the rendering texture and the depth texture simultaneously.

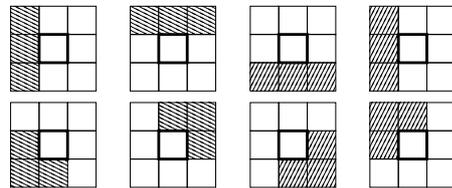


Figure 3: Masks of size 3×3 for detecting pixels exhibiting holes in the projected point cloud surface representation.

To fill pixels that exhibit occluded surface layers, we need to be able to distinguish between pixels from different surface layers. In order to decide whether two pixels belong to the same surface layer, we introduce a parameter d_{min} denoting the minimum distance between two consecutive layers. The parameter depends on the dataset and is typically determined empirically. The occluded pixel filling operation is analogous to the background pixel filling operation. The neighborhood of the candidate pixel is also checked against the masks in Figure 3, only that the bright and the dark pixels in the masks have a different meaning. If the candidate pixel's depth is d , bright pixels correspond to points that have depth values greater than $d + d_{min}$. Dark pixels may have any depth. If the neighborhood satisfies any of the masks, the pixel is not changed. Otherwise, its color and depth information is replaced by the color and depth information of the pixel with smallest depth within the stencil of the mask. Also this second gap filling step may have to be iterated.

To improve the quality of the surface rendering, two additional steps may be applied. The two gap filling steps always replace the gap with the information from the pixel closest to the viewer. A weighted average of the information of those neighboring pixels that belong to the same surface layer would have been preferable. As it would have been too cumbersome to detect all those neighbors, a more efficient way to obtain a simi-

lar result is to apply a subsequent smoothing filter. We apply a Gaussian filter of size 3×3 . This smoothing step may be iterated.

However, the smoothing step does not smooth across the silhouette of the projected surface. The silhouettes and feature lines are treated in a separate step that has explicitly been introduced for anti-aliasing purposes. From the depth image, we can easily detect silhouettes and feature lines by checking the depth difference of neighboring pixels against parameter d_{min} (edge detection filtering). All those pixels whose neighborhood exhibit a significant jump in the depth values are marked as contributing to a feature line. To all these pixels, we apply a smoothing that reduces aliasing along the feature lines.

A result of the described pipeline may be seen in Figure 4. We used the Turbine Blade dataset (Data courtesy of Visualization Toolkit) and extracted the first three surface layers. The results have been obtained by applying in each depth peeling pass one iteration of the background pixel filling, occluded pixel filling, Gaussian smoothing, and anti-aliasing.

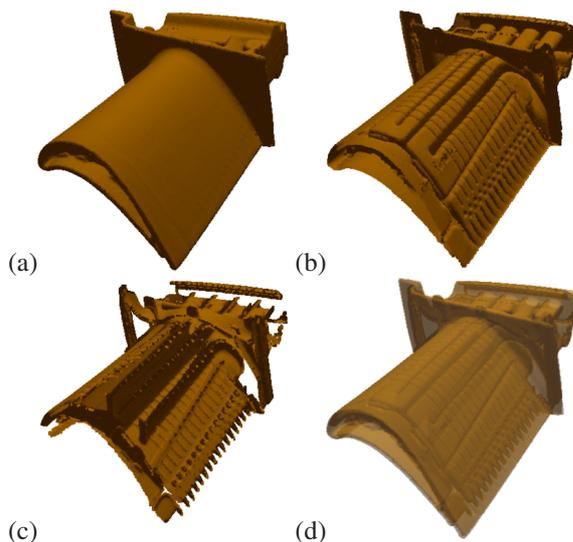


Figure 4: Depth peeling applied to the Blade dataset to extract the (a) first, (b) second, and (c) third layer. The layers are represented as point clouds. The gaps between projected points have been filled using only image-space operations. Blending the layers allows for transparent surface renderings (d).

4 TRANSPARENT SURFACES

Rendering of transparent surfaces is a direct application of depth peeling. It only requires to blend the acquired layers in the order of extraction. However, since point clouds are typically dense, it frequently happens that two or more adjacent points of one surface layer project to the same fragment. Without taking special care of this case, they would be recorded in separate

layers by the depth peeling technique such that consecutive layers contain points that should belong to the same surface layer. Figure 5(a) illustrates this problem in the 2D case. Points of the first surface layer are depicted in blue and of the second surface layer in red. Multiple blue points are mapped to one pixel of the screen.

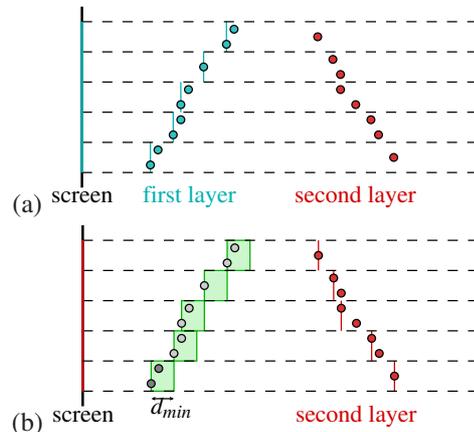


Figure 5: Depth peeling for transparent rendering: (a) first rendering pass records closest points and their depths; (b) second rendering pass again records the closest points and their depths, but ignores points less than d_{min} away from the reference depths obtained in the preceding run.

We tackle this problem by using, again, parameter d_{min} , i.e., the minimum distance between two surface layers, to perform ϵ -z culling: in each rendering pass, depth peeling records the color of the closest point \mathbf{p} for each pixel along with its depth d that serves as a reference for the next run. All points that project to the same pixel as point \mathbf{p} and have a depth less than $d + d_{min}$ must belong to the same surface layer as \mathbf{p} . Figure 5(b) illustrates this idea for the example from Figure 5(a). The green boxes of width d_{min} indicate the area that is considered as one surface layer. Hence, the second depth peeling pass discards all points with depth less than $d + d_{min}$ and correctly detects only points belonging to the second (red) surface layer, see Figure 5(b).

This procedure of skipping points within depth range $[d, d + d_{min}]$ has already been used to generate the three layers of the Blade dataset shown in Figure 4. All that is left to do for point cloud rendering with transparency is to blend the layers front to back with an application-specific opacity value α . The result can be seen in Figure 4(d). The opacity value used for all layers was $\alpha = 0.5$.

5 SHADOW TEXTURES

Point cloud shadow textures are basically Boolean arrays that store which points are lit and which not. Once the shadow texture is determined, lit points are drawn

properly illuminated with ambient, diffuse, and specular reflection components using Phong's illumination model, while unlit points are only drawn using the ambient reflection component. This illumination creates the effect of shadows, as only those points are marked unlit where the light source is occluded by other surface parts.

To determine which points are visible from the light source, we render the scene with the light source's position being the viewpoint with depth testing enabled. All visible points are marked in an array. However, as in Section 4 we observe that, due to the high point density, it is not unusual that several adjacent points of one surface layer project to the same fragment position. The suggested procedure would only mark the closest point for each fragment as lit, which would lead to an inconsistent shadow textures. Figure 6 illustrates the problem for a scene with only one surface layer and no occlusion. The points of the entire surface should be marked as lit. However, due to the described issue, only the closest points (red) are marked as lit, while the others (blue) remain unlit. When observing the scene from a position different from the position of the light source, the unlit points become visible and the rendering exhibits strange shadow patterns.

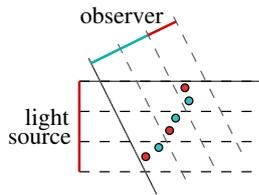


Figure 6: Inconsistent shadow texture in case of high point density: marking only the closest points to the light source as lit, leaves unlit points on the same surface part. The unlit points become visible when positions of observer and light source do not coincide.

Again, depth peeling is the key to solve this problem, but we apply it differently. While for transparent surface rendering our goal was to extract different surface layers, now we want to find all the points that belong to a single surface layer, namely the closest one.

To decide, which points belong to one layer, we consider again parameter d_{min} , i.e., the minimum distance between two surface layers. We render the point cloud from the position of the light source. Let d be the depth of the closest point \mathbf{p} for a given pixel. Then, we consider all points that project to that pixel and have depth values less than $d + d_{min}$ as belonging to the same surface layer as \mathbf{p} . Therefore, we mark them as lit.

However, since depth is measured as the distance to the viewing plane, applying the same offset d_{min} for all points would result in an inconsistent shadow texture. The reason is that the depth of the lit layer should always be taken perpendicularly to the surface, and not

along the viewing direction. In order to account for the change in the offset, we scale d_{min} by a factor that depends on the surface normal. Let \mathbf{v} be the viewing direction and \mathbf{n} be the surface normal in the light source domain. Then, the offset is given by $\Delta d = \frac{d_{min}}{\langle \mathbf{v}, \mathbf{n} \rangle}$. Given that the viewing direction in the light source domain is $(0, 0, -1)$, we obtain that $\langle \mathbf{v}, \mathbf{n} \rangle = -n_z$. To avoid division by zero, this factor is truncated at some maximum value.

As a first step of the algorithm, we obtain the shadow map for the light source, i.e., we record the depth of the closest points as viewed from the light source. As some of the recorded depths might correspond to occluded surface parts, we apply the occluded pixel hole-filling filter on the shadow map. This way pixels, which belong to an occluded surface, will be overwritten in the shadow map and, hence, remain in shadow.

Then, we project all points from the dataset to the light domain and compare their depth values to the ones stored in the shadow map. The points, whose depth is less than the reference depth plus threshold Δd , are recorded as lit in the shadow texture. The rest are left unlit. This operation can very efficiently be implemented on the GPU by using a shader, which takes an array (a texture) of all point positions as input and outputs a boolean array of the same size. The values in the boolean array determine whether the respective point from the input array is lit or not. The shader reads the position of each point from the input texture and projects it in the light domain. Then it compares its depth with the one stored in the shadow map and outputs the result of the comparison to the same texture position as in the input texture.

Figure 7(a) shows a point cloud rendering with shadows applied to the Blade surface shown in Figure 4. It can be observed that the binary marking whether a point is lit or not results in hard shadows with crisp, sharp edges. To create more appealing renderings with softer shadows, we approximate the complex computation of illumination by an area light source using Monte-Carlo integration methods. A number of randomly chosen sample points, lying in the plane perpendicular to the light direction and within the area of the light source, are used as point light sources. A separate shadow texture is computed for each of them. The resulting binary decision values are averaged. The resulting shadow texture is the average of all the shadow textures for the different sample points. It contains no longer just zeros or ones, but floating-point numbers out of the interval $[0, 1]$. These numbers determine to what extent the diffuse and specular components are taken into account.

Let k_a , k_d , and k_s denote the ambient, diffuse, and specular components of the illuminated surface at a specific point. Moreover, let $m \in [0, 1]$ be the value in the shadow texture stored for that particular point.

Then, the surface color at that point is computed as: $c = k_a + m \cdot (k_d + k_s)$. Figure 7(b) shows the result of point cloud rendering with soft shadows using Monte-Carlo integration methods for the scene that has been shown in Figure 7(a). We have used 30 samples to compute the shadow texture. In the lower left of both figures, we provide a zoomed view into a shadow/no-shadow transition region. The shadows appear much softer in Figure 7(b) and their edges are much smoother.

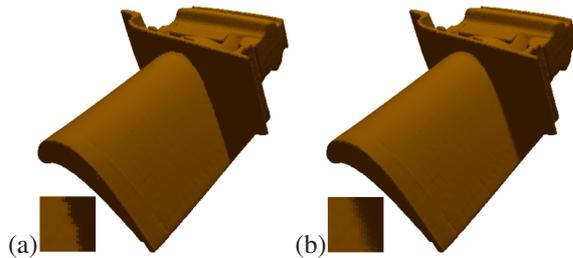


Figure 7: Point cloud rendering with shadows for the Blade dataset: (a) hard shadows using one point light source; (b) soft shadows using Monte-Carlo integration methods with 30 samples to compute the point cloud shadow texture.

6 RESULTS & DISCUSSION

We applied our approach to three types of point cloud data: The model of the Turbine Blade (883k points), given as an example throughout the paper, is from the category of scanned 3D objects. Other datasets from the same category that we have tested our approach on are the Dragon (437k points) and Happy Buddha (543k points) models¹. Although polygonal representations of these objects exist, any information beside the point cloud was not considered. A synthetical dataset we applied our algorithm to is a set of three nested tori (each 2M points). Finally, we tested our method on two point clouds obtained from isosurface extraction: one from an electron spatial probability distribution field referred to as “Neghip”² (128k points) and the other from a hydrogen molecule field³ (535k points for 3 nested isosurfaces).

All results have been generated on an Intel XEON 3.20GHz processor with an NVIDIA GeForce GTX260 graphics card. The algorithms were implemented in C++ with OpenGL and OpenGL Shading Language for shader programming. All images provided as examples or results in the paper have been captured from a 1024×1024 viewport. One iteration of each of the image-space operations described in Section 3, i.e., background pixels filling, occluded pixels filling, smoothing, and anti-aliasing, was used

when producing each rendering. A detailed list of computation times for different datasets, number of layers, number of samples, and resolutions is given in Table 1.

The frame rates for point cloud rendering with local Phong illumination are between 102 fps and 7.8 fps for datasets of sizes between 128k and 6M points and a 1024×1024 viewport. The computation times exhibit a linear behavior in the number of points and a sub-linear behavior in the number of pixels. There is no pre-computation such as local surface reconstruction necessary. All methods directly operate on the point cloud. All operations are done in image space.

For rendering with transparency, the computation times depend linearly on the number of transparent layers. For three transparent surface layers, we obtained frame rates ranging from 28 fps to 2.7 fps. No pre-computations are required. Zhang and Pajarola [15] report better performance for their *deferred blending* approach than depth peeling, but it is only applicable to locally reconstructed surfaces using splats and requires pre-computations. Moreover, it relies on an approximate solution to compute transparency. The frame rates they achieve on an NVidia GeForce 7800GTX GPU are around 37fps for a 303k points dataset and 23 fps for a 1.1M points dataset. As a comparison, our approach renders a 437k points model with 3 layers of transparency at 35fps and a 883k points one at 17.6. Unfortunately, no information about the resolution of the view port used to capture their results is stated to be able to perform a fully adequate comparison.

Figure 8(a) shows a transparent rendering of three nested tori, each drawn with a different color and having a different opacity value. The required number of layers to achieve this kind of rendering is six, such that all surface parts of all three tori are captured and blended. When rendering all six layers of this 6M point dataset, the frame rate drops to 1.3 fps. During navigation it may, therefore, be preferable to render just the first layer.

Figures 8(b) and (c) show examples of how our approach can be applied in the context of scientific visualization. When a scalar field is only known at unstructured points in space, an isosurface can be computed by interpolating between neighboring points. The result is given in form of an unstructured set of points on the isosurface, i.e., a point cloud. The datasets we used actually represent scalar fields defined over a structured grid, but for a proof of concept we re-sampled the datasets at uniform randomly distributed points in space. In Figure 8(b), we extracted an isosurface with many components and 128k points, whereas in Figure 8(c) we used three isovalues to extract multiple nested isosurfaces with a total of 535k points. Some surface parts are completely occluded by others. A transparent rendering helps the user to fully observe

¹ Data courtesy of Stanford University Computer Graphics Lab

² Data courtesy of VolVis distribution of SUNY Stony Brook

³ Data courtesy of SFB 382 of the German Research Council

Dataset # points Resolution	Blade 883k		Happy Buddha 543k		Dragon 437k		3 nested tori $3 \times 2M$		Neghip 128k		Hydrogen 535k in total	
	512 ²	1024 ²	512 ²	1024 ²	512 ²	1024 ²	512 ²	1024 ²	512 ²	1024 ²	512 ²	1024 ²
Local illumination	52	52	83	64	103	68	8	8	235	82	72	48
Transparency (3 layers)	17.6	17.5	28	22	35	23	2.7	2.7	83	27	24	15
Transparency (6 layers)	8.8	8.8	14	11	18	12	1.4	1.4	43	14	12	8
Shadows (1 sample)	26	25	40	39	50	49	4	3.7	145	64	40	31
Shadows (5 samples)	9	9	14	14	18	17	1.3	1.1	62	35	14	14
Shadows (10 samples)	5	5	7	7	9.6	9	0.6	0.6	35	22	8	7.5

Table 1: Frame rates in frames per second (fps) for rendering of point clouds with local illumination only, with transparency (using 3 and 6 blending layers), and with shadows computed with 1, 5, and 10 samples used for approximation of an area light source. One step for each hole filling filter was applied. No pre-computations are necessary.



Figure 8: Image-space point cloud rendering with transparency: (a) Transparent rendering of three nested tori (2M points each) with six blended layers. Each of the tori is drawn in a different color (blue, green, brown) and with a different opacities ($\alpha = 0.3, 0.5, 1.0$). (b) Point cloud with 128k points obtained by isosurface extraction of the volumetric scalar field “Neghip” is rendered with transparency ($\alpha = 0.7$) at 25 fps. (c) Three nested isosurfaces are extracted from a hydrogen molecule scalar field in form of point clouds with a total of 535k points. The visualization (at 9.8 fps) with semi-transparently rendered surfaces ($\alpha = 0.3, 0.5, 1.0$) allows the user to observe surfaces that are entirely occluded by others.

the isosurface extraction results. The transparent point cloud renderings use four and six surface layers, respectively, and run at frame rates of 25 fps and 9.8 fps.

The frame rates for generating renderings with shadows by first computing a shadow texture are also presented in Table 1. For low number of samples for Monte-Carlo integration, we achieve interactive rates for most tested models. For comparable models, our frame rates are higher than what has been reported for interactive ray tracing on splats [14] and similar to the ones reported for using shadow maps on splats [2]. These approaches, however, require a local surface reconstruction from the point cloud representation in a pre-processing step. For large datasets such local surface reconstructions can have a substantial computation time. Wald and Seidel [14] report performance of about 5 frames per second for comparable models with shadows and Phong shading, using a view port of 512x512 on a 2.4GHz dual-Opteron PC. On modern day hardware their approach would still be slower than what we have achieved (26 fps), since it utilizes only the CPU. The GPU accelerated EWA splatting approach of Botsch et al. [2] achieved a frame rate of about 23 fps on a GeForce 6800 Ultra GPU for rendering a model of 655k points with shadows. For comparison, our approach renders a 543k points model at 40 fps with one sample for shadows computation. On today’s GPUs, their approach would achieve sim-

ilar performance, but it still requires a pre-processing step to compute the splats. Moreover, for objects and light sources that do not change their relative position our approach also allows the shadow texture to be pre-computed and loaded along the point cloud. This way soft shadows, computed with lots of samples, can be rendered at highly interactive rates, imposing almost no load on the rendering pipeline.

A limitation of our approach comes from the resolution of the shadow map used to generate the shadow texture. If the resolution is chosen high, it is likely that the shadow texture will contain more “holes” and hence require more steps of the hole-filling filter to be applied. If the resolution is chosen lower, such that a couple of steps suffice, the edges of the shadow appear crisp and jaggy. This problem can be alleviated by using more samples for the area light source integration, which will provide soft anti-aliased shadows. If the scene cannot be rendered with multiple samples at interactive rates, an interactive rendering mode can be used: while navigating through the scene, i.e., rotating, translating or zooming, only one sample is used for shadow computation to provide high responsiveness. When not interacting, soft shadows are computed with a given number of samples.

A rendering of the Dragon dataset with shadows is shown in Figure 9. Ten samples were used for the shadow texture computation. The frame rate for that

rendering is 9.6 fps, which allows for smooth interaction.



Figure 9: Interactive rendering of the Dragon point cloud model with soft shadows at 9.6 fps. 10 samples are taken for the Monte-Carlo integration over the area light source.

Although all operations were executed without any computations in object space, we only introduced one intuitive parameter, namely the minimum distance d_{min} between two consecutive surface layers. This parameter was used at multiple points within our rendering pipeline. An improper choice of this parameter can produce severe rendering artifacts. For many datasets there is a wide range of values from which a suitable value for d_{min} can be chosen. Only when consecutive layers happen to get close to each other as, for example, for the Blade dataset, one has to choose d_{min} carefully. However, as the impact of the choice becomes immediately visible, an empirical choice was quickly made for all our examples.

7 CONCLUSION

We presented an approach for interactive rendering of surfaces in point cloud representation that supports transparency and shadows. Our approach operates entirely in image space. In particular, no object-space surface reconstructions are required. Rendering with transparency is achieved by blending surface layers that have been computed by a depth peeling algorithm. The depth peeling approach is also applied to compute point cloud shadow textures. A Monte-Carlo integration step was applied to create soft shadows. We have demonstrated the potential of our approach to achieve high frame rates for large point clouds. To our knowledge, this is the first approach that computes point cloud rendering with transparency and shadows without local surface reconstruction.

ACKNOWLEDGEMENTS

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under project grant LI-1530/6-1.

REFERENCES

- [1] Marc Alexa, Markus Gross, Mark Pauly, Hanspeter Pfister, Marc Stamminger, and Matthias Zwicker. Point-based computer graphics. In *SIGGRAPH 2004 Course Notes*. ACM SIGGRAPH, 2004.
- [2] Mario Botsch, Alexander Hornung, Matthias Zwicker, and Leif Kobbelt. High-quality surface splatting on today's gpus. In *Eurographics Symposium on Point-Based Graphics*, pages 17–24, 2005.
- [3] Frédéric Cazals and Joachim Giesen. Delaunay triangulation based surface reconstruction. In Jean-Daniel Boissonnat and Monique Teillaud, editors, *Effective Computational Geometry for Curves and Surfaces*. Springer-Verlag, Mathematics and Visualization, 2006.
- [4] Florent Duguet and George Drettakis. Flexible point-based rendering on mobile devices. *IEEE Comput. Graph. Appl.*, 24(4):57–63, 2004.
- [5] Cass Everitt. Introduction interactive order-independent transparency. White Paper, NVIDIA, 2001.
- [6] J. P. Grossman and William J. Dally. Point sample rendering. In *Rendering Techniques '98*, pages 181–192. Springer, 1998.
- [7] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 23, New York, NY, USA, 2007. ACM.
- [8] Ricardo Marroquim, Martin Kraus, and Paulo Roma Cavalcanti. Efficient point-based rendering using image reconstruction. In *Proceedings Symposium on Point-Based Graphics*, pages 101–108, 2007.
- [9] Tom Mertens, Jan Kautz, Philippe Bekaert, Frank Van Reeth, and Hans-Peter Seidel. Efficient rendering of local subsurface scattering. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, page 51, Washington, DC, USA, 2003. IEEE Computer Society.
- [10] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 497–500, New York, NY, USA, 2001. ACM.
- [11] Paul Rosenthal and Lars Linsen. Image-space point cloud rendering. In *Proceedings of Computer Graphics International (CGI) 2008*, pages 136–143, 2008.
- [12] Gernot Schaufler and Henrik Wann Jensen. Ray tracing point sampled geometry. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 319–328, London, UK, 2000. Springer-Verlag.
- [13] R. Schnabel, S. Moeser, and R. Klein. A parallelly decodable compression scheme for efficient point-cloud rendering. In *Symposium on Point-Based Graphics 2007*, pages 214–226, September 2007.
- [14] Ingo Wald and Hans-Peter Seidel. Interactive ray tracing of point based models. In *Proceedings of 2005 Symposium on Point Based Graphics*, pages 9–16, 2005.
- [15] Yanci Zhang and Renato Pajarola. Deferred blending: Image composition for single-pass point rendering. *Comput. Graph.*, 31(2):175–189, 2007.
- [16] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Surface splatting. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, New York, NY, USA, 2001. ACM.

Tracking single channel in protein dynamics

Petr Beněš
Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic
xbenes2@fi.muni.cz

Petr Medek
Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic
medek@fi.muni.cz

Jiří Sochor
Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic
sochor@fi.muni.cz

ABSTRACT

In this paper we present a new approach to the processing of molecule dynamics. The method performs the tracking of a channel in a sequence of molecule snapshots which represent atom positions in the molecule in certain time intervals. The centerline of the tracked channel is refined using the Delaunay triangulation from the actual snapshot resulting in a new optimized centerline. This method allows us easily to animate the behaviour of the channel in the sequence. The method can also be used to detect the channel geometry in snapshots, where recent methods are not able to find this channel. In addition, the method yields information about channel parameters which vary over time. We can evaluate opening and closing of the input channel.

Keywords

channel, protein dynamics, tracking, Voronoi diagram, Delaunay triangulation

1. INTRODUCTION

Biochemists usually want to observe the behaviour of a protein in a particular part of the molecule, e.g. to observe the exit route of a substrate. Channels as defined in [Med07] can be used to visualize this information. A channel which leads through an empty space in the molecule can for example be wide for a significant period of time or the substrate might initiate the opening of a narrow channel when passing by. This information helps chemists to predict the behaviour of a molecule before performing real experiments.

Most of the methods of channel computation are designed to process a single static protein molecule. There are only a few methods for analysing the dynamics of protein molecules. Since the dynamics of a protein molecule is a continuous movement, it is sampled into a sequence of snapshots representing atom positions in given time intervals. The snapshots

are usually aligned so that the global position and rotation of the molecule is fixed in all snapshots and the snapshots only represent local movement of atoms.

Recent methods typically process each of these snapshots separately as static molecules and cluster obtained results at the end of the computation. Therefore, none of these methods is specialized for tracking a certain channel throughout the whole sequence. The visualization of this information can improve the process of protein analysis significantly.

The method proposed in this paper is able to detect a particular channel in each snapshot of the dynamics and the resulting channels are spatially close to each other. This allows us to animate the progress of a channel over time easily.

We can use also this method to compute a channel in the snapshots, where the classic approaches are not able to detect this channel since they compute only limited number of channels in each snapshot. However, there are situations where we need to know the channel geometry in each snapshot. Using the proposed method, the missing channel can be computed from the surrounding snapshots where the channel geometry is known.

The main advantage of the proposed method is not only to improve the visualization of channel progress over time. As demonstrated in the results section, the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

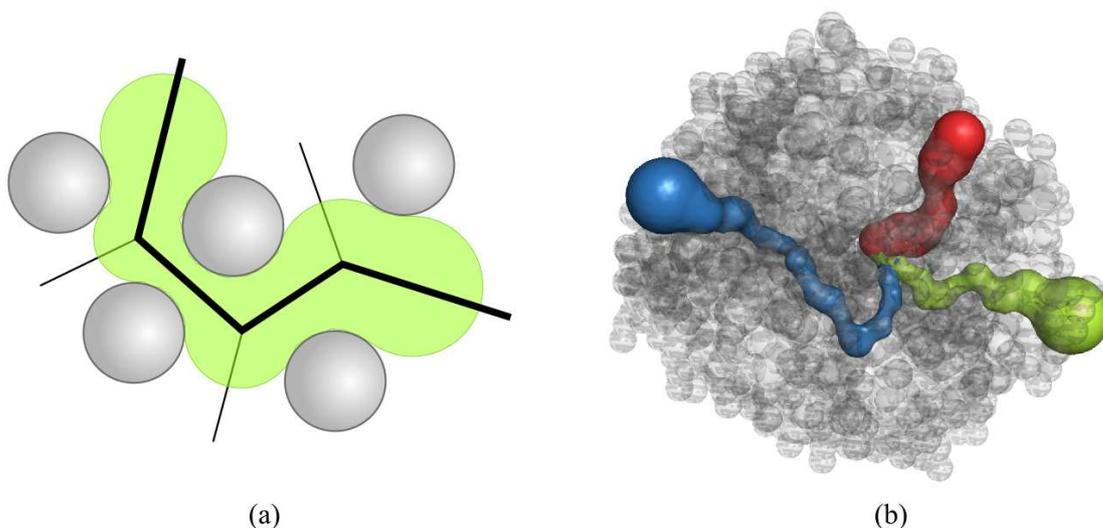


Figure 1. (a) Demonstration of a channel. This channel is ideal, i.e. its centerline leads along Voronoi edges. (b) Channels computed in a real static molecule and visualized using pyMol software.

overall evaluation of channels computed by this method can also bring new information about molecule and channel behaviour.

2. RELATED WORK

For the computation of channels in static molecules there have been many different methods proposed. All these methods process the molecule as a geometric model where each atom corresponds to a sphere with given position and radius in the three-dimensional space. Other biochemical properties are not considered. A channel in the molecule (also referred to as tunnel) is defined as a centerline and the volume ([Med07], Fig. 1). A centerline is a continuous curve and the volume is formed by the union of spheres inserted at each point of the centerline. The radius of all these inserted spheres is maximal so that it does not intersect any other atom in the molecule.

An approach introduced in [Pet06] is based on space rasterisation. This approach suffers from several disadvantages resulting from discrete sampling.

Other methods [Med07, Pet07, Yaf08] are based on the Delaunay triangulation (DT) and the Voronoi Diagram (VD) computed for the molecule. These methods are faster, more precise and more efficient than rastering solutions. However, they are designed to process a single static molecule and so they are not able to return information about channel properties varying across snapshots (such as the progress of the width of a channel over time). Nevertheless, the channels and their trajectories in the static snapshots can be used as an input for the tracking method proposed in this paper.

A method which is able to determine the progress of channels in protein molecules over time is called molecular dynamics (MD) [Ald59]. A small molecule (substrate) is positioned inside the protein molecule and a physical simulation starts. Random forces are applied to the substrate during the simulation and collisions and interactions of the substrate with the protein are evaluated. It is probable that the substrate molecule reaches the protein surface. This method is able to find certain exit route of a substrate leading from a given position inside the protein, so called active site, to the protein surface. Note that this method does not require the whole dynamics to be computed before starting the simulation (actually, it computes the dynamics itself in a run-time simulation). The movements of atoms are computed continuously during the simulation according to the result of force interactions. The method is immensely time consuming (one simulation takes hours to days to evaluate). Due to the application of random forces, it is not guaranteed that a channel is found. If a channel is not detected, it does not mean that this channel does not exist.

The complex approach proposed in [Ben09] requires a sequence of snapshots to be known in advance, either from some real screening or existing simulation. It computes channels in snapshots separately using any method for the computation of channels in a static molecule and clusters them afterwards for the whole dynamics. Each cluster represents the progress of a particular channel throughout the dynamics. Since the methods for the computation of channels in a single snapshot produce only a limited number of channels, it is probable that

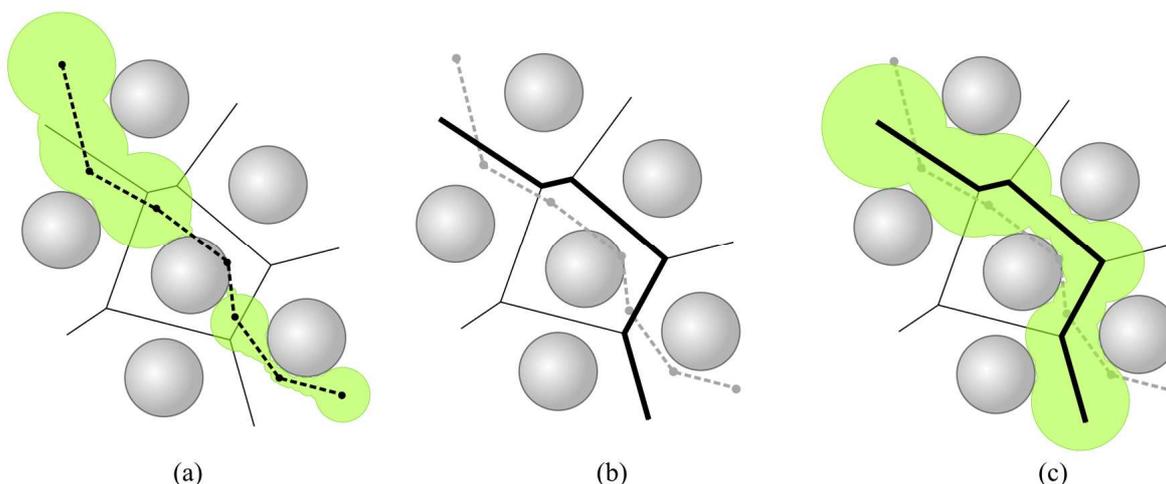


Figure 2. (a) Initial trajectory (dashed polyline) and the corresponding channel, (b) Voronoi diagram with optimized trajectory emphasized, (c) Channel defined by optimized trajectory

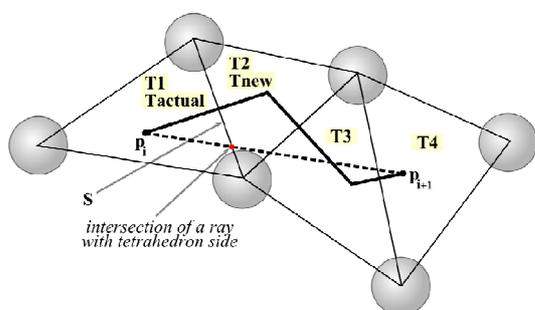


Figure 3. The demonstration of the algorithm for the segment $\langle p_i, p_{i+1} \rangle$

some clusters will not cover the whole sequence. We can consider a dynamic channel to be closed in the snapshots, where the cluster provides no information about the geometry of this dynamic channel. However, this complicates the visualization of the

<p>Input: initial trajectory $t = p_1..p_n$, tetras of DT for the actual snapshot</p> <p>Output: optimized trajectory</p>
<pre> for each $\langle p_i, p_{i+1} \rangle, i \in 1..n-1$ { $T_{actual} = \text{tetra containing } p_i$; while ($T_{actual}$ not contains p_{i+1}) { $s = \text{side of } T_{actual} \text{ intersected}$ $\text{by } \langle p_i, p_{i+1} \rangle$; $T_{new} = \text{tetra sharing } s \text{ with } T_{actual}$; // function $c(T)$ returns center // of gravity for tetrahedron T output ($\langle c(T_{actual}), c(T_{new}) \rangle$); $T_{actual} = T_{new}$; } } </pre>

Algorithm 1. The optimization of an initial trajectory in a single snapshot of a molecule.

channel dynamics.

3. PROPOSED METHOD

The method proposed in this paper is able to track the progress of a specific part of the protein molecule over time. The specific part is described by a channel. Its centerline is referred to as *initial trajectory* for further tracking.

The initial trajectory can be the centerline of an already known important channel or it can also be an exit route of a substrate computed by molecular dynamics.

Algorithm

In each snapshot, we optimize the initial trajectory so that the channel formed by this trajectory has the maximal possible volume. If we do not optimize the trajectory, the channel can be very narrow or it can even have zero or negative width (see Fig. 2a). The optimized trajectory follows edges of a Voronoi diagram of the protein molecule (Fig. 2b) and thus the resulting channel can be much wider (see Fig. 2c).

The algorithm utilizes the duality between VD and DT. The initial trajectory is mapped onto a sequence of tetrahedra in the DT. This sequence can be converted to Voronoi edges easily.

The initial trajectory is represented as a polyline with vertices p_1, \dots, p_n . These input points define $n-1$ line segments. For each of the segments $\langle p_i, p_{i+1} \rangle$ ($i=1, \dots, n-1$) the tetrahedron T_{actual} containing p_i is located and marked as actual. Then we determine the tetrahedron side s which intersects the ray between p_i and p_{i+1} . As the next step we move into the tetrahedron T_{new} which shares the side s with actual tetrahedron T_{actual} . Finally, T_{new} is marked as actual. The process is depicted in Fig. 3. The line segment

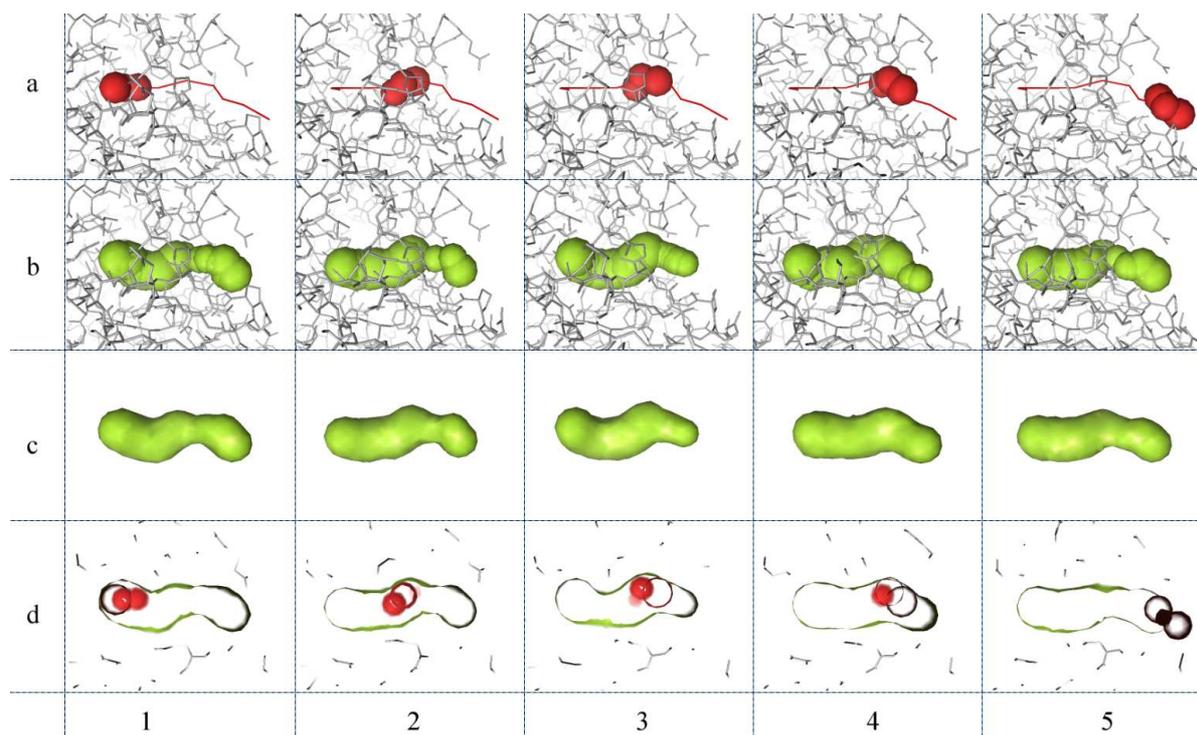


Figure 4. (a) The exit path of the substrate molecule in time (1-5) in 1mj5. This path was used as the initial trajectory. (b-d) Different types of visualization of a channel dynamics in certain snapshots (1-5).

$\langle p_i, p_{i+1} \rangle$ is substituted by the polyline which is formed by Voronoi edges adjacent to the tetrahedra in the DT which were traversed during the above procedure. The procedure is summarized in Alg. 1.

Note that the implementation of Delaunay triangulation enables to determine all neighbours of a given tetrahedra in constant time. In addition, the geometry test required for each processed tetrahedron is the calculation of the intersection of a line segment and a triangle (tetrahedron side) in three dimensions, which is fast and simple.

Since each segment $\langle p_i, p_{i+1} \rangle$ is replaced by the Voronoi edges dual to the tetrahedra intersected by $\langle p_i, p_{i+1} \rangle$, the spatial distance between initial and optimized trajectory is the minimal possible.

Notice that also the approach presented in [Med07] with minor changes could be used to get the widest channel between each two segment endpoints. Nevertheless, the computation would be more time consuming as the Dijkstra's algorithm would be used instead of fast following the ray-tetrahedra intersections in the DT. In addition, the channel might be much longer and its centerline might lead far from the initial trajectory. This fact would certainly complicate the smooth and continuous animation of a channel over time.

Time complexity

The time required to compute the Delaunay triangulation is quadratic with respect to the number

of atoms in the molecule [Pre85]. The subsequent tracking of an input trajectory is linear with respect to the number of atoms. The overall complexity is $O(k*n^2)$ where n is the number of atoms and k is the number of molecule snapshots.

The computation time can be reduced by computing only a subset of Delaunay triangulation which would be located near the input trajectory. If a trajectory covers only a small part of the molecule the computation time can be reduced significantly.

4. RESULTS

The first dynamics analysed by the proposed method is the protein molecule 1mj5 consisting of 50 snapshots. This sequence was achieved by MD simulation of the molecule and substrate. Therefore, the exit route of the substrate is known in this case. This exit route is used as the initial trajectory in the computation. When we visualize the results of the analysis, we can observe the substrate initiates opening of the channel, i.e. the channel gets wider in places where the substrate passes.

Different types of visualization of these results are depicted in Fig. 4. Five snapshots (1-5) are chosen from the dynamics to illustrate the progress over time. The results are visualized in different ways (a-d). The first of them (a) shows the exit route of a substrate and the trajectory of this route. The others show the resulting channel with a centerline located on the optimized trajectories in different snapshots displayed

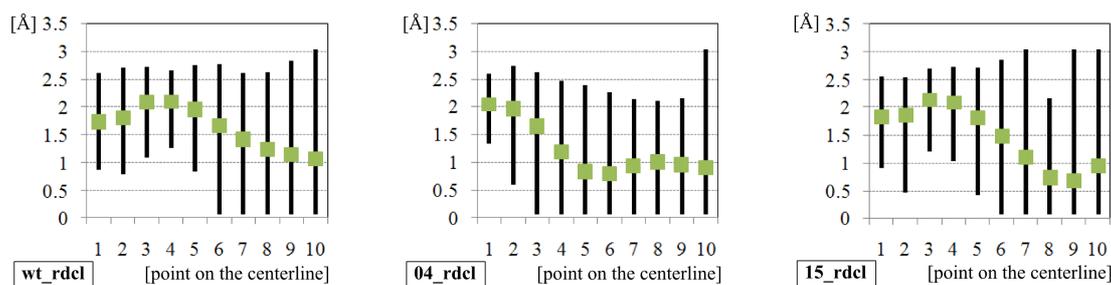


Figure 5. Charts depicting channel statistics for selected protein dynamic sequences. The x-axis denotes uniformly distributed points on a centerline and the y-axis denotes the variation of channel width in these points throughout the dynamics: maximum width, minimum width and average width in the whole sequence.

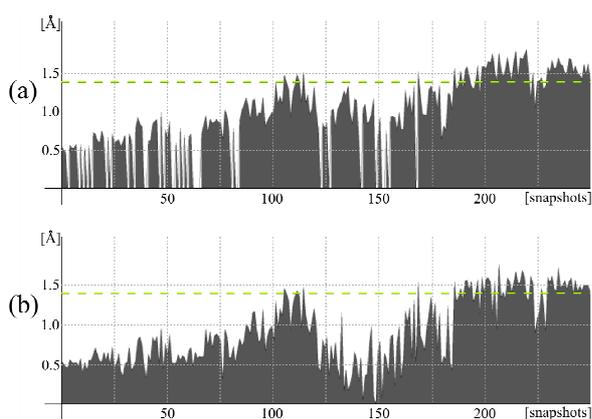


Figure 6. The analysis of the width of a channel in the sequence of 250 snapshots using (a) the clustering method, (b) the proposed method. The dashed green line denotes the biochemically important value 1.4Å

as a set of spheres (b) or a surface (c). In (d), the whole scene is clipped using the front clipping plane approximately in the middle of the channel. In this case we can observe the substrate molecule passing through the channel.

Notice that the previous example demonstrates a possible use of this method on dynamics where an exit route is known before. If such route is not known, chemists have to define the initial trajectory they want to observe. The definition can be done by hand or the widest channel from a single snapshot of the dynamics can be used.

The behaviour of the channel in the analysis indicates whether a certain substrate would be able to pass through this channel.

The second data set consisted of a set of nine molecules of type *rdcl*, which were structurally similar. They were mutants of the same protein molecule (only a few residues were different in each of the mutants). The dynamics of each mutant consisted of 400 snapshots. We tracked the same initial trajectory in all dynamic sequences.

The behaviour of resulting channels was analyzed in ten uniformly distributed points along their centerlines. The first point refers to the channel endpoint in the active site and the tenth point is the channel endpoint located near the molecule surface.

For each dynamics, statistics about the pulsing of a channel in each of these segments were computed. The statistic for selected molecules is shown in Fig. 5. The average width, minimal width and maximal width (y-axis) are shown for each of the ten points on the x-axis. One of the possible interpretations of the data in these charts is the following. All channels tend to be more stable near the active site whereas certain opening and closing of a channel happens near the molecule surface. It can be seen that in case of *wt_rdcl* (Fig. 5, *wt_rdcl*) the first half of a channel remained open during the whole sequence with radius varying from 0.8\AA^1 to 2.6\AA whereas the radii in the second half varied more significantly.

This information helps chemists to estimate which of the mutants is the most suitable for a certain substrate molecule to penetrate into the protein.

The visualizations in Fig. 4 were created using pyMol software [DeL02].

As a third test case, we have analysed the width of a channel in the sequence of 250 snapshots of *21_rdcl.cl* using the clustering method [Ben09] and the proposed method. In Fig. 6, it can be seen that both methods provide similar results. In the case of clustering method (Fig. 6a) the width of a channel is usually slightly larger. However, there are snapshots in which the channel is not detected. On the contrary, the proposed method (Fig. 6b) detects the channel in all snapshots. Therefore we can use results of this method to add the missing channel data.

We have also evaluated the distances between channels in all consecutive snapshots according to the distance function defined in [Ben09]. In comparison with the graph cutting clustering method, the distance

¹ $1\text{\AA} = 10^{-10}\text{ m}$

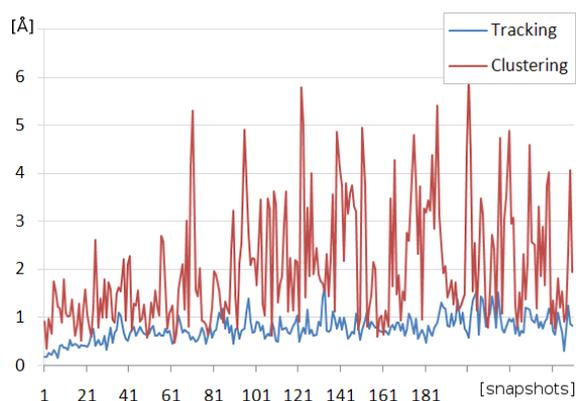


Figure 7. Distances between channels in consecutive snapshots for tracking and clustering method. First 250 snapshots containing channels were considered.

between channels computed using the proposed method is much smaller. The example for the sequence *21_rdcl.cl* can be seen in Fig. 7. The average distance for the proposed tracking method was 0.77\AA whereas the average distance for the clustering method was 2.03\AA . Due to the fact that the average distance is small, the progress of a channel over time can be easily animated.

5. CONCLUSION

The proposed method allows tracking an initial trajectory in the dynamics. The method is based on computational geometry and is fast and robust. Except for computation of the Delaunay triangulation, it uses only basic geometry tests.

We have also presented several applications of this method. The optimization is of key importance when performing smooth animation of channel progress across snapshots over time.

The properties of resulting optimized trajectories can provide useful information about the protein molecule. The possible interpretation of such results has been suggested. The presented method can also be used in snapshots where recent methods are not capable of detecting the channel geometry.

As for the future work, we plan to integrate this method within the complex software application Caver Viewer (<http://loschmidt.chemi.muni.cz/caver/>)

which is designed for the visualization and analysis of protein molecules.

6. ACKNOWLEDGMENTS

This work was supported by The Ministry of Education of The Czech Republic, Contract No. LC06008 and by The Grant Agency of The Czech Republic, Contract No. 201/07/0927.

7. REFERENCES

- [Ald59] Alder, B. J., and Wainwright, T. E., Studies in molecular dynamics. i. general method. The Journal of Chemical Physics, vol. 31, no. 2, pp. 459–466, 1959.
- [Ben09] Beneš, P., Medek, P., and Sochor, J. Computation of channels in protein dynamics. IADIS International Conference Applied Computing 2009, Rome, pp. 251–258, 2009
- [DeL02] DeLano, W.L. The PyMOL Molecular Graphics System (2002) on World Wide Web <http://www.pymol.org>
- [Med07] Medek P., Beneš P., Sochor J.: Computation of tunnels in protein molecules using Delaunay triangulation. Journal of WSCG 15, 1–3, pp. 107–114, 2007.
- [Pet06] Petřek, M., Otyepka, M., Banáš, P., Košinová P., Koča, J., and Damborský J. CAVER: a new tool to explore routes from protein clefts, pockets and cavities. BMC Bioinformatics, 2006.
- [Pet07] Petřek, M., Košinová, P., Koča, J., and Otyepka M.: Mole: A Voronoi diagram-based explorer of molecular channels, pores, and tunnels. Structure 15, 11, pp. 1357–1363, 2007.
- [Pre85] Preparata, F.P., and Shamos, M.I. Computational Geometry: An introduction. Springer-Verlag, 1985.
- [Yaf08] Yaffe, E., Fishelovitch, D., Wolfson, H. J., Halperin, D., and Nussinov, R.: Molaxis: Efficient and accurate identification of channels in macromolecules. Proteins, 73(1):72–86, 2008.

A framework for User-Assisted Sketch-Based Fitting of Geometric Primitives

Davide Portelli
VCL - ISTI, Pisa
davide.portelli@gmail.com

Paolo Cignoni
VCL - ISTI, Pisa
paolo.cignoni@isti.cnr.it

Fabio Ganovelli
VCL - ISTI, Pisa
fabio.ganovelli@isti.cnr.it

Matteo Dellepiane
VCL - ISTI, Pisa
matteo.dellepiane@isti.cnr.it

Marco Tarini
Univ. dell'Insubria, Varese
marco.tarini@isti.cnr.it

Roberto Scopigno
VCL - ISTI, Pisa
roberto.scopigno@isti.cnr.it

ABSTRACT

In this paper, we present a user-assisted sketch-based framework to extract hi-level primitives (e.g. columns or staircases) from scanned 3D models of an architectural complex. The framework offers a unified level of representation of the hi-level primitives, so that new types of primitives can be easily added as plug-ins to the main engine. Primitives are fitted with a user-assisted procedure: the user suggests the approximate location of the primitive by means of simple mouse gestures, sketched over a rendering of the model. The viewpoint that was selected prior to the sketching is also taken in consideration as hints on the orientation and size of the primitive. The engine performs a GPU assisted fitting and the result is shown in real time to the user. Ad-hoc gestures cause the system to add and fit groups of primitive in one go (e.g. a column complex, or a sequence of windows).

Keywords: 3D segmentation, fitting geometric primitives

1 INTRODUCTION

Before the advent of scanning devices, 3D digital models of architectural buildings were mainly obtained via manual modeling. This operation is typically guided by 2D data, like sections and prospects. A modeler usually proceeds by decomposing the structure in a set of primitives, then “builds” the model by adding the primitives.

The increasing availability of 3D range scanning devices, the development of software increasingly efficient and user-friendly for the creation and manipulation of complex 3D digital models and the drop of the scanning technology costs, are the main reasons of the recent fast proliferation of scanning campaigns for the acquisition of the shape of real world objects. Along with other application fields, 3D range scanning [CM02] is increasingly used in architecture.

The result that can be obtained using 3D scanning, organized as clouds of points or as triangle meshes, is a far more accurate description of the actual shape of the building or the façade than the one obtained with manual modeling, but it does not carry any

information on what the object or its parts are.

The possibility to decompose an architectural model in a set of higher level primitives (which are very often repeated on the same façade) is extremely important for a number of possible applications: analysis, archival, comparison with other models. This would combine the flexibility of direct 3D modeling to the accuracy of 3D scanning. The primitive extraction can also be applied to different approaches aimed at recovering the 3D information of buildings [BSZF99, SB03].

In this paper we present a framework for a user-assisted extraction of geometric primitives. The intervention of the user is limited to a few sketches over a rendering of the low-level model. The sketches roughly define the size, orientation and position of the intended primitive. The approach is robust with respect to incomplete geometry, and is also capable automatically identifying and extract repeated instances of a single primitive. As a result, the user can decompose a complex 3D models in a few minutes, without the need of picking accurate positions, and obtain good results.

The next subsections will briefly review several state-of-the-art automatic and semi-automatic approaches for primitive fitting. Then, the proposed framework will be shown. A discussion on the obtained result will be presented before the conclusions.

2 RELATED WORK

The literature on reverse engineering from 3D data is vast. In this section we will only give a brief overview of the approaches more closely related to our domain. We will subdivide the approaches in *segmentation* approaches and *fitting* approaches. In the first class we put the approaches for finding low level features, such as lines, planar regions or high curvature points in the 3D dataset. These methods do not aim to give the information about the nature of an object, instead they try to convert a raw geometric description (i.e. a point cloud or a triangles mesh) into a more abstract description. Usually these techniques rely on discrete local curvature operators to detect features [OBS04, WB01, HHW05, CSAD04], the biggest challenge being making the algorithm robust to geometrical noise. Extract-

ing features from an irregular 3D point cloud or from triangle mesh produced by 3D scanning is made difficult by the inherent ambiguities of the task as well as by the presence of geometrical noise, holes in the model, and other inconsistencies. Once basic geometric features such as lines and planes have been found, they can be grouped to describe higher level structures. In [SWWK07] this is done by creating a graph of relations where sub parts of the graph define structural elements and arcs describe the constraint between elements. In the class of fitting approaches we place those methods which use parametrized description of higher level primitives and try to "place" them in subparts of raw data by means of minimizing an error function. The function being minimized can be defined ad hoc for a given type of primitive (e.g. planes, cones, cylinders) [MLM01, Ben02]. In the general case, however, it consists in some form of distance between the surface of the primitive being fitted and the real model. In [USF08] the authors give a GML parametric description for the model being fitted and the minimization performs the fitting using the given parameters. In [PMW*08] the case of repeated regular structure in manufactures or natural objects is studied, such as a series of windows or a snow flake. The approach uses a sequence of operation consisting of partitioning the object, finding a set of transformations between parts and clustering them to extract geometric relations in the model.

3 OUR FRAMEWORK

Our framework falls in the group of fitting approaches. Rather than trying a fully automatic approach, we aim at reducing user intervention down to few mouse gestures. The gestures are used to reduce the search domain the the minimization required by the fitting process, so to avoid the most computationally demanding phase which is often carried out with RANSAC based algorithms. Figure 1 shows the steps required for the user to identify and fit a set of 5 columns. The user selects a view of the 3D raw dataset by manipulating a mouse-controlled trackball. Then he perform a sign over the current rendering with the mouse, as shown in Figure 1-(a). With this information a column shape (in this case, a trunk of cone) is fitted over the 3D dataset – the surface shaded in red in Figure 1-(b); once the first column has been fitted the user may perform a second gesture to indicate that there is a series of similar columns, as shown Figure 1-(c); those columns are automatically fit (also see attached video).

Our main concern is to make the system easily extendable, so that the process of adding new types of primitives is easy and the system is not tied to a predefined set of primitive types. The fitting problem is approached as a generic minimization problem. All primitive types are defined likewise as a parametric *shape*

function Sh , which takes as input a variable amount of intrinsic and extrinsic parameters, and returns in output a set of 3D points. More precisely, $Sh(x_1, \dots, x_n, RT) = \{p_1, \dots, p_m\}$, where m is the number of produced samples on the surface, and n is the number of scalar *intrinsic parameters*, and RT is a roto-translation matrix, or *extrinsic parameters*, which specify the location in space of the shape. Specification of a primitive type also include an interval for each intrinsic parameter. Note that choosing to express the shape as a parametric point set does not allow to exploit non geometric information that we may know about the primitive. On the other hand it gives generalization of the primitive description and allow us to write a extendable framework.

While any primitive type has the same extrinsic parameters, the intrinsic parameters vary from type to type, both in number and in range of values.

For example the primitive type *Column* is defined by the shape function:

$$Column(r_{bottom}, r_{top}, len, RT)$$

where r_{bottom} and r_{top} are the two radii of trunked cone with length len .

The minimization problem can now be defined independently of the type of the primitive being fitted:

$$\min_{x_i \in Constr(i)} Err(Sh(x_1..x_n, RT), M) \quad (1)$$

where Err is a measure of the difference between the primitive and the scanned model and $Constr(i)$ is the constraint defined for the parameter i (for example in the case of the column we have $0 < r_{bottom}, r_{top}, len$).

Figure 2 show a scheme of the whole process. The user select a shape and perform a mouse gesture so providing the input for the module that computes a first estimation of the parameters. Then the minimization process start by sampling the surface generated by the parameters, computing its distance from the model and updating the parameters to decrement the error, until a satisfactory fitting is found.

Being that our method relies also on user intervention, it may reminds to many user assisted techniques for segmentation of medical datasets for which a vast literature is available (see [PXP00] for a recent survey). However there are important distinctions both in finalities and in adopted strategies. The first difference is that for architectural manufactures we do not need a tool for supporting the recognition of a shape, as usually is for medical images, but only a tool for converting a raw description (a point cloud) in a structured one (union of architectural elements). Many techniques in medical segmentation are based on energy minimization methods but the exploitation of a known parameterization of the object to segment brings less advantages than in our case for the simple reason that human organs are much

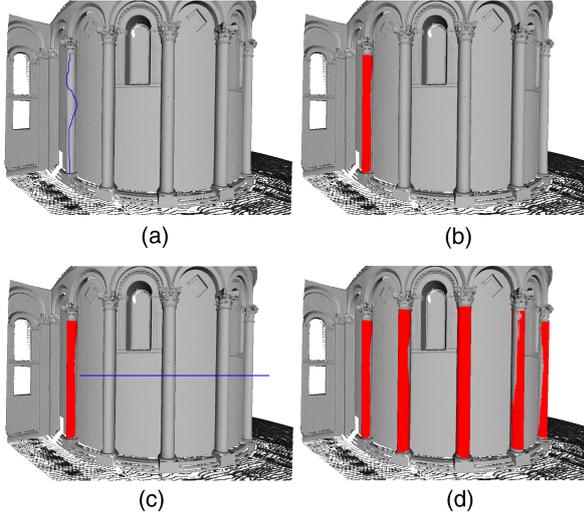


Figure 1: application example of fitting of 5 columns.

more difficult to parametrize than architectural buildings.

3.1 From gesture to parameters estimation.

The goal of using the mouse gesture is to reduce the search space for in the minimization process. However, in order to be effective, the gesture must be simple to do and not necessarily precise. The first task is to interpret a 2D mouse gesture in a selection of a 3D subpart of the original 3D data (mesh or point-cloud).

Figure 3.(a) shows the example of the column where the sign of the mouse is partly over the column (shaded in red) and partly over the background (shaded in green). In Figure 3 we see how the selected points are distributed in space.

We compute the distribution of the distance of these points from the viewer and use it to remove what we consider to be outliers (see Figure 3.(c)), in the assumption that the majority of points will be coherently on the part of the dataset that corresponds to the primitive being fitted. Then we take the bounding box of these points to infer an initial estimation of parameters for the shape. In the most general case, i.e. with no assumption neither on the type of dataset nor on the type of primitive, the only information that we could use from the bounding box is its volume, so we can solve a minimization problem:

$$\min \|Volume(Sh(X_0, \dots, x_n, RT) - Volume(BBox))\|$$

and use the solution as the initial estimation for the problem 1. The computation of a solution is made less computationally intensive by taking in account the view transformation that was chosen by the user in order to have a suitable view of the intended feature:

- the view transformation selected by the user before he performs a mouse gesture is assumed to be such that the feature has a natural orientation (e.g. the column is not upside down in screen space);
- similarly, the intended instance of the primitive is oriented, in view-space, as facing the camera.

taking advantage of these reasonable assumptions, we infer a correspondence between the frame centered in the center of the bounding box and oriented with its sides, and the frame where the shape is defined for the initial to obtain the parameters estimation.

3.2 Minimization

At a first glance, we could take the function to minimize, referred as *Err* in the problem 1 as the sums of Euclidean distance between the primitive and the model. Unfortunately this is not enough, because we may have architectural elements which subparts are also instances of the same type of element. For example a portion of a plane is also a plane and a portion of a column is also shaped as a column. Of course this also depends on the definition of the primitive types. Consider for example how a column including a basement and the capital we would not have these ambiguities (however that primitive type could not be fitted, for example, over a 3D point cloud featuring a broken column, a case for which we would need an ad hoc primitive).

For these reasons, we aim at the *maximal* portion of dataset that matches with the primitive. Therefore we redefine our error function as:

$$Err(Sh, M) = \frac{1}{Area(Sh)} \sum_{j=0}^{j < k [Area(Sh)]} \max(t, w_i D(s_i, M))^2 \quad (2)$$

where s_i , $i = 0 \dots k$ is a sampling of the surface of the primitive, $D(s_i, M)$ is a measure of the distance from s_i to the model M , t is the minimum error that is assigned to each sample to smooth out the contribution due to the noise of the scanned model and w_i is a $[0, 1]$ weight associated with the i^{th} sample that is used to discard outliers that are created if the model misses portion of surface that are represented in the shape (e.g. a column with a missing piece). Essentially *Err* takes into account the distance between the primitive and the model **and** the area of the shape and decreases both if the distance decreases and if the area grows. Note that the distance measure is squared in order to express both parts of the fraction in the same scale, and that the number of samples is proportional to the area so that each sample accounts approximately for a constant area.

Computing $D(s_i, M)$. We can define the distance function as the Euclidean distance to the closest point on M ,

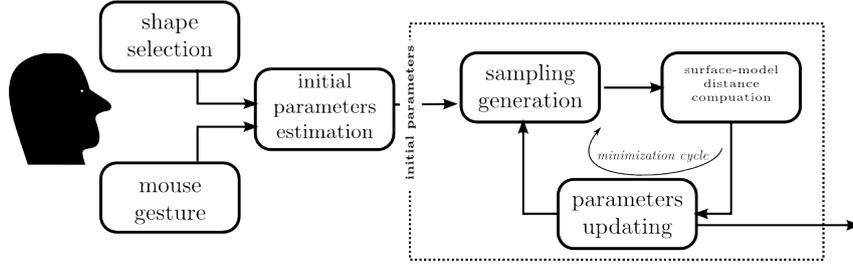


Figure 2: A scheme of the fitting framework

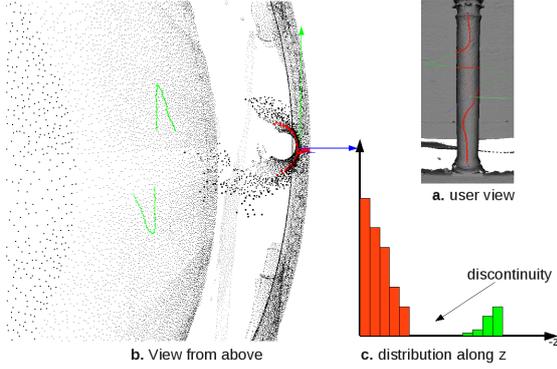


Figure 3: From mouse gesture to initial parameters.

just like in classic IPC algorithm [BM92] D . However, since we have an estimation of the normals both for the shape and for the model, we can achieve better results including the normals in the estimation and defining the distance as:

$$D(s_i, M) = \min_{p_i \in M} D(s_i, p_i), \quad p_i \in M \quad (3)$$

where:

$$D(s_i, p_i) = E(s_i, p_i) + \frac{\alpha (1 - \vec{n}_1 \cdot \vec{n}_2)^{2\beta}}{E(s_i, p_i) + 1} \quad (4)$$

the function D is simply the Euclidean distance E plus a positive bounded contribution En (the right part of the sum) which accounts for the normals in the distance computation. The expression is formulated so that the weight of the normal only comes into play where the two points are close to each other and the magnitude of their contribution is proportional to the angle between them. It can be easily seen that the maximum contribution (found when $E(s_i, p_i) = 0$ and $\vec{n}_1 \cdot \vec{n}_2 = -1$) is $\alpha 2^{2\beta}$. We can set β to determine how fast the contribution of this term grows and α to relate the term to the density of the dataset. The value of α is important because the contribution of the term must be proportional to the density of the sampling to affect the minimization. Typically a good choice is to set it to the average inter point distance. So if, say, $\beta = 2$ and the average inter point distance is 0.5, we will have a term that may increase the distance estimation from the Euclidean value at most by $0.5 \cdot 2^{2 \cdot 2} = 8$, when points with

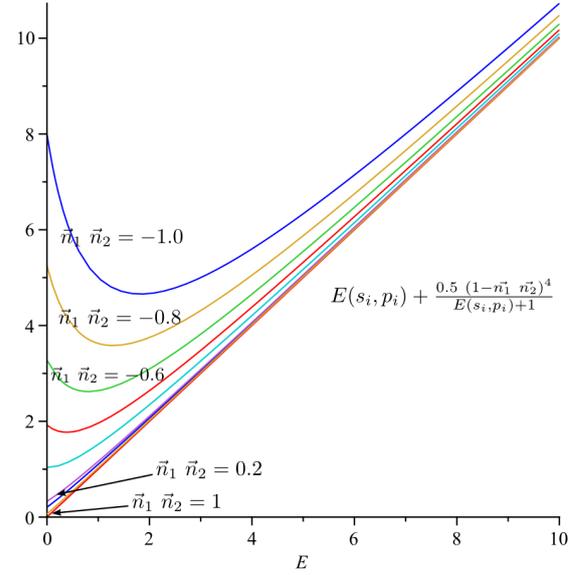


Figure 4: A plot of the distance function for $\beta = 2$, $\alpha = 0.5$ and several values of angle between the points' normal.

opposite normals coincide. Figure 4 shows a plot of En for different values of the product $\vec{n}_1 \cdot \vec{n}_2$ and $\beta = 2$ where this behavior can be observed.

Al though the distance function En is a $5D$ function, the closest point on M with respect to En can be found using only data $3D$ space indexing data structures for Euclidean distance by:

1. finding the closest point p_i with respect to the metric E
2. taking the closest point with respect to D among those which euclidean distance is less than $D(s_i, p_i)$.

It is easy to see that the algorithm returns the closest point w.r.t. D , because

$$E(s_i, p') > D(s_i, p_i) \rightarrow D(s_i, p') = E(s_i, p') + En(s_i, p') > D(s_i, p_i)$$

Minimization cycle. At this point we have defined both the parameters and the function to minimize and may apply any non linear minimization algorithm to find a hopefully optima solution. However, we exploit

the knowledge of a closed form solution for the extrinsic parameters alone [BM92] and decompose the minimization cycle in three steps:

- 1 for each sample in the shape Sh , find the closest point in the model
 - 2 find the rototranslation that minimizes the squared distances between all the pairs (only explicit parameters involved)
 - 3 iterate a non linear minimization procedure (only implicit parameters involved)
- if $Err(Sh, M)$ is under a user selected threshold return, otherwise goto 1

We used both *Levenberg-Marquardt* [Lou09] and *Newuoa* [Pow08], with similar results.

4 EXTENDING TO MULTIPLE INSTANCES

When an architectural element has been fitted, it is likely that other similar elements (i.e. of the same type and size) are present. Examples are the columns, the steps of a stair or a series of window. Therefore we wanted to spare to the user to repeat the same mouse gesture for all the elements and simply make a single gesture which says *here there are other elements of this type*. As shown in Figure 1.(c), the gesture required is two mouse clicks to define a line segment. From this gesture the initial parameters for all the other columns are found and the minimization process just described is launched on each instance.

4.1 From gesture to parameters estimation.

Since we have fitted the first element, we already have the estimation of the initial implicit parameters for the other instances of the same type of element. The user may define a segment $(seg(t)_x, seg(t)_y)$ to indicate where these other instances are, as shown in Figure 6. Therefore the information we need to extract is *how many* other elements there are and, for each one of them, an estimation of the extrinsic parameters. Furthermore we can exploit the fact that in architectural manufactures repeated elements usually differ by a translation but are oriented in the same way and reduce the missing extrinsic parameters to a translation.

In principle we could sample the segment and, for each sample, launch the optimization taking the projection of the sample onto the scene as a starting point for translation. Unfortunately the minimization process requires few seconds to complete and therefore we need to reduce the set of candidate translations.

We harness the rasterization process in order to quickly reduce the candidate translations. More precisely we exploit the z-fighting artifact. The z-fighting

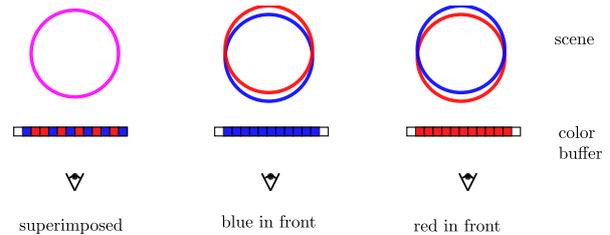


Figure 5: A schematic representation of z-fighting quantification

is the rendering artifact that happens when the depth values of the rasterization of different polygons falls in a range of values close or under the precision of the z-buffer, so that the pixels are evenly written by the conflicting polygons.

The idea is that if we consider the 3D point $(seg(t')_x, seg(t')_y, seg(t')_z)$ where $(seg(t')_x, seg(t')_y)$ are 2D points belonging to the segment and $seg(t')_z$ is the projection on the model and render an instance of the shape translated by $seg(t')$ together with the scene, the presence of z-fighting indicates a superimposition of the rendered shape with the model, at least from the view used to draw the segment.

Normally the z-fighting is a symptom of a weakness of the geometric representation or of the rendering algorithm, therefore if not quantified but only, possibly, avoided. In our approach, however, the z-fighting is an estimation of matching between a shape and the model and therefore we are interested in quantifying it.

Figure 5.(a) shows a schematic example representing the section of a column in the model (shaded in blue) and a section of the shape being fitted (shaded in red). Since they are perfectly superimposed, we see part of the pixels red and part blue, in the proportion which is essentially random and cannot be used directly to quantify the superimposition. However, if we apply a small displacement of the shape towards the viewer we see that all the pixels are red and, vice versa, displacing the shape away from the viewer the pixels will all be blue. In other words, the more the shape and the model are superimposed, the more the two renderings with the displaced shape will be different. Therefore we quantify the z-fighting as:

$$Z_{fight}(Sh, M, T_i) = \frac{\|F_{Sh}(+\epsilon) - F_{Sh}(-\epsilon)\|}{F_{Sh}}$$

where $F_{Sh}(+\epsilon)$ is the number of fragments belonging to the shape when is displaced by $+\epsilon$ and F_{Sh} is the number of fragments of the shape Sh alone. The upper half of Figure 6 shows two examples of a fitted shape, a column and a step, and the segments defined by the user, while in the lower part are shown the plots obtained by setting t (the parameter of the segment with range $[0, 1]$) as ascissa and (Sh, M, t) , so $(Sh, M, 0.5)$ is the value of the z-fighting when the shape is placed

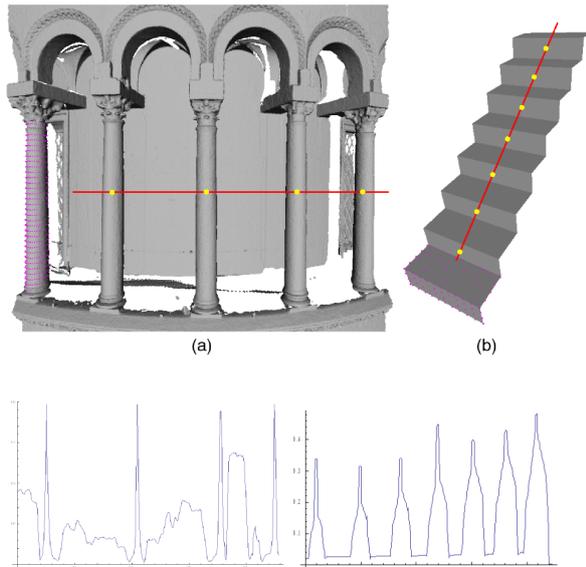


Figure 6: Example of estimation of extrinsic parameters from mouse gestures for a series of columns and a stairs.

on the projection of the middle point of the segment. Quantifying the z-fighting is very efficient because it requires only one rendering of the model and two renderings of the shape for each pixel of the segment, while the number of fragment for the displaced shape are counted by means of the hardware occlusion query.

Note that, being based on the rasterization, this technique is dependent on the window resolution, therefore it will be generally more effective with higher resolutions, simply because more translations are evaluated. It should be clear that the resolution to which we perform the z-fighting computation can be different (higher) than the resolution used by the application for rendering.

5 DEFINING NEW PRIMITIVE TYPES

As stated in Section 3, our framework is not restricted to a given set of primitives but uses an abstraction layer the *sees* a primitive as a sampling of its surface dependent on a set of implicit parameters. Therefore a developer user may add new type of primitive by deriving from a base class Primitive and implementing two methods:

```
struct MyPrimitiveType: public Primitive{
  int N_params();// returns the number of the implicit parameters of the
  primitive

  points Samples(float * params): // returns a sampling of the surface ←
  with the passed parameters
};
```

6 RESULTS AND DISCUSSION.

We tested our framework implementing a few types of primitives, summarized in Table 1.

We fitted the primitives to a scanned model of the Dome of Pisa and reported the timing for various runs in Table 2. Some of the runs are related to the figures we

Name	n.params.	meaning
Column	3	bot. rad., top rad., len.
SquareColumn	3	width, depth, leng.
Stap	3	width, depth, length
Arch	3	radius, angle, depth
Window	3	width, height,depth

Table 1: A few primitives defined to test the framework.

fig	n. pts	nI	nM	extr.(s)	intr.(s)	tot.
1	2M	1	1	53	4.9	58.7
	1.5M	5	1+1	132	18.5	1.51m
	309K	3	3	21.4	16.3	37.82
	122K	1	1	1.4	0.34	1.78
	226K	1	1	1.7	0.6	2.39
7(up)	730K	5	1+1	57.7	4.7	62.5
7(bt)	200K	4	4	12.4	0.43	12.8

Table 2: Time for fitting the primitives. **n. pts**: number of points of the model included in the user hint, **nI**: number of primitives fitted, **nM**: numbers of mouse gestures, **extr.** **intr.** time spent for minimization of extrinsic and intrinsic parameters, respectively, **tot.**: total time

used in the paper, in which case a reference is reported in the first column of the table. The second column reports the size of the portion of the model hinted by the user with the gesture and the third the number of primitives found with the run. The last three columns report the computation time. Note that the time for making the gestures are not reported in this table, since the experiments have been run only by an expert user and therefore not very meaningful. We took the number of mouse gestures as a measure of the user effort.

The table says that, thanks to the replication gesture, 5 architectural elements have been found with 2 gestures (second and sixth row), while where the replication is not used we need a mouse gesture per element, as for the arches referred in the last row. Conversely, indicating manually each and every element gives better starting points for the minimization and therefore the computation time are lower.

Since we performed minimization by alternating minimization of extrinsic parameters, for which we have a closed form solution, and extrinsic parameters, the time spent on each one is reported separately. The result may appear surprising at first, because the easiest side of the problem, i.e. finding the roto-translation between two sets of points, is actually the most expensive, in some cases almost by an order of magnitude. On the other hand, we must consider that the extrinsic step is performed many more times, in that we solve a mesh alignment problem for each iteration. It goes without saying that tweaking the thresholds of the minimization algorithms we may obtain different ratios between the two timings, we simply tuned their values to have robust fitting in reasonable time. The time for

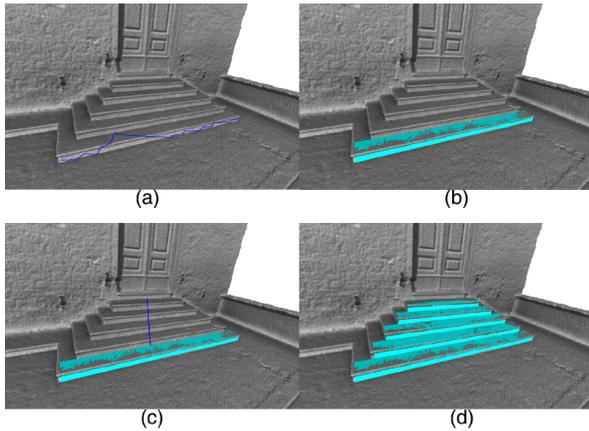


Figure 7: Above: example of selection of stair steps: (a) mouse gesture for the first step (b) fitting (c) mouse gesture for replicating the fitting (d) result. Below: final results for a set of arcs.

the analysis of the mouse gestures are not reported explicitly since they amount to few milliseconds both for the single primitives than for replication with the z-fighting computation.

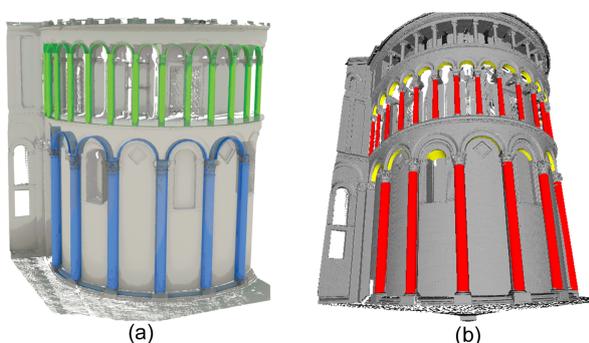


Figure 8: Fitting of columns and arches. (a) results from [USF08] (b) results of our framework on a similar model.

7 CONCLUSIONS

In this paper we presented a framework for user assisted fitting of geometric primitives on scanned architectural models. The main advantage of our framework is the generalized description of the primitive to fit that allows to include new type of primitive with minimal effort. We also devised efficient and practical solutions for enabling the user to hint the approximate position of the primitives, for improving the assessment of primitive models distance with a novel measure and for quantifying the superimposition of primitive and model by exploiting the rasterization hardware. Although requiring user assistance is in general a drawback, we made this choice motivated by two facts: 1) For a human it is very easy to indicate where an architectural component is, while is much more difficult to manually superimpose the CAD model of a component on the raw data; conversely, the analysis of raw data to locate architectural components is computationally time consuming while the minimization for finding the exact placement is an efficient process. 2) The process to digitize and entire building still take many man-hours and the reverse engineering is done once for all in a fraction of the time required for the rest of the scanning pipeline. In other words the little interaction used in this approach is hardly the bottleneck of the whole process.

From the work carried out so far, we can envisage at least two independent improvements.

The first one is to exploit more deeply the z-fight quantification to define a faster minimization algorithm only based on the rendering and therefore taking advantage of the rasterization hardware.

The second one is to derive the parametric primitive directly from a known model, that would allow a non-developer to define new type of primitives. The idea is that the user could provide a sample of the primitive as a geometric model (from a CAD or 3D scanning) and we should derive a parametric description of it, either automatically or providing a tool to do it. In this manner we could include complex shapes for which to find a parameterization is too complicated. With some approximation this would allow to include artifacts as statues when if they are copies of statues for which the digital counterpart is available.

REFERENCES

- [Ben02] BENKO P.: Constrained fitting in reverse engineering. *Computer Aided Geometric Design* 19 (March 2002), 173–205.
- [BM92] BESL P. J., MCKAY N. D.: A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and machine Intelligence* 14, 2 (Feb. 1992), 239–258.
- [BSZF99] BAILLARD C., SCHMID C., ZISSERMAN A., FITZGIBBON A.: Automatic line

- matching and 3d reconstruction of buildings from multiple views. In *ISPRS Conference on Automatic Extraction of GIS Objects from Digital Imagery* (Munich, 1999), pp. 69–80.
- [CM02] COLOMBO L., MARANA B.: 3d building models using laser scanning. *GIM - Geomatics Info Magazine* 16, 5 (2002), 32–35.
- [CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. *ACM Transactions on Graphics* 23 (August 2004), 905.
- [HHW05] HILDEBRANDT K., HILTHIER K., WARDETZKY M.: Smooth feature lines on surface meshes. In *Symposium on Geometry Processing* (2005), pp. 85–90.
- [Lou09] LOURAKIS M. I. A.: *Levenberg-Marquardt nonlinear least squares algorithms in C/C++*. Online, Apr. 2009.
- [MLM01] MARSHALL D., LUKACS G., MARTIN R.: Robust segmentation of primitives from range data in the presence of geometric degeneracy. *IEEE Trans. Pattern Anal. Mach. Intell.* 23, 3 (2001), 304–314.
- [OBS04] OHTAKE Y., BELYAEV A., SEIDEL H.-P.: Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph.* 23, 3 (2004), 609–612.
- [PMW*08] PAULY M., MITRA N. J., WALLNER J., POTTMANN H., GUIBAS L.: Discovering structural regularity in 3D geometry. *ACM Transactions on Graphics* 27, 3 (2008), #43, 1–11.
- [Pow08] POWELL M. J. D.: Developments of NEWUOA for minimization without derivatives. *IMA Journal of Numerical Analysis* 28, 4 (Oct. 2008), 649–664.
- [PXP00] PHAM D. L., XU C., PRINCE J. L.: A survey of current methods in medical image segmentation. In *Annual Review of Biomedical Engineering*, vol. 2. 2000, pp. 315–338.
- [SB03] SCHINDLER K., BAUER J.: A model-based method for building reconstruction. In *First IEEE International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis* (2003), pp. 74–82.
- [SWWK07] SCHNABEL R., WAHL R., WESSEL R., KLEIN R.: *Shape Recognition in 3D Point Clouds*. Tech. Rep. CG-2007-1, Universität Bonn, May 2007.
- [USF08] ULLRICH T., SETTGAST V., FELLNER D. W.: Semantic fitting and reconstruction. *JOCCH* 1, 2 (2008).
- [WB01] WATANABE K., BELYAEV A. G.: Detection of salient curvature features on polygonal surfaces. *Comput. Graph. Forum* 20, 3 (2001).

Real time accurate collision detection for virtual characters

Andoni Mujika, David Oyarzun, Aitor Arrieta, María del Puy Carretero

VICOMTech - Visual Interaction and Communication Technologies Center

Mikeletegi Pasealekua, 57 - Parque Tecnológico

E-20009 Donostia - San Sebastián, Spain

{amujika, doyarzun, aarrieta, mcarretero}@vicomtech.org

ABSTRACT

This paper presents an accurate real time collision detection algorithm for interactively animated virtual characters using sphere-trees as Bounding Volume Hierarchies. We build upon a fast mathematical method for on-demand sphere refitting during the animation and improve it for being applicable to any object, without dependency on its geometrical level of detail or its dynamic/static behavior. It uses sphere-plane intersection test as the exact test in the collision detection algorithm instead of the usual triangle-triangle one. Corner-trees, a special hierarchy that ensures the utilization of the plane-sphere intersection test is right, are also presented. In the worst case, the optimization decreases in 25% the time needed to process a frame in extreme conditions. The algorithm has been successfully tested on a real time and collaborative 3D virtual world.

Keywords

Real-time Collision Detection, Bounding Volume Hierarchies, Virtual Character Animation.

1. INTRODUCTION

Collision detection (CD) is a key issue in almost all fields of computer graphics. Real time virtual objects and virtual characters' animation are not exceptions. In most of cases they need to have realistic behaviors that imply CD, i.e. not to penetrate other objects. Therefore, many algorithms have been proposed in recent years.

Accurate algorithms are usually very expensive computationally speaking. Then, applications that make use of collision detection algorithms have to balance between preciseness of the detection and velocity of the algorithm.

For instance, a very fast performance of the collision detection algorithm is needed in Massively Multiplayer Online Games (MMOGs) and a very precise detection is crucial in serious games and virtual prototyping. On the other hand, continuous collision detection (CCD) was presented to solve the main problem that discrete algorithms presented, the tunneling effect, i.e. the miss of some collisions. However, the velocity of the algorithm obtained was

not appropriate for real time purposes.

Although the problem has been widely studied for rigid bodies, there is a lot to do regarding CD for real time deformable objects such as clothes, interactive virtual humans, etc. The problem increases in case of collaborative virtual worlds, with lot of avatars interacting among themselves and with objects at the same time. It is usual to see very fast but imprecise CD algorithms.

Therefore, in this article, we focus on real time humanlike animation in collaborative virtual worlds and propose an algorithm to obtain a fast and precise CD for interactive virtual characters of high level of detail. In order to be used in both, virtual worlds and precise simulations, the algorithm is based on these features:

- A fast update of the spheres in the Bounding Volume Hierarchy.
- Utilization of the sphere-plane intersection test instead of the slower triangle-triangle test.
- Implementation of the corner-trees, a novel hierarchy for the correct and fast performance of the algorithm.

The paper is structured as follows. In section 2 we summarize the related work. In section 3 we describe our virtual character animation platform and its collision detection algorithm. In section 4 we analyze the major problem we found for a fast performance of the algorithm and in section 5 we describe two

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

methods to solve it. In section 6 we present and compare the obtained results and finally, in section 7, we analyze future extensions to improve the performance of our CD algorithm.

2. RELATED WORK

When detecting collision detection between a virtual character and its environment, first, the character's movement has to be computed, i.e. the new position of the avatar's vertices has to be calculated. And then, the second stage will be the CD itself, taking into account the new positions of the vertices. Sections below deal with related work in each of these stages.

Virtual Character Animation

Virtual character animation has been widely studied in computer graphics. In this research field, one of the main goals is the realistic simulation of human movements. Especially in 3D animation, many efforts have been done in recent years. Although there are some methods such as Blend Shape Deformation [Moh03a] and Free-Form Deformation [Sed86], skeletal animation systems are the most used. The primitives that form the virtual character are transformed depending on the movements of a skeleton. We can classify these methods by the way they skeleton affects the primitives. Linear Blend Skinning [Moh03b] manipulates the triangle-mesh associating each vertex to a group of joints of the skeleton and giving a weight for each joint (the sum of the weights is one). Then, the transformation of the vertex is a linear combination of the joints' transformations. Spherical Blend Skinning [Kav05a] works similarly, but the relation between joints' transformations and vertices' transformations is not linear. It is based on Spherical Linear Interpolation.

Collision Detection

When detecting collisions between two objects, testing each primitive-couple is too costly. Detecting collisions between two objects with n and m primitives would cost $n * m * b$ operations, where b is the number of basic operations needed in an intersection test between primitives. Therefore, a method that detects which primitives are more likely to be colliding (broad phase) is used before executing the exact test between primitives (narrow phase) [Mol97, Tro05].

Usually, Bounding Volume Hierarchies (BVHs), i.e., sets of volumes that bound the object getting different levels of tightness, are used in the broad phase. During the collision detection, the volumes in the hierarchies of the objects are tested to be colliding. If they don't collide, all the primitives inside the volumes don't collide, but if they do collide, next levels of tightness are checked. Once the algorithm finds two colliding leaf-nodes, i.e. volumes

that enclose only one primitive, the exact intersection test between primitives is called.

The number of operations needed to detect collisions between bounding volumes is much lower than between primitives. For instance, a collision test between spheres consists of 10 operations and the best collision test between triangles consists of 96 operations.

We can sort these methods by the type of volume they use:

- Spheres [Qui94, Hub96]
- Axis-Aligned Bounding Boxes (AABBs) [Van98]
- Oriented Bounding Boxes (OBBs) [Got96],
- k-Discrete Orientation Polytopes (k-DOPs) [Klo98]

Most of these methods were presented for collision detection between rigid objects. Nevertheless, CD for deformable objects also makes use of BVHs. Once again, different types of BVHs appear such as spheres [Bro01] and AABBs [Lar01, Zac06].

Regarding collision detection for avatars, i.e. virtual characters, there have been different approaches in recent years. Kavan et al. use spheres to create the BVH. They refit the sphere-tree for bodies that are moved based on a skeleton. They proposed collision detection methods for Linear Blend Skinning [Kav05b] and Spherical Blend Skinning [Kav06].

All the results shown so far are discrete, i.e. they sample objects' motions. As opposed to these methods, continuous collision detection (CCD) methods compute the first time of contact during the collision detection. Six different approaches to CCD have been presented in the literature: algebraic equation-solving [Cho06], swept volumes [Abd02], adaptive bisection [Red02], kinetic data structures (KDS) [Aga01], the configuration space approach [Van04], and conservative advancement [Cou06]. However, these methods performance is not as fast as is required.

There are also some continuous collision detection results for avatars. Zhang et al. [Zha07] use OBB-trees and create AABBs during the motion interpolation using Taylor Models, i.e. a generalization of interval arithmetic. Instead, Redon et al. [Red04] use swept volumes (SV) for CCD in scenes with a simple articulated avatar.

3. ALGORITHM OVERVIEW

The developed collision detection algorithm is a discrete collision detection method and works with spheres as bounding volumes. Spheres were chosen because of the fast performance of the sphere-sphere

intersection test and the low space needed to store the data.

Virtual Character Animation

Regarding the animation stage, in our system, the vertices are associated to a unique joint and the transformation of a vertex is obtained computing the product of the transformations of all joints upon the associated joint in the skeleton-tree and the weighted transformation of the associated joint.

$$C_v = \left(\prod_{i \in I} C_i \right) * w_a * C_a$$

where I is the group of joints upon the associated joint in the skeleton-tree, C_v the transformation of the vertex, C_i s the transformations of the joints in I , C_a the transformation of the associated joint and w_a the weight associated to the vertex. This way of animation provides an adequate balance between performance and realism for its use in collaborative virtual worlds.

Collision detection

The collision detection algorithm begins with the sphere-tree construction. This construction of the sphere-tree is based on Quinlan's work [Qui94]. First a binary tree is constructed: in each step, the triangles of a sphere are divided in two groups and two spheres are constructed enclosing each group [Gae99]. In this case, to make the division, the triangles are ordered depending on their position in one of the axes, so as to get two spheres as far as possible one from the other. Moreover, the axis is chosen to be the one where the spheres are most spread. As in [Kav05b], the binary tree is turned into a n-ary tree eliminating the spheres the radius of which is similar to their parent's radius. This way, when testing for collision, tests between similar spheres are avoided.

Since each vertex is associated to a single joint in our platform, instead of creating a unique tree, a tree is constructed for the group of vertices associated to each joint, so as to prevent the algorithm having spheres affected by no-adjacent-joints. To merge all the trees, an enclosing sphere for all vertices is computed as the root of the main tree and a sphere for each extremity to form the second level are created.

Sphere update

The sphere update of our algorithm is inspired by the main contribution of Kavan and Zara [Kav05b]. In the preprocess, all the vertices of a sphere are visited to compute the minimum and maximum weights for each joint affecting this sphere.

Then, during the animation, when a joint is visited to update the vertices associated to it, the spheres containing vertices associated to this joint are also visited. For each visited sphere, two new spheres (one if maximum and minimum weights are the same) are created applying the same transformation as to the vertices to the center of the sphere but using the maximum and minimum weights. The radii are the same as the original sphere.

$$c_{min} = \left(\left(\prod_{i \in I} C_i \right) * min_s * C_a \right) * c_s$$

$$c_{max} = \left(\left(\prod_{i \in I} C_i \right) * max_s * C_a \right) * c_s$$

where c_s is the center of the sphere, c_{min} and c_{max} are the new centers and max_s and min_s are the precomputed maximum and minimum.

Finally, the enclosing sphere of the new spheres is created, ensuring that all the vertices are inside the new sphere.

$$c_u = \frac{\sum_{i=1}^n c_i}{n}$$

$$r_u = \max_{1 < i < n} (\|c_u - c_i\| + r_i)$$

where c_u and r_u are the center and the radius of the final updated sphere and c_i and r_i are the centers and the radii of the spheres obtained with all the maximum and minimum weights.

Narrow phase

During the collision detection, when two spheres in the lowest level of the hierarchies are colliding, an exact collision test between the triangles enclosed by those spheres is called. We use the fast algorithm presented by Tropp et al. [Tro05]. When detecting intersection between edges of a triangle and the other triangle, all the redundant operations to calculate determinants are discarded.

4. OPTIMIZATION

Since we want our platform to cope with virtual characters containing more than 40000 vertices, the algorithm needs some optimization. It has to be able to handle the big amount of spheres generated with this number of vertices.

In order to reduce the number of triangles that take part in the collision detection algorithm, we implemented an optimization proposed by Curtis et al. [Cur08]. They realized that many collision tests

between primitives are made more than once and developed a method to avoid these duplications. In our case, we assume that each edge of the triangle-mesh has to be tested once. Then, taking into account a triangle surrounded by three triangles that have already been taken into account is not necessary (see Figure 1). Therefore, we assume this triangle doesn't exist for the collision detection. One may think that some collision may be skipped this way. In fact, the penetration of a smaller triangle in a "not existing" triangle without touching its edges wouldn't be detected, but we have seen that in practice, this extreme case doesn't occur with avatars of so high level of detail. After implementing the optimization, the number of triangles used for the collision detection decreased 40%.

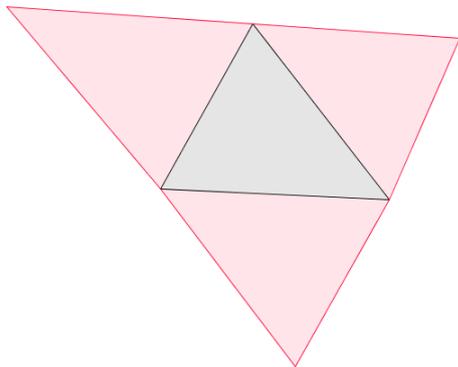


Figure 1 The triangle among the other 3 triangles is not taken into account in the CD algorithm.

Big triangles, a problem

When an avatar is walking in an environment, usually the triangles that compose the environment (walls, tables, windows, etc.) are much bigger than the ones that compose the avatar. This fact is a serious drawback when trying to get a fast performance of the collision detection system.

The leaf-node of the sphere-tree that corresponds to a big triangle is a big sphere. So, when the avatar is near a big triangle, it's possible that all the spheres in the BVH of the avatar are inside the big enclosing sphere of the triangle. This leads to a huge number of collision tests between spheres and a huge number of exact collision tests between triangles, since all the leaf-nodes of the avatar hierarchy are inside the leaf-node of the environment. We have checked that the algorithm can't cope with this number of operations, especially because of exact tests.

5. SOLVING BIG TRIANGLES' PROBLEM

A solution for the problem with big triangles could be just to divide big triangles in smaller triangles. Nevertheless, it is not always possible to manipulate the model received and dividing all big triangles until the leaf-nodes are small enough can increase the weight of the model drastically.

From now on, we denote the small triangle in the avatar's triangle mesh that takes part in an exact intersection test as t and the big triangle of the object in the environment as T . We denote their enclosing spheres, i.e. their leaf-nodes in the sphere-tree as S_t and S_T respectively.

Sphere Division

Although the division of the model's vertices may be impossible to carry out, a similar approach can be applied.

We want the avatar not to be inside S_T . So, we create a hierarchy inside the enclosing sphere of the big triangle to ensure that when exact test is called the primitives are really close. When the triangle is too big (we use a border value for the lengths of the edges), the triangle is divided in four new triangles joining the intermediate points of the edges and four new enclosing spheres are created to form the next level in the hierarchy (see Figure 2). The division finishes when the triangles are smaller than the threshold.

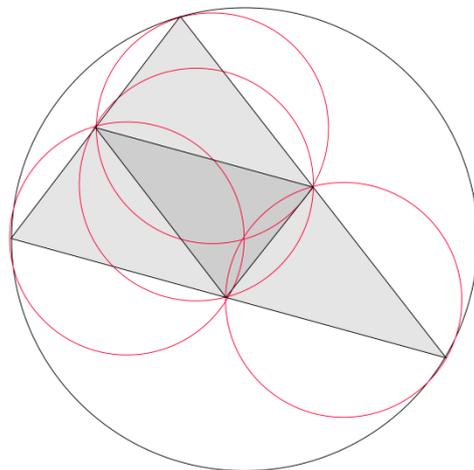


Figure 2 The enclosing sphere (black) of a triangle. The triangle divided in four triangles and their enclosing spheres (red).

During the animation, the collision detection algorithm runs as before, calling the spheres of the lower levels if the ones in upper levels collide, but in this case, the leaf-nodes doesn't enclose a triangle. They point to the big triangle T .

This way, when the avatar is not really close to T , only intersection tests between spheres are called. So the algorithm's performance is much faster. Moreover, when the avatar is close to the object that contains T , the exact collision test is only called for those triangles that are really close, avoiding the huge number of exact intersection test we had before.

The results obtained with this implementation were satisfactory, but we saw that a better performance could be obtained. Results will be shown in section 6.

Plane-Sphere intersection test

Virtual characters with high level of detail are composed by very small triangles comparing with the triangles that compose some objects of the environment. When the enclosing sphere S_t , of the small triangle, t , is colliding with a big triangle, T , t is colliding with T or it is very near. Therefore, testing t and T and testing S_t and T are nearly the same.

If the intersection test between a triangle and a sphere is not very costly, it is worth to use it instead of the exact test between two triangles. Nevertheless, we can see in [Eri05] that the sphere-triangle test is quite costly.

However, a simple and very efficient collision test between spheres and planes is presented in [Eri05] (see Algorithm 1) and it seems that T can be considered as a plane when testing with S_t . That way, the biggest bottle-neck in our algorithm would be solved due to the substitution of the exact test between triangles.

```
bool SpherePlaneTest(sphere s, triangle t){
    edge1 = t.v1 - t.v0;
    edge2 = t.v2 - t.v0;
    p = edge1 × edge2;
    n = s.center - t.v0;
    return ( |p · n| < s.radius * ||p|| );
}
```

Algorithm 1 Plane-sphere intersection test for sphere S and triangle t

The problem of this substitution is that it is usual to find a leaf-node in the hierarchy of the avatar inside S_T which is not colliding with T , but colliding with the plane defined by T . This leads to a not existing collision detection.

So, before calling the plane-sphere collision test, we have to ensure that the sphere S_t is in front of the triangle T and it is not in the part of the enclosing sphere that the triangle doesn't occupy.

Working as in the latest subsection, we can create a quaternary tree inside the enclosing sphere S_T . Then, when the collision detection algorithm reaches leaf-nodes and calls the plane-sphere test, we can be sure that the sphere is in front of T and we can consider it as a plane.

Besides, a smaller tree than the quaternary-tree can be used without losing any property. For instance, when dividing the triangle in four smaller triangles, we can assume that if S_t is colliding with the enclosing sphere of the central triangle it is in front of T .

So, in the algorithm that recursively creates the hierarchy inside S_T , only triangles that have an edge that matches one of T 's edges are divided into four new triangles again. We call the new hierarchy Corner-tree (see figure 3).

One may think that it is better to continue dividing the central triangles, because of the higher cost of the plane-sphere test. Nevertheless, it is less costly one plane-sphere test (30 operations) than four sphere-sphere tests (4x10 operations).

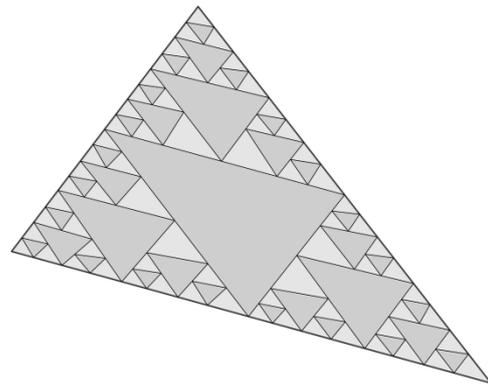


Figure 3 Division of a triangle to create the corner-tree.

Moreover, the number of the spheres in the hierarchy decreases drastically with this new algorithm. If n is the number of levels of the hierarchy, in the quaternary-tree the number of spheres in the n th level is 4^n . A huge number comparing to the new algorithm, which creates $12 * (2^{n-2} - 1)$ spheres in each level (see table 1).

Level	1	2	3	4	5	6	7
Corner-tree	1	4	12	36	84	180	372
Quaternary	1	4	16	64	256	1024	4096

Table 1 Number of spheres in the n th level in the Corner-tree and in the quaternary-tree

6. RESULTS

The collision detection algorithm has been applied in our platform for the animation of virtual characters in collaborative virtual worlds successfully [Oya07] (see Figure 4).

We have checked our algorithm with an avatar composed by 44345 vertices in two different scenarios: a virtual museum with 9482 vertices and virtual living-room with 133139 vertices.

Both virtual worlds have triangles that are bigger than the avatar and we have checked the performance of the algorithm in extreme conditions, i.e. when the avatar is very close to these triangles without colliding. Moreover, the collision detection algorithm was run without any optimization and with the sphere division optimization to compare them with the latest version. All the tests were made with an Intel Core 2 Duo CPU at 2.20 GHz.



Figure 4 A virtual character in a virtual living-room.

First, we counted the basic operations (sum and multiplication) needed in each basic collision test: 10 operations in the sphere-sphere test, 96 in the triangle-triangle test [Tro05] and 30 in the plane-sphere test. Then, we ran the animation platform counting the number of these basic tests per frame so as to obtain the maximum number of operations made in a frame.

Table 2 and table 3 show the results obtained. The space needed to store sphere hierarchies, maximum times the intersection tests are called in one frame, the sum of basic operations in those maxima and the duration of the frame in the case of maximum operation.

In both cases, the space to store the information about the sphere hierarchies is much bigger when the algorithm has an optimization. Nevertheless, the space needed is not big enough to be a problem. As we stated before, we can see that the corner-tree is smaller than the quaternary-tree.

Virtual Museum	Original	Sphere Division	Plane-Sphere
Data	548 KB	65 MB	24 MB
Sph-Sph tests	≈200000	≈15000	≈5000
Tri-Tri tests	≈150000	≈150	
Pla-Sph tests			≈2000
Operations	≈17000000	≈165000	≈110000

Table 2 Results obtained for the animation in the virtual museum.

Living-room	Original	Sphere Division	Plane-Sphere
Data	3.21 MB	49.2 MB	37.5 MB
Sph-Sph tests	≈80000	≈80000	≈8000
Tri-Tri tests	≈70000	≈50000	
Pla-Sph tests			≈275
Operations	≈7000000	≈5000000	≈90000

Table 3. Results obtained for the animation in the virtual living-room.

As wished, sphere division optimization decreases the number of exact tests, especially in the virtual museum. This leads to a decrease in the duration of a frame.

Moreover, the plane-sphere optimization decreases the number of tests made in both the broad phase and the narrow phase. Combining this with the lower complexity of the plane-sphere collision test, we obtain a very fast performance.

In conclusion, we can see in the tables that increasing the stored data, i.e. creating bigger sphere hierarchies, we can decrease the time spent detecting collisions. In the virtual museum, the difference between the optimizations is not considerable, but in the living room, the time gained with the plane-sphere optimization is twice as the time gained with the sphere division optimization.

7. CONCLUSIONS AND FUTURE WORK

This article presents a fast and precise collision detection algorithm for real-time virtual character animation.

The utilization of the intersection test between a sphere and a plane instead of the triangle-triangle test resulted in a much faster performance of the algorithm.. We also presented the corner-tree, a novel sphere hierarchy that makes the algorithm detect collisions correctly.

We implemented the algorithm in a virtual world composed of several interactive avatars of high level of detail and objects of different levels of detail. The velocity obtained is fast and the collision detection is precise enough. We also implemented the collision detection for an online version of our platform.

Since discrete collision detection methods sometimes miss collisions (tunneling effect), continuous collision detection is becoming an important topic of research. Most of the new CCD methods are based on discrete methods, so it seems natural to try to convert our contribution into a CCD algorithm.

In recent years, the utilization of the GPUs has become very important when accelerating algorithms' performance. Since collision detection is one of the most important bottle-neck in animation, it is important to study how GPUs can accelerate the collision detection.

8. REFERENCES

- [Abd02] Abdel-Malek, K., Blackmore, D. and Joy, K. Swept volumes: foundations, perspectives, and applications. *International Journal of Shape Modeling*. 2002.
- [Aga01] Agarwal, P. K., Basch, J., Guibas, L. J., Hershberger, J. and Zhang, L. Deformable free space tiling for kinetic collision detection. In *Workshop on Algorithmic Foundations of Robotics*, 83–96. 2001.
- [Bro01] Brown, J., Sorkin, S., Bruyns, C., Latombe, J. C., Montgomery, K. and Stephanides, M. Real-time simulation of deformable objects: Tools and application. *Computer Animation 2001*, 2001.
- [Cho06] Choi, Y.-K., Wang, W., Liu, Y. and Kim, M.-S. Continuous collision detection for elliptic disks. *IEEE Transactions on Robotics* 22, 2. 2006
- [Cou06] Coumans, E. *Bullet Physics library*. <http://www.continuousphysics.com>. 2006.
- [Cur08] Curtis, S., Tamstorf, R. and Manocha, D. Fast collision detection for deformable models using representative-triangles. *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, 61-69. 2008
- [Eri05] Ericson, C. *Real-time Collision Detection*. The Morgan Kaufmann Series in Interactive 3-D Technology. 2005
- [Gae99] Gaertner B.: Fast and robust smallest enclosing balls. In *ESA '99: Proceedings of the 7th Annual European Symposium on Algorithms*, Springer-Verlag, pp. 325–338. 1999.
- [Got96] Gottschalk, S., Lin, M. C. and Manocha, D. Obb-tree: A hierarchical structure for rapid interference detection. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171 – 180, 1996.
- [Hub96] Hubbard, P.M. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics (TOG)*, Volume 15 , Issue 3:179 – 210, 1996.
- [Kav05a] Kavan L. and Zara J. Spherical blend skinning: a real-time deformation of articulated models. *Proceedings of the 2005 symposium on Interactive 3D graphics and games*. 9 – 16. 2005.
- [Kav05b] Kavan, L. and Zara J. Fast collision detection for skeletally deformable models. *Computer Graphics Forum*, 2005.
- [Kav06] Kavan, L., O'Sullivan, C. and Zara, J. Efficient collision detection for spherical blend skinning. *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia, Fast graphics*:147 – 156, 2006.
- [Klo98] Klosowsky, J. T., Held, M., Mitchell, J.S.B., Sowizral, H. and Zikan, K. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Transactions on Visualization and Computer Graphics*, Volume 4 , Issue 1:21 – 36, 1998.
- [Lar01] Larsson, T. and Akenine-Möller, T. Collision detection for continuously deforming bodies. *Eurographics*, pages 325–333, 2001.
- [Moh03a] Mohr, A. and Gleicher, M. Building Efficient, Accurate Character Skins from Examples, *ACM Trans. Graph.*, Vol. 22, No. 3, pp. 562-568. 2003.
- [Moh03b] Mohr, A., Tokheim L. and Gleicher M. Direct manipulation of interactive character skins. *Proceedings of the 2003 symposium on Interactive 3D graphics*. 27 – 30. 2003.
- [Mol97] Möller, T. A fast triangle-triangle intersection test. *journal of graphics tools*, 2(2):25–30, 1997.
- [Oya07] Oyarzun, D., Lehr, M., Ortiz, A., Carretero, M. P., Ugarte, A., Vivanco, K. and García-Alonso, A. Using Virtual Characters as TV Presenters. *Technologies for E-Learning and Digital Entertainment*, 225-236. 2007.
- [Qui94] Quinlan, S. Efficient distance computation between non-convex objects. *International Conference on Robotics and Automation*, 1994.
- [Red02] Redon, S., Kheddar, A. and Coquillart, S. Fast continuous collision detection between rigid bodies. *Proc. Of Eurographics (Computer Graphics Forum)*. 2002.
- [Red04] Redon, S., Kim Y.J., Lin, M.C., Manocha, D. and Templeman, J. *Interactive and Continuous*

- Collision Detection for Avatars in Virtual Environments. IEEE Virtual Reality Conference 2004 (VR 2004). 2004.
- [Sed86] Sederberg, T. W. and Parry, S.R. Free-form deformation of solid geometric models, In SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques, ACM Press, pp. 151-160. 1986.
- [Tro05] Tropp, O., Tal, A. and Shimshoni, I. A fast triangle to triangle intersection test for collision detection. Journal of Graphics Tools, Volume 2 , Issue 2:25 – 30, 2005.
- [Van98] Van Den Bergen, G. Efficient collision detection of complex deformable models using aabb trees. Journal of Graphics Tools, Volume 2 , Issue 4:1 – 13, 1998.
- [Van04] Van Den Bergen, G. Ray casting against general convex objects with application to continuous collision detection. Journal of Graphics Tools. 2004.
- [Zac06] Zachmann, G. and Weller, R. Kinetic bounding volume hierarchies for deformable objects. Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications, Session F5:189 – 196, 2006.
- [Zha07] Zhang, X., Redon, S., Minkyong, F. and Kim, Y.J. Continuous collision detection for articulated models using Taylor models and temporal culling. International Conference on Computer Graphics and Interactive Techniques. 2007

Fast Approximate Visibility on the GPU using pre-computed 4D Visibility Fields

Athanasios Gaitatzes	Anthousis Andreadis	Georgios Papaioannou	Yiorgos Chrysanthou
University of Cyprus 75 Kallipoleos St. P.O.Box.20537 Cyprus (CY-1678), Nicosia	Athens University of Economics & Business 76 Patission St. Greece (10434), Athens	Athens University of Economics & Business 76 Patission St. Greece (10434), Athens	University of Cyprus 75 Kallipoleos St. P.O.Box.20537 Cyprus (CY-1678), Nicosia
gaitat@yahoo.com	anthousis@gmail.com	gepap@aueb.gr	yiorgos@cs.ucy.ac.cy

ABSTRACT

We present a novel GPU-based method for accelerating the visibility function computation of the lighting equation in dynamic scenes composed of rigid objects. The method pre-computes, for each object in the scene, the visibility and normal information, as seen from the environment, onto the bounding sphere surrounding the object and encodes it into maps. The visibility function is encoded by a four-dimensional *visibility field* that describes the distance of the object in each direction for all positional samples on a sphere around the object. In addition, the normal vectors of each object are computed and stored in corresponding *fields* for the same positional samples for use in the computation of reflection in ray-tracing. Thus we are able to speed up the calculation of most algorithms that trace rays to real-time frame rates. The pre-computation time of our method is relatively small. The space requirements amount to 1 byte per ray direction for the computation of ambient occlusion and soft shadows and 4 bytes per ray direction for the computation of reflection in ray-tracing. We present the acceleration results of our method and show its application to two different intersection intensive domains, ambient occlusion computation and stochastic ray tracing on the GPU.

Keywords

indirect lighting, pre-computed visibility, uniform distribution, hemisphere, tracing rays.

1. INTRODUCTION

The acceleration of the computation of the lighting equation in real-time on the GPU and especially the visibility term, one of the most intensive parts of the computation, is still a very active field of research. Ambient occlusion computation and real-time ray tracing are just two of the fields where the fast computation of the visibility queries is very important.

Ambient occlusion is defined as the attenuation of ambient light due to the occlusion of nearby geometry. It gives perceptual clues of depth, curvature, and spatial proximity and thus is important for realistic rendering. It is a technique that

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

approximates the effect of indirect global illumination without trying to simulate the interplay of incident and reflected light.

Ray tracing is a general and versatile algorithm that performs image synthesis by shooting rays through each pixel, finding the closest intersection with the scene geometric entities. The generic backwards ray tracing algorithm is capable of capturing both local illumination and basic indirect specular effects such as mirror-like reflections and refraction.

In this paper we improve and expand the method proposed by Gaitatzes et al. [Gai08] by moving the implementation to the GPU, taking advantage of the shader units parallelism and demonstrating significant performance gains. While the core of the visibility queries mechanism remains the same, the paper shows how the method is adapted to both interoperate with a generic ray tracing system and accelerate the generation of high quality ambient occlusion. First, at pre-processing time, we construct the visibility field (Figure 1). It stores the intersection distances of a hemisphere of rays originating from sample points on the bounding sphere of an object

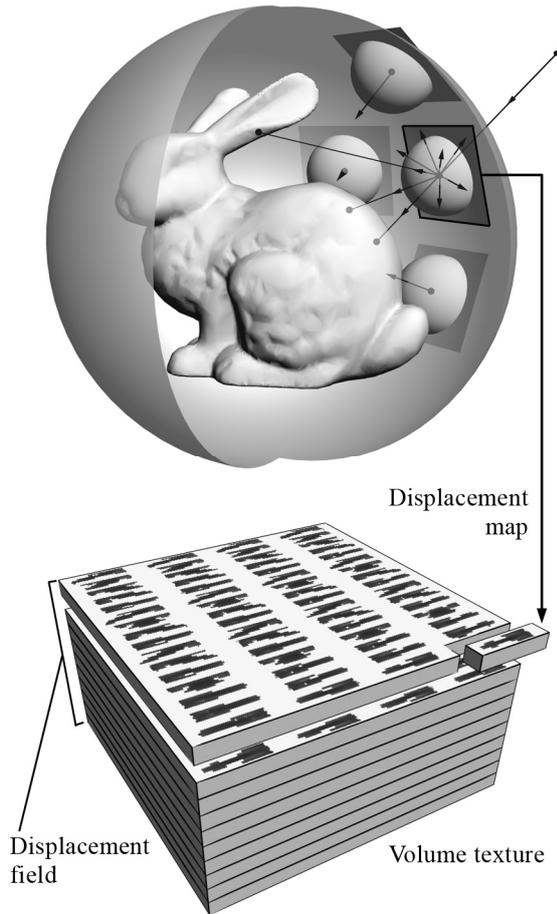


Figure 1: A hemisphere of rays emanating from the bounding sphere towards the object is pre-computed for a large number of sample points on the sphere. Bottom: Volume texture of the visibility field. Row by row each map is placed into a slice of the volume texture thus minimizing the volume space requirements. As a result a 512^3 volume will hold four 256^2 maps per slice.

and directed towards the model itself. We construct one map for each sample point (see Section 3.1). After the construction of the visibility field maps, we compactly fit them in one volume texture (see Section 4.1) for easy access on the GPU. In addition, all mesh information (i.e. coordinates, normals and materials) are stored in maps and passed on to the GPU. Then, at run time, when a ray from the environment towards an object (or vice versa) intersects its bounding sphere, we perform a simple ray-sphere intersection test and recover from the pre-computed maps the rest of the ray distance for the ray-object intersection test.

The advantage of the method described in Gaitatzes et al. [Gai08] is that the bulk of the computation is moved to a pre-processing stage. The results are stored in compact gray-scale textures; 1 byte per ray direction for the computation of ambient occlusion

and soft shadows and 4 bytes per ray direction for the computation of reflection in ray-tracing, providing for each object a constant size of additional information, independent of the complexity of the original model. Then the real-time algorithm performs a simple intersection test with the bounding sphere of the object and a constant-time map lookup (see Section 3.2).

For dynamic scenes with rigidly moving objects, visibility fields accelerate the computation of the approximation of the indirect lighting term of the rendering equation to real-time frame rates as well as the computation of soft shadows and reflection in ray-tracing. The performance of this approach does not depend on the polygon count to a large extent; instead, it is directly related to the number of visible pixels shaded by the GPU. This is a significant advantage over existing approaches. In addition, our acceleration structure is flat by nature and thus more suited to the GPU architecture.

In Section 2 we give an overview of the previous work, followed by a description of our method in greater detail in Section 3. In Section 4 we discuss the GPU implementation and in Section 5 our results from the application of the visibility fields method in ray tracing and especially the benefit of shadow rays and secondary rays as well as secondary diffuse illumination (termed ambient occlusion).

2. BACKGROUND AND PREVIOUS WORK

We distinguish the previous work in two areas that both share the computation of the visibility function; the acceleration of the computation of ambient occlusion on the GPU and the acceleration of stochastic ray tracing algorithms on the GPU. Note that we apply our method only to a GPU-based ray tracing algorithm in order to compare timings with the fastest approach.

2.1 Ambient Occlusion on the GPU

In ambient occlusion the indirect component can be computed as:

$$A(\mathbf{x}, \bar{\mathbf{n}}) = \frac{1}{\pi} \int_{\Omega} V(\mathbf{x}, \bar{\omega}_o) [\bar{\omega}_o \cdot \bar{\mathbf{n}}] d\bar{\omega}_o$$

Where $V(\mathbf{x}, \bar{\omega}_o)$ is an empirical function that maps distance from surface point \mathbf{x} to the closest surface along direction $\bar{\omega}_o$ to visibility values between 0 (no occlusion) and 1 (full occlusion).

By tracing rays outward from a given surface point \mathbf{x} over the hemisphere around the normal $\bar{\mathbf{n}}$, ambient occlusion measures the amount that a point is obscured from light. This average occlusion factor is used to simulate soft-shadowing.

Ambient occlusion (AO) computation on the GPU was first used by Bunnell [Bun05], who approximates the AO by modelling the receiver surface as disk-based occluders and evaluates the ambient occlusion caused by the disks using an analytic method. He uses a heuristic method to combine the shadows cast from multiple disks into a noise free image but requires high tessellation of scene geometry and a big pre-computation step.

Shanmugam et al. [Sha07] compute ambient occlusion as a post-processing pass based on a depth buffer from the eye's point of view. They split the AO computation into two phases, one for high frequency near detail, and another phase for low frequency detail with a wider search. The second phase allows large objects to inter-occlude as they pass next to each other. Their approach requires no scene-dependent pre-computations. On the downside, over occlusion artefacts might show up when multiple neighbouring spheres contribute occlusion to the same pixel.

Mittring [Mit07] does a full screen post-processing pass where z-buffer data is sampled around each pixel and an AO value is computed based on depth differences. Sampling occurs randomly in a sphere around each pixel, and AO is proportional to the number of sampled occluders. Like other screen space techniques, such as [Bav09], this view-dependent approach is fast, requires minimal or no pre-calculation, but cannot model AO correctly, because depth discontinuities, such as object edges and buffer boundaries, produce popping effects.

2.2 Real-time Ray Tracing on the GPU

Most GPU ray-tracing methods accelerate already established mechanisms for limiting the number of intersection tests. On the other hand, our approach provides an alternative and fast ray-surface intersection test, while it can certainly take advantage of the mentioned methods, to further improve final performance.

Carr et al. [Car02], Purcell et al. [Pur02], [Pur04], Karlsson et al. [Kar04] and Christen et al. [Chr05] implemented a streaming ray-triangle kernel on the GPU, fed by buckets of coherent rays and proximate geometry organized by a CPU process. However, there was a frequent communication of results from the GPU to the CPU over a narrow bus, negating much of the performance gained from the GPU kernel. Streaming geometry to the GPU became quickly the bottleneck.

To improve the performance of the GPU ray tracing, different acceleration structures have been widely adopted, such as the incorporation of kd-trees by Havran [Hav00] and Ernst et al. [Ern04]. However, these approaches had limited performance; by far not

reaching the frame rates of the CPU based ray tracers. The main problem was the limited GPU architecture. Only small kernels without branching were supported. In addition a stack was usually required, which was poorly supported on GPUs. Foley et al. [Fol05] presented two implementations of a stack-less kd-tree traversal algorithm for the GPU, namely kd-restart by Kaplan [Kap85] and kd-backtrack. Foley showed, that on graphics hardware, there are scenes for which a kd-tree yields far better performance than a uniform grid. Although better suited for the GPU, the high number of redundant traversal steps led to relative low performance.

Besides grids and kd-trees there are also several other approaches that use a BVH as an acceleration structure on the GPU. Carr et al. [Car06] implemented a limited ray tracer on the GPU that was based on geometry images but it required careful parameterization of the geometry. It could only support a single triangle mesh without sharp edges. The acceleration structure was a predefined bounding volume hierarchy which could not adapt to the topology of the object. To alleviate the need for a stack Thrane et al. [Thr05] presented stack-less traversal algorithms for a BVH. They conclude that on the GPU, the bounding volume hierarchy traversal method is up to 9 times faster than that of a uniform grid and a kd-tree. Also, the technique proves the simplest to implement and the most memory efficient.

Horn et al. [Hor07] reduced the number of redundant traversal steps of kd-restart by adding a short stack. With their implementation on modern GPU hardware they achieved a high performance of 15–18M rays/s for moderately complex scenes. At the same time, Popov et al. [Pop07] presented a parallel, stack-less kd-tree traversal algorithm without the redundant traversal steps of kd-restart but with a poor GPU utilization of below 33%. With over 16M rays/s, their GPU ray tracer achieved similar performance as CPU based ray tracers. However, both GPU ray tracing implementations demonstrated only medium-sized, static scenes. Günther et al. [Gün07] presented a BVH based GPU ray tracing method for large models achieving close to real time rates using hard shadows.

3. APPROXIMATE VISIBILITY COMPUTATION

The computation of exact visibility is a time consuming task even for the new GPU architectures. We briefly describe here the visibility field acceleration method that follows that of Gaitatzes et al. [Gai08] but emphasizing the GPU architecture.

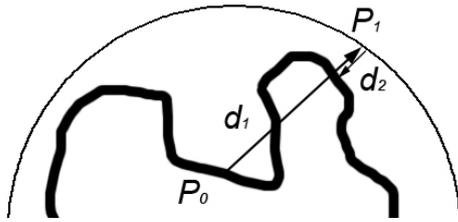


Figure 2: Visibility computation for intra-object occlusion.

3.1 Visibility Field Computation

The main idea of encoding visibility fields into maps is as follows. Consider a rigid object possibly moving through a scene. At a pre-processing step, from a discrete set of sample points on the objects bounding sphere, described as spherical coordinates (u, v) , a hemisphere of rays is cast around the inward normal direction (Figure 1). For each ray (u, v, θ, ϕ) , the closest distance between the bounding volume and the model surface is found and recorded as a compact integer value after being normalized by twice the sphere radius. Thus, for each sample point (u, v) a visibility gray-scale map is obtained that represents the distance travelled along the ray in the direction (θ, ϕ) before hitting the model surface. We define the *visibility field* of the object to be the collection of all visibility maps generated from all sample points on the bounding sphere of the object.

3.2 Visibility Field Indexing

During the real-time part of the execution an incident ray to the object intersects its bounding sphere and the distance between the ray origin and the intersection point is recorded. The intersection point \mathbf{q} is transformed into the object coordinate system: $\mathbf{q}' = \mathbf{M}^{-1} \cdot \mathbf{q}$, where \mathbf{M} is the transformation matrix with respect to the reference frame of the ray. We need to acquire the closest point (u, v) on the sphere for which we have a visibility map and therefore the index of the corresponding visibility map. In addition we need to transform the corresponding (θ, ϕ) of the incident ray into a visibility map cell coordinates. The indexing is performed following the methodology proposed in Gaitatzes et al. [Gai08]. We can now index into the *visibility field* for the

```

1: for all emanating rays do
2:   if ray intersects bounding sphere of occluder object
3:     discretize intersection point  $(u, v)$ 
4:     discretize ray  $(\phi, \theta)$ 
5:     access distance in visibility field volume
6:   end
7:   use distance for occlusion approximation
8: end
9: compute occlusion at pixel  $x$ 

```

Algorithm 1: Pseudo code of shader algorithm for AO rendering, using visibility fields.

given ray (u, v, θ, ϕ) and extract the distance information which is then added to the intersection distance above and this is our approximated distance value of the ray origin from the object's surface.

A special case arises when the rays originate from the object being queried for visibility. As we can see in Figure 2, when a ray originates on the object at point \mathbf{p}_0 , the distance d_1 in direction $\overline{p_0 p_1}$ is computed and compared to distance d_2 in direction $\overline{p_1 p_0}$ which is extracted from the visibility map at point \mathbf{p}_1 . If d_1 is greater than d_2 then point \mathbf{p}_0 is occluded.

4. Visibility Fields on the GPU

4.1 Ambient Occlusion

Directional ray samples on a reference hemisphere aligned with the z-axis are pre-computed and stored in a texture for passing to the GPU. In the fragment shader (Algorithm 1), the pre-computed ray directions are transformed according to the local normal vector and intersected with the bounding sphere of each occluder. We are able to handle both rays originating outside and inside the bounding sphere for inter-object and intra-object occlusion respectively. The only difference in the computation is the respective step to compute the final ray-object intersection distance at line 7 of Algorithm 1.

The indexing of the visibility fields is executed entirely on the GPU as is the Monte Carlo ray casting to evaluate the resulting ambient occlusion. The visibility maps are compacted and stored into a single 3D texture as slices, as shown in Figure 1. As the number of positional samples (i.e. visibility maps) can exceed the maximum volume texture dimension supported by the hardware, we compact as many visibility maps on each 2D slice of the volume as the texture hardware permits.

4.2 Ray tracing

For our proof-of-concept case study, we wanted to further improve ray-tracing timings of an already fast ray tracer. We used the method of Amit Ben-David et al. [Ami07] that implemented both a CPU and a fast GPU ray tracer by exploiting a BVH acceleration structure that has been proven to work better in some cases [Gün07] and is better suited for dynamic scenes. We did not replace the primary ray intersection tests because the regularity of the ray distribution emphasized the sampling pattern on the bounding sphere. Furthermore GPU rasterization provides better timings for the primary rays pass. In conjunction with the fact that for complex (and therefore time consuming) scenes with elaborate materials, most time is spend on secondary rays, we applied the *visibility fields* method only to secondary

rays, including shadow rays. To capture the intricate reflection effects of non-perfect reflection surfaces and to highlight the advantage of our method when intersection tests increase significantly, we extended the implementation to stochastic ray-tracing.

As in the case of the ambient occlusion computation, the rays are stored in a 2D map but this time are re-computed for each running pass. For the ray-object intersection the visibility maps are used in a fragment shader on the GPU (similar to Section 4.1) along

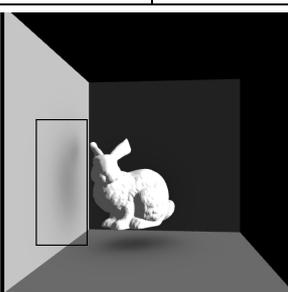
		Visibility field directional samples		
		32 x 32	64 x 64	128 x 128
Visibility field positional samples	1090			
		81.2 ms, RMS 0.59578	82.7 ms, RMS 0.58886	82.9 ms, RMS 0.58606
	4226			
		83.2 ms, RMS 0.45392	83.3 ms, RMS 0.42404	
16642				
	84.2 ms, RMS 0.42054			

Figure 3: Inter-object AO of a bunny model using the visibility fields method with 256 rays per pixel implemented on the GPU. We report the draw time and the RMS error. On the bottom right the reference image rendered on the GPU using 256 rays per pixel in 7126 ms. The model itself is rendered using fixed-pipeline direct rendering.

with the additional pre-computed maps of normals. The generated fragments correspond to intersection test results and the fragment shader returns the intersection point and distance to the actual surface as extracted from the visibility field. These results are used for shading or for spawning secondary rays for the next ray-tracing iteration.

5. Tests and Results

We implemented the real-time part of the above algorithm using the OpenGL[®] Shading Language [Kes06] on a 32bit Intel Core 2 Quad Q6600 at 2.4 GHz CPU and 4GB of main memory equipped with a GeForce 8800 GTS GPU with 512MB of texture memory. The window size was set to 512x512 for a total of 262144 pixels.

5.1 Ambient Occlusion

For most of the test runs the active pixels were about 200000 as only 75% of the window was rendered (the rest being black).

To acquire a reference image against which to compare our acceleration method in speed but mainly in image quality, we implemented ambient occlusion on the GPU using the uniform grid acceleration structure (see Figure 3 bottom-right).

We observe (in Figure 3) that the RMS error of the images compared to the reference image of the bunny, is very low and the achievable draw time, even for large models, is real-time. Based on the RMS error using 4226 64x64, visibility maps gives the same results as using maps of size 16642 32x32. We also infer from Figure 4 that the draw time is unaffected by the number of maps used thus the space required for the *visibility maps* depends only on the image quality that we would like to achieve.

In Figure 5 the visibility fields were used for the generation of intra-object occlusion but because the ray sphere intersection algorithm always succeeds at finding an intersection (worst case since we are inside the bounding sphere of the object) the rendering times are up to 4 times slower than the

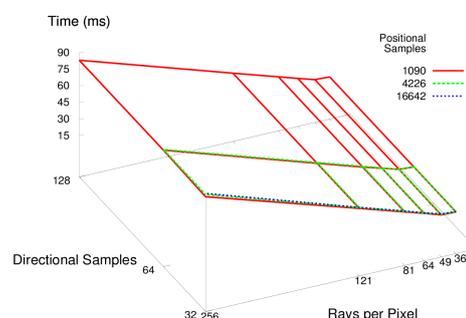


Figure 4: The draw time of the bunny model (39000 tris) plotted against different rays/pixel versus the size of the visibility maps.

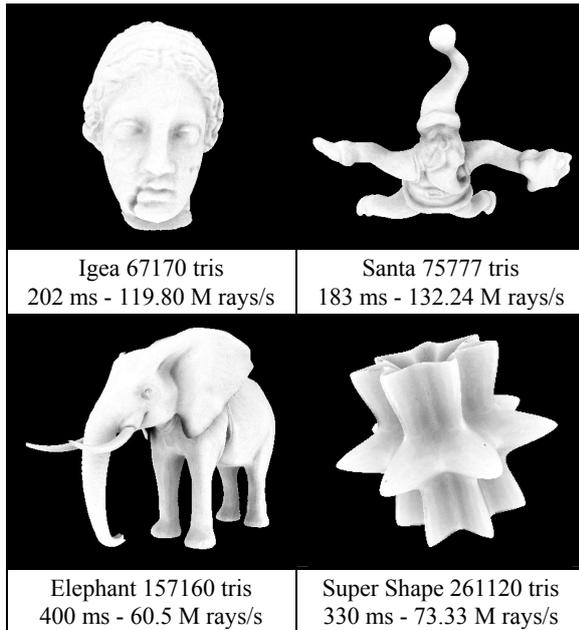


Figure 5: Intra object ambient occlusion rendered on the GPU using 16642 64x64 visibility maps requiring 65 MB of space and 121 rays per pixel.

inter-object occlusion case. Still the performance rate is above the one reported by Horn et al. [Hor07]. We also observe that more visibility maps are required in this case in order to render a believable image. We attribute this to the fact that multiple rays, with small angular differentiation, originating on close points on the object, hit the same sample point on the objects bounding sphere. Thus the same visibility map is used and the occlusion result looks grainy. When more maps are used the problem is alleviated.

In Figure 6 we show the Sponza Atrium rendered with several large polygon models inside it. The



Figure 6: A scene of the Sponza Atrium with a bunny (38889 tris), a cow (92864 tris) and an elephant (157160 tris) rendered in three passes (one per object) with the visibility fields algorithm using 4226x64x64 maps and rendering in 2.5 frames per second.

resulting draw time is contributed to the rendering method that uses one pass for each caster model. Just before each caster model is drawn, we enable subtractive blending (with OpenGL blend equation `GL_FUNC_REVERSE_SUBTRACT`), in effect, removing colour from the image. The poor draw time is also attributed to the fact that non-visible pixels (the Sponza Atrium has a lot of non-visible geometry) are not culled before the fragment shader is executed on the GPU.

Even though the *visibility fields* method is only an approximation, it does a very good job at preserving image quality given the low memory requirements and achieved draw time.

5.2 Ray tracing

In Figure 7 we show a close-up of the bunny ears of using the visibility-fields method. We show that very good results of soft shadows can be achieved while using 20 shadow ray samples along with 4226 64x64 visibility maps (i.e. 16.51MB of memory).

In Figure 8 we render a slightly more complex scene using 3 light sources of radius 2. As in the previous cases, the rendering time is almost completely affected by the primary rays which perform triangle intersection tests. Our method completes the rendering in 3268 ms, of which 70% is for the shadow rays. It produces a very good approximation of soft shadows using 20 shadow rays per pixel. For the total of 11,838,600 shadow rays, this corresponds to $1.9323 \cdot 10^{-4}$ ms per shadow ray which is a very encouraging result. In the corresponding BVH GPU method to produce sharp shadows using just 1 shadow ray per pixel, the draw time is 48047 ms to compute the final image. Of that time 70% is used for the 591930 shadow rays yielding $5.682 \cdot 10^{-2}$ ms per shadow ray.

In Figure 9 we use the visibility fields algorithm to render non-perfect-mirror reflections. The polished reference image is rendered with 4 rays for each reflective pixel leading to slower rendering times. However, we notice from the images and the RMS factor that the reflected sub region of our method is much closer to the result of the brushed metal reference image than the perfect mirror reference image. This strengthens our position that the proposed method is suitable for stochastic ray-tracing, as the quality of the rendered image is comparable to the reference image. Furthermore, the rendering time, even using 4 rays per reflective pixel, is very close to ray-casting without secondary rays.

6. Limitations of the Visibility Fields

The *visibility fields* method is not very well suited for elongated models. The occlusion produced, even when using 16642 maps is pretty grainy. In addition

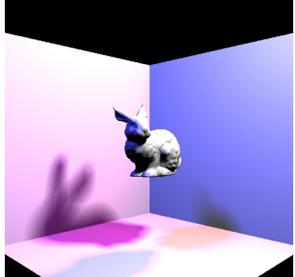
		Visibility field directional samples		
		32 x 32	64 x 64	128 x 128
Visibility field positional samples	4226	 348.0 ms, RMS 5.95, 4.127 MB	 348.5 ms, RMS 4.82, 16.508 MB	 348.7 ms, RMS 4.44, 66.031 MB
	16642	 348.5 ms, RMS 5.94, 16.252 MB	 348.6 ms, RMS 4.80, 65.000 MB	
			Close-up of the bunny ears from the reference image.	
				

Figure 7: Close-up of the bunny ears rendered using the visibility fields for the generation of soft shadows using 3 lights and 20 shadow ray samples on the GPU. We report the required time, the RMS error and the total space requirements. Bottom: Reference image rendered using the BVH method with 3 lights and 256 rays per pixel taking 913,210 ms on the GPU.

models that are highly concave would fail to produce accurate visibility maps as it would not be possible to record all of the tight concavities of the model.

7. Conclusions

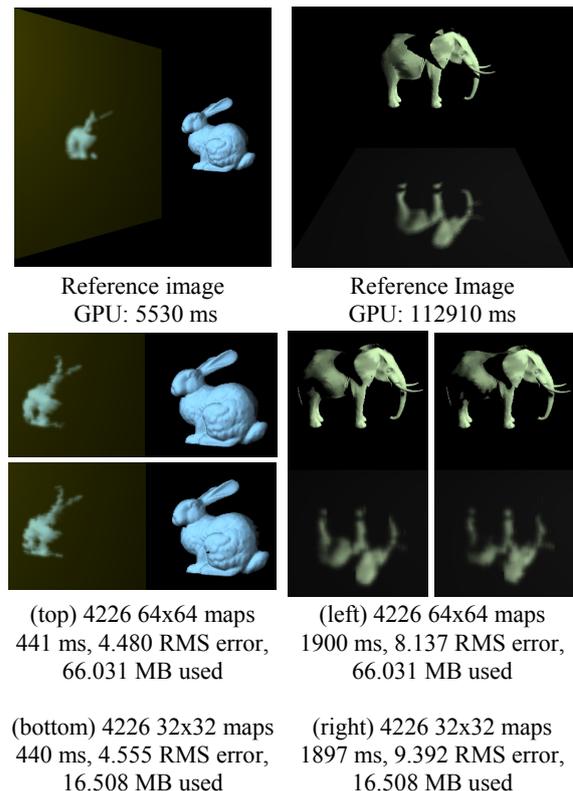
We have presented the *visibility fields*, a discretization of the visibility around an object, implemented on the GPU. We have shown how it can be used for an interactive inter-object ambient occlusion approximation computation. For the intra-object occlusion case the number of required maps is large and the draw time needs improvement when the model covers a lot of pixels on the screen. But in a



Figure 8: Close-up of a more complex scene using 3 point lights and 20 shadow ray samples rendered in 3268 ms using the visibility fields method. The BVH GPU method for sharp shadows takes 48047 ms.

game environment where several models exist on the screen and their coverage is not very big, the intra-object occlusion method can be used even for high triangle count models.

The method especially favours large model data sets, where we maintain a constant computation time, independent of the model complexity. Our method is robust, has a relatively small memory footprint



(top) 4226 64x64 maps 441 ms, 4.480 RMS error, 66.031 MB used (left) 4226 64x64 maps 1900 ms, 8.137 RMS error, 66.031 MB used
(bottom) 4226 32x32 maps 440 ms, 4.555 RMS error, 16.508 MB used (right) 4226 32x32 maps 1897 ms, 9.392 RMS error, 16.508 MB used

Figure 9: Polished reflection of the elephant (157160 tris) and the bunny (39000 tris) using 4 rays per reflective pixel. First row: Reference images using the BVH method (GPU draw times). Second row: Close-up view of our visibility fields GPU method where we report the draw time, the RMS error and the space requirements.

against comparable existing methods and the time required to generate the visibility maps depends only on the complexity of the occluder geometry. In addition, the number and resolution of the maps used in the *visibility fields* can be adjusted depending on the required accuracy and the available memory. The same maps can be used for both inter and intra-object ambient occlusion computation.

Furthermore, our algorithm can be applied to ray tracing calculations where exact ray hits are not critical, for example for shadow and secondary ray intersection tests, such as soft shadow rays and Monte Carlo ray tracing.

We have shown that in the above mentioned cases the production of the desired image is accelerated while the results remain close to the reference images. The hybrid method we propose favours large model data sets as in ambient occlusion. This result is expected as all triangle intersection tests for shadow and secondary rays are replaced with constant time operations. In this way rendering time is affected mostly by the primary rays that give us the visibility of the scene.

8. REFERENCES

- [Ami07] Amit B., Elber G.: GPU Ray Tracing. Master's Thesis, Technion Israel Institute of Technology, 2007.
- [Bav09] Bavoil, L., Sainz, M.: Image-space horizon-based ambient occlusion. In *ShaderX7 - Advanced Rendering Techniques*, Delmar, 2009.
- [Bun05] Bunnell M.: Dynamic ambient occlusion and indirect lighting. In *GPU Gems 2*, pages 223–234. Addison Wesley, 2005.
- [Car02] Carr A. N., Hall D. J., Hart C. J.: The Ray Engine. In *Proc. Graphics Hardware 2002*, pg. 37–46, Sep. 2002.
- [Car06] Carr A. N., Hoberock J., Crane K. Hart C. J.: Fast GPU Ray Tracing of Dynamic Meshes using Geometry Images. In *Proceedings of Graphics Interface 2006*, Quebec, Canada, June 07-09, 2006.
- [Chr05] Christen M., Engel W., Hudritsch M.: Ray Tracing on GPU. Diploma Thesis Univ. of Applied Sciences Basel (FHBB), 2005.
- [Ern04] Ernst M., Vogelgsang C., Greiner G.: Stack Implementation on Programmable Graphics Hardware. In *Proceedings of the Vision, Modelling, and Visualization Conference 2004 (VMV 2004)*, pp. 255–262.
- [Fol05] Foley T., Sugerma J.: Kd-tree acceleration structures for a GPU ray tracer. In *Proc. Graphics Hardware*, pages 15–22, 2005.
- [Gai08] Gaitatzes A., Chrysanthou Y., Papaioannou G.: Presampled Visibility for Ambient Occlusion. In *Proc. of the 16-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG '2008)*, Czech Republic, February 2008.
- [Gün07] Günther J., Popov S., Seidel H.-P., Slusallek P.: Realtime Ray Tracing on GPU with BVH-based Packet Traversal. In *Proc of the IEEE / Eurographics Symposium on Interactive Ray Tracing 2007*, pp. 113–118.
- [Hav00] Havran V.: Heuristic Ray Shooting Algorithms. Ph.D. thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, November 2000.
- [Hor07] Horn R. D., Sugerma J., Houston M., Hanrahan P.: Interactive k-D Tree GPU Raytracing. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*, ACM Press, pp. 167–174, 2007.
- [Kap85] Kaplan R. M.: Space-Tracing: A Constant Time Ray-Tracer. In *Proc. Computer Graphics 19*, 3 (July 1985), pg. 149–158. (Proceedings of SIGGRAPH 85 Tutorial on Ray Tracing).
- [Kar04] Karlsson F., Ljungstedt C. J.: Ray tracing fully implemented on programmable graphics hardware. Master's Thesis, Chalmers Univ. of Technology, 2004.
- [Kes06] Kessenich J., Baldwin D., Rost R.: The OpenGL Shading Language. Version 1.2.8. 3Dlabs, Inc. Ltd. 2006.
- [Mal88] Malley T. J. V.: A shading method for computer generated images. In *Master's Thesis, Computer Science Department, University of Utah*, June 1988.
- [Mit07] Mittring M.: Finding next gen: CryEngine 2. In *ACM SIGGRAPH 2007 Courses*, San Diego, California, August 05-09, 2007.
- [Pur02] Purcell J. T., Buck I., Mark R. W., Hanrahan P.: Ray tracing on programmable graphics hardware. In *Proc. SIGGRAPH*, 2002.
- [Pur04] Purcell J. T.: Ray Tracing on a Stream Processor. Ph. D. Dissertation, Stanford University, 2004.
- [Pop07] Popov S., Günther J., Seidel H.-P., Slusallek P.: Stackless KD-Tree Traversal for High Performance GPU Ray Tracing. In *Proc. of Computer Graphics Forum 26(3)*, pp. 415–424, 2007, (Proceedings of Eurographics)
- [Sha07] Shanmugam P., Arikan O.: Hardware accelerated ambient occlusion techniques on GPUs. In *Proceedings of the 2007 Symposium on interactive 3D Graphics and Games*, Seattle, Washington, April 30 - May 02, 2007.
- [Shi97] Shirley P., Chiu K.: A low distortion map between disk and square. In *Journal of Graphics Tools* 2, 3 (1997).
- [Sla02] Slater M.: Constant time queries on uniformly distributed points on a hemisphere. In *Journal of Graphic Tools* 7, 1 (2002), pp. 33–44.
- [Thr05] Thrane N., Simonsen L.O.: A comparison of acceleration structures for GPU assisted ray tracing. Master's Thesis, University of Aarhus, Denmark, 2005.

Parcel's information visualization on mobile Device

Andriamasinoro Rahajaniaina¹

Jean-Pierre Jessel²

VORTEX Group

Institut de Recherche en Informatique de Toulouse
Université de Toulouse - Paul Sabatier, 118 Route de Narbonne

31062, Toulouse, France

{rahajani¹, jessel²}@irit.fr

ABSTRACT

This paper presents the use of Augmented Reality system for visualizing a distributed parcel's information in a mobile device using wireless network. Our system offers useful information (number of parcel, name of the owner, agronomic structure, juridical state, and adequate plant) related to each parcel geo-referenced according to the user's position and orientation. This information augmented the live images of the real environment surrounding the user. For receiving these live images, we used two kinds of camera: webcam for an Ultra Mobile Personal Computer and notebook, and SD camera for Pocket Pc hardware. We used inertial sensor MTi and Global Positioning System receiver to achieve user's position and orientation. Internet Communication Engine, an object-oriented middleware is used to ensure the connection between database servers and clients. The users can interact with the images of surrounding environment using classic interaction tools (stylus, buttons ...).

Keywords

Distributed Augmented Reality, Visualization of GIS data, Interaction with GIS data, Ubiquitous system.

1. INTRODUCTION

In Human Computer Interaction, quality of user interface is important. Augmented Reality (AR) is among of the technique used to perform an user interface for ubiquitous application. AR presents information in its context within a 3D environment. The goal is to create the impression that the virtual objects are part of the real environment. Geographical Information System (GIS) database takes an important place for an outdoor AR system.

Most of previews researches [Höl99][Käh06][Rei07] used GIS database corresponding for building, streets in order to enhance the experiences of users (e.g. tourists, visitors). These systems overlaid digital information (such as building name, road name) on the real world in order to perceive remote or local geographical information.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Instead, the system presented here talks about visualization of parcel's characteristic depending on location and context. As GIS database stores numerous data, retrieving information from it is among of one critical point in a distributed AR system because it may generate latency during exchange. To overcome this problem, we show in detail our approach about retrieving information related to a Parcel from GIS database, and the adequate metaphor of visualization and interaction of them.

For receiving live images of real environment, we used two kinds of camera: webcam for an Ultra Mobile Personal Computer (UMPC) and notebook, and SD camera for Pocket Pc hardware.

In this paper, section 2 reviews the related work. After that, we describe in detail our *ARGisUbiq* system in Section 3. Finally, we conclude and provide possible perspectives for future investigations.

2. RELATED WORK

Outdoor AR systems have traditionally been reserved to use GIS databases related to buildings and streets in order to provide help to users. Several systems are presented hereafter.

First, Mars (Mobile Augmented Reality Systems) project [Höl99] presented in 1999 by Columbia University was one of the first truly mobile augmented reality setups which allowed the user to freely walk around while having all necessary equipment mounted onto his back. It allowed user to arrange the multimedia information according to chronological order. This system used a campus database to overlay labels on buildings seen through a tracked head-worn display. Users was able to request additional overlaid information, such as the names of a building departments, and to view related information, such as a department web page, on a hand-held display.

Archeoguide project [Gle06] was designed to increase real images of user's environment with virtual story information related to them.

Next, in [Lia05], the authors presented a prototype of an interactive visualization framework specifically designed for presenting geographical information in both indoor and outdoor environments. They used ESRI Shapefiles as input of their system. They represented 3D building geometry and others attributes. Participants can visualize 3D reconstructions of geographical information in real-time based on two visualization clients: a mobile VR interface and a tangible AR interface.

Then, ARscouting system [Rei07] introduces an outdoor AR system witch has run on UMPC using a camera and a GPS receiver to collect information about the environment. Mobile system has been used as a thin client. While exploring the environment, the scout takes several images for instance of a target building. These images are automatically annotated by current positioning data. The enriched data are then transmitted to a custom database (multimedia database) store. Whenever a new image is stored in the database, the reconstruction engine gets a notification and triggers the reconstruction process. The engine requires at least three different views in order to generate an initial 3D model. Each further image is added in an iterative way and updates the model accordingly within seconds. Once the reconstruction task is over, the server stores the virtual object and transmits it to the mobile client (scout) in order to increase user interface. The last one is based on the Studierstube platform.

The claimed MARA [Käh06] system implements hand-held, video-see through Augmented Reality for Nokia S60 mobile imaging devices equipped with additional sensors like a GPS receiver, accelerometers and a tilt compensated magnetometer. The system allows users to interact with their surrounding environment using the standard mobile device inputs. It allowed users to place hyperlink at their current location in order to give information

about an object. The users could share or exchange all data with others connected users.

The following Section expands our *ARGisUbiq* platform.

3. ARGISUBIQ SYSTEM

ARGisUbiq system is an improved version of [And08] which runs on Windows Vista, Windows mobile XP and Windows CE dedicated for desktop PC or notebook, UMPC and Pocket Pc hardware devices. [And08] was a new architecture for a multiplatform AR which allowed the users to change in dynamic way their virtual workspace. The work plan is augmented by the virtual workspace. Each virtual workspace relied to several virtual objects. For adding virtual objects, we used a virtual menu inspired by the metaphor of forward and next buttons.

The main goal of *ARGisUbiq* system is to propose a new application AR GIS in agronomic domain that shows all information about a parcel according to the user's location. In our knowledge, this is the first AR system using parcel's information to enhance user's visualization interface.



Figure 1. The UMPC visualization tool

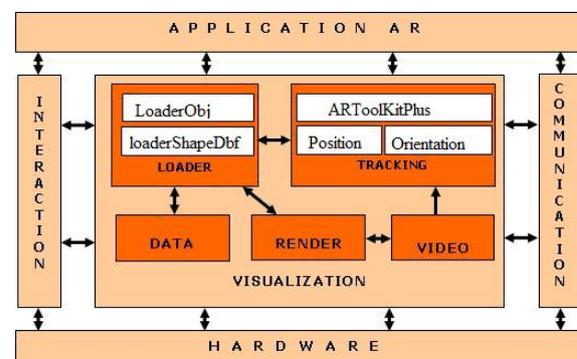


Figure 2. The software architecture

We added some module in the software architecture of [And08] in order to enhance its functionalities (see Figure 2):

- *Communication* module ensures the exchange between clients and servers, and between all modules.
- *LoaderShapeDbf* module is responsible of parcel's information loading from the Data module using the database's structure (see Figure 3).
- *Position* module and *orientation* module retrieve information from GPS receiver and MTi inertial sensor.

Our system is made of two modules: server module and client module (see Figure 5). About 90% of the task was run on the client part. Visualization module and communication module are the two main modules of the client. Data module may be available on the client and/or on the server module. The servers are used as database servers.

3.1 Information source

In the following section we present our database structure and explain the information's selection mode.

3.1.1 Database's structure

Like others outdoor AR systems, *ARGisUbiq* system uses GIS data as data source. As parcel's information related to agronomic and type of plant is unavailable on producer's map, we create our own database inspired from parcel database (using a vector format formed by shape files, index one and dbf one) (see Figure 3). In the Figure 3:

- *Parcel's* table stores information about parcel in the public register of lands. It has five attributes: the *numParcel* indicates the parcel's number, *numFeuille* designates the number of page, *numSection* is the number of section, *codeCom* signify common's code and *nomCom* is the name of common.
- *JuridicalState's* table stores data related to the juridical state of parcel. *Num_Situat* and *libelle* are its attributes: the first one is number of the juridical state and the second one designates the label of the juridical state.
- *AgronomicalState* table is designed to stock data associate to the state agronomic. The attribute *Num_prte_a* is the number of the agronomical state. *Type_sol* indicates the structure of the parcel and *ph* is the ph of the land.
- *Adequateplant* table stores data related of all type of tilling. It has three attributes: *id_cult* indicates the identity of the tilling, *libelle* is

the name of the tilling and *detail* relates to the detail of the tilling's feature.

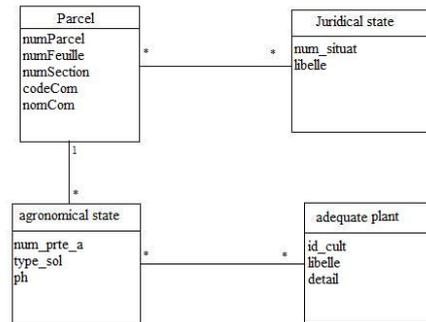


Figure 3. The database's structure

We can note that this database may be available on the clients and/or on several replica servers. The clients establish a connection to the servers using the communication module based on Internet Communication Engine (Ice) [hen03] when local database is unavailable. This later case occurred while the device have not enough space disk (UMPC or notebook) or space memory (Pocket Pc) for storing the database or it is deleted.

3.1.2 Information's selection mode

Information retrieval procedure occurs when the camera's orientation turns to the ground. GPS receiver (TomTom wireless GPS receiver) and MTi inertial sensor are used for tracking the camera's position and orientation. The GPS receiver exchanges information with the client using a Bluetooth connection whereas the connection between client and MTi sensor is established by a serial communication. Orientation values provide by MTi relate to the orientation of its coordinate system $S(x, y, z)$ according to the fixed global coordinate system $G(X, Y, Z)$ (see Figure 4).

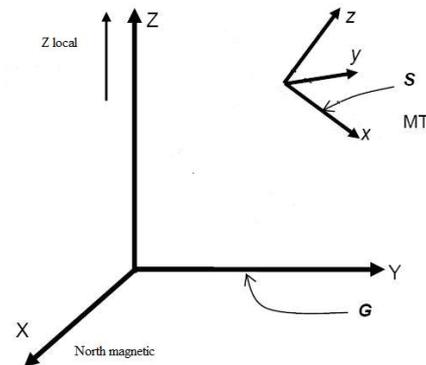


Figure 4. The coordinate system of the MTi

Each polygon in the shapefile is delimited by a bounding box. In order to know the existence of parcel according to user's location, we check if user's position is in the bounding box of its record. When his location is included in the limit, we save the number of row and bounding box of all corresponding record. Whether the record number is more than one, we take account only the record (polygon) having its barycenter nearby the camera's position. After that, we use the row number related to the selected parcel for searching others information (agronomical state, adequate plant, juridical state ...) in the associated tables. While the user's location is always in the bounding box, we work (research a new nearest barycenter of polygons) with these existing number rows and bounding box values of each record. We made it in order to reduce the amount of requests to the database which produce latency indeed for lightweight hardware devices.

3.2 Distributed architecture

As we describe above, *ARGisUbiq* system is composed of two modules: client module and server module which use geo-referenced GIS database. The Clients connect to the servers using WLAN network. Our framework uses Internet Communication Engine (ICE) and IceE (Ice Embedded: a lightweight version of Ice for mobile devices) to ensure the connection between the clients and the servers.

With the aim of having flexible data distribution, we duplicate on several servers our database and IceGrid services is used to establish load balancing between the client and all replica servers. It provides a convenient way to distribute an application to a set of computers, without the need for a shared file system or complicated scripts. Each server may have one registry which control one or several node's activities. Registry implements locator service and the locator object is available on the registry client endpoints (IP address or hostname and port number). A Node monitors the load of their computers (servers) and reports this information to the registry. This one uses this information to decide which endpoints of the object adapters to return to a client. In general, the server selects one or a set of endpoints which have the least-load statistics.

In presence of several Ice servers, one which have master registry is the master server and others one are slaves. Slave or master server property is specified in their configuration files. A first locate request activates the application server automatically (starting the Ice server process). Activation usually occurs as a side effect of indirect binding, and is completely transparent to the client. Node is responsible of this activation task when it receives registry's order. Node sends responses to a registry

according to its configuration file (the replica's number to include in the registry's response is specified in this file).

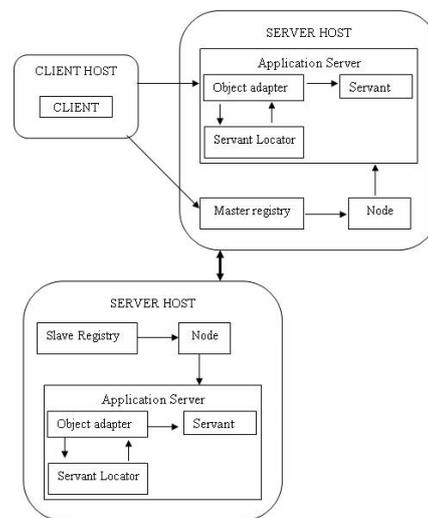


Figure 5. The distributed architecture

Selected application server uses object adapter in order to obtain information about an object (parcel) requested by a proxy client. Then, object adapter attempts requests to a servant which is a direct responsible of one or more objects.

Notice that multithreading is supported by Ice server. In fact, this property allows more clients to establish connections in the same time.

The master replica knows all of its slaves, but the slaves are not in contact with the others. If the master replica fails, the slaves can perform several vital functions that should keep most applications running without interruption. Eventually, however, a new master replica must be started to restore full registry functionalities. For a slave replica to become the master, the slave must be restarted.

Client module uses Ice/IceE in order to retrieve parcel's information from the replica server when a local database is unavailable. We deactivate proxy cache (that contains all information about previous server) and set a timeout to cache locator in order to make load balancing. IceGrid's load balancing capability assists the client in obtaining an initial set of endpoints for the purpose of establishing a connection. Before attempting locate request to a server location, client checks its cache locator. We randomize selection of object adapter's endpoints used by proxy client to establish connection in order to collect all object's information. One Client's request is formed by endpoint and object id.

As these properties are unavailable on IceE, in fact, to simulate the functionality of load balancing, we set a timeout for an established connection and close it in order to establish another one to other endpoints when the timeout is expired.

3.3 Visualization metaphors

With the aim of seeing augmented view, users must hold account information's selection principle. Once this one is respected, users saw real video augmented by parcel map, juridical situation, agronomical state and adequate kind of tilling related to the user's location. To achieve it, we propose two visualizations metaphors: *textview* mode and *hybridview* mode. In the *textview* mode, the scene is augmented by virtual text and aural information related to a parcel and the position of user. In the *hybridview* metaphor, user interface is enhanced by virtual text, parcel map and audio information. The blue point on the map is the user's position.



Figure 6. The hybridview mode

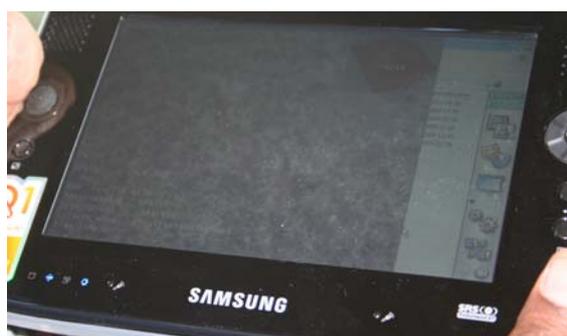


Figure 7. The hybridview mode on the UMPC

We combine *landscape* and *portrait* mode with *textview* and *hybridview* when users use lightweight hardware as visualization tools. The transition

between the two metaphors depends on the way which the user holds his PDA.



Figure 8: The landscape hybridview mode



Figure 9: The portrait hybridview mode



Figure 10. The landscape Textview mode

The *hybridview* is the default visualization metaphor for UMPC and PC notebook, and the *textview* metaphor is for PDA.

3.4 Interaction metaphor

We propose a possibility for user to choose visualization metaphor using *textview* and *hybridview* menus. We decide to use classical interaction tools like menu, stylus, and button because these are available on each device that we use as visualization tools. When the user selects one of both menus using his stylus, the user interface changes according to the menu item selected. After that, the menu item changes to another one.

As described above, when using a PDA, the transition between the two metaphors depends on how the user holds his PDA and the value of pitch angle (Θ) from the inertial sensor MTi (Θ value between -5.0° and 0.0° for portrait mode and landscape mode for others values).

If the user needs additional information related to the kind of tilling, he selects "more info" and listens the aural information.

4. Experimentation and Results

This first prototype was tested with three users: the first client has used Q1 Samsung with 800 Mhz Celeron M ULV processor, 256 Mo RAM and the two other clients have used a Pocket PC dell axim x51v with 624Mhz Intel xscale processor, 64Mo RAM. We have used a database of common formed by 180,000 parcels and each parcel is formed by 10 up to 20 vertices. We have tested two different scenarios: first, we have used a local database: as we have loaded the database in the memory at the first time, the Q1 client has run after 5s of the database loading and 22s for the two PDA clients. After this step, the Q1 client was able to achieve 25-30 fps (frame per second) and 17-20fps for the PDA clients during the exchange with the data in memory. In the second test, the database is duplicated on three replica servers. One master registry and two slave registries. Each slave registry has had its own node which monitors two applications servers. The locate request from the client to the registry has spent 0.3s for Q1 client and 0.7s for PDAs clients. After that, the Q1 client was able to 23-28fps and 15-20fps for the PDAs clients during the exchange with the replica server.

After these tests, we asked the users about ergonomic of user interface and about the visualization hardware device: 80% of the users are satisfied about user interface but 50% only for hardware device.

5. Discussion

As we saw, the difference between the results using the local database and replicate database was small. It's not surprising because we have added to the client a functionality to reduce the number of exchange with the replica server (see section 3.1.2). It's also due to the performance of our replica servers. 50% of users only are satisfied for hardware device ergonomic because most of users prefer using wireless inertial sensor instead of using MTi. It is easy to use.

From these results we can deduce designs guidelines for choose of hardware device in future AR application.

6. Conclusion and future work

In this paper, we addressed the problem of enhancing user's contextual perception of the real world using GIS data on several hardware and software platform. To tackle this, we have proposed the *ARGISUbiq* multiplatform architecture which exploits Mobile Augmented Reality principles to improve user's interaction with GIS data. As we use specifically built GIS data, we described our

database's structure and how to select appropriate information related to user's position. Our distributed application is based on Internet Communication Engine, an object-oriented middleware, used to ensure the connection between database servers and clients. To avoid eventual problem with database server, we duplicate our database on several servers and we use Icegrid services to provide load balancing between all servers. Some clients are able to access concurrently to a selected server.

We are entirely satisfied with our first results. In the future work, instead using MTi sensor we plan to use low cost or embedded inertial sensor and image based techniques to compute the user's orientation

7. REFERENCES

- [And08] Andriamasinoro Rahajaniaina and Jean-Pierre Jessel, A new architecture for a multiplatform Augmented Reality System, Proc. In International Conference on Signal processing and multimedia Applications, Porto Portugal, 2008.
- [Gle06] Gleue and P. Daehne, Design and implementation of a mobile device for outdoor augmented reality in the archeoguide project, In Virtual Reality, Archaeology, and Cultural Heritage International Symposium, Glyfada, Nr Athens, Greece, 2001.
- [hen03] Henning et al., Distributed Programming with Ice, ZeroC: <http://www.zeroc.com/Ice-Manual.pdf>, 2003.
- [Höl99] Höllerer, S. Feiner, T. Terauchi, G. Rashid, and D. Hallaway, Exploring MARS: Developing Indoor and Outdoor User Interfaces to a Mobile Augmented Reality System, Computers and Graphics, 23(6), Elsevier Publishers, 1999.
- [Käh06] Kähäri and D. J. Murphy, MARA-Sensor based Augmented Reality System for Mobile Imaging Device, 5th IEE and ACM International Symposium on Mixed and Augmented Reality, Santa Barbara, 2006.
- [Lia05] Liarokapis, I. Greatbatch, D. Mountain, A. Gunesh, V. Brujic-Okretic and J. Raper, Mobile Augmented Reality techniques for GeoVisualisation, Proc. 9th International Conference on Information Visualisation, IEEE Computer Society, London, 2005.
- [Rei07] Reitinger, C. Zach and D. Schmalstieg, Augmented RealityScouting for interactive 3D reconstruction, in Proceedings of IEEE Virtual Reality Conference, Charlotte NC, USA, 2007.

Modeling and rendering heterogeneous fog in real-time using B-Spline wavelets

Anthony Giroud
University Paris Est Marne-la-Vallée
giroud@univ-umlv.fr

Venceslas Biri
University Paris Est Marne-la-Vallée
biri@univ-umlv.fr

ABSTRACT

Heterogeneous fogs are often modeled with several layers of different density or using particle systems. However, layers are limited to vertical variations and using particles can involve a long computation time with large outdoor scenes. In this article we present a simple method to render heterogeneous fog in real-time. The extinction function of our fog, related to its density, is first modeled in a B-Spline function basis. Then, a wavelet transform is applied on this function to obtain a decomposition in both space and frequency domains. A grid traversal is used to render the fog in real time using the GPU. Since no precomputation is required concerning the position of the camera or the fog, we can freely navigate or move the fog into the scene.

Keywords: Participating medium, Fog, Rendering, GPU.

1 INTRODUCTION

Fog is massively used in rendering both for aesthetic purposes and to increase performances by providing an efficient way to cull surfaces that are far from the camera. Simple fog models are straightforward to implement but, like OpenGL's fog model, only allow a basic representation of homogeneous fog as can be seen on figure 1. Most of the time, these models are barely convincing visually, as we know that natural fogs never reach such perfect homogeneity. Considering latest advances in GPU programming, design of heterogeneous fog should be simple, and its rendering reachable in real-time.

The fog phenomenon is due to small particles of water in suspension. Because it interacts with light rays, fog is considered as a participating medium in computer graphics. Fog effects take into account attenuation, caused by absorption and out-scattering, and also consider multiple scattering of light as isotropic and constant over the scene. If we consider an homogeneous fog in its simplest form, equations are simple enough to allow an analytical integration of its effects along a view ray. When rendering heterogeneous fog, the density of water particles is varying across the scene, thus dramatically complexifying the model, involving local changes in physical properties of the fog, such as its extinction coefficient. Therefore, in order to compute the light-fog interaction, we have no other choice than per-



Figure 1: Comparison between OpenGL's homogeneous fog (left) and our heterogeneous fog (right)

forming the integration of the density along each view ray from the eye to the nearest object.

Considerable work has been achieved in the development of real time solutions to handle participating media. Physical simulations taken aside [12, 14, 7, 11, 6], which do not reach realtime, researchers have been working on rendering complex exchanges of light within the medium, dealing, for example, with single scattering. They also considered simpler forms of fogs, with a density either varying along horizontal layers, defined by Perlin noise, or using particles. But few tried a direct and continuous mathematical representation of its density.

In this paper, we present a new method helping to shape and render complex heterogeneous fog in large outdoor scenes, lighted by a single light source (the sun). First, the fog is modeled in a B-Spline function basis, which allows a simple and efficient construction of its extinction function. As a preparation before rendering, Mallat's wavelet decomposition is applied on the extinction function in order to automatically generate different resolutions, enabling an optimized real-time rendering using the GPU. The use of wavelets offers several advantages :

- An easy modelization leading to a smooth and continuous fog density by opposition to particles ap-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

proaches that are discrete. Analytical representation compresses data more efficiently and are, for example, easier to animate.

- Wavelet modelization is generic. It includes naturally, using Haar wavelets, discrete approaches like quad tree or octree representation.
- Wavelet decomposition leads to sparse data that can be used to improve rendering time.

Therefore the contribution of this paper is :

- Establishing a wavelet framework for the definition and modelization of an heterogeneous fog.
- Rendering the fog in real time using this representation without precomputation involving camera or fog position.
- Allowing a tradeoff between correctness and speed using the multiresolution offered by the wavelet decomposition.

In the next section, we review previous methods to render, in real time, the effects of participating media in a scene. Then, we briefly introduce the wavelet theory along with the equation of transfer inside a participating medium. Section 4 presents our modeling scheme and our implementation for rendering. In section 5, we expose and discuss our results.

2 PREVIOUS WORK

Rendering participating media such as fog in real-time has been well studied. We will not consider global illumination algorithms concerning participating media. For more information on this subject, the readers should refer to the excellent survey of Cerezo et al. [2]. Algorithms dealing with single scattering, including volume based approaches [17] or direct representation [1], also handle fog naturally but due to complexity problems these techniques only consider homogeneous mediums (except [19] discussed bellow). Therefore, we limit our overview to other real time approaches for heterogeneous fog which can roughly be divided in, on the one hand, particle approaches and, on the other hand, layered or bounded approaches.

Particles provide a natural way to handle heterogeneous fog. They have been used efficiently in numerous works [4, 15, 9, 3]. The idea is to consider particles as groups of water drops, allowing real time rendering of effects like smoke or physically based simulation. But is not well adapted to large scale fog recovering a whole scene. Moreover, animating all particles in a large scene is computer time consuming. The same drawbacks hold for the hybrid approach of Zhou et al. [19] which handles single scattering in a heterogeneous participating medium combining particles and spherical harmonics. We can also cite the work of Zdrojewska [18] which uses Perlin noise to alter the homogeneous density of the fog. Despite this good idea, the use of 3D random noise forbids any animation of this fog.

The idea behind layered or bounded approaches is to enclose fog density variations into layers [8, 5] or bounded volumes [10]. These works consider homogeneous fog enclosed in volume, inducing a discontinuous density function and creating artifacts on the border of these volumes. Moreover, intuitive or physically based animations of this kind of representation could be difficult to handle. Despite these limitations, it is often the kind of solution we can find in common graphic engines, along with particle rendering. Nevertheless, none of the previous methods offers a simple and efficient mathematical representation of heterogeneous fog adapted for both animation and rendering.

3 THEORETICAL BACKGROUND

3.1 Fog's illumination model

Our main goal is to render our fog in real-time, using conventional graphics cards. Although performances of GPUs have never been increasing so fast, we have to slightly simplify our fog model. Between points O and P , fog induces an attenuation (due to out-scattering and absorption) of the luminance L of P and an increase (in-scattering and emission) of light along the ray \vec{OP} . We start directly with the integral transfer equation, see [13] :

$$L(O) = \tau(O, P)L(P) + \int_O^P \tau(O, u)K_t(u)J(u, \vec{\omega})du \quad (1)$$

$L(O)$ being the radiance received by the observer, $J(u, \vec{\omega})$ being the incoming radiance along the ray, K_t the extinction coefficient and $\tau(u, v)$ the transmittance of the fog along the ray going from u to v :

$$\tau(u, v) = e^{-\int_u^v K_t(s)ds} \quad (2)$$

First, when daylight passes through fog, it is immediately scattered such that light in-scattering can be simplified by a constant amount L_{fog} . Moreover, if we consider that the light emitted by the fog itself can be neglected, the incoming radiance $J(u, \vec{\omega})$ equals L_{fog} , and then equation (1) becomes :

$$L(O) = \tau(O, P)L(P) + \int_O^P \tau(O, u)K_t(u)L_{\text{fog}}du \quad (3)$$

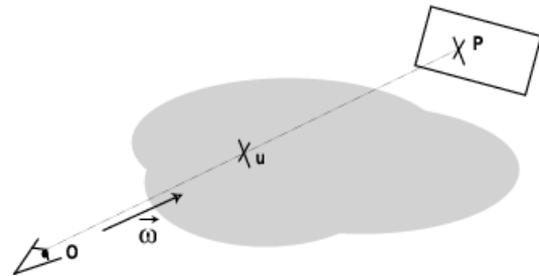


Figure 2: View ray \vec{OP} through a participating medium.

The second part of equation (3) can be analytically integrated to obtain :

$$L(O) = \tau(O, P)L(P) + L_{\text{fog}}(1 - \tau(O, P)) \quad (4)$$

3.2 Wavelets

From equation (3), we can see that the density variation could be represented efficiently by the extinction function. Therefore, K_t will be modeled using the wavelet framework whom principal used characteristics are detailed in this section. More details on the wavelet framework can be found in [16].

The wavelet framework. In a multiresolution analysis, data is represented using several approximation spaces. Different functions bases are used to represent a single signal, and each functions basis corresponds to a different resolution. Moreover, all basis functions are obtained by translating and scaling a single original pattern function $f \in \mathbb{L}^2(\mathbb{R})$, in other words :

$$f_{j,k}(x) = f(2^j x - k), \text{ with } j \in \mathbb{N}, k \in \mathbb{Z} \quad (5)$$

where $f_{j,k}$ represents the basis functions and j the resolution level. If we define F_j as the closed subspace of \mathbb{L}^2 using basis functions $\{f_{j,k}\}_{k \in \mathbb{N}}$, the closure of $\bigcup_{j \in \mathbb{N}} F_j$ is the space \mathbb{L}^2 and represent all square integrable functions.

The wavelet framework uses, to build basis functions of spaces F_j , a particular function called scaling function and often denoted by ϕ . It verifies equation (5) and generates a ϕ_{jk} family, $j \in \mathbb{N}, k \in \mathbb{Z}$. This function ϕ also presents the property of being written as a linear combination of $k/2$ translated and $1/2$ scaled versions of itself. It is the *scaling relation* of the scaling function, given by :

$$\phi(x) = \sum_{k=-\infty}^{\infty} p_k \times \phi(2x - k) \quad (6)$$

where $\{p_k\}$ are the coefficients of the *scaling sequence* of ϕ . Note that each subspace F_j , $j \in \mathbb{N}$ will in fact use the same and unique function ϕ translated and scaled.

The particularity of the wavelet framework is its ability to decompose a function of F_{j+1} using several functions of F_j and of its orthogonal complement G_j . Therefore, if J is the maximum resolution level, the F_J space can be written :

$$F_J = F_0 \cup \bigcup_{j=0}^{J-1} G_j \quad (7)$$

This equation means that a function (up to a resolution J) can be described using only one scaling function (space F_0) and several functions of spaces G_j . The basis functions of spaces G_j are called wavelet function and verify equation (5). They can also be built using

the *scaling relation for wavelets*, which we will call the *wavelet relation* :

$$\psi(x) = \sum_{k=-\infty}^{\infty} q_k \times \phi(2x - k) \quad (8)$$

where $\{q_k\}$ are the coefficients of the *wavelet sequence* of ψ . Note that, similarly to F_j , each subspace G_j uses the same and unique function ψ translated and scaled.

Decomposition and multiresolution using wavelets. The advantage of the wavelet framework is that it provides an efficient way to decompose a function into multiresolution spaces. The fast decomposition can be assured by the Mallat's wavelet transform which uses, as entry data, coefficients of the function modeled directly in the maximum resolution level. Therefore, our fog extinction function will be modeled using scaling function.

Mallat's algorithm takes advantage of equation (7) and consists, for each step, in extracting from the approximation at level n (represented in a scaling functions basis) first the approximation at level $n-1$ (F_{n-1} which is twice less precise), and then the corresponding layer of details (G_{n-1} represented by a wavelet basis). We simply repeat this process until we obtain the approximation at level 0. Mallat's transform is lossless, therefore when we simply sum up the coarsest approximation with all layers of details, we recover the original signal untouched.

Wavelets in two dimensions. Now that we know how to build scaling functions and wavelets in one dimension, going 2D will actually be quite straightforward. In a nutshell, it simply consists in assigning the corresponding 1D function to each axis, and the result is given by the product of these two 1D functions. Basically :

$$\phi\phi(x, y) = \phi(x)\phi(y) \quad (9)$$

where $\phi\phi$ is a 2D scaling function and ϕ is the corresponding 1D scaling function. Things go exactly the same way with wavelet functions.

Obtaining a 2D wavelet transform is slightly harder and requires to process rows and columns separately. There are two different decomposition methods : the *standard decomposition* and the *nonstandard decomposition*. These two types of decomposition output exactly the same kind of result :

- A single coarse approximation at level 0, modeled with 2D scaling functions $\phi\phi(x, y) = \phi(x)\phi(y)$.
- $J-1$ layers of vertical details, modeled with hybrid functions $\phi\psi(x, y) = \phi(x)\psi(y)$.
- $J-1$ layers of horizontal details, modeled with hybrid functions $\psi\phi(x, y) = \psi(x)\phi(y)$.
- $J-1$ layers of 2D details, modeled with 2D wavelets $\psi\psi(x, y) = \psi(x)\psi(y)$.

For example, our fog's extinction function can be written as :

$$K_t = \sum_{ij} \alpha_{ij} \phi_{ij} + \sum_{n=1}^{J-1} \left[\sum_{ij} \beta_{ij}^n \psi_{ij}^n + \delta_{ij}^n \psi \phi_{ij}^n + \gamma_{ij}^n \psi \psi_{ij}^n \right] \quad (10)$$

4 OUR METHOD

4.1 Modeling the fog

The two-dimensional framework. Unlike other types of participating media from the same family, fog almost always appears in large outdoor scenes as a horizontal layer of varying thickness. This is quite different from smoke, which can evolve indifferently in all directions in terms of shape and movement, and thus really need to be defined with the same precision along all three dimensions.

For this reason, and in order to ease the shape definition as much as possible and, later, the rendering step, we have chosen to restrict our main framework to two dimensions. The optical properties of our fog, similarly to most other participating media rendering techniques, are proportional to its density, which depends itself on its extinction function. Therefore, the fog's main shape will actually be modeled as horizontal layers containing horizontal extinction function projected in a two-dimensional function basis. Further parameters, starting with a vertical extinction coefficient, will then thicken the fog vertically and give its final appearance.

Designing the fog's shape. Horizontal variations of our fog's density are modeled by specifying the value of each coefficient in the extinction function basis. These coefficients can be adjusted by hand, or be, for example, the result of a simulation, which was exported as a fogmap (see figure 3), i.e. a greyscale image, and then loaded back in our implementation.

Compared with other techniques such as RBF or particle-based methods, shaping our fog using a grayscale image is straightforward. The fogmap represents, in some extent, a direct preview of its aspect, what can be interesting for some applications where great intuition is needed. To ease the manual setting of

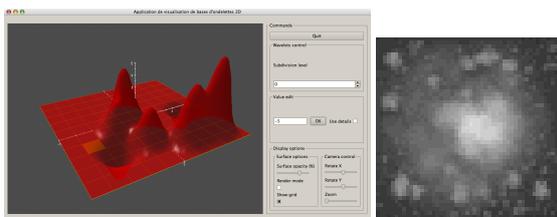


Figure 3: Left : snapshot of our modeling tool. Right : greyscale image representing the highest resolution coefficients

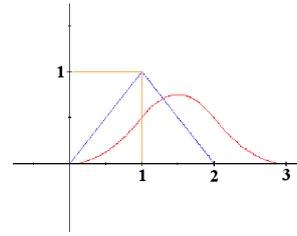


Figure 4: Shape of Haar, Linear and Quadratic B-Splines.

the coefficients, we also developed a small application where the values of the density can be directly adjusted using a drag-and-drop interface.

Choosing the basis functions. The appearance of the fog's density is a key criteria to choose our basis function. It is clear that abrupt changes in density would not look natural, so we would ideally like continuous functions to design smooth fogs using as few coefficients as possible. In order to avoid border effects, the scaling function must tend to zero on both sides of its support, which eliminates, for example, Legendre scaling functions.

For design and optimisation purposes, our rendering algorithm also needs the scaling function never to oscillate under zero. Whatever the trajectory of the ray within the function in 2D, and more generally within the fog, we would like to be sure that the sum of the density it intersects can only increase as it traverses the fog from the observer to the nearest object. Daubechies wavelets, which, by the way, are not symmetrical, might not be the way to go.

Finally, we have to consider the fact that, as will be discussed in the next section, the cost of using a particular type of wavelet is quadratically proportional to the support of the scaling function in one dimension.

According to their shape, the most adapted candidates seem the linear or quadratic B-Splines, which are shaped like a hill (see figure 4), and have a relatively compact support.

Although we are limited to wavelet scaling functions for the fog's representation, our method is not reduced to a particular type of wavelet. Our implementation specifically handles all degrees of B-Spline wavelets, but can be extended to other families, as long as they are compatible with Mallat's decomposition.

4.2 Preparing data for rendering

Generating multiple resolutions. One of our main goals is to take profit of multiresolution. Indeed, multiresolution helps to omit details that could be expensive to render, while being of limited visual importance. Therefore, perform a wavelet decomposition on our fog, which generate multiple level of details (i.e. multiple resolutions) from the original extinction function, and use them at the rendering phase. The most

adapted solution seems Mallat's fast wavelet transform, which is lossless, but requires data to be modeled in a scaling functions basis of the same type as the wavelets used for the decomposition. Therefore, each pixel of the fogmap will represent the coefficient of a scaling wavelet function.

Computing textures. From the fogmap we generate four multiple-level function bases : the approximation on a single level (i.e. a single 2D grid of values), and three different kinds of details for each level which was decomposed. All details bases have the same depth, which corresponds to the number of decomposition steps that were executed, value which must be decided by the user, depending on how much details can be omitted. Coefficients from the approximation and details basis will be stored in packed textures, and transmitted to the GPU under this form.

4.3 Rendering the fog

Overview. The purpose of our algorithm is to alter the original color of each pixel of the image using equation (3), blending $L(P)$, the color of the object behind the fog, and the fog color to obtain $L(O)$ the new color to compute.

For each pixel, we perform a ray-marching from the camera to the nearest surface, in which we integrate over the fog's extinction function to obtain the transmittance $\tau(O, P)$ along the view ray \vec{OP} .

The grid. As a result from the wavelet decomposition, the fog's density is scattered in several multiple-level function bases, having their own vector space and definition domain in 2D. Each single level can be assimilated to a rectangular grid, each cell being associated to both a coefficient and a basis function. Since all bases have the same definition domain, grids from different bases match at a given level.

Since our fog is only modeled in two dimensions, we do not take into account vertical variations and consider the fog as homogeneous on that direction. However, a vertical extinction coefficient taken as parameter allows to fade the fog out while its vertical distance from the viewer increases. But note that this is only a quick approximation over the exact equations.

Integration along the ray. The algorithm is iterative, but instead of advancing regularly along the ray, we move cell by cell. Each step corresponds to a new intersection between the ray and the grid, thus we always integrate between two intersections, i.e. between two positions on the perimeter of a square cell. This is a brute-force method, and some optimisations will be discussed in the next section.

We start by transposing both positions of the camera and the object from the scene to the fog's vector space. Our algorithm performs the entire integration level by level, and then, for each single function basis level, cell by cell.

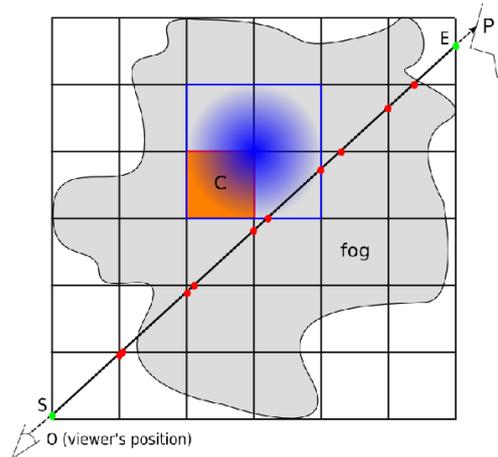


Figure 5: Ray-marching through a single level, designed with linear B-Splines scaling functions (support=2).

To initiate the integration on a given level, we first determine both entry and exit points of our integration on the grid. The entry point corresponds to either the nearest intersection between the ray and the current level's bounding box, or the viewer's position in case he stands within the fog. Similarly, the exit point corresponds to the intersection with either the farthest plane of the bounding box, or with the nearest object if situated within the fog.

When integrating a given level, the contribution of each single cell can be obtained by the product of both the function basis coefficient and the integral of the basis function associated to that cell along the view ray.

Mathematically, considering each cell c intersected by OP and using the extinction function decomposition of (10), we have :

$$\tau(O, P) = \sum_{cell:c \cap OP} K_t = \sum_c \left[\int_{c \cap OP} \alpha_c \phi \phi_c + \sum_{n=0}^{J-1} \int_{c \cap OP} \beta_c^n \phi \psi_c^n + \delta_c^n \psi \phi_c^n + \gamma_c^n \psi \psi_c^n \right] \quad (11)$$

J being the maximum decomposition level of our fog. Thanks to multiresolution analysis, each function indexed by cell c and level n is indeed a translated and scaled version of $\phi \phi$, $\phi \psi$, $\psi \phi$ or $\psi \psi$.

Therefore, we can precompute on the CPU a bunch of integrals for a set of sampled paths (complete or partial) within 1×1 squares on each function's definition domain, so that these values are directly available at runtime, transmitted on the GPU in packed textures. Integration on partial paths allow handling particular cases when the ray either starts and/or ends at the center of a cell within the fog's bounding box.

Figure 5 shows ray \vec{OP} traversing a single level's grid from entry point S to exit point E . Integration steps (i.e. intersections with the grid) are shown in red. The

basis function (in this example : linear B-Spline scaling function) associated to the orange cell's coefficient c is shown in blue.

When using basis functions which are supported on an 1×1 square (e.g. Haar scaling functions and wavelets), their contribution area matches exactly that of the cell it is attached to, therefore we know that the cells which contribute to the pixel being rendered are exactly those traversed by the ray.

When the functions are supported on a domain larger than 1×1 , part of the contribution of each cell gets superimposed on that of its neighbouring cells, thus also contributing to rays which do not necessarily pass through those cells themselves. Actually, a ray passing through a cell must take into account the contribution of that cell, plus the contributions of the $dx - 1$ previous cells on the X axis, times the $dy - 1$ previous cells on the Y axis, where dx and dy are the dimensions of the basis function's definition domain.

When the ray encounters a new function, we only integrate the density on the portion of that function which overlays the current cell, and then resume the integration for another 1×1 square of the same function when the ray traverses the next cell. If we directly integrate on the whole function's support at once, we omit the contributions of the functions attached to cells which are not encountered by the ray.

When the ending point has been reached, the whole process must be repeated with each level of each wavelet basis that was generated by Mallat's wavelet transform.

4.4 Optimizations & multiresolution

Our idea consists in omitting an increasing quantity of details from layers whose resolution is above a threshold which decreases as the observer moves away from the fog. When integrating the fog's density from the observer O to point P , the maximum integration distance d_{\max_l} on level $l \in \mathbb{N}$ is given by :

$$d_{\max_l}(\vec{OP}) = \|\vec{OP}\| \times \mu^l \quad (12)$$

where $\mu \in [0, 1]$ is the optimization coefficient. When $\mu = 1$, the integration is performed entirely on all levels ; on the contrary, when $\mu = 0$, only the upper level of the basis is rendered.

As seen previously, when using scaling functions that are defined on more than an 1×1 square, the integration cost is no longer proportional to the fog's size, since more than each single particular cell traversed by the ray brings a contribution on these cell's area. That's why although the total number of coefficients modeling the fog stays almost unchanged, the rendering cost increases dramatically after the wavelet decomposition, since B-Spline wavelets always have a larger support than their scaling function.

Algorithm 1 Pseudo code of the shader

```

for each pixel do
  sum = 0
  for l = 0 to nb_levels do
    compute 2D entry point on grid
    compute 2D exit point on grid
    while pos  $\neq$  exit do
      inter = compute next intersection with grid
      if (l = 0) then
        coef = get cell coef on approx basis
        approx = integrate on  $\phi\phi$  between pos &
        inter
        sum += coef*approx
      end if
      coef = get cell coef on details1 basis
      det1 = integrate on  $\phi\psi$  between pos & inter
      sum += coef*det1
      coef = get cell coef on details2 basis
      det2 = integrate on  $\psi\phi$  between pos & inter
      sum += coef*det2
      coef = get cell coef on details3 basis
      det3 = integrate on  $\psi\psi$  between pos & inter
      sum += coef*det3
      pos = inter
    end while
  end for
  pixel color=sum*obj color + (1-sum)*fog color
end for

```

When using such basis functions, for example linear or quadratic B-Splines, it can be interesting to use the two-scale relation for wavelets 8 to *deconstruct* the three wavelet bases. This turns them back into scaling function bases, which can then be merged (i.e. added) together. When using scaling functions with a large support, this operation, performed on the CPU just after the decomposition, can reduce the rendering cost by up to 2, while keeping the multi-resolution aspect brought by the decomposition. Moreover, if we perform a deconstruction, we can stop the integration as soon as the sum reaches a particular threshold, close to a great opacity. Deconstruction is important since it assures that each new cell will only add opacity.

5 RESULTS AND DISCUSSION

This algorithm has been implemented using GLSL, an Intel Core 2 Quad 2.8Ghz processor and a NVidia GeForce GTX 280 graphics card. Screen resolution is 800x600.

5.1 Performance

Table 1 show FPS results obtained when using our ray-marching alone to directly render raw Haar, linear or quadratic fogmaps, without any decomposition. Note also that a classical numerical integration along the ray,

Fog resolution	Haar	Linear	Quadratic
16×16	199	142	71
32×32	124	83	31
64×64	90	45	15

Table 1: FPS results with our ray-marching without optimizations.

Nb levels \ μ	1	0.8	0.6	0.4	0.2	0
	0	45	-	-	-	-
1	35	39	47	55	66	83
2	31	45	55	71	90	124
3	27	39	66	76	99	166

Table 2: FPS results with our optimization, using a 64×64 linear B-Spline fogmap.

with 256 samples, runs at 99 FPS and suffers from severe aliasing artifacts. This is clearly outperformed by our method : Haar without decomposition gives 124 FPS and we can achieve similar FPS using a linear base. Each type of basis functions is defined on an area which size is increasing linearly in 1D, which involves a quadratically increasing number of neighbouring cells contributing to the density on each 1×1 square on the grid.

Table 2 show FPS results obtained when rendering a 64×64 linear B-Spline fog using our details dropping optimization, for different values of the tolerance parameter μ . With $\mu = 1$, no details are dropped, and we are performing a simple ray-marching. If, in addition, we do not apply any decomposition step, we are directly rendering the fogmap, like in table 1, therefore this value stands for the threshold above which we have a substantial acceleration.

In table 3 we show, with arrows, the performance gain induced by our optimisations.

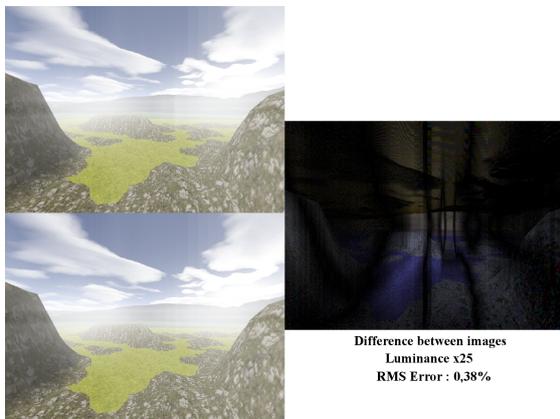


Figure 6: Quality difference when removing all layers of details (bottom left) from an Haar 32x32 (top left). Difference image (right) is shown (x25).

Nb levels	1	2	3
Linear 16x16	58 \rightarrow 111	47 \rightarrow 99	35 \rightarrow 83
Linear 32x32	20 \rightarrow 62	15 \rightarrow 55	13 \rightarrow 49
Quad. 32x32	7 \rightarrow 23	5 \rightarrow 20	5 \rightarrow 18

Table 3: FPS improvement when turning back into scaling function bases the four b-spline/wavelet generated by the decomposition (before \rightarrow after).

5.2 Visual quality

The higher the degree of the B-Spline wavelet is, the smoother each basis function looks. With Haar wavelets, we can see, in figure 7.B, that the visual result is a bit unsatisfactory, with abrupt changes in density which betray the discontinuity of Haar functions. With linear B-Spline wavelets (figure 7.C) the framerate decreases but the visual result is a lot smoother and artifacts and peaks are now practically imperceptible. Finally, with quadratic B-Spline wavelets (figure 7.D), we loose in performance but this time, the quality gain is relatively low compared to linear B-Spline wavelets.

5.3 Discussion

Linear B-Spline seems a good trade-off between speed and quality but Haar could be used if rendering time is an issue. The advantage of using wavelets, beside their property of good data compression, is to have a mathematical representation of heterogeneous fog from physical simulation to rendering. Indeed, animating such fogs is easy, since wavelet decomposition can be performed in real-time. Moreover, unless previous approaches, we perform a precise numerical integration of density along the view ray, without any approximation. In comparison to particle approaches like [19], our method is more adapted to large outdoor scenes when camera is moving in the fog, and the modelling is far more intuitive than using particles.

6 CONCLUSION AND FUTURE WORK

In this paper, we presented a new method for modeling heterogeneous fog using wavelet scaling functions. Rendering is performed through a simple decomposition scheme of the fog density function represented in a scaling function basis leading to sparse data. Wavelets and scaling functions allow and ease a certain number of precomputations, such as the integrals of the wavelets along each ray. A brute force rendering algorithm using the GPU has been presented allowing real-time rendering for moderated complex fog along with an optimized version taking profit of the sparsity of data induced by the wavelet decomposition. We have shown that our method outperforms brute force integration and allows exact computation of the effects of fog, without exotic approximations. Moreover, our method

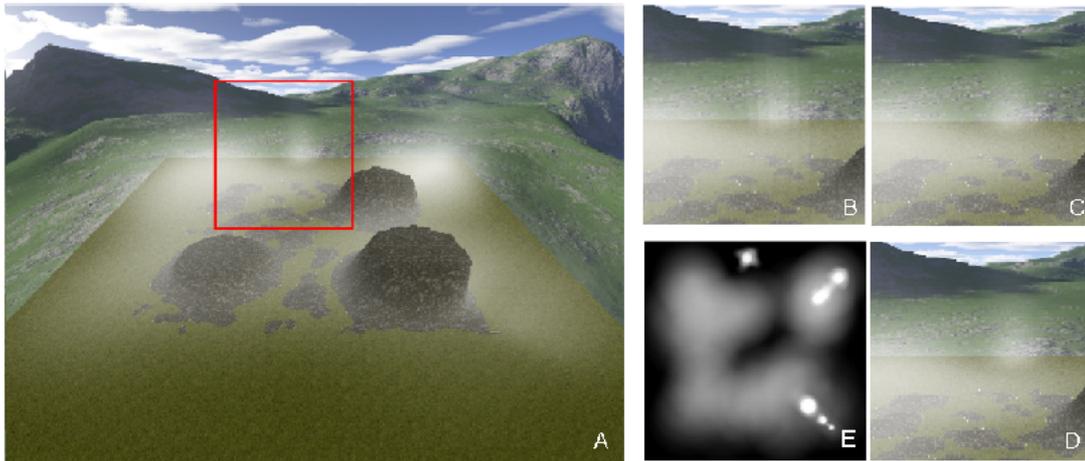


Figure 7: Quality difference with a large 30x30 fog. Fog taken from above (A), and the associated fogmap (E). Zoom on the red part when using Haar (B), Linear (C) and Quadratic (D) wavelets.

do not depends on the position of either the light or the fog, allowing simple transformations of the fog.

The use of wavelets opens the door to other major optimisations for our method. Mainly, the rendering algorithm can be improved by focusing only on the grid's cells which actually contain a non-negligible value, in order to be able to directly jump to the interesting zones of the fog when performing the integration along the ray. For this purpose, we aim at designing a simple GPU traversal of the graph generated by the wavelet decomposition. Since wavelets can be used to solve fluids equations, we also plan to link our rendering algorithm to a physical simulation involving wavelets, allowing a real-time physical animation and rendering of heterogeneous fog. Finally, we plan to add single scattering and volumetric shadows in our model.

REFERENCES

- [1] V. Biri. Real Time Single Scattering Effects. In *Best Paper of 9th International Conference on Computer Games (CGAMES'06)*, pages 175 – 182, November 2006.
- [2] Eva Cerezo, Frederic Perez-Cazorla, Xavier Pueyo, Francisco Seron, and François Sillion. A survey on participating media rendering techniques. *the Visual Computer*, 2005.
- [3] R. Fedkiw, J. Stam, and H.W. Jensen. Visual Simulation of Smoke. In *proceedings of SIGGRAPH'01, Computer Graphics*, pages 15–22, August 2001.
- [4] N. Foster and D. Metaxas. Modeling the motion of a Hot, Turbulent Gas. In *proceedings of SIGGRAPH'97, Computer Graphics*, pages 181–188, August 1997.
- [5] Wolfgang Heidrich, Rüdiger Westermann, Hans-Peter Seidel, and Thomas Ertl. Applications of pixel textures in visualization and realistic image synthesis. In *ISD '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 127–134, New York, NY, USA, 1999. ACM.
- [6] Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. Irradiance Gradients in the Presence of Participating Media and Occlusions. *Computer Graphics Forum (Proceedings of EGSR 2008)*, 27(4):xx–xx, 2008.
- [7] H. W. Jensen and P.H. Christensen. Efficient Simulation of Light Transport in Scenes with Participating Media using Photon Maps. In *Proceedings of SIGGRAPH'98, Computer Graphics*, pages 311–320, August 1998.
- [8] J. Legakis. Fast multi-layer fog. In *Siggraph'98 Conference Abstracts and Applications*, volume Technical sketch, page 266, 1998.
- [9] N. Adabala and S. Manohar. Modeling and rendering of gaseous phenomena using particle maps. *The Journal of Visualization and Computer Animation*, 11(5):279–293, December 2000.
- [10] Nvidia. Fog polygon volumes - rendering objects as thick volumes, 2004.
- [11] Mark Pauly, Thomas Kollig, and Alexander Keller. Metropolis light transport for participating media. In B. Péroche and H. Rushmeier, editors, *Rendering Techniques 2000 (Proceedings of the Eleventh Eurographics Workshop on Rendering)*, pages 11–22, New York, NY, 2000. Springer Wien.
- [12] H. Rushmeier and K. Torrance. The zonal method for calculating light intensities in the presence of participating medium. In *proceedings of SIGGRAPH'87, Computer Graphics*, volume 21(4), pages 293–302, 1987.
- [13] R. Siegel and J.R. Howell. *Thermal Radiation Heat Transfert*. Hemisphere Publishing, 3rd edition, 1992.
- [14] F.X. Sillion. A Unified Hierarchical Algorithm for Global Illumination with Scattering Volumes and Object Clusters. In *IEEE Trans. on Vision and Computer Graphics*, volume 1(3), pages 240–254, September 1995.
- [15] J. Stam. Stable Fluids. In *proceedings of SIGGRAPH'99, Computer Graphics*, pages 121–128, 1999.
- [16] E. J. Stollnitz, A. D. DeRose, and D. H. Salesin. Wavelets for computer graphics: a primer.1. *Computer Graphics and Applications, IEEE*, 15(3):76–84, 1995.
- [17] B. Sun, R. Ramamoorthi, S.G. Narasimhan, and S.K. Nayar. A practical analytic single scattering model for real time rendering. In *proceedings of SIGGRAPH'05, Computer Graphics*, volume 24 (3), pages 1040–1049, 2005.
- [18] D. Zdrojewska. Real time rendering of heterogeneous fog based on the graphics hardware acceleration. In *proceedings of CESC'04*, 2004.
- [19] Kun Zhou, Qiming Hou, Minmin Gong, John Snyder, Baining Guo, and Heung-Yeung Shum. Fogshop: Real-time design and rendering of inhomogeneous, single-scattering media. In *PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, pages 116–125. IEEE Computer Society, 2007.

Efficient Reconstruction From Scattered Points

Helton Hideraldo BÍscaro

University of São Paulo, Brazil

EACH-USP

Av. Arlindo Bettio, 1000.

São Paulo - SP - Brazil

CEP: 03828-000

Tel.: 55 (11)-3091-1020

heltonhb@usp.br

Abstract

Most algorithms that reconstruct surface from sample points rely on computationally demanding operations to derive the reconstruction, beside this, most of the classical algorithm use a kind of three-dimensional structure to derive a two-dimensional one. In this paper we introduce an innovative approach for generating two-dimensional piecewise linear approximations from sample points in \mathbb{R}^3 that simplify significantly the numerical calculation and the memory usage in the reconstruction process. The approach proposed here is an advancing front approach that uses rigid movements in the three-dimensional space and a bidimensional Delaunay triangulation as the main tools for the algorithm. The principal idea is to use a combination of rotations and translations in order to simplify the calculations and avoid the three-dimensional structure used by the most of the algorithms. Avoiding those structures, this approach can reduce the computational cost and numerical instabilities typically associated with the classical algorithm reconstructions.

Keywords: Algorithm, Reocnstruction, Rigid Movements, three-dimensional structures, Delaunay triangulation.

1 INTRODUCTION

Given a set of samples P extracted from a smooth closed surface S in \mathbb{R}^3 , the reconstruction problem consists in reconstruct F , a piecewise linear approximation of S , using the points of P . The surface F must be equivalent to S topologically and as close as possible to S .

In the last decades, surface reconstructions have been focus of extensive investigation not only because the number of practical applications in engineer and virtual museums but also by the challenges that need to be faced. In general only the three-dimensional coordinates of the points are known. Despite of lack of information about the topology and geometry, several algorithms has been proposed to solve this problem [8, 7, 9]. Some of the existing methods can even ensure a correct reconstruction as long as an adequate sampling rate is employed, such as those by Amenta *et al.* [2, 3]. However, in spite of considerable theoretical advances many algorithms fail to accomplish a successful reconstruction in practical situations.

In general, algorithms in literature use three-dimensional structures as Delaunay triangulations,

or a kind of "immersion" three-dimensional space to derive a two-dimensional reconstruction. This paper introduces an advancing front approach, called LDT (Local Delaunay Triangulations) which runs entirely in two-dimensions. The main idea is to start from a boundary edge e and use the n nearest neighbors of one of end points of e to build a two-dimensional Delaunay triangulation in order to choose the better triangle to be glued in e . Avoiding those three-dimensional structures, not only the calculations are simplified, but also the amount of memory used is considerably reduced.

Prior to introducing the LDT algorithm, this work discuss related work in Section 2 and introduce some mathematical fundamentals required to lay out the proposed approach in Section 3. In Section 4 the reconstruction algorithm is described. Reconstruction results with LTD are given in Section 5. Finally, conclusions and further work are addressed in Section 6.

2 RELATED WORK

Surface reconstruction from sample points has deserved considerable attention from researchers in both Computer Graphics and Computational Geometry. The problem became popular after the paper by Hoppe *et al.* [23], who presented an algorithm for reconstructing the surface as the zero set of a signed distance function. However, that approach is unable to capture fine surface details. A related algorithm was developed by Curless and Levoy [14] that is more effective in capturing surface details; nevertheless, it relies on additional information than just the sample points. An alternative

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WSCG 2010 conference proceedings, ISBN 80-903100-7-9
WSCG'2010, February 1 – February 4, 2010
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

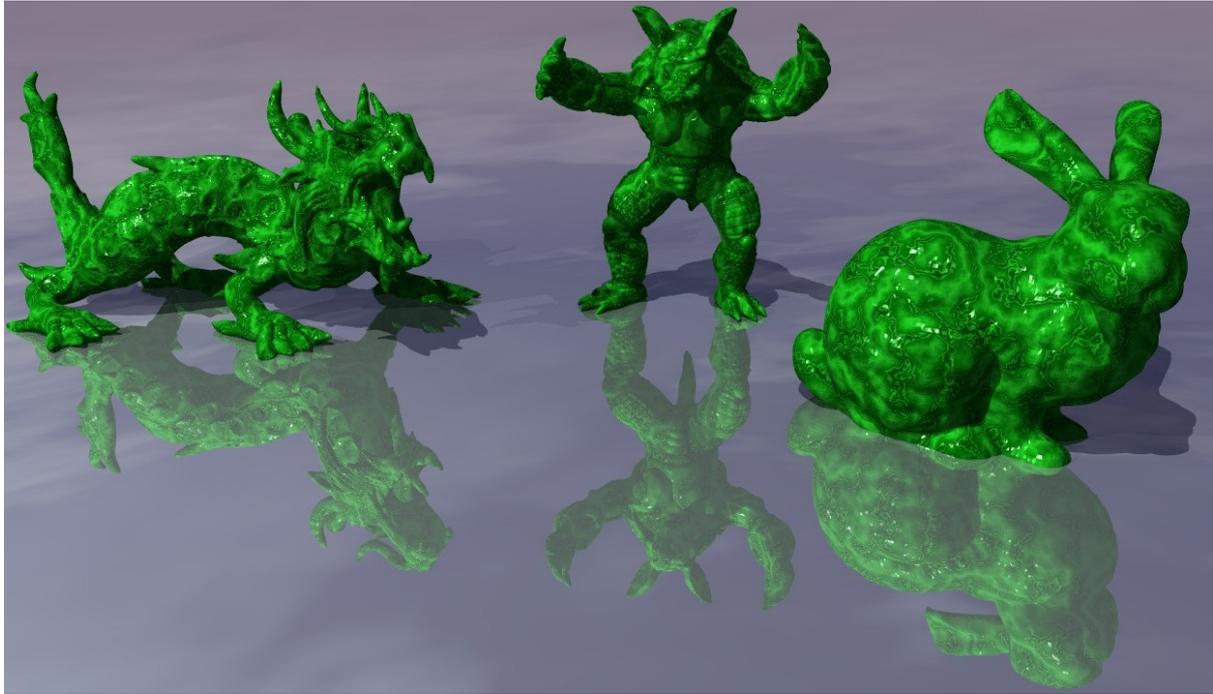


Figure 1: Models reconstructed with LDT algorithm

approaches for reconstructing a surface from the zero set of a distance function have been proposed. Carr et al. [13], for example, employ radial basis functions to approximate the signed distance. Their algorithm, though computationally expensive, can handle gaps and capture fine model details. Ohtake et al. [29] and Alexa et al. [1] use local fitting by employing partition of unit and moving least-squares approximation to estimate the approximating surface. The ability of handling large data sets is a major strength of such implicit approaches. However, the surfaces produced do not interpolate the given samples, which may be undesirable in some applications.

Researchers in Computational Geometry adopted a different approach towards the problem, some of them have proposing reconstruction algorithms based on a Delaunay complex generated from the sample points. The rationale behind such algorithms is to sculpt the surface from the Delaunay complex; others have proposed advancing fronts approaches. Boissonnat [11] proposed the first Delaunay based reconstruction algorithm, which operates by removing tetrahedral and triangles that violate certain geometrical conditions. Unfortunately, it is applicable only to surfaces of genus zero. The α -shape algorithm [17] starts with the Delaunay tessellation of the sample points and removes all simplices that are not contained in an empty ball of radius $\frac{1}{\alpha}$. The α -shape is simple to implement, but it works properly only on evenly sampled point sets, as a single α value applies to the whole data set. Teichmann and Capps [33] introduced a density scaled α -shape to handle this problem. Nonetheless, their ap-

proach requires the normal vectors at the sample points. The Crust, by Amenta and Bern [2], is the first three-dimensional algorithm with theoretical guarantees of reconstruction. For a suitably sampled object it computes a piecewise linear surface approximation that is homeomorphic and geometrically close to the original one. The Crust handles non-evenly sampled point sets and requires little user intervention during reconstruction. A drawback is that the geometrical calculations required to compute the Voronoi vertices introduce numerical instabilities. Furthermore, the algorithm has high computational cost because it builds two Delaunay tessellations, one to compute the Voronoi vertices and a second one to generate the Crust. The Cocone algorithm, by Amenta et al. [4], is an elegant and fast simplification of the Crust that holds the same theoretical guarantees. However, in practical applications it generates undesirable holes in the reconstructed surface. This problem has been solved by Dey and Goswami in the Tight Cocone algorithm [15]. Nonetheless, unlike its predecessor Tight Cocone does not capture internal components. Moreover, it requires pole estimates, cell labeling and, in some cases, triangle size estimates are also necessary. Power Crust [5] also improves on the Crust algorithm. It computes a piecewise linear approximation of a smooth surface employing a weighted Voronoi diagram called Power Diagram. Power Crust is also theoretically guaranteed to generate a correct reconstruction under proper conditions, and its computational performance is superior to that of the Crust. But it still faces numerical instability problems due to the geometrical calculations required to construct the Power

Diagram. Kolluri et al. [25] introduced the Eight Crust algorithm for reconstructing a watertight surface from noisy point cloud data. Starting from the Delaunay tessellation it uses a variant of spectral graph partitioning to decide whether each tetrahedron is inside or outside the original object. The reconstructed surface consists of the set of triangular faces shared by both internal and external tetrahedral. The spectral partition makes local decisions based on a global view of the model and therefore the algorithm can ignore outliers, patch holes and under-sampled regions. The high computational cost is still a major disadvantage.

The ball pivoting algorithm by Bernardini et al. [9] is very simple and fast. Three points form a triangle if a ball of user-specified radius touches them without containing any other point. Starting from a seed triangle, the ball pivots around an edge – i.e., it revolves around the edge while keeping in contact with the edge's endpoints until it touches another point, forming another triangle. The process proceeds until all reachable edges have been tried, and then it starts over from another seed triangle, stopping when all points have been considered. The process can be repeated with a ball of larger radius to handle uneven sampling densities. A major advantage of ball pivoting is that it does not compute the Delaunay tessellation of the sample points. On the other hand, it is user-dependent and needs the normals at the samples. Advancing front strategies have been employed in reconstruction algorithms by several authors, such as Schreiner et al. [30] and [31], but computational implementation of such methods can be quite intricate.

Edelsbrunner [16] derived an algorithm for fitting a surface to a set of sample points that relies on classical Morse theory. Although it relies on a topological background, topology is employed just to deduce the geometrical calculations. Another approach that uses Morse theory, in its discrete version is the work of Bís-caro *et al.* [10] which uses a discrete Morse function defined in a three-dimensional Delaunay triangulation to guide the reconstruction process. Also, the main drawback of this work is the three-dimensional structure required to extract a two-dimensional one.

Finally, Gopi *et al.*[20] has proposed a similar approach that uses a local Delaunay triangulation. However, their approach selects a set of candidate points which might be possible neighbors of a vertex in the final triangulation using a kind of sample criteria. They also compute the local Delaunay triangulation in the tangent plane without using any kind of simplification in its computation.

In fact, most of the classical algorithms derive the reconstruction from a subset of the three-dimensional Delaunay tessellation. This approach avoid to construct a three-dimensional structure to derive a two-dimensional piecewise linear approximation of the sur-

face. Avoiding this immersion space, the algorithm presented here reduces the amount of ram memory used in the process as well as number of geometrical calculations. Another advantage of avoiding a three-dimensional Delaunay triangulation is absence of sliver tetrahedrons, which is a classical problem in three-dimensional triangulations.

3 BASIC CONCEPTS

This Section introduces the basic concepts and the terminology used in the remainder of the text.

A Delaunay triangulation for a set P of points in \mathbb{R}^n is a triangulation of $DT(P)$ such that no point in P is inside the circumsphere of any simplex in $DT(P)$. In the plane, each vertex has on average six surrounding triangles; also, this triangulation maximizes the minimum angle. Compared to any other triangulation of the points, the smallest angle in the Delaunay triangulation is at least as large as the smallest angle in any other [21, 18].

Let S be a smooth closed surface in \mathbb{R}^3 , i.e., S is C^1 -continuous and divides \mathbb{R}^3 into open solids. A ball B is said to be empty (with respect to S) if its interior contains no point of S . The set of centers of the maximal empty balls touching S in at least two points make up the medial axis of S . The local feature size of a point s in S , denoted $lfs(s)$, is the distance from s to the medial axis of S . An important property of $lfs(\cdot)$ is that $lfs(p) \leq lfs(q) + |pq|$, where $|pq|$ is the distance between p and q . A set of points $P \subset S$ is an r -sample of S if the distance from any point $s \in S$ to the closest point in P is at most $r \times lfs(s)$. In this case S is said to be r -sampled; in general, good results in reconstructions are achieved for $r \leq 0.1$.

Quaternions (four numbers) are a kind of number system that extends the complex numbers. A quaternion number $q = (w, x, y, z)$, or correspondingly, $w + ix + jy + kz$, where hold the following identities; $i^2 = j^2 = k^2 = -1$, $ij = k - ij$ and $w, x, y, z \in \mathbb{R}$ [22]. They also provide a useful mathematical notation for representing and rotations of objects in three dimensions. When compared to Euler angles, they are simpler to compose and have an advantage of not present the problem of "gimbal lock". Also, they are more numerically stable a more efficient than rotations matrices. To represent a rotation of an angle θ around the axe n , a unit vector, is enough to define the quaternion $q = (\cos(\frac{\theta}{2}), \sin(\frac{\theta}{2})n)$.

This work also need an efficient and effective way of find the n nearest neighbors of a three-dimensional point p . To accomplish this, the work of Lin and Yang [27] was used. Their work offer a high accuracy nearest neighbor search by their ANN-Tree (Approximate Nearest Neighbor Tree) which is a tree based structure that works for arbitrary dimension.

Another important calculation present in this work is the angles between two vectors. According to Jonathan

Shewchuck [32], given two vectors with the same origin r and s , the best way of calculate the angle between r and s is to use the formula $\tan(\theta) = \frac{2A_f}{\langle r,s \rangle}$, where A_f is the area of the triangle with sides r and s . The computation of A_f can be done making $A_f = \frac{|r \times s|}{2}$, where $r \times s$ is the cross product of r and s .

This paper uses the Hausdorff distance to compare the meshes generated by the LDT algorithm and the meshes of the classical algorithms. Hausdorff distance is a generic technique that defines a distance between two nonempty sets; and has been used as an efficient tool to evaluate distances between three-dimensional meshes [6].

In next section, this paper presents details of the algorithm developed in this work.

4 ALGORITHM

The algorithm LDT uses the normal vector at each sample point. There are several strategies to estimate this vector, but is important to include, in such estimation, the impact of the point's distance. The influence of the sample points must be inversely proportional to its distance. This work uses weighted principal component analysis (WPCA). The weight average of a point $p \in \mathbb{R}^3$ is given as follow:

$$M(p) = \sum_{i=1}^n \frac{w_p(p_i) p_i}{\sum_{i=1}^n w_p(p_i)} \quad (1)$$

where n is the number of the nearest neighbors of p , the function $w_p(x)$ specifies the influence of the point x in point p . According to Levin [26], a good choice is $w_p(x) = e^{-\frac{\|x-p\|}{H}}$, where H estimates the local density in p , $H = \sum_{i=1}^n \frac{\|p_i-p\|}{n}$. The 3×3 covariance matrix C for a point p if given by:

$$C = \begin{pmatrix} p_1 - M(p) \\ p_2 - M(p) \\ \vdots \\ p_{n-1} - M(p) \\ p_n - M(p) \end{pmatrix}^T \begin{pmatrix} p_1 - M(p) \\ p_2 - M(p) \\ \vdots \\ p_{n-1} - M(p) \\ p_n - M(p) \end{pmatrix} \quad (2)$$

Let $\lambda_1 \leq \lambda_2 \leq \lambda_3$ be the three eigenvalues of C , and α_1, α_2 and α_3 the three associated eigenvectors. Jolliffe, in his work [24] establish that α_3 is the direction of greatest variance in a neighborhood of p , α_2 represents the direction of second greatest variance and α_1 the direction that minimizes the variance. As the set of points P is a subset of the surface S , the geometric interpretation is that α_2 and α_3 approximates the main directions of the tangent plane at p and α_1 approximates the normal direction.

Data: A set of samples $P \subset \mathbb{R}^3$

```

1 for each  $p \in P$  do
2   | Approximate the normal vector in  $p$ 
3 end
4 Find an initial triangulation  $F$ ;
5 Store in  $E$  the boundary edges of  $F$ ;
6 while  $E \neq \emptyset$  do
7   | Remove  $e$  from  $E$ ;
8   | if  $e$  still is a boundary edge then
9     |    $f \leftarrow \text{FindNewFace}(e)$ ;
10    |    $F \leftarrow F \cup \{f\}$ ;
11    |   Add to  $E$  the boundary edges of  $f$ ;
12  | end
13 end
14 return  $F$ 

```

Algorithm 1: Algorithm LDT - Local Delaunay Triangulation

The main idea in the LDT algorithm is to execute an advancing front approach to achieve the reconstruction. This advancing front technique uses a two-dimensional Delaunay to get the next triangle from a boundary edge. The pseudo-cod 1 shows the main loop of the algorithm developed here. In the step 2, the weighted principal component analysis is used to approximate the normal vectors in the samples points. The initial triangulation (step 4 in the algorithm 1) is acquire choosing an initial point p , projecting its n nearest neighbors in its tangent plane, computing the Delaunay triangulation in the plane and re-projecting the triangulation in the surface. The two-dimensional Delaunay triangulation was implemented using the only the first and the second coordinates of the samples. Considering this, the algorithm must rotate p and its neighbors such that the tangent plane in p coincide with the XY plane. By doing this rotation, the projection operation is expressively simplified. The Figure 2 illustrates this initial step, showing a normal vector in an initial point of a paraboloid, the blue points are the nearest neighbors of the initial point and the orange plane is the tangent plane where the neighbors are projected.

The set E store the boundary edges of the triangulation F and can be interpreted as a list of active edges that guide the reconstruction process. The main loop of the algorithm is repeated while E is not an empty set. It is worth to mention that when an edge e is removed from E , it is possible that e is not a boundary edge anymore. Also is possible that a new face f returned in the step 9 of the algorithm 1 has no boundary edges.

The pseudo-code 2 illustrates the procedure to expand the frontier of F , and is a variation of the procedure to achieve the initial triangulation. The idea is also to project the neighbors in the tangent plane π , execute a two-dimensional Delaunay DT triangulation with the

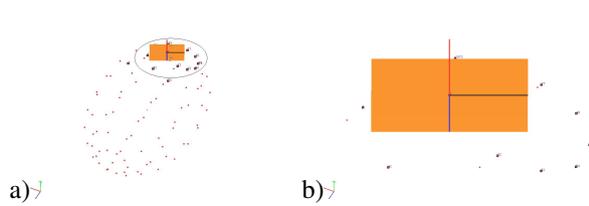


Figure 2: a) Set of samples of a 3D object b) Zoom view of the initial point.

Data: A boundary edge e .

- 1 Let p be one of the end points of e ;
- 2 Rotate p and its n nearest neighbours to align the normal in p with the Z axis and the edge e with Y axis ;
- 3 Find p_e , the opposite vertex to e ;
- 4 Project in the tangent plane only the vertex p_i such that $p_{e_x} * p_{i_x} \leq 0$;
- 5 Find DT , a two-dimensional Delaunay triangulation with the projected vertices ;
- 6 Find f , the triangle of DT that contain e as a boundary edge ;
- 7 **return** f

Algorithm 2: Algorithm FindNewFace

project points, to choose from DT , the face f that has e as boundary edge, and re-project f in the surface.

To ensure that the projection of e appear in the local triangulation, consider $f_e \subset F$ the face of F containing e , and p_e the vertex of f_e opposite to e . When the vertex p , which is one of the end points of e , and its nearest neighbors are rotate to align the edge e with the Y axis, the x coordinate of p_e is either positive or negative depending of its relative position. After that, only the neighbors that has x coordinate with opposite signal when compared with p_e are projected in π . This procedure is enough to ensure that the projection of e appear in the boundary of the Delaunay triangulation DT . It is worth to mention that at this point of the algorithm (step 4 of the algorithm 2), only the boundary vertex in the neighborhood of p or vertices that are not contained in a face are considered to be projected in the tangent plane π .

Two steps of a paraboloid reconstruction can be seen in the Figure 3 a) and b). The distinct face represents the last face glued in the mesh and the wider edge represents the first edge in the list E of active edges. In the Figure 3 c) the complete reconstruction is showed. The figure 4 exhibit a local Delaunay triangulation's example for a set of sample points, and again, the distinguished face is the one captured to be re-projected in the surface.

Although this algorithm does not need to handle sliver tetrahedral, which is a very common problem in sculpturing techniques, it is possible that the algorithm

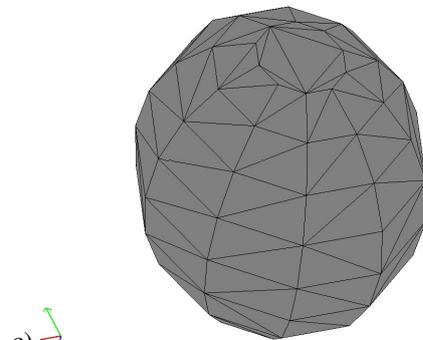
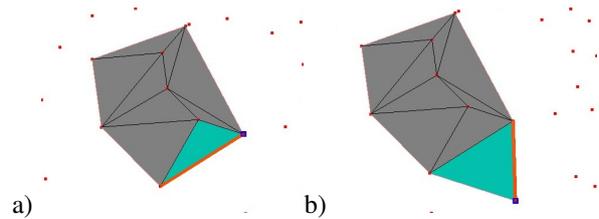


Figure 3: Two steps in the reconstruction of a paraboloid

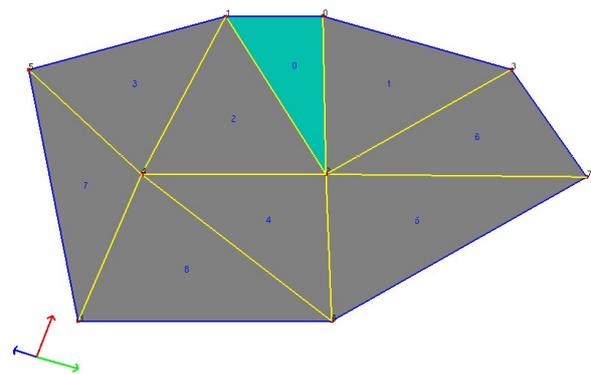


Figure 4: Local Delaunay triangulation to a set of sample points

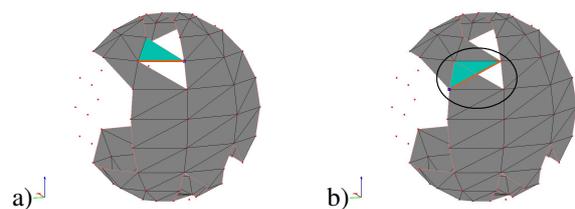


Figure 5: Small dihedral angles

glue faces with small dihedral angles as is showed in the Figure 5 The Figure presents two consecutive steps of the paraboloid reconstruction. However, according to the work of Mederos *et al.* [28], the dihedral angle between two adjacent faces approximates to π when the sample rate increases. To avoid this problem, a dihedral angle calculation, given by the work of Jonathan Shewchuck [32], must be done before glue a new face in the mesh.

4.1 Discussions

There are some crucial points in the LDT algorithm. The estimation of normal vectors in the samples points for instance, plays a crucial role in all reconstruction process (algorithm 1 step 2). Of course that the LDT algorithm does not have intention of reconstructs arbitrary surfaces with arbitrary sampling rate. In order to achieve a good normal estimation in all samples; is acceptable that a minimum sampling rate be respected. However, this is a theoretical study that will be subject of a future work. Another consideration, is about the projection effect over the algorithm's results. According to Amenta's work [3], for an adequate sample rate, in general, a r -sampled surface with $r \leq 0.1$, the correct reconstructions lies in a subset of the Delaunay complex of the samples points. Therefore, respecting this sampling condition, for an arbitrary sample point p , its neighbors must lie close to the tangent plane in p ; not causing ample movements in the projection operation as well.

It is also worth to mention that in the initial triangulation (cod:algorithm step 4) no glue operation is needed. Therefore, the Delaunay triangulation computed I this step can be re-projected directly in the output surface.

The next section presents some of results obtain with the LDT algorithm.

5 RESULTS

This section shows some examples of models reconstructed with LDT algorithm as well as some comparisons with classical algorithms in the literature. For the comparisons was used in-house implementations, based on CGAL [12], of the Crust and Power Crust developed as part of a master dissertation project [19]; the TSR implementation was part of a previous work [10] and the original implementations of Cocone and Tight Cocone were kindly provided by Tamal Dey. The reconstructions were performed on a dual Pentium 4 with 3 GHz and 1GB RAM.

The figure 6 give an idea of the quality of the mesh generated by the algorithm LDT in a reconstruction of a bitorus. The Figures 7 and 8 show additional reconstructions examples, the dragon model is rendered with a jade texture and the hand model with a stone texture, and the Figure 10 shows the Lucy model reconstructed from a large data set (921085 points).

In the table 1 the usage of memory, in Kbytes, of some classical algorithm is exhibit. The algorithms are Crust, Power Crust, Cocone Tight Cocone, TSR and that of LDT, for a set of standard sample sets, identified in the top Table line (models shown were generated with LDT). The Crust and the Power Crust Algorithm produces no output to the Isis model, the fourth in the table. The Figure 9 represents the running times, in seconds, of three traditional reconstruction algorithms,

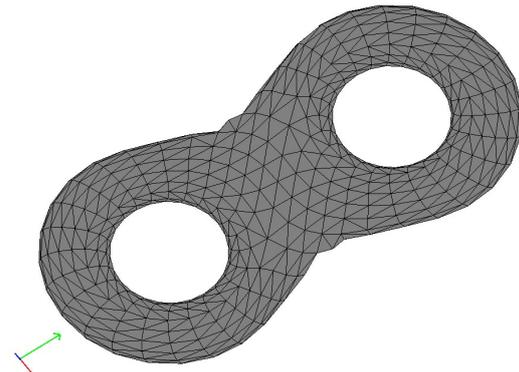


Figure 6: Mesh generated with LDT algorithm

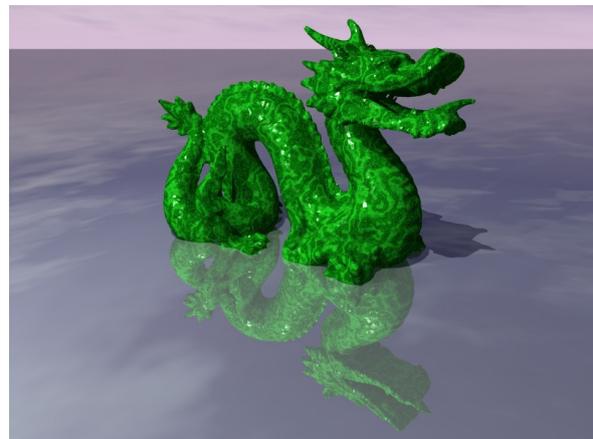


Figure 7: Dragon Model generated with LDT algorithm and rendered with a jade texture



Figure 8: hand Model generated with LDT algorithm and rendered with a stone texture

Cocone, Tight Cocone , TSR and LDT. As the table 1 reveal, the LDT algorithm, due its optimizations and its advancing front approach, lean to use less memory than the others. One observes that the running times were improved, particularly when the models are bigger than 50,000 points. One can see in Figure 9 that the LDT al-

gorithm is faster than the classical algorithms compared with it, especially when reconstruct large data sets.

				
	9697 pts	35947 pts	54707 pts	187644 pts
Cr	21.90	53.13	72.21	No output
PC	41.94	79.50	110.54	No output
Co	15.10	51.24	78.98	265.32
TC	19.20	66.10	101.68	343.50
TSR	20.65	71.28	111.72	376.19
LDT	12.48	38.75	59.17	192.22

Table 1: Usage of memory in k bytes to reconstruct the models shown Legend: Cr - Crust; PC - Power Crust; Co - Cocone; TC - Tight Cocone; TSR - Topological Surface Reconstructor; LDT - Local Delaunay Triangulation

The table 2 exhibits the Hausdorff distances between the LDT' meshes and meshes generated by other classical algorithms (cocone, Tight-cocone and TSR). The distances are quite small; suggesting that the output meshes are very similar.

				
	9697 pts	35947 pts	54707 pts	187644 pts
Co	0.003235	0.001485	0.213477	0.000561
TC	0.002139	0.001181	0.446078	0.000525
TSR	0.002480	0.001018	0.006002	0.000074

Table 2: Hausdorff distance between the meshes generated with LDT algorithm and the follow ones : Co - Cocone; TC - Tight Cocone; TSR - Topological Surface Reconstructor

6 CONCLUSION AND FUTURE WORK

This work introduces an innovative approach, called LDT - Local Delaunay Triangulation, to reconstructing piecewise linear approximations of surfaces in \mathbb{R}^3 that are defined by set of samples. The approach present here is an advancing front approach which make use of a two-dimensional Delaunay triangulation to choose the adequate triangle to be glued in the mesh. The main advantage of this kind of technique is to avoid the use of three-dimensional structures when the goal is to derive a two-dimensional one. Principal component analysis is used to estimate the normal vector in the samples points, and rigid movements are used to optimize the projections operations.

By avoiding those three-dimensional structures, the LDT algorithm improves not only the running times,

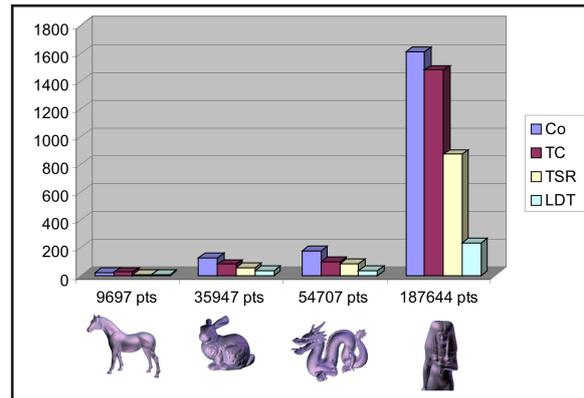


Figure 9: Running time in seconds to reconstruct the models shown Legend: Co - Cocone; TC - Tight Cocone; TSR - Topological Surface Reconstructor; LDT - Local Delaunay Triangulation



Figure 10: Lucy model reconstructed with LDT

but also the amount of memory used in the reconstruction process, which enable it to reconstruct models with considerable quantity of points, as showed in the lucy model (Figure 10).

Unfortunately, was not possible to compare the LDT algorithm with the one developed by Gopi *et al.*[20], which is another two-dimensional approach that uses Delaunay triangulation. Basically, the Gopi' approach uses a sample criteria to select the candidate points in computation of the Delaunay triangulation. The comparison of the two techniques must be subject of a future work.

Another step to be analyzed is the possibility of substitute the two-dimensional Delaunay triangulation by another kind of calculation, which cam makes this algorithm even faster. Another possibility for future work is to produce theoretical guarantees of the reconstruction, that is, to investigate for which value of r the LDT produces a correct reconstruction of a r -sampled surface.

ACKNOWLEDGEMENTS

The authors are grateful to Tamal K. Dey for providing his implementations of the Cocone and Tight Cocone; and to João Paulo Gois for his implementation of Crust and Power Crust.

REFERENCES

- [1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, 2003.
- [2] N. Amenta and M. W. Bern. Surface reconstruction by voronoi filtering. In *Symposium on Computational Geometry*, pages 39–48, 1998.
- [3] Nina Amenta, Sunghee Choi, Tamal K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *International Journal of Computational Geometry and Applications*, 12(1-2):125–141, 2002.
- [4] Nina Amenta, Sunghee Choi, Tamal K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *International Journal of Computational Geometry and Applications*, 12(1-2):125–141, 2002.
- [5] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry*, 19(2-3):127–153, 2001.
- [6] Nicolas Aspert, Diego Santa-Cruz, and Touradj Ebrahimi. Mesh: Measuring errors between surfaces using the hausdorff distance. In *Proc. of the IEEE International Conference in Multimedia and Expo (ICME) 2002*, volume 1, pages 705–708, Lausanne, Switzerland, August 2002.
- [7] D Attali. r -regular shape reconstruction from unorganized points. *Computational Geometry Theory and Applications*, 10:239–249, 1998. Elsevier.
- [8] Chandrajit L. Bajaj, Fausto Bernardini, and Guoliang Xu. Automatic reconstruction of surfaces and scalar fields from 3d scans. *Computer Graphics*, 29(Annual Conference Series):109–118, 1995.
- [9] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999.
- [10] Helton Hideraldo BÍscaro, Antonio Castelo Filho, Luis Gustavo Nonato, and Maria Cristina Ferreira de Oliveira. A topological approach for surface reconstruction from sample points. *Vis. Comput.*, 23(9):793–801, 2007.
- [11] J D Boissonnat. Shape reconstruction from planar cross-sections. *Computational Vision Image*, 44:1–29, 1988.
- [12] Jean-Daniel Boissonnat, Olivier Devillers, Sylvain Pion, Monique Teillaud, and Mariette Yvinec. Triangulations in cgal. *Computational Geometry - Theory and Applications*, 22:5–19, 2002.
- [13] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76. ACM Press, 2001.
- [14] B Curless and Levoy M. A volumetric method for building complex models from range images. In *Siggraph*, pages 303–312, 1996.
- [15] Tamal K. Dey and Samrat Goswami. Tight cocone : A water-tight surface reconstruction. Technical Report OSU-CISRC-12/02-TR31, The Ohio State University, december 2002.
- [16] Hebbert Edelsbrunner. Surface reconstrution by wrapping finite point set in space. *Discrete and Computational Geometry. The Goodman-Pollack Festschrift*, pages 379–404, 2003.
- [17] Herbert Edelsbrunner and Ernst P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43 – 72, 1994.
- [18] Fortune. Voronoi diagrams and delaunay triangulations. In *Computing in Euclidean Geometry, Edited by Ding-Zhu Du and Frank Hwang, World Scientific, Lecture Notes Series on Computing – Vol. 1.* 1992.
- [19] João Paulo Gois. Reconstrução de superfícies a partir de nuvens de pontos. -in portuguese. Master's thesis, Universidade de São Paulo - Instituto de Ciências Matemáticas e Computação., 2004.
- [20] M. Gopi, S. Krishnan, and C. T. Silva. Surface reconstruction based on lower dimensional localized delaunay triangulation. In M. Gross and F. R. A. Hopgood, editors, *Computer Graphics Forum (Eurographics 2000)*, volume 19(3), 2000.
- [21] Leonidas Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi. *ACM Trans. Graph.*, 4(2):74–123, 1985.
- [22] W.R. Hamilton. *Elements of Quaternions*. Chelsea Publishing Company, third edition, 1969 - The original was published in 1866.
- [23] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, 1992.
- [24] Ian T. Jolliffe. *Principal components analysis*. Springer Us, 2002.
- [25] Ravikrishna Kolluri, Jonathan Richard Shewchuk, and James F. O'Brien. Spectral surface reconstruction from noisy point clouds. In *Sigraph*, pages 11–22, 2004.
- [26] D. Levin. Mesh-independent surface interpolation. *Geometric Modeling for Scientific Visualization*.
- [27] King-IP Lin and Congjun Yang. The ann-tree: An index for efficient approximate nearest neighbor search. In *DASFAA '01: Proceedings of the 7th International Conference on Database Systems for Advanced Applications*, pages 174–181, Washington, DC, USA, 2001. IEEE Computer Society.
- [28] Boris Mederos, Luiz Velho, and Luiz Henrique de Figueiredo. Moving least squares multiresolution surface approximation. In *SIBGRAPI 2003*, 2003.
- [29] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. *ACM Trans. Graph.*, 22(3):463–470, 2003.
- [30] John Schreiner, Carlos E. Scheidegger, Shachar Fleishman, and Cláudio T. Silva. Direct (re)meshing for efficient surface processing. *Computer Graphics Forum*, 25(3):527–536, 2006.
- [31] Andrei Sharf, Thomas Lewiner, Ariel Shamir, Leif Kobbelt, and Daniel Cohen-Or. Competing fronts for coarse-to-fine surface reconstruction. In *Eurographics*, pages 389–398, Vienna, september 2006.
- [32] Jonathan Richard Shewchuk. Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete & Computational Geometry*, 18:305–363, 1996.
- [33] Marek Teichmann and Michael Capps. Surface reconstruction with anisotropic density-scaled alpha shapes. In David Ebert, Hans Hagen, and Holly Rushmeier, editors, *IEEE Visualization '98*, pages 67–72, 1998.

Creating Continuous Force Feedback for Haptic Interaction of Volume Data Sets

Y. Liu S. D. Laycock
School of Computing Sciences, University of East Anglia
Norwich, NR4 7TJ, UK
{yu.liu|s.laycock}@uea.ac.uk

ABSTRACT

Interacting with volumetric models via a haptic device presents an effective way of perceiving details concerning the models internal structures. Approaches to facilitate this range from interacting directly with the volume data to interacting with a polygonal surface derived from the data. Previous approaches have utilised a force field to provide continuous forces such as the Force-Map method which assigns a force vector at any position in the virtual environment. Nevertheless, the Force-Map method is still limited in simulating fast moving drilling due to the fact that there are no forces inside the volume. It suffers from a pop through problem when the virtual drill quickly moves against the volume object. To circumvent this problem, the work presented in this paper introduces a Level-Box method to improve the Force-Map method by encoding the object's internal area into a number of levels which not only enables the user to touch the volume object by using a Force-Map, but also accelerates the Force-Map update procedure when drilling. Users can select from a variety of virtual tools to gain continuous and smooth force feedback during the drilling of volumetric data which increases the applicability of the approach.

Keywords

Volume haptics, Marching cubes, Force-Map haptic rendering, Level-Box,

1 INTRODUCTION

The potential for the use of volumetric data in medical applications has been well established. Recent developments in graphics accelerator cards have enabled systems to render large and complex volumetric data sets in a variety of different rendering styles, aiding the observer's perception of the data. Previous work in interactive simulation of volumetric data has focused primarily on visualization. By integrating haptic technology, an important emerging area related to volumetric visualization has developed to build up a visual haptic system which enables the user to interact with the volume data via a haptic feedback device. The visualizations that were linked with haptic feedback devices to enable the user to touch the volumetric data were introduced in 1993 by Iwata and Noma. They used their approach for the haptic interaction of data produced in

Computational Fluid Dynamics. In this case a force could be mapped to the velocity and torque mapped to the vorticity [5]. Virtual Sculpting systems linked to haptic feedback devices have been available for many years; however, these often do not ensure the modified data remains faithful to the characteristics of the original volumetric data. In this paper, a Level-Box approach to improve the Force-Map haptic rendering method for drilling into surfaces based on the volumetric data is presented.

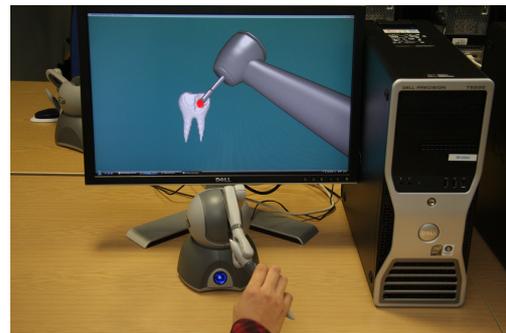


Figure 1: The visual-haptic system illustrating drilling into a volumetric object constructed from CT data.

The major objective for the design of the visual haptic system is to gain a fast haptic and graphic refresh rate at which the calculations must be efficiently performed. Based on the results of analyzing human fac-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

tors, an update rate of 1KHz is required in order for a user to perceive stable and smooth haptic feedback from the visual haptic system. This is in contrast to the visualization which must update at approximately 30Hz to ensure the graphic scene is perceived as a smooth and continuous animation. If the haptic update frequency is lower than 1KHz, an obvious vibration can be felt from the haptic device. One objective of this work is to create a system which can accurately render volume data at sufficient rates for both the visualization and the haptics. For the field to move beyond today's state of the art, researchers must surmount a number of technological barriers. Firstly, the volume data updating algorithm must be fast, especially considering the fact that the surface representation of the volume data may be constructed from millions of triangles. Secondly, the haptic feedback should be rendered such that when the probe point is moving across the voxel boundaries a continuous force is returned to the user. Lastly, since the haptics and visualization calculations will be performed in separate threads, mechanisms are required to ensure that each thread can be updated in a safe manner.

2 PREVIOUS WORK

A large proportion of the previous volume haptic rendering approaches have concentrated on the use of a surface-based haptic rendering technique. An intermediate surface can easily be extracted using Marching Cubes to enable forces to be calculated utilizing a standard constraint-based method [15, 4]. However, this suffers from stability problems which occur when the surface is updated. This motivates researchers to develop algorithms which directly haptically render the isosurface extracted from the volumetric data. The direct volume haptic rendering approach is capable of providing a way to generate force feedback directly from the volume data without extracting an intermediate representation. Even though it is able to represent the force at any position in the volume data, the haptic feedback generated by this method suffers from force instabilities since it is difficult to properly decide the rendering parameters in the force function. This is especially the case when the function is changing during the process, such as when drilling or milling, in real applications. Moreover, forces may vary significantly in strength and direction which sometimes can not be represented by a simple mapping method.

Morris et al. [12] simplifies the computations for drilling through the use of another point-shell method to compute haptic interactions and bone erosion for spherical drill bits. In contrast to the work of Pflesser et al. [13], Morris et al. use the data within the spherical tool to perform bone removal as opposed to sampling points on the tool's surface. Both of these approaches limit the user to drilling with a spherical drill. Eriksson et al. [2] proposed a haptic milling surgery simulator using a

localized Marching Cubes algorithm for the visualization. To improve the stability they employed a direct haptic rendering method with mechanisms to remove fall-through issues. The data inside the virtual drill is set to a vector pointing to the centre of the voxel. The output force is the sum of all those vectors. This approach works well when the drilling tool moves in a small area, but a "kicking" would result when the haptic test points move across the cubes' boundaries.

A Force-Map method is proposed by Liu and Laycock [6] to solve these problems which encode the whole virtual 3D space in an invisible map for haptic rendering and is able to generate smooth force feedback. It allows arbitrary shapes of drilling tools. But simulators are still limited to haptic rendering methods which use the surface based haptic rendering approach for touching the object. What is more, the force calculation suffers from the pop through problem due to the Force-Map only being calculated near to the surface. This is particularly likely to occur when the operation is performed by a fast moving drilling tool. In order to alleviate these issues, the work presented in this paper introduces a Level-Box method to improve the Force-Map haptic rendering algorithm which enables the visual haptic system to use a single approach to rendering for the standard interaction and also when drilling. Additionally, it can more efficiently update the Force-Map to gain smooth and stable force feedback during drilling into the volume data.

McNeely et al [10] proposed a distance field method to give an advance warning of any potential contacts between the tool and the objects. They extend the voxelization of an object beyond its surface into free space surrounding the polygonal object, marking free-space voxels with different integer values that represent a conservative estimate of distance-to-surface expressed in units of voxel size. The work presented in this paper uses a similar distance field idea to encode the non-surface free-space voxels into a number of layers according to the Euclidean distance to the surface. In contrast to McNeely's work, we encode the internal voxels of the volume object in this work with our Level-Box approach. The method is described in detail in Section 5.

Yau et al [14] also proposed a visual haptic system for training dental students by using surfel models. They use an octree based box to define the internal area of the teeth, when the drilling changes the shape of the teeth models, the internal boxes are dynamically updated which increases the octree level to create a modified surface. In spite of the advantages of using variable shapes of drilling tools, the haptic rendering update occurs under 1 kHz which does not meet the requirement of a stable haptic rendering system.

3 VOLUME DATA MODIFICATION

The volume-based representation is a natural choice for rendering a collection of digital images produced by medical scanning technologies such as Magnetic Resonance Imaging (MRI) or Computed Tomography (CT). There are a variety of graphical rendering techniques for visualizing the three dimensional data, often with options to display the material properties such as density and viscosity within the voxels. This has the potential to greatly enhance a user's performance in medical and scientific three dimensional data exploration.

When using the Marching Cubes algorithm [8], a volume can be interpreted by generating polygons representing the surface, typically constrained to a specified value of the data. But extracting the global iso-surfaces from the volume data based on Marching Cubes can be time consuming especially when the volume data is derived from many high resolution digital images. However, in this work a local Marching Cubes algorithm is employed to enable the surface to be updated efficiently. The values of the volume data surrounding the haptic stylus can be adjusted to less than a surface threshold value depending on the application. By considering the material properties of the data contained within a voxel the rate at which the data is removed can be adjusted. Once the data has been updated, the local Marching Cubes approach recomputes the surface surrounding the stylus. The volume that is updated depends on the resolution of the volume data and the shape of the tool used for the interaction.

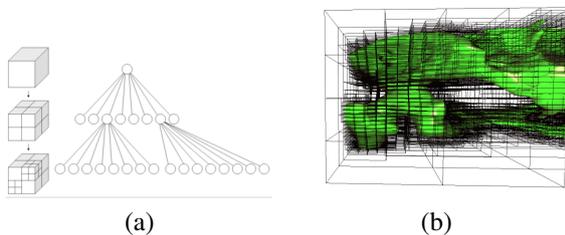


Figure 2: (a) Octree data structure, (b) Pelvis data construction using Octree data structure.

To handle large data sets, an Octree based structure [3] is employed which enables the data to be changed dynamically in an efficient manner. The Octree based structure uses a hierarchical representation of the data to efficiently detect and update localized changes to the data [11]. Each node in the octree represents a cell which contains triangles. Initially paths in the octree from the root to a leaf (voxel) will only be created if triangles forming the surface reside in the voxel, Figure 2. If the haptic stylus reaches a region and edits the data where no surface triangles are present then a new surface is likely to result. At this point the octree is updated by traversing from the root to the leaf containing the modified data, creating any new cells for the octree that do not previously exist. If the data changes such

that an octree cell no longer contains triangles on the surface, then the triangles and octree cells are removed from the structure.

The efficiency of the approach is affected by the chosen depth of the octree. There is a trade-off between the quality of the visualization and the efficiency of the approach. If a small octree depth is used fewer voxels containing large triangles will result, which can often exhibit undesirable edge aliasing. Conversely, too many voxels caused by higher octree depths will increase the computational load of updating the surface during tool-object intersection. The Octree depth selection also depends on the size of the volume data. If the grid is too small then the visualization is more complex when dealing with a huge number or a large area of volume data.

4 SURFACE EXTRACTION AND MODIFICATION

The visual haptic system presented in this paper is able to function with an arbitrarily shaped drilling tool composed of polygons. This extension strives further than other work which only employs simplistic objects, such as single spheres or cylinders as the drilling tools represented by implicit functions.

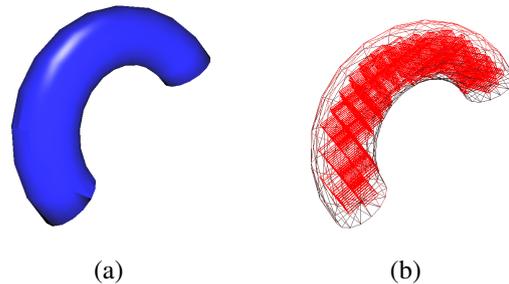


Figure 3: (a) Original Polygonal tool, (b) Identification of internal boxes via flood fill.

A grid of cells is constructed to encompass the whole object. Then a flood fill algorithm can be used to determine the cells that are inside the virtual tool. This method starts by choosing a cell known to be inside the tool object. Subsequently, it iteratively checks the 26 surrounding boxes until the boundary ones are reached. The approach results in all the interior boxes being labelled as interior. Figure 3 shows the steps for voxelising the internal volume of an arbitrary polygonal tool. The scale of the tool may also be easily adjusted to satisfy the specific requirements of a given application.

During the running of the program the polygonal tool interacts with the object derived from the volume data. To be able to effectively modify the data whilst drilling the volume, data points within the tool's bounding box are tested to determine if they are inside the tool's volume. Firstly, each data point must be checked with the three dimensional grid of cells to detect if the point is

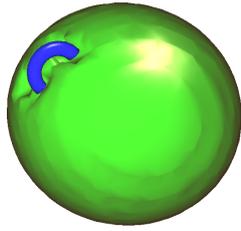


Figure 4: Polygonal tool and object interaction.

either in a boundary or interior cell. If a data point is located in a boundary cell, then it will be further checked against the tool's surface triangles located in the cell. After these steps, the values of all the data points inside the tool will be modified. After the data has been changed, the bounding box volume around the modified data points can be utilised to perform a local Marching Cubes algorithm to generate a new surface from the modified volume data.

The efficiency of the method discussed above largely depends on the size of the tool. The larger the tool used to interact with the data, the more voxels that need to be updated and recalculated by the Marching Cubes approach. This limits the use of the complicated tool implementation. Typically the haptic stylus moves slowly during drilling, especially when the tool interacts with rigid objects such as bones. The volume of data that must be changed between the adjacent graphic frames may differ by only a small amount, or indeed maybe exactly the same when the drilling tool does not move across a small voxel.

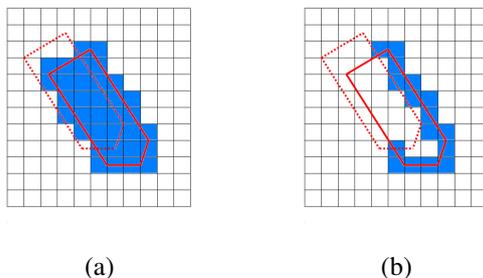


Figure 5: The red outlines represent the tools between two adjacent graphic frames when drilling. The dotted outline represents the drill tool at the previous frame, whilst the solid outline represents the drill at the current frame. The blue boxes represent areas that need to be calculated by the Marching Cubes algorithm. (a) represents the full update, whilst (b) illustrates our approach.

In this situation, it is not necessary to update the whole bounding box in each graphic frame because of the largely overlapping area. Alternatively, the update step can only consider the new area compared to the data area in the previous frame, as shown in Figure 5(b), which avoids calculating the overlapping voxels twice

in two frames. By using this method, the computation of the tool-object interaction is dramatically improved even when dealing with large polygonal tools. First of all, the modified data is detected for later use. Then the voxels containing the modified data are chosen to regenerate the new surface, as shown in Figure 5(b).

5 HAPTIC RENDERING

5.1 Force-Map Algorithm

The haptic rendering method described by Eriksson et al. [Eri05] suffers from force discontinuities when the tool moves between the encoded cubes. Sample points in this work are tested for contact with the volume data. Given a sample point position, a vector calculated from the occupancy force-map can be output. By using this method, the force feedback is stable and smooth even though it has a similar force cube encoding system. The force vectors stored in the data are calculated based on the local surface, which also benefits from the advantages of the surface based haptic rendering approach. The synchronisation of updating the graphic and haptic loops enhances the fidelity of the virtual visual-haptic system when applied to real applications. The following steps outline the Force-Map haptic rendering method adopted for a surface representation of dynamically changing voxel data.

Initially all the normals of the triangles contained in each octree leaf node (voxel) are averaged to result in a single force vector representing the data in the voxel. The larger the voxel is, the more volume data points lie within it. Additionally, only the data inside the voxel is assigned to a force vector while others are set to none. After this initialisation step, all the data near to the surface is set to a force vector which approximately equals the closest surface normal.

When the surface is updated in the haptics thread the data points that are found to lie inside the new voxel are set to a force value based on the triangle's face normal. If there is more than one triangle in the voxel, the averaged face normal will be used. Some force values in the old surface might also need to be updated since the triangles forming the surface in the voxel may have changed.

The force vectors stored in the data must be combined appropriately before being returned to the haptic device. When the virtual drilling tool moves into the volume data, a haptic test point checks the surrounding eight data values in the three dimensional space. These eight data values are referred to as the force cube in this work. The corners of the force cube contain the force vectors stored in the data. Tri-linear interpolation is employed here to enable an interpolated force vector to be calculated for any position inside the force cube. Another advantage of using the tri-linear interpolation method is that the haptic test point can be

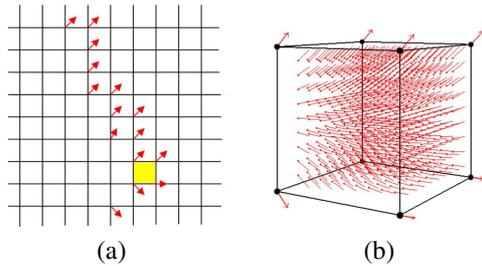


Figure 6: Force-Map haptic rendering, the red arrows represent the force vector. (a) The yellow square indicates one force cube displayed in two dimensions. (b) The same single force cube in three dimensions.

smoothly moved from one force cube to another without any force discontinuities occurring between them.

5.2 Level-Box Method

In our previous work [7], two different haptic rendering methods are employed depending on the user interaction with the volume data. When touching, a surface based method is employed. The Force-Map method is only used for the drilling. The system needs to switch between two totally different haptic rendering methods which can cause problems with regard to the consistency of the forces. Previously the Force-Map method only set force vectors close to the surface, preventing it from being employed when the user is touching the surface. If the user quickly pushes the tool toward the volume object, it will pop through the Force-Map.

In order to overcome this problem and enable the system to use one haptic rendering method, this paper introduces a new Level-Box method as an enhancement to the Force-Map approach. The area inside the volume will be partitioned into different layers. The data points in each layer will be assigned a force vector. The deeper the layer is, the larger the force vector will be set to the data in that layer. Firstly, the whole volume is partitioned into small boxes which are called Level-Boxes in this paper. The size of each Level-Box matches the size of the Octree leaf used to construct it.

5.2.1 Level-Box Construction

The Level-Boxes outside the volume object are labelled as level -1, as shown by the empty boxes in Figure 7. The boxes with the surface triangles are then labelled as level 0, as shown by the yellow boxes. After that, the neighbouring boxes of level 0 are set to level 1 (represented by green boxes). This step is repeated a number of times, until the level box reaches the centre of the volume and every box has been assigned to a level.

The next step is to assign a force vector to each data point. Basically, the data in the high level boxes will be set to a larger force vector. Figure 8 (a) shows one corner of the whole volume. Figure 8(b) shows the data position which is also the Force-Map corner position in 2D. The data in the higher level is set to a force vector

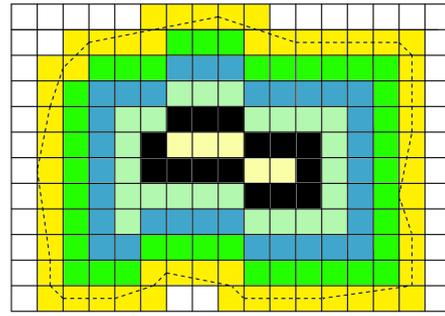


Figure 7: Level-Box construction.

with the direction of the average of the neighbouring lower level boxes. As shown in Figure 8(c), the yellow box (level 1) has one data point inside. The force vector direction will be decided by the neighbouring yellow boxes but with larger scale. Then the data in level 2 is decided by level 1. Following this logic, the centre data has the greatest force vector.

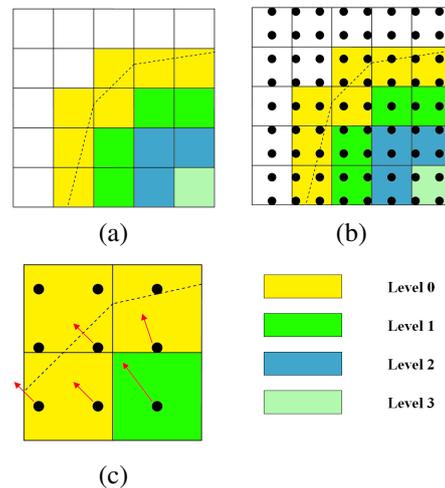


Figure 8: . Level-Box construction. (a) A small area of the level boxes, (b) The data position in the level boxes, the black points represent the data, (c) The force in the high level green box is decided by the force in low level yellow boxes.

When the Level-Boxes are constructed, the tool is able to gain the correct force feedback. The deeper it goes into the volume object, the larger the force will be which is sent to the haptic device. The Force-Map makes sure that the force is continuous and smooth.

5.2.2 Level-Box Updating

In the Level-Box construction step, the system also sets up a link between adjacent lower and higher level boxes. The force vectors in the higher level boxes are decided by the lower level ones, thus any changes in the lower box will affect the Force-Map in the neighbouring higher level box. In this circumstance, if the drilling tools modify the surface level boxes, the interior high level boxes get updated correspondingly. This

link helps the low level boxes to quickly find the high level box related to it.

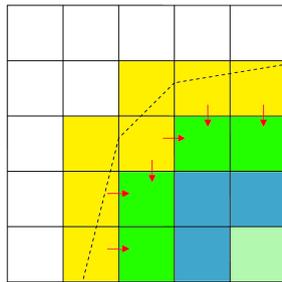


Figure 9: Level-Box link between high level and low level boxes. The arrows represent the links between a level 0 box and its neighbouring level 1 boxes.

When the tool moves towards the volume object during drilling, the surface will be recalculated based on the position of the sphere. If there is no surface in the level boxes anymore, they are changed to level -1, while the surface boxes are set to level 0. By using the link, the neighbouring ones will reduce the level because it is closer to the surface. Since the level numbers are updated, the scalar of the force vectors inside is also changed based on which level they are located in. The user is able to detect the difference of the surface after the drilling.

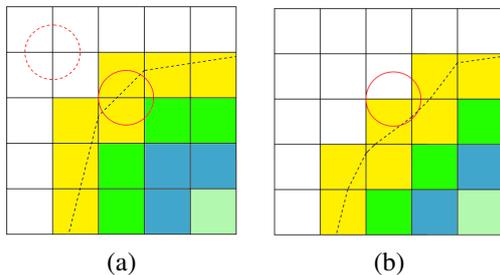


Figure 10: Level-Box updating. (a) The tool drills into the volume object. The dotted sphere represents the previous position of the drilling tool, while the solid one represents the current position, (b) The surface is updated and the level 0 boxes are changed. Consequently, the high level boxes are affected.

5.3 Multi-point Haptic Rendering

For any real application, drilling with a single point does not lead to a realistic result. An approach involving multiple test points approximating the drilling tool is usually preferred. In this work, a number of haptic points are distributed approximately around the surface of the drilling tool. At each time step, each haptic point is tested in the constructed Force-Map to calculate the contribution to the overall haptic force.

5.4 Arbitrary Tool Haptic Rendering

Pflesser et al. [13] proposed a haptic system for virtual temporal bone surgery which uses a modified version

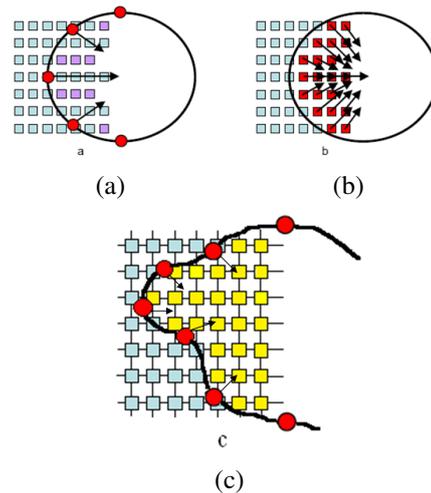


Figure 11: Arbitrary tool haptic rendering. (a) Peter-sik's et al. [13], (b) Morris et al. [1], (c) Arbitrary tool for changing data. Red points represent the haptic test points. Yellow points represent the data removed by the of drilling tool.

of the Voxmap-Pointsshell algorithm [9]. Their approaches sample the surface of the drilling instrument and then generate appropriate forces at each sampled point. A number of samples are distributed around the drill and a ray-tracing approach is then employed to calculate the force vectors towards the tool centre, which can subsequently be combined to generate the overall force returned to the haptic feedback device. The ray tracing algorithm has the potential to miss voxel data located between two rays due to an insufficient sampling as Figure 11(a) shows. Morris et al. [1] also present a method which calculates the force by counting the data points inside the tool. The force direction points to the centre of the drilling tool. Unfortunately, the haptic rendering method only allows sphere drilling (Figure 11(b)). In this work, the multiple points are located on the surface of the tool to calculate the force in the Force-Map respectively. All the force vectors inside the drilling tool are set to none and when the tool touches and drills the volume the next time, the user can detect the previously modified area.

5.5 Multi-Layer Rendering

In many applications, the properties of the simulated materials differ depending on the location being drilled. This is particularly the case in medical and dental applications where the material properties of each voxel must be considered. For example drilling through soft tissue should be very different to drilling through rigid bone. We demonstrate that the Force-Map haptic rendering method can be extended to use Multi-Layer volume data so that the trainee can feel underlying structures and material properties, such as teeth and bones. In detail, the Force-Map method can easily incorporate

this issue by simply setting a scaled force vector where the scaling factor is related to the neighbouring voxel data.

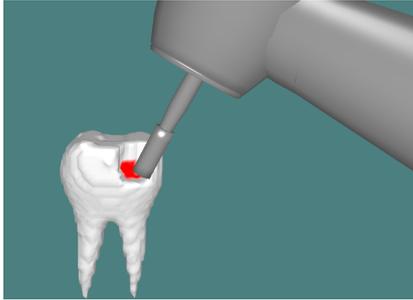


Figure 12: Multi-Layer haptic rendering.

5.6 Tangent Force Rendering

In order to enhance the force fidelity, this work also implements the tangential force on tools which is an important property of the drilling application by using the Force-Map haptic rendering algorithm. The direction of the tangent force is opposite to the tools rotation direction on the surface of the volume object.

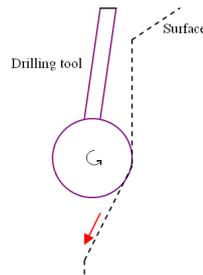


Figure 13: Tangent force implementation.

The tangent force also depends on the drilling speed of the tool and the properties of the drilling material. This haptic system allows users to choose a range of the haptic drilling speed from 200000 R/min to 400000 R/min. A faster speed will result in a greater tangent drilling force in the tangent direction of the tool-surface-contact points. Different material properties also affect the tangent force. This work allows multi-layer applications; the tangent force differs when the drilling tool moves through different materials.

6 RESULTS

Figure 14 illustrates a procedurally generated sphere along side a surface representation of a human pelvis. The surface was extracted from 87 CT slices obtained at the Norfolk and Norwich University Hospital, UK.

The work has been tested on a Two Quad Core 2.26 GHz processor PC with a NVIDIA Quadro FX580 graphics card. To provide haptic feedback a PHANToM Omni device, produced by SensAble Technologies has been employed. By using the system,

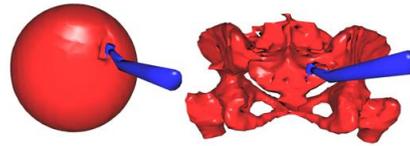


Figure 14: The left sphere-like object is created procedurally whilst the right hand image was extracted from 87 CT image slices. Each slice contains 256 X 256 pixels.

a user can drill into rigid objects using arbitrary types of tools constructed from polygons.

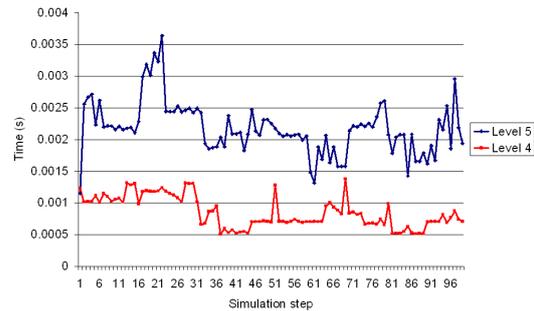


Figure 15: A graph presenting the time taken to update the surface during drilling with a polygonal tool. The blue line shows the result which uses octree level 5. The red line shows the result which uses octree level 4.

The volume of the tooth has been calculated from a data set. This data has been sampled to create a triangular surface mesh. Figure 15 shows the time required to perform the surface modification and Force-Map updates during rendering, which allows users to efficiently obtain visual cues. The Force-Map can be sampled at a higher rate in the haptic feedback loop to obtain stable force feedback. In Figure 15 the blue line shows that if the octree depth is five, the display has higher resolution but this increases the update time.

7 CONCLUSION

In this paper a Level-Box method is introduced for assisting a Force-Map haptic rendering algorithm to achieve real-time drilling of volumetric objects. In order to gain more realistic force feedback for drilling applications, arbitrary tool model selection has been implemented in this work, for tools based on implicit equations.

This paper addresses some of those challenges, specifically in the context of simulating stable and smooth force feedback. To further ensure that the fidelity of the simulator is at an acceptable level, the future work will involve the integration of drilling sound and drilling dust simulation.

A video demonstrating the program can be downloaded from the following link. <http://www.urbanmodellinggroup.co.uk/drilling.wmv>.

REFERENCES

- [1] E. Ruffaldi E, D. Morris D, T. Edmunds T, F. Barbagli, D. K. Pai, S. Superiore, and S. Anna. Standardized evaluation of haptic rendering systems. *Proceedings of the 14th IEEE Haptics Symposium*, 2006.
- [2] M. Eriksson, H. Flemmer, and J. Wikander. A haptic and virtual reality skull bone surgery simulator. *Proceedings of World Haptics*, March 2005.
- [3] J. D. Foley, A. V. Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principle and practice*. Addison-Wesley, 1996.
- [4] C. Ho, C. Basdogan, and M.A. Srinivasan. Haptic rendering: Point and ray based interactions. In *Proc. of the Second PHANTOM Users Group Workshop*, 1997.
- [5] H. Iwata and H. Noma. Volume haptization. In *Proc. of IEEE Symp. on Research Frontiers in Virtual Reality*, pages 16–23, 1993.
- [6] Y. Liu and S.D. Laycock. The force-map haptic rendering algorithm for drilling into volume data. In *WSCG 09*, pages P17–20, 2009.
- [7] Y. Liu and S.D. Laycock. A haptic system for drilling into volume data with polygonal tools. In *TPCG 09*, pages 10–16, 2009.
- [8] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM SIGGRAPH*, pages 163–169, 1987.
- [9] William A. Mcneely, Kevin D. Puterbaugh, and James J. Troy. Six degree-of-freedom haptic rendering using voxel sampling. In *ACM SIGGRAPH*, pages 401–408, 1999.
- [10] William A. Mcneely, Kevin D. Puterbaugh, and James J. Troy. Six degree-of-freedom haptic rendering improvements. In *Haptics-e,3*, 2006.
- [11] M.Eriksson, M.Dixon, and J.Wikander. A haptic vr milling surgery simulator using high resolution ct data. *Proceedings of Medicine Meets Virtual Reality*, 14:138–144, 2006.
- [12] Dan Morris, Christopher Sewell, Nikolas Blevins, Federico Barbagli, and Kenneth Salisbury. A collaborative virtual environment for the simulation of temporal bone surgery. In *In MICCAI*, pages 319–327, 2004.
- [13] A. Petersik, B. Pflesser, U.Tiede, and K. H. Höhne. Haptic rendering of volumetric anatomic models at sub-voxel resolution. In *10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, page 66, 2002.
- [14] H. T. Yau, L. S. Tsou, and M. J. Tsai. Octreebased virtual dental training system with a haptic device. *Computer-Aided Design & Applications*, 3:415–424, 2006.
- [15] C. Zilles and J. Salisbury. A constraint based godobject method for haptic display. *IEEE Conference on Intelligent Robots and Systems*, pages 146–151, 1995.

Surface Curvature Effects on Reflectance from Translucent Materials

Konstantin Kolchin

NVIDIA Corporation, Russian branch
Arbat 10, Moscow 119002, Russia
kkolchin@nvidia.com

ABSTRACT

Most of the physically based techniques for rendering translucent objects use the diffusion theory of light scattering in turbid media. The widely used dipole diffusion model [JMLH01] applies the diffusion-theory formula derived for the planar surface to objects of arbitrary shapes. The purpose of this communication paper is to present the very first results of our investigation of how surface curvature affects the diffuse reflectance from translucent materials.

1 INTRODUCTION

Translucent materials, such as human skin, marble, wax, fruits, more scatter light than absorb it. Therefore, when a photon enters such a material, it undergoes many scattering events under the surface before it leaves the material. Such a light behavior is well described by the Bidirectional Surface Scattering Distribution Function (BSSRDF) [NRH⁺77]. Based on the light diffusion theory, Jensen et al. [JMLH01] suggested the dipole diffusion model for BSSRDF. This model applies an expression for reflectance from a turbid half-space to arbitrarily shaped objects. The multipole [DJ05, DJ06] and quadpole [DJ08] models have been suggested to describe more complicated geometries - a multilayered slab (or half-space) and a right-angle corner, respectively. Jensen et al. [DJ08] showed that a big variety of shapes can be rendered by combining photon tracing and a scheme for interpolating between dipole and quadpole and between quadpole and multipole models wherever appropriate. However, they do not focus on how the BSSDRF itself changes as a flat surface is replaced with a curved one. It is difficult to devise how their interpolation scheme can be used with approaches that do not use photon tracing - for example, the curvature-based method [Kol07]. Our goal is to investigate how inclusion of curvature may change the diffusion BSSRDF model. A BSSRDF model that includes curvature effects could be easily incorporated into many existing approaches for rendering translucent materials. We present here preliminary results of our study.

2 DIFFUSION EQUATION

Under the assumption that light scattering in a turbid medium dominates absorption, light transport in it is

well described with the diffusion theory [Far92]. The fluence rate $\Psi(\mathbf{r})$ obeys the modified Helmholtz equation [Far92]

$$\Delta\Psi - \sigma_{tr}^2\Psi = -D^{-1}\delta(\mathbf{r} - \mathbf{r}_0) \quad (1)$$

where $\sigma_{tr} = \sqrt{3\sigma_a(\sigma_s' + \sigma_a)}$ is the effective transport coefficient, σ_s' is the reduced scattering coefficient, σ_a is the absorption coefficient, $D = \frac{1}{3(\sigma_s' + \sigma_a)}$ is the diffusion coefficient. We refer the reader to [JMLH01, Far92] for explanation of the physical meaning of the quantities. In the above equation, we assume that there is a single source in the medium, and it is located at a point \mathbf{r}_0 .

Let us first consider the case of translucent material occupying the half-space $z > 0$. The point source is at $\mathbf{r}_0 = (0, 0, z_0)$. Farrell et al. [Far92] showed that quite an accurate solution can be obtained by using the boundary condition $\Psi|_{z=-z_b} = 0$ and putting the image source at the point $\mathbf{r}_0 = (0, 0, -z_0 - 2z_b)$, where $z_b = 2AD$, and A is calculated as described in [JMLH01, Far92]. The resulting fluence is

$$R(\rho, z_0) = \frac{1}{4\pi D} \left[\frac{e^{-\sigma_{tr}r_1}}{r_1} + \frac{e^{-\sigma_{tr}r_2}}{r_2} \right],$$

where r_1 and r_2 are the distances to the source and image source, respectively; that is,

$$r_1 = [(z - z_0)^2 + \rho^2]^{1/2} \quad (2)$$

$$r_2 = [(z + z_0 + 2z_b)^2 + \rho^2]^{1/2} \quad (3)$$

The reflectance is calculated from the fluence using the formula

$$R = -D\nabla\Psi \quad (4)$$

where the gradient is evaluated at the interface. In the planar case, this gives

$$R(\rho, z_0) = \frac{1}{4\pi} \left[z_0 \left(\sigma_{tr} + \frac{1}{r_1} \right) \frac{e^{-\sigma_{tr}r_1}}{r_1^2} + (z_0 + 2z_b) \left(\sigma_{tr} + \frac{1}{r_2} \right) \frac{e^{-\sigma_{tr}r_2}}{r_2^2} \right] \quad (5)$$

where r_1 and r_2 are calculated for $z = 0$.

The dipole diffusion model [JMLH01] applies the above formula to an arbitrary shaped air-material interface by calculating r_1 and r_2 as the distance from

a point being shaded to the source and image source, respectively.

3 EXACT SOLUTION FOR A SPHERE

Suppose the turbid medium is confined within a sphere having the radius R_0 and the center at $z = R_0$. In addition to Cartesian coordinates, we will also use the polar system of coordinates with r counted from the sphere center and θ counted from the z axis. We assume that R_0 is much bigger than the mean free path for photons scattered in the medium, we can use the same boundary condition as in the planar case - namely, the fluence rate vanishes at a distance of z_b from the sphere surface. In other words, Ψ is zero at a sphere of the radius $R = R_0 + z_b$. We will solve eq. (1) with the boundary condition $\Psi|_{r=R} = 0$ following the method described in [Mat71]. The solution of the modified Helmholtz equation 1 with the zero boundary condition on the sphere $r = R$ can be written as

$$\Psi(r, \theta) = \begin{cases} \sum_{m=0}^{\infty} A_m \frac{I_{m+1/2}(\sigma_{tr} r)}{\sqrt{r}} P_m(\cos \theta), & r < r' \\ \sum_{m=0}^{\infty} B_m \frac{1}{\sqrt{r}} [I_{m+1/2}(\sigma_{tr} r) \times \\ \times K_{m+1/2}(\sigma_{tr} R) - K_{m+1/2}(\sigma_{tr} r) \times \\ \times I(\sigma_{tr} R)] P_m(\cos \theta), & r > r' \end{cases} \quad (6)$$

where r' is the distance of the point source from the sphere center; that is, we suppose that \mathbf{r}_0 has the polar coordinates $r = r'$ and $\theta = 0$. The functions $I_\nu(r)$ and $K_\nu(r)$ are the modified Bessel functions [MA70]. The constants A_m and B_m are determined by stitching the solutions 6 at the sphere $r = r'$. The function Ψ is continuous, but its derivative is not. In a manner similar to that used in [Mat71], we integrate eq. 1 over an infinitesimally thin region confined by parts of spherical surfaces with radiuses $r = r' + \varepsilon$ and $r = r' - \varepsilon$ and containing the point \mathbf{r}_0 . We utilize the Gauss theorem and get

$$\left(\frac{\partial \Psi}{\partial r} \Big|_{r'+\varepsilon} - \frac{\partial \Psi}{\partial r} \Big|_{r'-\varepsilon} \right) = \frac{1}{r'^2} \delta(\Omega) \quad (7)$$

where Ω is the solid angle variable. The delta function $\delta(\Omega)$ can be decomposed in terms of the Legendre polynomials as [Mat71]

$$\delta(\Omega) = \sum_{m=0}^{\infty} \frac{(2m+1)}{4\pi} P_m(\cos \theta) \quad (8)$$

Substituting eq. (8) into eq. (7) and calculating the derivatives from eq. (6), we arrive at an equation for A_m and B_m . One more equation for them is obtained by requiring continuity of Ψ at $r = r'$. Solving the resulting system of two equations, we get

$$\Psi(r, \theta) = \frac{1}{4\pi D} \left[\sum_{m=0}^{\infty} \frac{(2m+1)}{\sqrt{r r'}} I_{m+1/2}(\sigma_{tr} r') \times \right. \\ \left. \times I_{m+1/2}(\sigma_{tr} r) \frac{K_{m+1/2}(\sigma_{tr} R)}{I_{m+1/2}(\sigma_{tr} R)} P_m(\cos \theta) - \frac{e^{-\sigma_{tr} \tilde{r}}}{\tilde{r}} \right]$$

where

$$\tilde{r} = [r^2 + r'^2 - 2rr' \cos \theta]^{1/2},$$

and we used equality 10.2.35 from [MA70].

To find the reflectance, we choose $r' = R_0 - z_0$, apply eq. 4 and set $r = R_0$ and get

$$R(r, \theta) = \frac{1}{4\pi} \sigma_{tr} \left\{ \sum_{m=0}^{\infty} \frac{(2m+1)}{\sqrt{R_0 r'}} I_{m+1/2}(\sigma_{tr} r') \times \right. \\ \left. \times I'_{m+1/2}(\sigma_{tr} R_0) \frac{K_{m+1/2}(\sigma_{tr} R)}{I_{m+1/2}(\sigma_{tr} R)} P_m(\cos \theta) + \right. \\ \left. + [z_0 \cos \theta - R_0(\cos \theta - 1)] \left(\sigma_{tr} + \frac{1}{r_1} \right) \frac{e^{-\sigma_{tr} r_1}}{r_1^2} \right\}$$

4 RESULTS

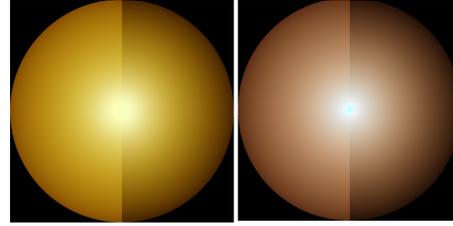


Figure 1: A spherical potato (left) and a marble sphere (right) illuminated with a stencil beam, which enters at the image center, normally to the image plane. Each of the spheres is rendered using the exact solution proposed (left part of a sphere) and the dipole diffusion model (right part of a sphere).

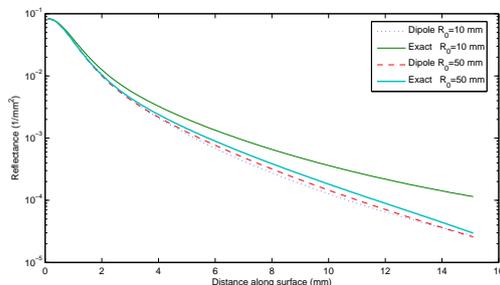
We calculated the reflectance from translucent spheres of various radiuses. The incident light is a pencil beam entering a sphere at $x = 0, y = 0$. Ideally, we should consider a line of sources situated along the z axis. But it was shown in [Far92] that they all can be replaced with a single source located $z = 1/(\sigma'_s + \sigma_a)$. The plot below shows how the reflectance depends on the distance from the point of light entrance measured along the surface (that is, the length of a geodesic connecting the entrance point and the point of interest). The calculations were done for the scattering coefficient $\sigma'_s = 1 \text{ mm}^{-1}$ and absorption coefficient $\sigma_a = 0.01 \text{ mm}^{-1}$ (note that in [Far92], the same quantities are designated as μ'_s and μ_a , respectively). These values of the scattering and absorption coefficients are typical for human tissue (see [JMLH01]). It can be seen that in this

case, the difference between the exactly computed reflectance and that found by the dipole diffusion model becomes noticeable only when the radius approaches 1 cm.

Figure 1 above shows visualization of light reflection from spheres having a radius of 1 cm in two cases - a potato, on the left, and marble, on the right. As for the plot given below, we assume that a sphere is lit up by a stencil beam entering the sphere at the center of the image. The left part of each of the image corresponds to the exact calculation we describe above. The right part is computed using the diffuse dipole approximation. We used the measured values σ'_s and σ_a reported in [JMLH01]. Because the amount of reflected light decays with distance from the entrance point very rapidly, we applied the tone mapping operator to a calculated HDR image. We chose the logarithmic mapping operator [DMAC03], as it is simple and robust, and a source code for its implementation is available on the web.

As we could anticipate in advance, the diffuse dipole model underestimate the reflectance. However, our investigation shows that this underestimation is small when curvature radiuses are of the scale of several centimeters and more for such materials as marble, potato, human tissue.

The program for computing the solution given by the last formula of the previous section was written using CUDA [NVI], which allowed a roughly 10x speed-up as compared to a CPU implementation.



5 FUTURE WORK

The investigation presented here definitely lacks comparison of analytical results with Monte-Carlo simulations. We are working on this and plan to report them elsewhere when the work is complete. Also, we would like to consider the case of arbitrarily curved surfaces. It would be interesting to try to build a phenomenological model for reflectance from a translucent material with an arbitrary surface. It can be sought as a function of principal curvatures at the point of light entrance. An approximate solution for slightly curved surfaces can serve as a base in attempts to construct a phenomenological model. Monte-Carlo simulations can be used for validation of such a model. A big potential of the phenomenological approach to constructing BSSRDF

models has been proven by successful development of an empirical BSSRDF model described in [DLR⁺09]. A BSSRDF model including surface curvature could be incorporated into the curvature-based method [Kol07]. It could be used for investigating perceptual effects, such as color shift at the terminator line [Gre04].

REFERENCES

- [DJ05] Craig Donner and Henrik Wann Jensen. Light diffusion in multi-layered translucent materials. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 1032–1039, New York, NY, USA, 2005. ACM.
- [DJ06] Craig Donner and Henrik Wann Jensen. Rapid simulation of steady-state spatially resolved reflectance and transmittance profiles of multilayered turbid materials. *J. Opt. Soc. Am. A*, 23(6):1382–1390, 2006.
- [DJ08] Craig Donner and Henrik Wann Jensen. Rendering translucent materials using photon diffusion. In *SIGGRAPH '08: ACM SIGGRAPH 2008 classes*, pages 1–9, New York, NY, USA, 2008. ACM.
- [DLR⁺09] Craig Donner, Jason Lawrence, Ravi Ramamoorthi, Toshiya Hachisuka, Henrik Wann Jensen, and Shree Nayar. An empirical bssrdf model. *ACM Trans. Graph.*, 28(3):1–10, 2009.
- [DMAC03] Frederic Drago, Karol Myszkowski, Thomas Annen, and Norishige Chiba. Adaptive logarithmic mapping for displaying high contrast scenes. In Pere Brunet and Dieter W. Fellner, editors, *Proc. of EUROGRAPHICS 2003*, volume 22 of *Computer Graphics Forum*, pages 419–426, Granada, Spain, 2003. Blackwell.
- [Far92] Patterson M.S. Wilson B. Farrell, T.J. A diffusion theory model of spatially resolved, steady-state diffuse reflectance for the noninvasive determination of tissue optical properties in vivo. *Med Phys.*, Jul-Aug, 19(4):879–88, 1992.
- [Gre04] Simon Green. *GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics*, chapter Real-Time Approximations to Subsurface Scattering, pages 264–266. Addison-Wesley Professional, March 2004.
- [JMLH01] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *SIGGRAPH '01: Proceedings of the 28th annual conference*

- on Computer graphics and interactive techniques*, pages 511–518, New York, NY, USA, 2001. ACM Press.
- [Kol07] Konstantin Kolchin. Curvature-based shading of translucent materials, such as human skin. In *GRAPHITE '07: Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and South-east Asia*, pages 239–242, New York, NY, USA, 2007. ACM.
- [MA70] Abramovitz M. and Stegun I. A. *Handbook of Mathematical Functions*. Dover Publications, 1970.
- [Mat71] Walker R. L. Mathews, J. *Mathematical Methods of Physics*. Addison Wesley Longman, 1971.
- [NRH⁺77] Fred E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. *Geometrical Considerations and Nomenclature for Reflectance*. Monograph number 160. National Bureau of Standards, 1977.
- [NVI] NVIDIA. Cuda architecture. http://www.nvidia.com/object/cuda_get.html.

Fingerprint Alignment Based on Local Feature Combined with Affine Geometric Invariant

Chuchart Pintavirooj
Department of Electronics
Faculty of Engineering
King Mongkut's Institute of
Technology Ladkrabang
Thailand 10520
kpchucha@kmitl.ac.th

Fernand S. Cohen
Department of Electrical and
Computer Engineering
Drexel University
Philadelphia, PA 19104

Woranut Iampa
Department of Electronics
Faculty of Engineering
King Mongkut's Institute of
Technology Ladkrabang
Thailand 10520
woranut_nuch@hotmail.com

ABSTRACT

In this paper we introduce a novel method of fingerprint alignment that uses the intrinsic geometric properties of minutiae-based triangles combined with the geometric invariant. The minutiae points are extracted from the fingerprint image and a Delaunay (DL) triangulation is constructed from these minutiae points resulting in a series of triangles. Corresponding minutiae points are established using local affine invariants constructed from the local minutia-based triangles. Triangles that are distorted by noise or have no counter part on the query are discarded. We rely only on "strong" matches that are reliable and present, for example, where the error metric between the local absolute invariants is below a set threshold. The correspondences of such matches are then used to estimate transformation parameters. The performance of our method is represented by computing the distance map error between a template and a query fingerprint after undoing the transformation, computed from the ridge structures of the two fingerprints. In conclusion, the proposed method can be used to find the corresponding minutiae and align any fingerprints considered into affine transformation, in the presence of noise including the partial occlusion.

Keywords

Fingerprint Alignment, Geometric Invariant, Delaunay Triangulation

1. INTRODUCTION

Biometric recognition based on distinctive anatomical and behavioral characteristics is used to recognize an individual in terms of verification and identification purposes. The biometric systems are applied to building access systems, authenticating person to access facilities, electronic access control including forensic identification. Commonly used biometric identifiers are human's face, fingerprint, iris, signature, and voice. The fingerprint is one of the most widely used biometric identifiers because of its uniqueness and immutability. The most evident structural characteristic of a fingerprint is a pattern of interleaved ridges and valleys.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Fingerprint matching can be categorized into 3 main approaches [Mal09] which are (i) correlation-based matching; (ii) minutiae-based matching; and (iii) non-minutiae feature-based matching. Correlation-based fingerprint matching is the simplest and earliest method. In the matching scheme, two fingerprint images consisting of template and query are superimposed in order to estimate the correlation between the corresponding pixels of the two images. For example, the differential matching rate based on the cyclic structure in the local area of fingerprint pattern was used to calculate the correlation value in the fingerprint verification algorithm [Hat02a]. Additionally, the correlation-based technique was applied based on coherence of the orientation field to match the fingerprints scanned from high resolution and touchless sensors [Lin09a]. Consequently, this technique was developed to the correlation filters tolerated to distorted fingerprints [Ven03a]. The performance of the correlation-based technique significantly relies on the alignment accuracy, which can be quite sensitive to transformation changes.

Minutiae-based matching is the technique for fingerprint matching to find the alignment of

minutiae feature sets between the template and the query fingerprint images. These minutiae, ridge endings and bifurcation, are characterized by their attributes such as location, orientation, and minutiae types. The local minutiae matching [Mal09] can be classified according to the local structures which are nearest neighbor-based structures [Jea05a], [Chi06a], fixed radius-based structures [Che06a], [Fen08a], minutiae triangles [Tan03a], [Che06b] and texture-based local structures [Tic03a], [Ben07a]. For the non-minutiae feature-based matching, due to the complexity of minutiae extraction in quite low-quality fingerprint images, other features of the fingerprints are extracted from the ridge pattern, for instance, local ridge orientation [Yag05a], [Liu06a] and frequency, shape, texture information [Jai00a], and sweat pores.

From these three approaches of fingerprint matching, the local minutiae-based matching is the most widely used technique. Moreover, many of minutiae-based matching construct Delaunay triangles from the minutia set and extract various features from these triangles. In [Beb99a], the ratio of side of triangle and cosine of the angle between the smallest two sides were used as the invariant features for the fingerprint indexing. The relative position and orientation of each minutia with respect to its neighbor of triangle structure were utilized in the fingerprint minutiae-matching algorithm [Par04a]. The invariant feature vectors consisting of the distance between the two minutiae and the relative radial angle between directions of each two minutiae were obtained from the minutia triangle and were then used together with the growing and fusing region of minutiae structures to match the fingerprints [Xu07a]. In addition, the minutiae type, the minimum and median angles, the length of the longest edge of the triangle including the difference between angles of two edges and orientation field at any minutiae were determined from the low-order Delaunay triangles to find the corresponding triangles for the fingerprint identification [Lia07a].

In this paper we introduce a novel method of fingerprint alignment that uses the intrinsic geometric properties of triangles constructed from minutia triplet to align minutiae points on the query and the template. Finding correspondences using invariants allows a fast non-iterative procedure for alignment, additionally, it is robust to noisy or missing data since these invariants are based on the local triangles constructed from the minutiae points. Triangles that are distorted by noise or have no counter part on the query are discarded. We rely only on “strong” matches that are reliable and present, for example, where the error metric between the local absolute invariants is below a set threshold. The

correspondences of such matches are then used to estimate transformation parameters.

This paper is organized as follows. Section 2 is related to Delaunay triangulation. Section 3 describes minutiae-based matching in the presence of affine transformation including estimation of linear transformation. Section 4 shows experimental results on the proposed algorithm. Discussion and conclusion are given in Section 5.

2. DELAUNAY TRIANGULATION OF MINUTIAE SET

Given a set \mathcal{R} of minutiae points m_1, m_2, \dots, m_N , a Voronoi diagram divides the region into sub-region about each point m_i such that all points around m_i are closer to m_i than any other minutiae point. By connecting an edge between each pair of centers in the Voronoi diagram, a Delaunay triangulation is formed. A Delaunay Triangulation possesses attractive properties that make them very suitable for fingerprint matching [Beb99a]. The following properties in particular are extremely relevant to fingerprint matching:

- (i) Affine invariance: A Delaunay Triangulation constructed from minutiae subjected to an affine transformation is still a Delaunay Triangulation whose minutiae points are obtained by subjecting the original minutiae points to that affine transformation.
- (ii) Local shape controllability: Any local deformation of minutiae is locally confined. This is very important when trying to deal with fingerprint identification in the presence of missing parts or noise.
- (iii) Uniqueness: A Delaunay Triangulation is unique. The same set of minutiae always generates the same Delaunay Triangulation.
- (iv) Robustness: A Delaunay Triangulation is immune to noise. Any disturbance to the vertex does not significantly affect the triangulation pattern.
- (v) Linear computational time complexity: This makes the algorithm suitable for on-line fingerprint matching system.

3. MINUTIAE-BASED MATCHING

An overview of fingerprint alignment using minutiae-based matching is shown in Figure 1. A Delaunay triangulation is constructed from the minutiae sets resulting in series of triangles. Then, the corresponding minutiae are established prior to determine transformation parameters and align the two fingerprint images.

In this paper, we present the fingerprint matching in the case of the fingerprint derived from a different

scanner and with a different shear on the query sample. It is considered into an affine transformation rather than a rigid transformation. Therefore, finding the matched triangle will be done based on absolute affine invariant.

From a relative invariant, the areas of the two corresponding triangles are related to each other through the determinant of linear transformation matrix as shown in Equation 1. From the equation, $A(k)$ are the area patches of sequence of triangles on the template and $A_a(k)$ are those of triangles on the query.

$$A_a(k) = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} A(k), \quad k = 1, 2, \dots, n \quad (1)$$

By taking the ratio of the consecutive area of the sequence, absolute invariants are obtained. These absolute invariants are applied to find the matched triangles between the two fingerprints according to following algorithm:

- (i) Obtain the set of triangles for the template and the query fingerprint using the Delaunay triangulation process.
- (ii) Find the list of triangles having the minutiae as a vertex and order their sequence in a counter-clockwise direction.
- (iii) Compute the absolute invariant by taking the ratio of the consecutive area of ordering triangles.
- (iv) Search for the longest string of absolute invariant that matched between the template and the query. Given one minutia in the unknown, searching the matched criterion between each pair of absolute invariant, says invariant i^{th} and j^{th} , is defined by Equation 2.

$$\% \varepsilon = \frac{|I(i) - I_a(j)|}{I(i)} \times 100 < \xi \quad (2)$$

- (v) Circular shift of the absolute invariant is performed both in each sequence of triangles corresponding to a vertex and the list of triangles having the minutiae as a vertex.

- (vi) Declare the match on the longest string (N) of triangles that yields minimum averaged error of $\frac{\sum_i \% \varepsilon_i}{N}$

Eventually, the matched triangles are obtained from the algorithm described above. The vertices of the matched triangles are considered as the corresponding minutiae. The corresponding minutiae between template and query fingerprint are used to compute the transformation matrix. The

matrix is estimated in a least square sense, from normal equation as shown in Equation 3.

$$T = (G^T G)^{-1} (G^T F) \quad (3)$$

Where

$$F = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ x_n & y_n & 1 \end{bmatrix} \quad G = \begin{bmatrix} x'_1 & y'_1 & 1 \\ x'_2 & y'_2 & 1 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ x'_n & y'_n & 1 \end{bmatrix}$$

and where F and G are the sets of corresponding minutiae between template and inquiry fingerprint.

4. EXPERIMENT AND RESULTS

Two fingerprints of thumb of the same individual shown in Figure 2 were scanned with a fingerprint scanner of L SCAN 100R. The resolution of the scanner is 500 pixels per inch (ppi). Prior to the computation of DL triangulation, the pre-process was performed including Gaussian blurring, Gabor filtering, thinning and minutia detection. Only two types of minutia were interested including ridge ending and bifurcation as represented in Figure 3. The DL triangulation of minutiae of the two fingerprints is shown in Figure 4.

As a result, according to the algorithm described in Section 3, the matched triangles of the two fingerprint images were found as shown in Figure 5. The vertices of the corresponding triangles were used as to estimate transformation parameters. The two fingerprint images before and after the alignment are shown in Figure 6a and 6b, respectively. Since the true correspondences of the scanned ridge points are not known, we elect to use the distance map that displays the distance between any point of one ridge coordinate and the closest point on the other image after undoing the transformation to the second image. The average distance map before and after the alignment are 39.8519 and 23.1273 pixels, respectively. The alignment errors on average before and after the alignment are 10.71% and 6.22% of the size of the finger, respectively.

Moreover, the results of fingerprint matching in the presence of noise are shown in Figure 7. The two fingerprint images before and after alignment are shown in Figure 8. The average distance map before and after the alignment are 2.0424 and 1.2852 pixels, respectively. The alignment errors on average before and after the alignment are 0.55 % and 0.35% of the size of the finger, respectively.

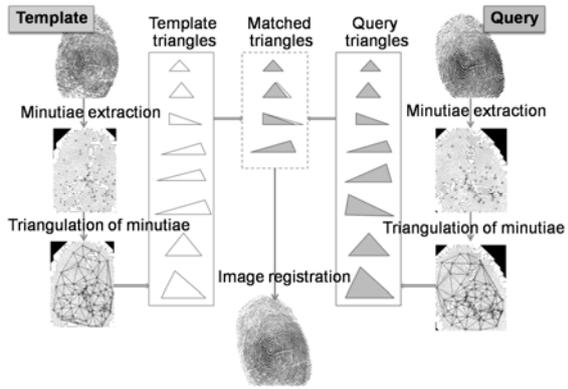


Figure 1. An overview of fingerprint alignment using minutiae-based matching.



Figure 2. Template (a) and query (b) fingerprints

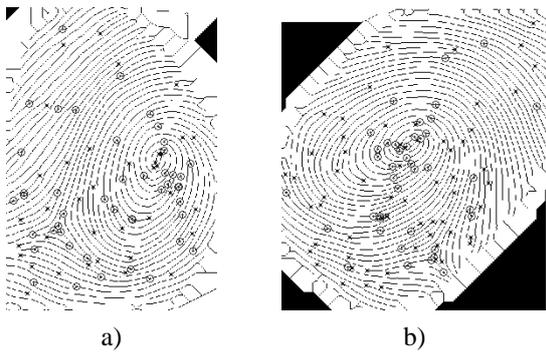


Figure 3. Minutiae position of template (a) and query (b) fingerprints. Cross signs indicate ridge endings and circle signs indicate ridge bifurcations.

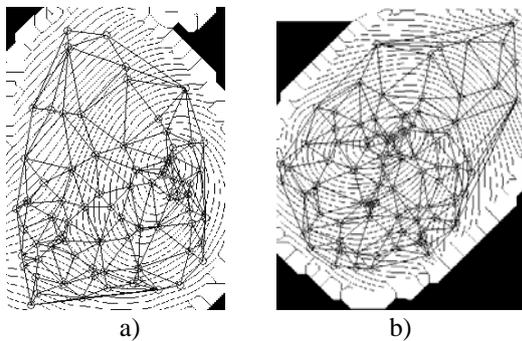


Figure 4. DL triangulation constructed from the minutiae of the template (a) and the query (b).

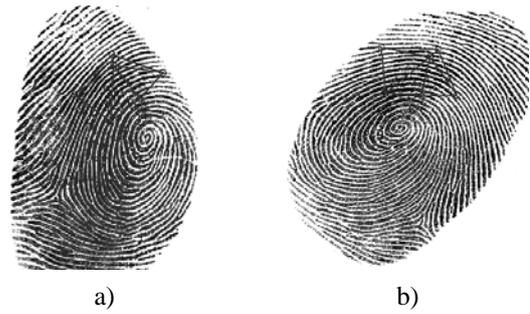


Figure 5. Matched triangles of the template (a) and the query (b) derived from the minutiae-based matching

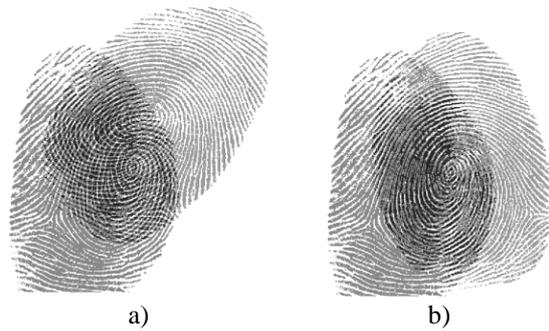


Figure 6. Two fingerprints before (a) and after (b) alignment.

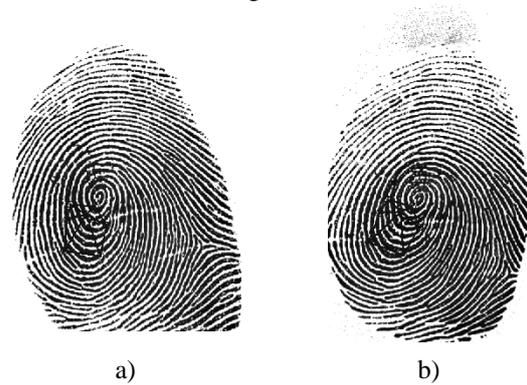


Figure 7. Matched triangles of the template (a) and the query (b) fingerprints in the presence of noise



Figure 8. Two fingerprints in the presence of noise before (a) and after (b) alignment

5. DISCUSSION AND CONCLUSION

In this paper, we introduced a geometric-based method to perform shape matching by aligning fingerprint image. For the 2D-to-2D alignment, a set of minutia points are extracted. The fiducial points were local and hence are well suited to deal with the partial alignment problem (occlusion). This is sharp contrast to other geometric invariant methods like moments and Fourier descriptors that are global in nature. To find correspondences between the minutia points on the two fingerprint images, a set of geometric invariants were determined based on the triangles constructed from sets of the minutia point triplets. After the correspondences were established, the parameters of a relevant transformation were estimated and the two images were aligned. The performance of our method is demonstrated by the ability to register the fingerprint image scanned under a host of shape transformations. In conclusion, the proposed method can be used to find the corresponding minutiae and align any fingerprints in case considered as the affine transformation, the presence of noise including the partial occlusion.

6. REFERENCES

- [Beb99a] Bebis, G., Deaconu, T., and Georgiopoulos, M. "Fingerprint identification using Delaunay triangulation," in IEEE Int. Conf. on Intelligence, pp. 452-459, 1999.
- [Ben07a] Benhammedi, F., Amirouche, M.N., Hentous, H., Bey Beghdad, K., and Aissani, M. "Fingerprint matching from minutiae texture maps," Pattern Recognition, vol. 40, pp. 189-197, 2007.
- [Che06a] Chen, X., Tian, J., and Yang, X. "A new algorithm for distorted fingerprints matching based on normalized fuzzy similarity measure," IEEE Trans. Image Process., vol. 15, no. 3, pp. 767-776, 2006.
- [Che06b] Chen, X., Tian, J., Yang, X., and Zhang, Y. "An algorithm for distorted fingerprint matching based on local triangle feature set," IEEE Trans. Inf. Forensics and Security, vol. 1, no. 2, 2006.
- [Chi06a] Chikkerur, S., Cartwright, A.N., and Govindaraju, V. "K-plet and coupled BFS: A graph based fingerprint representation and matching algorithm," Proc. Int. Conf. on Biometrics, LNCS 3832, pp. 309-315, 2006.
- [Fen08a] Feng, J. "Combining minutiae descriptors for fingerprint matching," Pattern Recognition, vol. 41, no.1, pp. 342-352, 2008.
- [Hat02a] Hatano, T., Adachi, T., Shigematsu, S., Morimura, H., Onishi, S., Okazaki, Y., and Kyuragi, H. "A fingerprint verification algorithm using the differential matching rate," in Proc. Int. Conf. on Pattern Recognition (16th), vol.3, pp. 799-802, 2002.
- [Jai00a] Jain, A.K., Prabhakar, S., Hong, L., and Pankanti, S. "Filterbank-based fingerprint matching " IEEE Trans. Image Process., vol. 9, pp. 846-859, 2000.
- [Jea05a] Jea, T.Y., and Govindaraju, V. "A minutia-based partial fingerprint recognition system," Pattern Recognition, vol. 38, no. 10, pp. 1672-1684, 2005.
- [Lia07a] Liang, X., Bishnu, A., and Asano, T. "A robust fingerprint indexing scheme using minutiae neighborhood structure and low-order Delaunay triangles," IEEE Trans. Inf. Forensics and Security, vol. 2, No. 4, pp. 721-733, 2007.
- [Lin09a] Lindoso, A., Entrena, L., Liu-Jimenez, J., and Millan, E.S. "Correlation-based fingerprint matching with orientation field alignment," in Proc. Int. Conf. on Biometrics, LNCS 4642, pp. 713-721, 2009.
- [Liu06a] Liu, L., Jiang, T., Yang, J., and Zhu, C. "Fingerprint registration by maximization of mutual information," IEEE Trans. Image Process., vol. 15, no. 5, 2006.
- [Mal09a] Maltoni, D., Maio D., Jain A.K., and Prabhakar S., Handbook of Fingerprint Recognition, 2nd ed., Springer-Verlag: London, pp. 168-233, 2009.
- [Par04a] Parziale, G., and Niel, A. "A fingerprint matching using minutia triangulation," Int. Conf. on Biometrics Authentication, LNCS 3072, pp. 241-248, 2004.
- [Tan03a] Tan, X., and Bhanu, B. "A robust two step approach for fingerprint identification," Pattern Recognition Letters, vol. 24, pp. 2127-2134, 2003.
- [Tic03a] Tico, M., and Kuosmanen, P. "Fingerprint matching using an orientation-based minutia descriptor," IEEE Trans. Pattern Anal. Mach. Intell., vol. 25, no. 8, pp. 1009-1014, 2003.
- [Ven03a] Venkataramani, K., and Vijaya Kumar, B.V.K. "Fingerprint verification using correlation filters," in Proc. Int. Conf. on Audio-and Video-Based Biometric Person Authentication, LNCS 2688, pp. 886-894, 2003.
- [Xu07a] Xu, W., Chen, X., and Feng, J. "A robust fingerprint matching approach: growing and fusing of local structures," Proc. Int. Conf. on Biometrics, LNCS 4642, pp. 134-143, 2007.
- [Yag05a] Yager, N., and Amin, A. "Coarse fingerprint registration using orientation fields," EURASIP J. Appl. Signal Process., no. 13, pp. 2043-2053, 2005.

Chrome, Gold and Silver on the Screen

Julia Wucharz
Hochschule Bremen
University of Applied Sciences
Flughafenallee 10
28199 Bremen, Germany
wucharz@gmx.de

Jörn Loviscach
Fachhochschule Bielefeld
University of Applied Sciences
Wilhelm-Bertelsmann-Str. 10
33602 Bielefeld, Germany
joern.loviscach@fh-bielefeld.de

ABSTRACT

A plausible rendering of metallic effects on a computer display is of high importance for 3D representations—as used in advertising and sales—and for pre-visualizing print designs that include special inks such as gold and/or silver. A human viewer recognizes these materials by their specific reflection properties. Hence, simulating them requires taking the illumination from the environment and the position of the viewer’s head into account. We demonstrate that this can be achieved in a Web-based application that leverages the webcam installed on the user’s computer. Thus, metallic color effects can be made available almost ubiquitously, in particular in Web shops.

Keywords

Lighting-sensitive displays, head-tracking, virtual reality, Web-based applications

1. INTRODUCTION

A car manufacturer’s Web site may show the newest model of that brand as an almost photorealistically rendered 3D object. Typically, a canned environment map is employed to simulate the look of parts made of chrome. The rendered image does not depend, however, on the viewer’s position so that the illusion breaks down when the user moves his or her head. The reproduction of metallic effects has been addressed even less in prepress applications, that is: applications that deal with simulating the look of a printed sheet of paper. Color management systems have been employed for more than a decade to ensure the optimal simulation of matte color prints on computer displays. Current color management systems do not, however, simulate metallic printing inks.

With 3D catalogs and 2D prepress as two fields of application in mind we have developed a Web-based system (see Figure 1) to address these issues in the reproduction of metallic colors. The system reads data from the user’s webcam, leveraging the fact that

webcams have become household items and mostly are already integrated in the screen bezels of notebook computers. Thus, the method cannot solely be used in software locally installed on the computer. Rather, it is also available to electronic product catalogs as used by Web shops and to online print services that want to show the effect of non-standard printing inks in advance. The contributions of this work to the state of the art comprise

- the use of the webcam to track the position of the viewer’s head—in addition to capturing the illumination—and
- the integration of all components into a Web-based application.

This paper is structured as follows: Section 2 outlines relevant related work on displays for virtual and augmented reality and on color reproduction. Section 3 describes the architecture of the prototype system, the implementation of which is covered by Section 4. Section 5 reports the results achieved; and Section 6 concludes the paper, indicating directions for future research.

2. RELATED WORK

Displays that react to their environment have been proposed at highly different levels of complexity:

Ropinski et al. [Rop04] create an environment map from the camera image to improve the look of 3D objects inserted into augmented reality displays, a

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.



Figure 1. The system takes the position of the viewer’s head (two positions shown) and the illumination into account to simulate metallic effects.

technique that was already outlined by Miller [Mil95]. Daniel [Dan05a] employs a camera with a fisheye lens to capture an environment map and illuminate 3D objects displayed on the screen through pre-computed radiance transfer. This is limited to diffuse lighting. The exposure time of the camera alternates between a long and a short setting to synthesize a higher dynamic range. Nayar et al. [Nay04a] describe a display that makes similar use of a fisheye lens but employs a large data-compressed collection of pre-rendered or pre-captured images for full re-lighting including specular highlights.

Fuchs et al. [Fuc08a] discuss options to build passive reflectance field displays, that is: displays that react to illumination—in this case illumination from behind. Using microlens arrays and LC display panels in a similar fashion, Koike and Naemura [Koi08a] demonstrate a “BRDF display,” in which the directional response to the incoming illumination can be controlled digitally. Reproducing metallic effects with such a system would, however, require a huge angular resolution to produce appropriately sharp reflections.

In a patent application [Ker09] that has been published after the submission of this paper, Kerr and

King of Apple, Inc., propose to track the user’s head—for instance through a camera—to simulate 3D effects on a 2D screen. The user may for instance “look around” the edges of window in the foreground to see what is behind; including reflections of the environment is mentioned, too. Mannerheim and Nelson [Man08] propose using a camera to track the location of the user’s head in order to adjust a binaural audio signal presented through loudspeakers.

Many goggle-free (i.e., autostereoscopic) 3D virtual reality displays employ head-tracking to project the left and right partial images onto the respective eye of the user; for an example, see [San05a]. The data thus gained can in principle be employed to render specular and mirroring reflections based on the actual position of the viewer.

Color management systems [Sto04a] are a standard amenity of current computer operating systems. They operate on the basic principle of converting colors from device-dependent spaces such as RGB and CMYK to device-independent spaces such as XYZ or CIELAB. This conversion is described through profiles for each input and output device such as camera, scanner, display, or printer. Current color management systems only support perfectly diffuse reflection

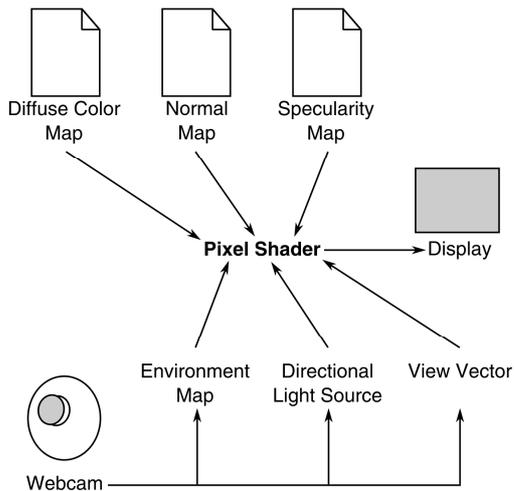


Figure 2. The system reads three two-dimensional maps and the image stream from the webcam to feed the pixel shader used for rendering.

models. Whereas color models for metallic inks have been researched into [Her03a], they have not yet found their way into off-the-shelf prepress software solutions.

3. ARCHITECTURE

This work focuses on rendering a sheet of paper or a single view of an object and trying to create as lean and hence Web-compliant a system as possible. Hence, we confine ourselves to working with two-dimensional maps instead of operating on complete three-dimensional meshes as has been done in former work on Mixed Reality. The input to the system consists of several maps, which typically are stored on the server side: the color data for diffuse reflection (a standard RGB image), a normal map (encoded as RGB image), and a specularity map (encoded as grayscale image) that defines the blend between matte and metallic behavior per pixel.

The non-metallic part of the model is rendered with the Lambertian model [Bli77]. The metallic part employs the Cook-Torrance model [Coo82] with fully editable parameters. In the software prototype, these are offered as controls on the graphical user interface. In an actual application, however, they would be set and frozen during the authoring phase and then be stored as part of the media file or in a configuration file.

To adapt the sharpness of the reflected environment to the selected sharpness of the highlights, the environment map can be sampled down by an adjustable power of two. Figure 2 shows the overall architecture.

To provide a system that works over the Web with a typical computer on the client side we elected to employ a standard webcam image instead of an image shot through a fisheye lens. The image taken by the webcam is used for three purposes:

First, a cube map of a plausible environment is build from the image. The front face of the cube is formed by the camera image as such, cropped from both the left and right side by the eighth part of its width. The remaining parts on the left and right are put into the left and right faces of the cube map and extended through repetition of the last pixel column. The bottom and the top face of the cube map are formed through repetition of the first row or the last row of the camera image, respectively. To partially hide the repetitions, the lateral faces are feathered toward black at their ends, see Figure 3.

Second, a coarse-grained version of the camera image is searched for the brightest spot. For the rendering, a light source with this color is placed accordingly. Thus, one strong specular reflection is taken into account without high dynamic range imaging and without complex rendering algorithms, see Figure 4.

Third, the user's head is found in the camera image using an existing software library (see Section 4). The center of the head is used to define the view direction for the rendering, see Figure 5. In case no webcam is available—for example, out of privacy concerns—the user can choose to steer the viewing position



Figure 3. The environment cube map is built from the camera's input through cropping, the repetition of the final rows and columns, and feathering.



Figure 4. To determine a position for a single dominant light source, the camera image (left) is sampled down to large blocks (right).

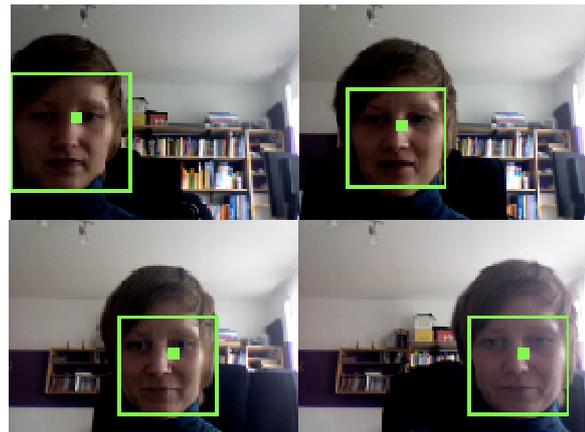


Figure 5. A point slightly above the centroid of the largest rectangle returned by the face detector controls the view direction of the lighting model.

through the mouse and apply one of several environment maps included with the software.

4. IMPLEMENTATION

The prototype has been developed in ActionScript 3 using the Adobe Flex Builder 3 development environment based on the Flex software development kit 3.4 targeting Adobe Flash Player version 10 or newer.

The face-tracking component employs the “Marilena” port [Mas09a] of the face detection in the OpenCV library. This detector [Vio01] employs a cascade of simple classifiers that use the contrast between averages over rectangular parts of the image. These averages can be computed quickly through a summed-area table. Consequently, the features being detected resemble Haar wavelets. The training data that has been generated upfront is based on a variant of the AdaBoost. In this case, during training the best features (that is: sets of rectangles) are found and the classifiers are adjusted, whereas a classical AdaBoost would only concern the latter step.

The rendering has been realized through a shader routine developed with Adobe’s Pixel Bender Toolkit 1.5 [Ado09a]. PixelBender comprises of a basic interactive development environment to build image processing routines (called “kernels”) in a programming language resembling the OpenGL Shading Language GLSL. The range of available functions corresponds to a pixel shader in standard GPU programming. The kernels thus created can be connected into dataflow graphs and can be compiled to byte-code to be loaded and executed in Flash Player 10.

As the Pixel Bender Toolkit as such offers GPU acceleration for the kernels, it is foreseeable that future

versions of Flash Player also execute kernels on the GPU instead of running them on the CPU as the current version does. Then a vital part of the acceleration offered by the graphics processor can be leveraged even in this Web-based software. The circumstance that Pixel Bender only targets pixel processing but not mesh processing fits nicely to the scope of our application.

5. RESULTS

We measured the performance of the system on an Apple MacBook computer, which runs Mac OS X 10.5.8., is equipped with an Intel Core 2 Duo processor running at 2.2 GHz and an integrated webcam. It does not contain a dedicated graphics chip but uses the Intel GMA X3100 chipset graphics instead.

At an image size of 512 x 384 pixels, the software prototype with all functions applied runs at 15 frames per second; at an image size of 615 x 461 pixels, this rate decreases to 8 frames per seconds. With all camera functions switched off, the rendering alone easily achieves 30 frames or more per second. This shows that the processing of the camera image is the step that limits the performance.

One may hope that future application frameworks grant direct access to head-tracking data and thus relieve the application from such computations. Most popular webcams already come with robust and computationally lean integrated head-tracking to add 3D items such as hats or sunglasses to the user’s face. Currently, however, there is no official way to access these head-tracking data from other software.

The .swf file that is transferred to the client computer and contains the complete code of the application possesses a size of 260 KB. The three maps (diffuse

color, normal, specular) add to this size; their byte count depends heavily on the compression used.

The pixel repetitions used to build the environment cube map (see Figure 3) may become visible in extreme situations, namely if large, flat and perfectly mirroring surfaces are viewed from head positions that are strongly off-center. In all other cases, the details of the texture and specular maps and/or the blurriness of the reflection hide these artifacts. This becomes apparent in Figure 6, which also demonstrates the use of our system with two-dimensional normal maps of three-dimensional meshes: Even though the object does not rotate, the look of polished metal is reproduced faithfully.

For speed and simplicity, the color computations are executed in RGB space and employ the automatic clamping of the RGB components. Thus, bright highlights—as they are more or less required for metallic effects—appear color-shifted toward white. For instance, the internally computed color (1.9, 1.5, 0.9) does not appear on the display screen as reddish orange but as (1.0, 1.0, 0.9), which is a slightly yellowish white. Even though this effect is only apparent to the trained eye, a color clamping that restricts the hue of the color to its original value could suppress it, at the cost of less brighter highlights.

6. CONCLUSION AND OUTLOOK

We have demonstrated a system that plausibly simulates metallic colors but remains inexpensive in terms of computer hardware and computational effort. In particular, the system leverages standard Internet technology and can thus be employed in Web shops, electronic advertisements, etc.

Future developments can target the precision of the simulation of the lighting, possibly turning the plausible result into an almost visually exact one. Doing so would require dealing with camera calibration, generating environment maps with a high dynamic range from a standard camera [Dan05a], and creating cheap but precise ancillary lenses to turn a standard webcam into a fisheye camera. The reproduction of perfectly mirrored reflections on extended flat surfaces could be improved through replacing the pixel repetition in the cube map by a synthesized texture. Strong highlights would benefit from bloom effects based on high-dynamic range computations of the colors.

An integration of 3D meshes looks straightforward from the algorithmic side. In terms of performance, however, Adobe Flash—running on the computer's CPU—may be overcharged with such a task. In future, a more general approach that requires no

browser plug-in may become possible through the advent of WebGL [Mar09].

A second avenue of development would be to focus on strengthening the connection to color management systems in their present form. A standard color management system could handle the diffuse illumination and a system similar to the prototype we have described could add gloss and mirror effects.

7. REFERENCES

- [Ado09a] Adobe. Adobe Pixel Bender technology. <http://labs.adobe.com/technologies/pixelbender/>, last accessed 2009-10-24.
- [Bli77] Blinn, J.F. Models of light reflection for computer synthesized pictures. *SIGGRAPH Computer Graphics* 11, No. 2, pp. 192–198, 1977.
- [Coo82] Cook, R.L., and Torrance, K.E. A reflectance model for computer graphics. *ACM Transactions on Graphics* 1, No. 1, pp. 7–24, 1982.
- [Dan05a] Daniel, T. Real-time video lighting. *ACM SIGGRAPH '05 Posters*, Los Angeles CA, ACM Press, p. 50, 2005.
- [Fuc08a] Fuchs, M., Raskar, R., Seidel, H.-P., and Lensch, H.P.A. Towards passive 6D reflectance field displays. *ACM Transactions on Graphics* 27, No. 3, Article No. 58, 2008.
- [Her03a] Hersch, R.D., Collaud, F., and Emmel, P. Reproducing color images with embedded metallic patterns. *Proceedings of ACM SIGGRAPH '03*, Los Angeles CA, ACM Press, pp. 427–434, 2003.
- [Ker09] Kerr, D.R., and King, N.V. Systems and methods for adjusting a display based on the user's position. United States Patent Application 20090313584, 2009.
- [Koi08a] Koike, T., and Naemura, T. BRDF display: interactive view dependent texture display using integral photography. *Proceedings of IPT/EDT '08*, Los Angeles CA, ACM Press, Article No. 6, 2008.
- [Mas09a] Masakazu, O. Marilena: port of the OpenCV face to ActionScript 3. <http://www.libspark.org/wiki/mash/Marilena>, last accessed 2009-10-24.
- [Man08] Mannerheim, P., and Nelson, P.A. Virtual sound imaging using visually adaptive loudspeakers. *Acta Acustica united with Acustica* 94, No. 16, pp. 1024–1039, 2008.
- [Mar09] Marrin, C. (editor). WebGL specification, working draft 10 December 2009. <http://www.khronos.org/webgl/>, last accessed 2010-01-02.



Figure 6. In addition to visualizing two-dimensional relief prints on paper, the system can also plausibly convey the look of metallic 3D objects as described through a 2D normal map. In this image sequence, the user's head has moved from left to right. For demonstration, a specularity map with less metallicity below the diagonal has been applied. (Stanford Bunny courtesy of <http://graphics.stanford.edu/data/3Dscanrep/>)

[Mil95] Miller, G. Volumetric hyper-reality, a computer graphics holy grail for the 21st century? Proceedings of Graphics Interface '95, Québec Québec, Canadian Information Processing Society, pp. 56–64, 1995.

[Nay04a] Nayar, S.K., Belhumeur, P.N., and Boulton, T.E. Lighting sensitive display. ACM Transactions on Graphics 23, No. 4, pp. 963–979, 2004.

[Rop04] Ropinski, T., Wachenfeld, S., and Hinrichs, K.H. Virtual reflections for augmented reality environments. Proceedings of Artificial Reality and Telexistence (ICAT04), Seoul, Korea, pp. 311–318, 2004.

[San05a] Sandin, D.J., Margolis, T., Ge, J., Girado, J., Peterka, T., and DeFanti, T. A. The Varrier autostereoscopic virtual reality display. ACM Transactions on Graphics 24, No. 3, pp. 894–903, 2005.

[Sto04a] Stone, M.C. Color in information display: principles, perception, and models. ACM SIGGRAPH '04 Course Notes, Los Angeles CA, ACM Press, Course 21, 2004.

[Vio01] Viola, P., and Jones, M. Rapid object detection using a boosted cascade of simple features. Proceedings of Computer Vision and Pattern Recognition (CVPR 2001), Kauai HI, IEEE Press, pp. I-551–I-518, 2001.

Decomposing the Contact Linear Complementarity Problem into Separate Contact Regions

Olexiy Lazarevych

lazarevych@vision.ee.ethz.ch

Gabor Szekely

szekely@vision.ee.ethz.ch

Matthias Harders

mharders@vision.ee.ethz.ch

Computer Vision Laboratory, ETH Zurich
Sternwartstrasse 7, CH-8092 Zurich, Switzerland

ABSTRACT

We present a novel approach to handling frictional contacts for deformable body simulations. Our contact model allows to separate the contact area into a set of detached contact regions. For each of them a separate mixed linear complementarity problem (MLCP) is formulated. Parallel processing of these independent contact regions may considerably improve the performance of the contact handling routine. Moreover, the proposed contact model results in sparse matrix formulation of the corresponding MLCP in the individual contact regions. For solving the MLCPs we propose an iterative method which combines the projected conjugate gradient approach and the projected Gauss-Seidel method.

Keywords

Linear complementarity problem, contact force, deformable object.

1 INTRODUCTION

Contact handling of interacting solid objects is a common research topic, for instance in computer animation or surgical simulation. Physically plausible responses to collisions and contacts potentially enrich the animation, especially if frictional effects are taken into account. Contact response methods aim at computing a set of contact forces that prevent the simulated objects from interpenetrating, while taking into account friction.

Several approaches have been proposed in the field of computer graphics and simulations to handle contacts. The majority of these can be split into two classes: penalty-based and constraint methods (note that further approaches exist, e.g. impulse methods). Penalty methods compute virtual spring forces that drive the interacting objects apart. The values of these forces are usually considered to be proportional to a geometrical measure of the interpenetration of the interacting bodies [HTK*04, KMH*04, HVS*09]. Therefore, penalty based methods not only allow interpenetrations but essentially depend on them. Despite the lack of physical plausibility caused by this simplified contact

model, they are still widely used because of the simplicity of their implementation and high computational efficiency.

In contrast, constraint methods aim at following the geometrical restrictions of non-penetration of the interacting objects based on their relative position and orientation [Bar89, DAK04, PPG04, Erl07]. The resulting system of equations can be solved by a large variety of methods among which the most preferable are fast iterative procedures. However, for complex systems which consist of many interacting objects the computation time of this approach becomes quickly prohibitive. Therefore, much effort is made to develop efficient algorithms [Bar96, GBF03, KEP05, KSJP08, OTSG09, HVS*09].

Contributions. We propose a new approach to resolving contacts for deformable objects by splitting the contact area into separate, independent regions. The deformation model together with the time-integration scheme we use allows the separate treatment of detached contact regions. Handling a number of local contacts instead of a single global contact system gives a significant gain in performance even without using parallel computation techniques. The proposed contact model results in a simple diagonal mass matrix as well as sparse constraint matrices.

In addition, we propose a novel iterative scheme for the mixed contact linear complementarity problems which combines a projected conjugate gradient method with the widely used projected Gauss-Seidel method. Although, the performance in our current implementation is not better than for the normal projected Gauss-Seidel method, our scheme demonstrated more stable convergence behavior and therefore was more reliable.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

2 RELATED WORK

Constraint methods are widely used in computer graphics as well as in computational mechanics due to their physical correctness. The theoretical basis of the underlying mechanics and related contact problems are thoroughly discussed by Stronge [Str90] and Wriggers[Wri02]. Classical works in constraint based dynamics in computer graphics are by Baraff [Bar89, BW92] and Witkin [Wit97].

Constraint based approaches for contact problems usually employ Signorini's law [WP99] of unilateral contact resulting in the formulation of the contact linear complementarity problem (LCP) [AP97]. Lagrange multipliers belong to the most widely used solution approaches for this kind of problems [WP99]. The LCP formulation in contact handling is used for obtaining contact responses between rigid bodies [Cat05, Erl07] or deformable objects [DAK04, DDKA06, OG07], as well as in cloth simulations [VMT97, VT00, HB00].

General approaches to the LCP solution can be split into two classes: direct and iterative methods [CPS92]. Although direct methods, *e.g.* Lemke's algorithm, Danzig's method, and other pivoting techniques [Cot90, CPS92, Mur88] are designed to give precise solutions, they are computationally demanding and slow. Therefore, in computer graphics applications almost exclusively iterative methods are used. Iterative methods for the LCP follow the scheme similar to the one used to solve a linear system of equations [CPS92, Mur88]. Therefore, projected versions of well-known iterative methods such as Jacobi, successive overrelaxation, and its special case – Gauss-Seidel – are used [Cat05, Erl07]. They work very well for rigid body simulations, however, applied to deformable body collisions they become computationally very expensive. Attempts to find a compromise were presented in [PPG04, DDKA06].

Many researchers are working on optimization and improvement of the performance of these basic iterative methods in different application areas. Exploiting the sparsity of the matrices involved in computations is one of the basic optimization approaches which works for almost any underlying model of simulated objects [GL89]. Other more sophisticated algorithms consider the LCP formulation tightly linked with the dynamical model. Baraff and Witkin employed implicit integration methods for large time step simulations of cloth [BW98]. Otaduy et al. [OTSG09] proposed an iterative solver that includes two nested relaxation loops (based on the constraint anticipation introduced in [Bar96]).

Using the conjugate gradient method for general LCP was proposed by researchers in the area of computational mechanics, like Renouf and Alart [RA05], and Li et al. [LNZL08]. We explore the combination of

the projected conjugate gradient approach with the projected Gauss-Seidel method.

3 DEFORMABLE CONTACT MODEL AND MLCP FORMULATION

In simulations of scenes with many interacting deformable objects, numerous pairs of objects or parts of the same object may be simultaneously in contact. The deformable nature of the simulated material provides non-instantaneous spreading of the contact forces from the contact area into the physical body. Therefore, simultaneous but spatially separated contacts may be considered independently as their effect spreads over the objects in contact during future simulation time steps. This is in contrast to rigid body simulations where all contacts have to be taken into account to correctly compute the reaction of the object. Following this reasoning we take advantage of considering spatially separated contacts between deformable objects independently. This should speed up the contact response computations in the simulations.

In our simulations deformable objects are represented as tetrahedralized meshes with mass points located in the nodes. Each object has a triangulated surface and contacts are treated between basic surface elements: point-triangle and edge-edge pairs. Point-edge and point-point contacts are treated as special cases of point-triangle contacts. For the sake of simplicity we omit edge-edge contacts and consider only point-triangle pairs in the further discussion.

Constraints Formulation

In the absence of friction the only constraint for the point-triangle collision is that contact points cannot penetrate planes of the corresponding contact triangles. Mathematically this can be described by the condition of non-negativity of the function $C(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ of the coordinates of the corresponding mass points.

$$C(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) = -((\mathbf{p}_1 - \mathbf{p}_0) \times (\mathbf{p}_2 - \mathbf{p}_0)) \cdot (\mathbf{p}_3 - \mathbf{p}_0) \quad (1)$$

The time derivative of this function gives the Jacobian matrix of the normal contact constraints.

$$\dot{C}(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) = \mathbf{J}_n \cdot \mathbf{u} \quad (2)$$

where $\mathbf{u} = [\mathbf{v}_0^T \mathbf{v}_1^T \mathbf{v}_2^T \mathbf{v}_3^T]^T$ is a generalized velocity vector of the corresponding points.

The principle of virtual work requires orthogonality of the constraint force and the constraint. Therefore, in the frictionless case for our model the constraint force is defined as

$$\mathbf{f}_n = \mathbf{J}_n^T \cdot \lambda_n \quad (3)$$

where the Lagrange multiplier λ_n is to be found.

According to Signorini's contact law [WP99] at a unilateral contact the following complementarity conditions have to be satisfied.

$$\mathbf{w}_n = \mathbf{J}_n \cdot \mathbf{u} \geq 0, \quad \lambda_n \geq 0, \quad \mathbf{w}_n \cdot \lambda_n = 0 \quad (4)$$

The conditions (4) pose a linear complementarity problem (LCP) for a frictionless unilateral contact.

In general, if N mass-points are involved in contacts with K constraints, the Jacobian of the whole system is easily assembled from the Jacobians of each individual constraint. Therefore, the global Jacobian consists of K lines of blocks \mathbf{J}_q^0 , \mathbf{J}_q^1 , \mathbf{J}_q^2 and \mathbf{J}_q^3 , where $q = 1, \dots, K$. Note, that in each line only the entries corresponding to the mass-points involved in the q -th contact are non-zero. This way the Jacobian of the contact system has the dimension $K \times 3N$.

Separation of the Contact Regions

The time integration scheme of the simulations uses the net force of the internal, global (*e.g.* gravitational), and contact forces to compute position and velocity of each simulated contact point at the next time step. Thus, a force applied to a particular mass point in the current time step will influence its neighbors only in the next time step through internal deformation.

The nature of the time-integration scheme and the discretized model of simulated objects allows us to separate two contact areas if they do not have any common simulated mass points simultaneously involved in constraint equations of both contacts. As will be shown later, this way the amount of computations becomes significantly smaller and the convergence rate for each individual contact problem increases.

The separation of the contact areas is performed by analysis of the constraint matrix \mathbf{J}_n which consists of the rows related to the normal contact constraints only. The element j_{ki} of the matrix is non-zero if and only if the i -th mass point is involved in the k -th constraint. Therefore, the area separating algorithm efficiently extracts sets of rows such that each pair of the sets does not have any non-zero elements in the same columns simultaneously. In terms of the contact graph of the current configuration which is encoded by the Jacobian, the region separation algorithm aims at finding a set of disconnected subgraphs.

Currently, a basic sequential algorithm is used to assign each contact to a contact region. Contacts corresponding to a line of the Jacobian \mathbf{J}_n are assigned to a particular region, such that any two different contact regions do not have contacts that share a simulated mass point. Thus, contacts that involve the same mass point belong to the same contact region. The outline of the contact region separation is presented in Algorithm 1. Here, $Contact[i][j]$ contains the index of the j -th point on the i -th contact, $i = 1, \dots, K$, $j = 1, \dots, 4$ and $\{Contact[i]\}$ is the set of points that belong to the

i -th contact. $Area[i]$ contains the index of the detached region to which the point i belongs. Note that more advanced, *e.g.* parallel, algorithms could be applied in this stage. Moreover, it should be mentioned that we consider contacts of deformable objects which usually are maintained over a number of successive simulation time steps, even in dynamic scenes. Thus, information about contact regions could be stored and updated on successive time steps as required.

Algorithm 1 Contact region separation

```

nextIndex ← 1
CheckedPointSet ← ∅
for i = 1 to K do
  if Area[i] not assigned then
    Area[i] ← nextIndex++
    CheckedPointSet ← {Contact[i]}
    for j = i + 1 to K do
      if Area[j] is assigned then
        continue
      endif
      if {Contact[j]} ∩ {Area[i]} ≠ ∅ then
        Area[j] = Area[i]
      endif
    endfor
  else
    for l = 1 to 4 do
      if Contact[i][l] ∉ CheckedPointSet then
        for j = i + 1 to K do
          if Area[j] > 0 then
            continue
          endif
          if {Contact[j]} ∩ {Area[i]} ≠ ∅
            then
            Area[j] = Area[i]
          endif
        endfor
        CheckedPointSet ⇒ Contact[j][l]
      endif
    endfor
  endif
endfor

```

Including Frictional Contact

Classically the frictional part of the contact force lying in the plane of the contact triangle is introduced having two components along two orthogonal vectors \mathbf{e}_1 and \mathbf{e}_2 [Bar94]. In the frame of our contact model the part of the Jacobian responsible for friction is

$$\begin{bmatrix} \mathbf{J}_{\mathbf{e}_1} \\ \mathbf{J}_{\mathbf{e}_2} \end{bmatrix} = \begin{bmatrix} -\mathbf{e}_1^T & \alpha \mathbf{e}_1^T & \beta \mathbf{e}_1^T & \gamma \mathbf{e}_1^T \\ -\mathbf{e}_2^T & \alpha \mathbf{e}_2^T & \beta \mathbf{e}_2^T & \gamma \mathbf{e}_2^T \end{bmatrix} \quad (5)$$

where (α, β, γ) are barycentric coordinates of the contact point at the time of collision.

Coulomb's friction model is often approximated by a 4-sided [Bar94] (in general, k -sided [KEP05, DDKA06]) pyramid with faces parallel to the orthogonal vectors \mathbf{e}_1 and \mathbf{e}_2 . This friction model leads to the following conditions to be satisfied at the contact.

$$\begin{aligned} \mathbf{J}_{e_i} \cdot \mathbf{u} > 0 &\Rightarrow \lambda_{e_i} = -\mu \lambda_n \\ \mathbf{J}_{e_i} \cdot \mathbf{u} < 0 &\Rightarrow \lambda_{e_i} = \mu \lambda_n \\ \mathbf{J}_{e_i} \cdot \mathbf{u} = 0 &\Rightarrow \lambda_{e_i} \in [-\mu \lambda_n; \mu \lambda_n] \end{aligned} \quad (6)$$

where $i = 1, 2$ and μ is the friction coefficient.

In addition, we also tested a friction cone model which more precisely follows Coulomb's law. We project the solution onto the friction cone domain. If the tangential component of the contact force is larger than $\mu \lambda_n$ then we scale the friction components to fit the friction cone without changing the direction of the friction force.

$$\|\lambda_{e_1} \mathbf{e}_1 + \lambda_{e_2} \mathbf{e}_2\| > \mu \lambda_n \Rightarrow \begin{cases} \lambda_{e_1} \leftarrow \frac{\lambda_{e_1} \cdot \mu \lambda_n}{\|\lambda_{e_1} \mathbf{e}_1 + \lambda_{e_2} \mathbf{e}_2\|} \\ \lambda_{e_2} \leftarrow \frac{\lambda_{e_2} \cdot \mu \lambda_n}{\|\lambda_{e_1} \mathbf{e}_1 + \lambda_{e_2} \mathbf{e}_2\|} \end{cases} \quad (7)$$

For a single point-triangle frictional contact the complementary conditions (4) together with (6) or (7) have to be satisfied. The general Jacobian of the system is built in the same way as in the frictionless case. The dimension of the matrix is $3K \times 3N$.

Dynamics Formulation

After separating the contact area into detached contact regions we formulate and solve the dynamic equations for each of the regions independently. In the following discussion we consider a part of the simulated system which corresponds to a particular contact region C . This part consists of the mass points involved in the contacts of that specific region. The simulated system obeys the following equation of motion.

$$\mathbf{M}_C \cdot \mathbf{u}_C = \mathbf{J}_C^T \cdot \lambda_C + \mathbf{f}_C \quad (8)$$

where \mathbf{M}_C is the mass matrix of the system, $\lambda_C = (\lambda_{n,j_1} \lambda_{e_1,j_1} \lambda_{e_1,j_1} \dots \lambda_{n,j_k} \lambda_{e_1,j_k} \lambda_{e_1,j_k})^T$ – the generalized vector of contact forces for the region, and $\mathbf{f}_C = (\mathbf{f}_1^T \mathbf{f}_2^T \dots \mathbf{f}_l^T)^T$ – the generalized vector of non-contact forces acting on each mass point. k and l are the number of constraints and mass points of the contact region C , respectively.

We employ the forward Euler integration scheme to relate the unknown general velocity at time $t + \Delta t$ to the known velocity at the previous time step t . For deformable object collisions we employ Newton's rule for changes of the normal component of velocity after the collision [Str90], *i.e.* $\frac{v_{reflected}}{v_{incident}} = \kappa$.

$$\mathbf{u}_C(t + \Delta t) = (1 + \kappa) \mathbf{u}_C(t) + \mathbf{M}_C^{-1} \mathbf{J}_C^T \cdot \lambda_C \Delta t + \mathbf{M}_C^{-1} \cdot \mathbf{f}_C \quad (9)$$

By pre-multiplying (9) with \mathbf{J}_C we connect the dynamics equation with the complementarity conditions (4) and (6) discussed above.

$$\mathbf{w}_C = \mathbf{J}_C \cdot \mathbf{u}_C(t + \Delta t) = \mathbf{A} \cdot \lambda_C + \mathbf{b} \quad (10)$$

where

$$\mathbf{A} = \mathbf{J}_C \mathbf{M}_C^{-1} \mathbf{J}_C^T \quad (11)$$

$$\mathbf{b} = (1 + \kappa) \mathbf{J} \cdot \mathbf{u}_C(t) + \mathbf{J}_C \cdot \mathbf{M}_C^{-1} \cdot \mathbf{f}_C \quad (12)$$

Note, that we included the factor Δt into λ_C and therefore λ_C is no longer the force but the impulse vector.

The above equations (11) and (12) together with general complementarity condition (6) or (7) constitute the MLCP that has to be solved for the values of the contact force components λ_C .

Unlike the usual formulation of the dynamics equations we explicitly consider only mass-points involved in each contact. Therefore, the generalized velocity vector does not include the angular velocity of the contact triangle and the mass matrix does not include 3×3 blocks corresponding to inertia tensors. This formulation provides a strictly diagonal form of the matrix \mathbf{M} allowing optimized matrix multiplications.

Each line of the constraint matrix \mathbf{J}_C consists of four 3×3 blocks. However, if the matrix \mathbf{J}_C is stored in a suitable reduced format [GL89, Cat05], the calculations of $\mathbf{J}_C \mathbf{M}_C^{-1} \mathbf{J}_C^T$ can be done very efficiently in linear time.

4 ITERATIVE METHODS FOR LCP

Here, we leave aside the underlying dynamics and consider iterative methods for solution of the LCP(\mathbf{A}, \mathbf{b})

$$\begin{aligned} \mathbf{A} \cdot \lambda - \mathbf{b} &> 0 \\ \lambda &> 0 \\ (\mathbf{A} \cdot \lambda - \mathbf{b}) \cdot \lambda &= 0 \end{aligned} \quad (13)$$

Projected Gauss-Seidel Iterative Method

A general splitting scheme for iterative LCP solving is described in [CPS92]. By splitting the matrix \mathbf{A} of the LCP(\mathbf{A}, \mathbf{b}) in different ways, iterative schemes similar to those for systems of linear equations are obtained. The projected Gauss-Seidel method is derived by splitting $\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}$, where \mathbf{L} , \mathbf{D} and \mathbf{U} are the strictly lower, diagonal, and strictly upper matrix components of \mathbf{A} .

According to the iterative scheme for solving the LCP(\mathbf{A}, \mathbf{b}) [CPS92] each iteration cycle consists of two steps. In the first a new approximation of the solution is found

$$\lambda_{k+\frac{1}{2}} = (\mathbf{L} + \mathbf{D})^{-1} \cdot (\mathbf{b} - \mathbf{U} \cdot \lambda_k) \quad (14)$$

In the second step this approximation is projected onto the set of feasible solutions.

$$\lambda_{k+1} = \max \left\{ 0, \lambda_{k+\frac{1}{2}} \right\} \quad (15)$$

Although, the projected Gauss-Seidel method demonstrates only first-order convergence, its computational efficiency and implementation simplicity have made it a common choice for many constraint based collision response methods in computer animation, *e.g.* [Cat05, DDKA06, Erl07, OTSG09].

Projected Conjugate Gradient Method

The conjugate gradient method [She94] can also be adapted for solving the LCP(\mathbf{A}, \mathbf{b}) [RA05]. The original conjugate gradient method has been widely used for optimization problems as well as for the solution of systems of linear and non-linear equations. For a linear system the method converges after at most n iterations, where n is the order of the system. If the method is applied to a non-linear system it gives successive approximations and is stopped if a particular condition is fulfilled, *e.g.* the residual \mathbf{r}_{i+1} is less than some predefined threshold. The general scheme of the conjugated gradient method as well as its detailed analysis can be found in [She94]. Nevertheless, some specific remarks related to the application to LCP are given below.

The expression for calculating the conjugate direction

$$\mathbf{d}_{i+1} = \mathbf{r}_{i+1} + \beta_{i+1} \mathbf{d}_i \quad (16)$$

usually takes the value of the coefficient β_{i+1} from Fletcher-Reeves' formula.

$$\beta_{i+1} = \frac{\mathbf{r}_{i+1}^T \mathbf{r}_{i+1}}{\mathbf{r}_i^T \mathbf{r}_i} \quad (17)$$

However, another possible approach is to calculate β_{i+1} using Polak-Ribiere's formula.

$$\beta_{i+1} = \frac{\mathbf{r}_{i+1}^T (\mathbf{r}_{i+1} - \mathbf{r}_i)}{\mathbf{r}_i^T \mathbf{r}_i} \quad (18)$$

Analysis of both approaches in our computations showed that the Fletcher-Reeves method converged if the initial approximation was sufficiently close to the solution, whereas the Polak-Ribiere method sometimes resulted in an infinite loop. However, the latter often converged faster.

To adapt the conjugate-gradient algorithm to our specific MLCP(\mathbf{A}, \mathbf{b}) formulation, we add an additional projection step (15) to the general scheme. Another important modification we introduce concerns the residual. Given the current solution λ_{i+1} of the MLCP(\mathbf{A}, \mathbf{b}) we denote the set of feasible $\mathbf{w} = \mathbf{A} \cdot \lambda - \mathbf{b}$ as $W(\lambda_{i+1})$. Since we are interested only in solutions lying in the feasible domain, we modify the intermediate residual $\tilde{\mathbf{r}}$ by projecting its value onto the set $W(\lambda_{i+1})$.

$$\mathbf{r}_{i+1} = \mathbf{Proj}(\tilde{\mathbf{r}}_{i+1}, W(\lambda_{i+1})) \quad (19)$$

This way, the direction for searching the solution on the current iteration step is lying in the feasible domain.

Moreover, if the current solution is close to the real solution then the projected residual \mathbf{r}_{i+1} is close to zero, which may not be the case for $\tilde{\mathbf{r}}_{i+1}$.

We did not carry out a rigorous theoretical investigation of the convergence of the obtained projected conjugate gradient-like method, but we thoroughly tested it experimentally. The complete algorithm for the projected conjugate gradient method is summarized in Algorithm 2.

Algorithm 2 Projected conjugate gradient algorithm

```

 $\mathbf{d}_0 \leftarrow \mathbf{b} - \mathbf{A} \cdot \lambda_0$ 
 $\mathbf{r}_0 \leftarrow \mathbf{b} - \mathbf{A} \cdot \lambda_0$ 
for  $i = 0$  to  $i_{max}$  do
   $\alpha_i \leftarrow \frac{\mathbf{r}_i^T \mathbf{r}_i}{\mathbf{d}_i^T \mathbf{A} \mathbf{d}_i}$ 
   $\tilde{\lambda}_{i+1} \leftarrow \lambda_i + \alpha_i \lambda_i$ 
   $\tilde{\mathbf{r}}_{i+1} \leftarrow \mathbf{r}_i - \alpha_i \cdot \mathbf{A} \cdot \mathbf{d}_i$ 
   $\lambda_{i+1} \leftarrow \mathbf{Proj}_{contact}(\tilde{\lambda}_{i+1})$ 
   $\mathbf{r}_{i+1} \leftarrow \mathbf{Proj}(\tilde{\mathbf{r}}_{i+1}, W(\lambda_{i+1}))$ 
  if error is small1 then
    exit
  endif
  if Polak-Ribiere then
     $\beta_{i+1} \leftarrow \frac{\mathbf{r}_{i+1}^T (\mathbf{r}_{i+1} - \mathbf{r}_i)}{\mathbf{r}_i^T \mathbf{r}_i}$ 
  else
     $\beta_{i+1} \leftarrow \frac{\mathbf{r}_{i+1}^T \mathbf{r}_{i+1}}{\mathbf{r}_i^T \mathbf{r}_i}$ 
  endif
   $\mathbf{d}_{i+1} \leftarrow \mathbf{r}_{i+1} + \beta_{i+1} \mathbf{d}_i$ 
endfor

```

Combined Iterative Method and Termination Criteria

In order to improve the iterative search for the solution of the MLCP(\mathbf{A}, \mathbf{b}) we combine the projected conjugate gradient and the projected Gauss-Seidel methods. One of the advantages of using the projected conjugate gradient is its fast convergence rate during the first iteration steps. The conjugate direction is chosen for optimal convergence, and therefore this method has a clear advantage over the projected Gauss-Seidel approach at this stage. However, the convergence rate decreases while approaching the solution and the projected Gauss-Seidel method becomes more preferable. Following this consideration we perform several steps of the projected conjugate gradient method and then use the resulting solution as the initial approximation of the projected Gauss-Seidel algorithm.

As termination criteria of the iterative loops we check the values of the successive approximations of the solution $\|\lambda_{i+1} - \lambda_i\|$ as well as the value of the projected residual $\|\mathbf{r}_{i+1}\|$. If either $\|\lambda_{i+1} - \lambda_i\| \leq \epsilon$ or $\|\mathbf{r}_{i+1}\| \leq \delta$

¹ The details of the exit criterion are discussed in the following section.

is fulfilled then the corresponding iterative loop is terminated. The error thresholds ϵ_{cg} , δ_{cg} and ϵ_{gs} , δ_{gs} for the conjugate gradient and Gauss-Seidel iterative loops respectively can be set to different values (obviously, $\epsilon_{cg} \geq \epsilon_{gs}$ and $\delta_{cg} \geq \delta_{gs}$).

Taking into account the physical meaning of the solution λ – in our case this is the contact impulse or force – it is reasonable to require a certain precision for each component of λ which is related to the accuracy of the computer simulation. Therefore, along with above criteria we also use

$$\|\lambda_{i+1} - \lambda_i\|_\infty \leq \epsilon_\infty \quad (20)$$

as well as

$$\|\mathbf{r}_{i+1}\|_\infty \leq \delta_\infty \quad (21)$$

In some cases the convergence rate of both iterative methods is slow. This is presumably a consequence of the numerical properties of the matrix \mathbf{A} and the limited numerical accuracy. For instance, for the projected Gauss-Seidel the convergence rate is small if $\|\mathbf{L} + \mathbf{D}\|$ is close to 1 [CPS92, Mur88]. In such cases the successive approximations of the solution may oscillate or even diverge. In order to prevent infinite loops we restrict the number of iteration within both phases of the combined method. The termination of the projected conjugate gradient loop is enforced after $2n$ iterations, where n is the size of the system in consideration, and the projected Gauss-Seidel loop is halted after a predefined number of N_{max} iterations.

In order to improve the precision in cases of forced termination we store the best solution approximation showing the smallest residual \mathbf{r} . The value is used as the outcome of the corresponding phase of the method, if it is better than the last approximation. Thus, we guarantee that the best approximation obtained in the conjugate gradient phase is taken for initializing the Gauss-Seidel phase. The final solution will correspond to the smallest residual among all of the obtained approximations. It should also be noted that according to the experimental results the portion of the cases with poor convergence, *i.e.* cases for which the iterative process did not terminate within the maximum number of iterations, is quite small – ranging from 0 to 0.9%. On the contrary, using a pure projected Gauss-Seidel method for the same simulating scenarios gave up to 3% cases with poor convergence.

5 RESULTS

In order to compare the performance of the proposed method for separated and non-separated contact treatment, several scenes were simulated.

Separated vs. Non-Separated Contact Region Handling

A scene of balls breaking a pyramid of bowling skittles with friction was used to test methods in a dynamic

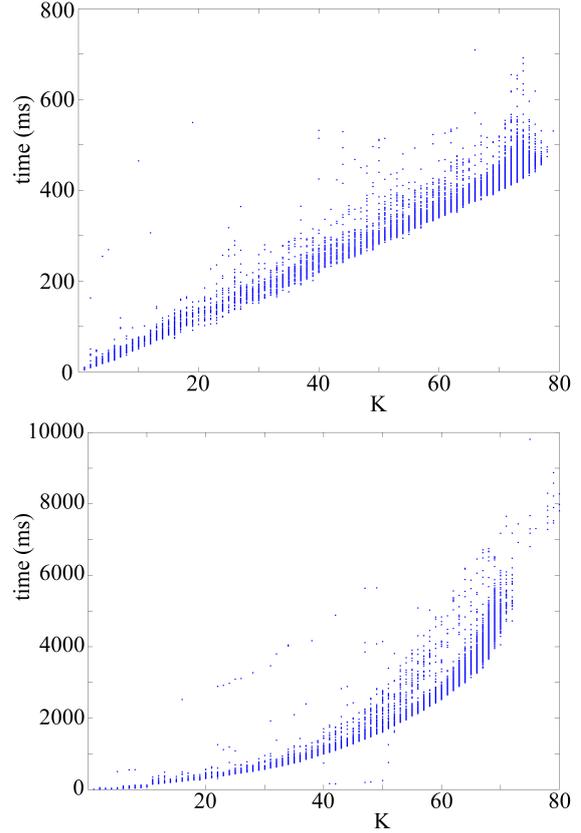


Figure 1: Static scene: Number of contacts K vs. computation time for separate (above) and non-separate (below) contact handling (the latter plot can be omitted)

simulation without any resting states because of the absence of gravity. A scene of balls stacking in a bucket under gravity was used to test the methods in mostly static conditions. The number of contacts varies from 1 to ~ 45 for the dynamic scene and from 1 to ~ 80 for the static scene. Note that all objects in the simulations are (slightly) deformable.

The advantage of the separation of the contact area into independent regions becomes apparent for MLCPs with larger numbers of contacts. The benefit is even present if the processing of the independent regions is performed sequentially for a method of complexity $O(n^2)$. The average total computation time is $\sim 2.5 - 3$ times less for the dynamic, and $\sim 7 - 8$ times less for the static scene.

Figure 1 shows the dependency of the computation time on the number of contacts. In case of non-separated contact handling the time increases much faster than for separated contact handling. Moreover, since the independent contact regions in the latter approach have similar sizes, an almost linear growth is obtained. Note that a further possible improvement could be achieved by processing the detached contact regions in parallel.

Friction Handling

Simple static scenes of deformable objects placed on an inclined plane were used to verify the correctness of the friction handling. Experiments showed that the critical inclination angle of the plane corresponds to the friction coefficient between the objects and the plane with high accuracy. Moreover, the number of separate contact areas between objects and the plane had no influence on the result. It was the same for global and separated contact area handling.

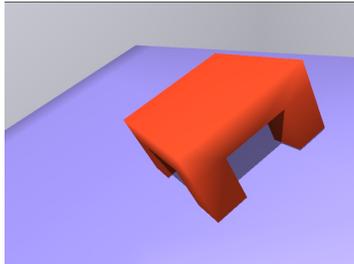


Figure 2: A table on the inclined plane

When simulating the sliding of a deformable plastic table on a plane (Figure 2), even a typical behavior found in reality could be reproduced. If the friction coefficient exceeds the critical value for the given inclination, a deformable table still can move downwards with its legs sliding in turns (*i.e.* the front legs slide while the back ones remain still, then the front legs stop and the deformation tension transfers to the back legs which start to slide until the opposite deformation tension cause them to stop and the cycle repeats). This phenomenon is a distinctive feature of certain objects made of plastic and can be easily observed in reality. It also has been described in related work dealing with contact friction [KSJP08].

Finally, both friction models were tested in more complex scenes – the 4-sided pyramid and the friction cone. The combined MLCP solving method demonstrated a considerably better performance when using the friction cone model – the convergence time decreased by $\sim 20 - 40\%$.

6 DISCUSSION AND CONCLUSION

We have presented an algorithm for the separation of detached contact regions in a simulated scene consisting of deformable objects. The experimental results demonstrated considerable gain in performance by using this approach. Moreover, the separate handling of the contact regions allows further acceleration by parallelization.

The presented contact model is based on simple constraint conditions and directly considers the mass points of the discretized deformable objects. This approach provides a simple diagonal mass matrix of the system which does not contain blocks related to the inertia tensors unlike most of previously proposed models. The

simplicity of the mass matrix combined with the sparsity of the constraint matrix potentially allows efficient implementation of matrix computations by employing known patterns of \mathbf{M} and \mathbf{J} . Therefore, no auxiliary routines or modifications, *e.g.* iterative constraint anticipation [OTSG09], are needed.

We also presented an iterative method for the solution of the contact MLCP which combines the projected conjugate gradient and the widely used projected Gauss-Seidel methods.

7 ACKNOWLEDGEMENTS

This work has been performed within the frame of the EU project PASSPORT ICT-223894 and the Swiss CTI project ArthroS.

REFERENCES

- [AP97] ANITESCU M., POTRA F.: Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics* 14, 3 (1997), 231–247.
- [Bar89] BARAFF D.: Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Computer Graphics, SIGGRAPH89* (1989), vol. 23, pp. 223–232.
- [Bar94] BARAFF D.: Fast contact force computation for nonpenetrating rigid bodies. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (1994), ACM, pp. 23–34.
- [Bar96] BARAFF D.: Linear-time dynamics using lagrange multipliers. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM, pp. 137–146.
- [BW92] BARAFF D., WITKIN A.: Dynamic simulation of non-penetrating flexible bodies. *Computer Graphics (Proc. Siggraph)* 26, 2 (1992), 303–308.
- [BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Computer Graphics Proceedings, Annual Conference Series* (1998), SIGGRAPH, pp. 43–54.
- [Cat05] CATTO E.: Iterative dynamics with temporal coherence. *Online Paper* (2005).
- [Cot90] COTTLE R. W.: The principal pivoting method revisited. *Math. Program.* 48 (1990), 369–385.
- [CPS92] COTTLE R., PANG J. S., STONE R. E.: *The Linear Complementarity problem*. Academic Press, 1992.
- [DAK04] DURIEZ C., ANDRIOT C., KHEDDAR A.: Signorini's contact model for deformable objects in haptic simulations. In *IROS, 2004. Proceedings.* (2004), vol. 4, pp. 3232–3237 vol.4.
- [DDKA06] DURIEZ C., DUBOIS F., KHEDDAR A., ANDRIOT C.: Realistic haptic rendering of interacting deformable objects in virtual environments. *IEEE Transactions on Visualization and Computer Graphics* 12, 1 (2006), 36–47.
- [Erl07] ERLEBEN K.: Velocity-based shock propagation for multibody dynamics animation. *ACM Trans. Graph.* 26, 2 (2007), 12.
- [GBF03] GUENDELMAN E., BRIDSON R., FEDKIW R.: Non-convex rigid bodies with stacking. *ACM Transaction on Graphics* 22, 3 (2003), 871–878.
- [GL89] GOLUB G. H., LOAN C. F. V.: *Matrix Computations*, second ed. Baltimore, MD, USA, 1989.
- [GMS04] GIROD B., MAGNOR M. A., SEIDEL H.-P. (Eds.): *Proceedings of the Vision, Modeling, and Visualization Conference 2004* (2004), Aka GmbH.

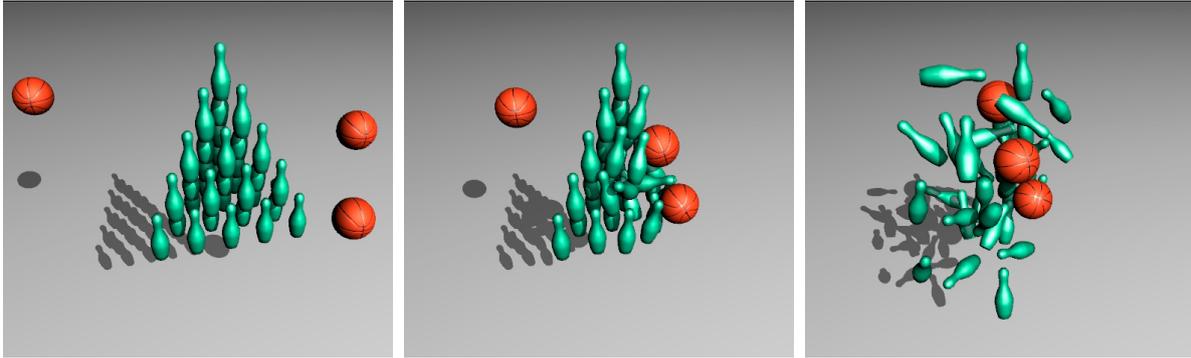


Figure 3: Dynamic scene



Figure 4: Static scene

- [HB00] HOUSE D. H., BREEN D. E. (Eds.): *Cloth modeling and animation*. A. K. Peters, Ltd., 2000.
- [HTK*04] HEIDELBERGER B., TESCHNER M., KEISER R., MÜLLER M., GROSS M. H.: Consistent penetration depth estimation for deformable collision response. In Girod et al. [GMS04], pp. 339–346.
- [HVS*09] HARMON D., VOUGA E., SMITH B., TAMSTORF R., GRINSPUN E.: Asynchronous contact mechanics. *ACM Trans. Graph.* 28, 3 (2009), 1–12.
- [KEP05] KAUFMAN D., EDMUNDS T., PAI D.: Fast frictional dynamics for rigid bodies. *ACM Trans. Graph.* 24, 3 (2005), 946–956.
- [KMH*04] KEISER R., MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M. H.: Contact handling for deformable point-based objects. In Girod et al. [GMS04], pp. 315–322.
- [KSJP08] KAUFMAN D., SUEDA S., JAMES D., PAI D.: Staggered projections for frictional contact in multibody systems. In *ACM SIGGRAPH Asia 2008 papers* (2008), ACM, pp. 1–11.
- [LNZL08] LI D.-H., NIE Y.-Y., ZENG J.-P., LI Q.-N.: Conjugate gradient method for the linear complementarity problem with s-matrix. *Mathematical and Computer Modelling* 48, 5-6 (2008), 918–928.
- [Mur88] MURTY K. G.: *Linear Complementarity, Linear and Nonlinear Programming*, vol. 3 of *Sigma Series in Applied Mathematics*. Heldermann Verlag, 1988.
- [OG07] OTADUY M. A., GROSS M.: Transparent rendering of tool contact with compliant environments. In *WHC '07: Proceedings of the Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems* (2007), IEEE Computer Society, pp. 225–230.
- [OTSG09] OTADUY M., TAMSTORF R., STEINEMANN D., GROSS M.: Implicit contact handling for deformable objects. *Computer Graphics Forum (Proc. of Eurographics)* 28, 2 (2009).
- [PPG04] PAULY M., PAI D., GUIBAS L.: Quasi-rigid objects in contact. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2004), Eurographics Association, pp. 109–119.
- [RA05] RENOUF M., ALART P.: Conjugate gradient type algorithms for frictional multi-contact problems: applications to granular materials. *Computer Methods in Applied Mechanics and Engineering* 194, 18-20 (2005), 2019–2041.
- [She94] SHEWCHUK J. R.: *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Tech. rep., 1994.
- [Str90] STRONGE W.: Rigid body collisions with friction. *Proceedings: Mathematical and Physical Sciences* 431, 1881 (1990), 169–181.
- [VMT97] VOLINO P., MAGNENAT-THALMANN N.: Developing simulation techniques for an interactive clothing system. In *VSMM '97: Proceedings of the 1997 International Conference on Virtual Systems and MultiMedia* (1997), IEEE Computer Society, p. 109.
- [VT00] VOLINO P., THALMANN N. M.: Accurate collision response on polygonal meshes. In *CA '00: Proceedings of the Computer Animation* (2000), IEEE Computer Society, p. 154.
- [Wit97] WITKIN A.: Physically based modeling: Principles and practice. In *Computer Graphics* (1997), pp. 11–21.
- [WP99] WRIGGERS P., PANATIOTOPOULOS P. (Eds.): *New Developments in Contact Problems*. SpringerWien-NewYork, 1999.
- [Wri02] WRIGGERS P.: *Computational Contact Mechanics*. John Wiley & Sons Ltd., 2002.

Interactive Stipple Rendering for Point Clouds

Naoyuki Awano

Osaka Institute of Technology
1-79-1, Kitayama
Hirakata, Osaka
Japan(573-0196)
awano@is.oit.ac.jp

Koji Nishio

Osaka Institute of Technology
1-79-1, Kitayama
Hirakata, Osaka
Japan(573-0196)
nishio@is.oit.ac.jp

Ken-ichi Kobori

Osaka Institute of Technology
1-79-1, Kitayama
Hirakata, Osaka
Japan(573-0196)
kobori@is.oit.ac.jp

ABSTRACT

Non-photorealistic rendering has been attracting attention in the field of computer graphics. A common approach to artistic rendering is using a shape model which utilizes mesh data. In recent years, the use of point clouds as shape models has increased due to the rapid development of 3D-scanners used to create them. Correspondingly, there has been an increase in the research on point clouds. We propose a stipple rendering method as a type of artistic rendering for point clouds, based on a hybrid image/object space. First, we eliminate hidden points based on an image space. Next, we apply a novel shading method to the visible points based on an image space. Lastly, we apply the above two results to the input point cloud. We implement the proposed method using a graphics processing unit to accomplish the interactive rendering. The experimental results show that we can achieve shading and shadowing interactively.

Keywords

Computer graphics, non-photorealistic rendering, stippling, point cloud, and graphics processing unit.

1. INTRODUCTION

Non-photorealistic rendering (NPR) has become a major focus for research in the field of computer graphics, because it is an effective conveyor of geometric features. Considerable artistic rendering has been proposed using a 3D-shape model utilizing mesh model [Zan04][Sat04][Lak00][Say06]. A mesh model is suggested because it has a topological data structure that can be used to extract features. However, the mesh model must first be constructed from point clouds before any suggested methods can be applied to it.

Over the past few years, point clouds have attracted attention as a new shape model, because they can be easily created using 3D-scanners, which have seen rapid development lately. Correspondingly, there has been an increase in the research on point clouds [Pau03][Pfi00] including NPR studies.

For example, Zakaria [Zak04] proposed a hybrid

image/object space method of interactive silhouette rendering. It can also do stipple rendering or user-drawn strokes on point set surfaces. Runions [Run07] proposed a novel rendering method for point clouds. It creates stripes, called “ribbons” to achieve photorealistic or non-photorealistic representations. Furthermore, it achieves NPR interactive silhouette rendering utilizing ribbons only. Rosenthal [Ros08] proposed an image space rendering method using a graphics processing unit (GPU). It is mainly used for photorealistic representations, and optionally for non-photorealistic representations. In addition, it can render silhouettes on photorealistic representations, and accordingly enables conspicuous representation.

In this paper, we propose an NPR method of stippling using a point cloud without normal vectors. We selected stippling because it is suitable for point clouds, which consist of points only. First, we eliminate the hidden points based on the image space. Next, we apply a new shading method to the visible points based on the image space; moreover, we can control the degree of shading. In addition, we implement all of the methods using a GPU to accomplish interactive rendering.

2. STIPPLE RENDERING

Stippling is an artistic rendering method which uses points only, and achieves shading by changing the density of the points. To get results for stippling

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

using a point cloud, we limit the background color to white and color all the points black. The input point cloud our method uses is evenly distributed without normal vectors and insufficiency.

Figure 1 shows the results of applying general shading to the mesh model. Figure 1(b) is the result of applying diffuse reflection to Figure 1(a). Figure 1(c) is the result of applying specular reflection to Figure 1(b). Typically, there are two steps to general shading; diffuse and specular reflection. Therefore, we apply our method, which also consists of diffuse and specular reflection, to the input point cloud. To get an effect similar to diffuse reflection, we thin out some of the points from Figure 2(a) as shown in Figure 2(b). In addition, we omit some local points for specular reflection as shown in Figure 2(c).

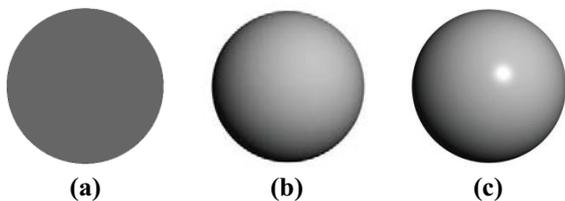


Figure 1. Shading; (a) original; (b) diffuse; (c) specular.

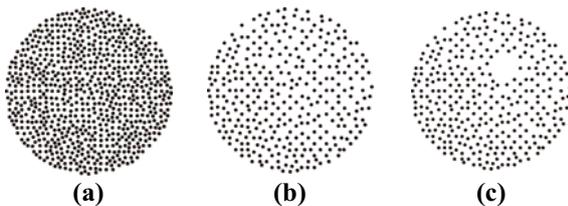


Figure 2. Shading using our proposed method.

Our method consists of four steps, which are outlined in Figure 3, and we implement all the methods using a GPU to accomplish interactive rendering.

- Step 1)** Creating a texture to eliminate the hidden points from the point set surfaces.
- Step 2, 3)** Creating textures for diffuse and specular reflections so that we can obtain similar effects.
- Step 4)** Applying the three textures applied to a point cloud and obtain the result.

2.1 Hidden points texture

In general, if we use the mesh data as input shapes, the back faces are eliminated from the front faces utilizing a Z-buffer. However, point clouds have points only, so the points on the back faces are not eliminated even if we apply a Z-buffer. Therefore, we apply following method so that we can apply the Z-buffer to eliminate the hidden points.

First, we place a texture plane at a viewpoint on the GPU at a size greater than the screen resolution. All points are then projected onto the texture plane. In Figure 4(a), the depth value of point A is 2, point B

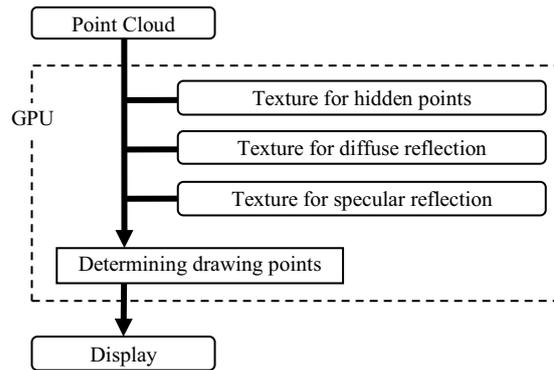


Figure 3. Outline of our method.

is 4, and point C is 3. So, each pixel has the smallest depth value of all the points which are points projected onto it. Next, as shown in Figure 4(b), in order to eliminate the hidden points in the texture plane, we store each depth value into the neighboring pixels, which also have the smallest depth value. After that, we apply the results of the texture plane to the point cloud; see Section 2.4.

However, there are cases where the hidden points on the back faces are not eliminated from the texture plane by the above process. In such cases, we repeat the above process until the hidden points are eliminated from the texture plane.

When B (Figure 4(b)) is eliminated from the texture plane, other points in high density parts of the texture plane also tend to be eliminated. As a result, too many points are eliminated from a high-density point cloud.

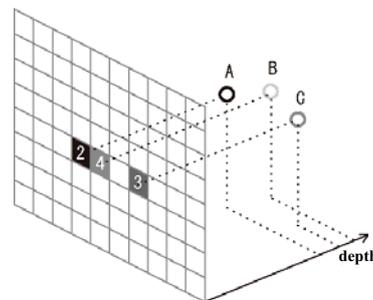


Figure 4(a). Projection onto a texture plane.

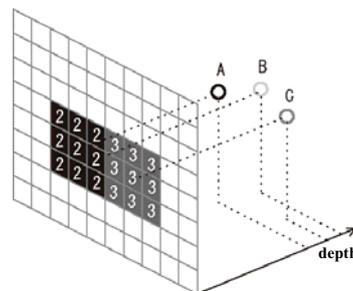


Figure 4(b). Storing the depth values into 8 neighboring pixels.

Hence, instead of storing the same depth value into the surrounding pixels, we store a slightly larger depth value, as shown Figure 5, so as to avoid cases where extra points are eliminated. In fact, if we assume that the maximum norm in all points is 1.0, we set the larger depth value is 0.05, which value was defined by our implementation. The final result of the hidden points texture is shown in Figure 6.



(a) Before changing. (b) After changing.

Figure 5. The centered pixel is projected by a point with depth value 3. In (a), the other pixels have the same depth value: 3. In (b), the other pixels have a slightly larger depth value of 6.

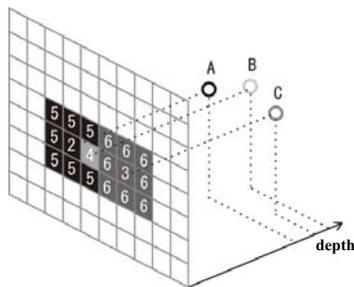


Figure 6. Improvement of storage.

2.2 Diffuse reflection texture

As shown in Figure 2(b), we thin out some points to represent diffuse reflection, and control the degree by changing the number of hidden points. In particular, we control the degree by changing the density of the points; a high-density part is low degree and a low-density part is high degree.

First, we place a texture plane at an illuminant on the GPU as a diffuse reflection texture. Next, all points are projected onto the texture plane as shown at the top of Figure 7. The density distribution of all the projected points on the texture plane is shown at the bottom of Figure 7. As a result of this, the lowest density part has the highest degree of diffuse reflection. Therefore, it is possible to achieve the effect of diffuse reflection by all points are projected only onto the texture plane. Additionally, we control the degree of diffuse reflection by changing the number of drawing points.

The idea is that each pixel on a texture has a hidden point, and we control the degree of diffuse reflection by changing texture size. In particular, after all visible points are projected onto the texture plane, each pixel has a minimum depth value, similar to the

hidden points texture. The texture consists of multi-resolution textures so that we can control the degree of diffuse reflection, as shown in Figure 8(a). Then we define a full quadtree, with LEVELS 0-n.

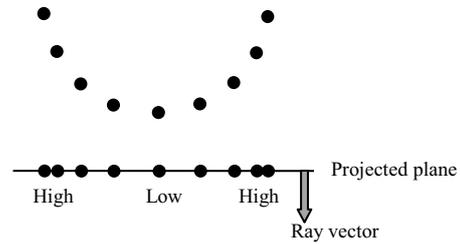
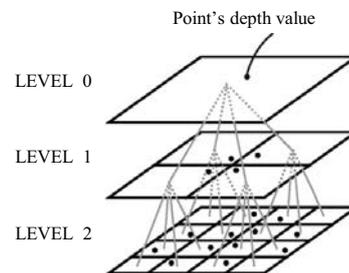


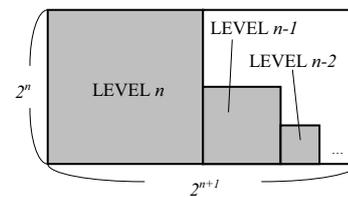
Figure 7. Density of projected points.

If we select a LEVEL, all stored points on the texture of the selected LEVEL are hidden. Moreover, the full quadtree is maintained on a texture so that we can effectively hold the textures on the GPU. In our method, we create the diffuse reflection texture, sized $2^{n+1} \times 2^n$, and store each LEVEL as shown in Figure 8(b). So, a pixel on LEVEL 1 has the minimum depth value of four pixels on LEVEL 2. Similarly, a pixel on LEVEL 2 has the minimum depth value of four pixels on LEVEL 3. Thus, all LEVELS can be created based on the texture of LEVEL n .

In particular, after the projection of all visible points onto the texture of LEVEL n , all of the other resolution textures are created in descending order,



(a) Multi-resolution texture.



(b) On a texture.

Figure 8. Full quadtree.

based on the texture of LEVEL n . So, e.g., a pixel (x, y) on the texture of LEVEL $(l-1)$ has the minimum depth value in (x', y') which is calculated by Formula (1), $(x'+1, y')$, $(x', y'+1)$, $(x'+1, y'+1)$.

$$(x', y') = (2(x - \sum_{k=l}^n 2^k), 2y) \quad (1)$$

2.3 Specular reflection texture

We refer to the Phong reflection model that is typically applied to a shape model. The degree of specular reflection is determined by the angle between the ray and normal vector of the surface. However, the input point cloud does not include normal vectors; therefore, instead of using normal vectors, we create parameter similar to the above angle.

First, we place a texture plane at an illuminant on the GPU as a specular reflection texture. Next, all points are projected onto the texture plane. Then, assume that all points have normal vectors as shown at the top of Figure 9. The angle between each normal vector and ray vector has the angle distribution shown at the bottom of Figure 9.

Note that the density distributions of the projected points in Figure 9 have a similar distribution to those in Figure 7. Therefore, we compare their distribution to achieve shading without a normal vector. In particular, we regard each of the density distributions on the projected plane to be the angle between a normal and ray vector. Similarly, if we replace the ray vector with the eye vector in Figure 9, we can regard the density distributions of each projected point with the angle between the normal and eye vector. Consequently, the differences of their distributions indicate the angle between the eye and ray vector.

In Figure 6, note that the depths based on point A were stored into 8 pixels; point C has 9 pixels. We have found that their numbers are almost proportional to the distributions on the hidden points texture.

Thus, we create a hidden points texture at an illuminant as a specular reflection texture on the GPU. Then, we determine the drawing points by referring to specular reflection texture and the hidden points texture; see the next section.

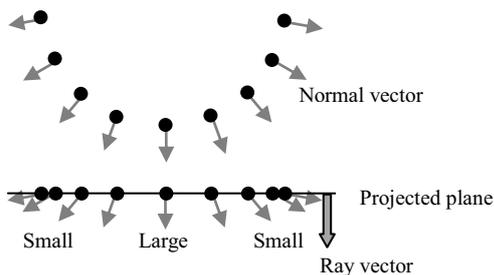


Figure 9. Angle between two vectors.

2.4 Determining drawing points

We determine the drawing points by utilizing all three textures. First, we determine the visible points by applying the results of the hidden points texture to the point cloud. Next, we apply the results of the diffuse

reflection texture and specular reflection texture to the visible points.

2.4.1 Eliminating hidden points

To eliminate the hidden points, all points are projected again onto the hidden points texture, as shown in Figure 10, where the depth value of point A is 2, point B is 4, point C is 3, and point D is 7. It shows that point A is projected onto a pixel whose depth value is 2, and each adjacent pixel has a depth value of 4-5. Then, since there is a larger depth value than point A's depth value of 2, point A is drawn.

However, point D is projected onto a pixel whose depth value is 6, and each adjacent pixel has a depth value of 3-6. Then, since there is no larger depth value than point D's depth value of 7, point D is not drawn. Using this process, all points are projected onto the hidden points texture to determine whether they should be eliminated or not.

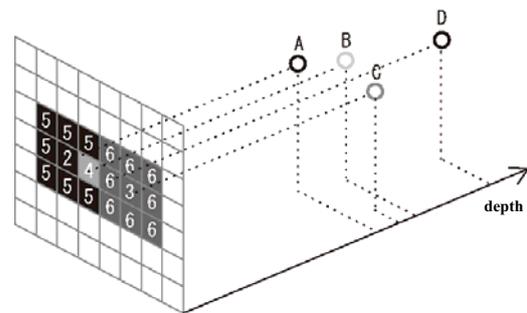


Figure 10. A result of hidden points texture.

2.4.2 Hiding points for diffuse reflection

We hide some of the points to represent diffuse reflection. This process is illustrated using Figure 11, where we can select any of the four hidden LEVEL (0-3).

We start by selecting hidden LEVEL 3. All visible points are projected onto the diffuse reflection texture of LEVEL 3, so as to represent the diffuse reflection with LEVEL 3. For example, if a point with depth value 4 is projected onto the pixel shaded with diagonal lines, the point is not drawn because it has the same depth value as that of the pixel. In contrast, when a point with depth value 5 is projected onto the same pixel, the point is drawn.

Next, when we select hidden LEVEL 2, all visible points are projected onto the diffuse reflection texture of LEVEL 3. For example, if a point with depth value 4 is projected onto the pixel shaded with diagonal lines, the pixel is related to a pixel such as the bold pixel in LEVEL 2 of Figure 11. Since, the bold pixel in LEVEL 2 has the depth value of 2, the point is drawn because its depth value is different from that of the pixel.

In case of selecting hidden LEVEL l , all points are first projected onto the location of LEVEL n . After that, when a point is projected onto a pixel (x, y) , we refer to the pixel as (x', y') , computed by Formula (2). If the depth in the referring pixel is equal to the projected point's depth, the point is not drawn.

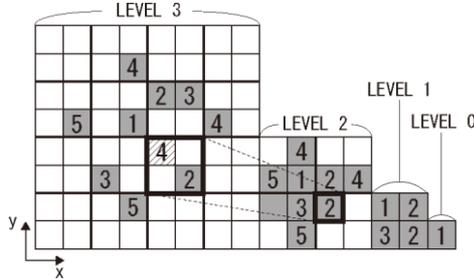


Figure 11. An example of diffuse reflection texture. (The numbers represent the depth value of each pixel.)

$$(x', y') = \begin{cases} (x, y) & (n = l) \\ \left(\left(\frac{x}{2^{n-l}} \right) + \sum_{k=l+1}^n 2^k, y / 2^{n-l} \right) & (n > l) \end{cases} \quad (2)$$

2.4.3 Hiding points for specular reflection

We hide some points to represent specular reflection in addition to diffuse reflection. First, all visible points are projected onto a specular reflection texture and a hidden points texture. As noted in Section 2.2, we count the pixels in each texture and calculate the difference between them. Then, we define a threshold ε to the difference so that we can control the degree of specular reflection; ε is the ratio of the difference to the number of pixels. The first iteration count is 9 and the second iteration count is 25. Next, all visible points that have a difference greater than ε are hidden.

3. RESULTS

We conducted experiments and verified our method. We tested our method as shown in Table 1. The size of the diffuse reflection texture is 2048×1024 ; the hidden points texture and specular reflection texture are 1024×1024 each. Additionally, we adopted Cg for implementing our method on GPU, and assign coordinate value XYZ of all points to color value RGB in all three textures.

CPU	Core2 Duo 2.66 GHz
RAM	2.0 GB
GPU	GeForce 8800GTX
VRAM	768 MB

Table 1. Experimental environment.

Figure 12 shows the results of eliminating hidden points with 72,027 (Bunny). This indicates that we can eliminate the hidden points by repeating the

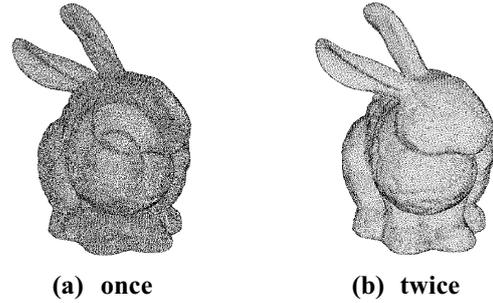


Figure 12. Eliminating hidden points.

process, if we cannot eliminate the hidden points the first time.

Figure 13 shows the results of applying diffuse reflection to two point clouds: (a) and (b) are 542,199 (Oil pump); (c) and (d) are 152,807 (Chinese dragon). They indicate that we can achieve diffuse reflection as shown in Figure 2(b) and adjust the diffuse reflection by changing the LEVEL. Figure 14 shows the results of specular reflection on the shapes shown in Figure 13. It shows that the high-light of specular reflection, as shown in Figure 2(c), appears locally by using our shading method. Furthermore, they indicate that we can adjust the specular reflection by changing ε . Therefore, they indicate that we achieve shading by stippling.

Figure 15 shows another result of shading with 172,974 (Armadillo). This indicates that our method can achieve shadowing as shown in the circle in Figure 15. Due to containing the eliminated hidden points in our shading method, our shading method has not been applied to the part in the shadow.

Figure 16 shows the results of processing speed. We have compared the implementation on a CPU against a GPU. It shows that on a GPU, the speed is 11 to 17 times faster than on a CPU. The reason is that our method can be implemented with an image space, and the GPU can perform in parallel with an image space. Thus, our method can achieve interactive rendering.

4. CONCLUSION

In this paper, we proposed a NPR method with stippling using point cloud without normal vectors.

First, we eliminated hidden points. Next, we applied a novel shading method consists of specular reflection and diffuse reflection. In addition, we implemented our method on GPU to accomplish interactive rendering.

In the future, we will refine our method so it can be applied to an unorganized point cloud.

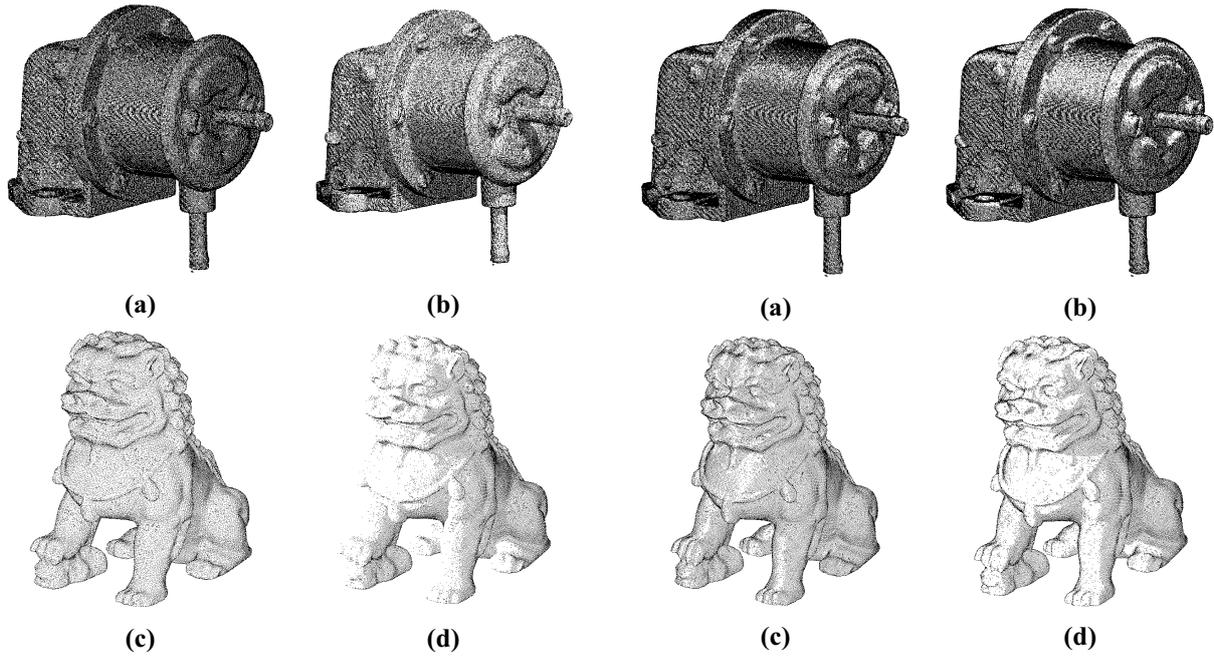


Figure 13. Results of after applying diffuse reflection. The hidden LEVEL of (a) and (c) are 7; (b) and (d) are 9.

Figure 14. Results of applying specular reflection. ϵ of (a) and (c) is 0.3; (b) and (d) is 0.2.

5. REFERENCES

- [Zan04] J. Zander, T. Isenberg, S. Schlechtweg, and T. Strothotte, High Quality Hatching, *Computer Graphics Forum*, 23(3), 2004, 421-430.
- [Sat04] Y. Sato, T. Fujimoto, K. Muraoka, and N. Chiba, Stroke-Based Suibokuga-Like Rendering for Three Dimensional Geometric Models, *The Journal of the Society for Art and Science 2004*, 3(4), 2004, 224-234.
- [Lak00] A. Lake, C. Marshall, M. Harris, and M. Blackstein, Stylized Rendering Techniques For Scalable Real-Time 3D Animation, *1st International Symposium on Non-Photorealistic Animation and Rendering*, 2000, 13-20.
- [Say06] R. Sayeed, T. Howard, State of the Art Non-Photorealistic Rendering (NPR) Techniques, *EG UK Theory and Practice of Computer Graphics*, 2006, 1-10.
- [Pau03] M. Pauly, R. Keiser, L.P. Kobbelt, and M. Gross, Shape Modeling with Point-Sampled Geometry, *Proceedings of SIGGRAPH 2003*, 2003, 641-650.
- [Pfi00] H. Pfister, M. Zwicker, J.V. Baar, and M. Gross, Surfels: Surface Elements as Rendering Primitives, *Proceedings of SIGGRAPH 2000*, 2000, 335-342.
- [Zak04] N. Zakaria and H.P. Seidel, Interactive Stylized Silhouette For Point-Sampled Geometry, *GRAPHITE 2004*, 2004, 242-249.
- [Run07] A. Runions, F. Samavati, and P. Prusinkiewicz, Ribbons: A representation for point clouds, *The Visual Computer: International Journal of Computer Graphics*, 23, 2007, 945-954.
- [Ros08] P. Rosenthal and L. Linsen, Image-space Point Cloud Rendering, *Proceedings of Computer Graphics International 2008*, 2008, 136-143.

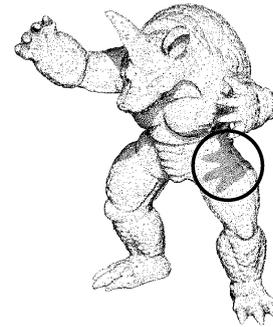


Figure 15. A result of shadowing.

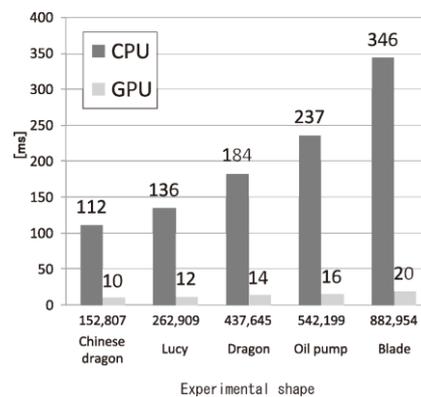


Figure 16. Processing speed.

New methods for progressive compression of colored 3D Mesh

Ho Lee

Université de Lyon, CNRS
Université Lyon 1, LIRIS,
UMR5205, F-69622, France
ho.lee@liris.cnrs.fr

Guillaume Lavoué

Université de Lyon, CNRS
INSA-Lyon, LIRIS,
UMR5205, F-69621, France
guillaume.lavoue@liris.cnrs.fr

Florent Dupont

Université de Lyon, CNRS
Université Lyon 1, LIRIS,
UMR5205, F-69622, France
florent.dupont@liris.cnrs.fr

ABSTRACT

In this paper, we present two methods to compress colored 3D triangular meshes in a progressive way. Although many progressive algorithms exist for efficient encoding of connectivity and geometry, none of these techniques consider the color data in spite of its considerable size. Based on the powerful progressive algorithm from Alliez and Desbrun [All01a], we propose two extensions for progressive encoding and reconstruction of vertex colors: a prediction-based method and a mapping table method. In the first one, after transforming the initial RGB space into the Lab space, each vertex color is predicted by a specific scheme using information of its neighboring vertices. The second method considers a mapping table with reduced number of possible colors in order to improve the rate-distortion tradeoff. Results show that the prediction method produces quite good results even in low resolutions, while the mapping table method delivers similar visual results but with a fewer amount of bits transmitted depending on the color complexity of the model.

Keywords: Progressive compression; Colored 3D mesh.

1. INTRODUCTION

Nowadays, 3D models are widely used in many applications such as virtual reality, entertainment, Computer-Aided Design, scientific simulation and e-commerce. Among the various existing representations, 3D triangular meshes are particularly appropriate to represent these models due to their algebraic simplicity so that the most part of manipulations can be processed by the graphic hardware. The increasing popularity and the increasing size of 3D meshes to respond to the needs of representing objects or scenes with more and more realism have become a critical issue, especially for

the end-users with limited bandwidth and storage capacity. In this context, compression is a good solution for this task; two different classes of techniques exist: *single-rate* and *progressive*. Single-rate techniques compress the mesh information as a whole and the visualization is possible only when the entire compressed file is received at the user-side. These techniques often have advantages in terms of compression ratio. On the other hand, progressive techniques are more flexible by providing the possibility of early visualization of the coarse version with very few bits transmitted and then more refined models can be rendered when more bits are received. This property of progressive reconstruction is useful especially for large models and for Internet-based applications.

A typical 3D mesh is composed by its geometry, connectivity and attribute data. Geometry data determine vertex positions in the 3D space. Connectivity data describe how these vertices are connected together and attribute data specify colors, surface normals or texture information for instance.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Among these mesh elements, attribute data is not often considered by the state-of-the-art mesh compression algorithms in spite of their visual importance and their considerable size, especially for the progressive algorithms.

In this paper, we propose two approaches to encode efficiently color data in a progressive manner. Our work can be seen as an extension of the progressive mesh compression algorithm from [All01a] which encodes only the connectivity and the geometry. We have chosen this algorithm, since it is the best state-of-the-art connectivity-driven algorithm. As it was observed in [Lee09], even the most efficient geometry-guided algorithm [Pen05] produces a poor visual quality at low and medium bit rate, due to the stair-like effects. Moreover, Alliez and Desbrun's algorithm which is based on the vertex removal allows a better prediction using more neighboring vertices than algorithms based on edge-contraction [Hop96] [Paj00] [Tau98a] [Kar02], leading to the better compression of the color data.

Related work

Single-rate techniques have been firstly studied by many researchers in order to reduce compactly the mesh data [Tau98b] [Tou98] [Gum98] [Baj99] [Ros99] [All01b].

Later on, research on progressive compression techniques have been introduced with the increasing popularity of web-based applications. The first progressive algorithm was proposed by Hoppe [Hop96]. This new mesh representation, *progressive mesh*, simplifies a given mesh by applying successively edge contraction operations. At each step, the edge to be contracted is properly chosen in order to reduce the approximation error as much as possible. At the decompression stage, the reconstruction is achieved by the inverse operation, vertex split. This method has been extended by several researchers to improve the compression efficiency and also the rate-distortion trade-off [Paj00] [Tau98a] [Kar02]. In their work, Cohen-Or et al. [Coh99] proposed the patch coloring algorithm for progressive transmission. This algorithm removes iteratively an independent vertex set – any two vertices of this set are not connected by an edge – using vertex decimation. Then, each hole left by vertex decimation is re-triangulated in a deterministic way. The set of these new triangles is called a patch. The authors applied 2-coloring and 4-coloring methods to the patches in order to permit the decoder to identify correctly each patch. This algorithm encodes the connectivity with an average of 6 bits-per-vertex (bpv). Alliez and Desbrun [All01a] extended the existing valence-driven single-rate

approaches [Tou98] [All01b] for progressive mesh encoding. Their algorithm, which is also based on vertex decimation, consists of two conquests: decimation and cleansing. The decimation conquest is successively applied alternating with cleansing conquest, building different levels of details. This algorithm encodes the connectivity with an average of 3.7 bpv.

All the progressive algorithms described above are connectivity-driven algorithms, meaning that the priority is given to the connectivity coding. Observing that the amount of geometry data in the compressed file is often larger than connectivity data, Gandoin and Devillers [Gan02] proposed the first geometry-driven approach based on the kd-tree space subdivision. In terms of lossless compression ratio, this algorithm outperforms connectivity-driven algorithms. Peng and Kuo [Pen05] proposed a more efficient geometry-guided technique by using the octree cell subdivision. An improvement is achieved by using efficient prediction methods for both connectivity and geometry. These geometry-driven algorithms give very impressive results in terms of lossless compression ratio, however they provide quite poor results at low resolutions, hence they are not fully efficient for progressive transmission. In [Lee08], the authors proposed key-frame based technique for the efficient transmission of animating meshes.

Up to present, the compression of the mesh attribute data such as colors, normals or texture coordinates plays a secondary role. Among the well-known single-rate techniques, only [Dee95] [Baj99] [Tau98b] proposed a method to encode vertex-bind color information in the RGB color space. However, the prediction and the quantization used for the color encoding are the same as for the geometry encoding regardless of its different nature. More recently, Ahn et al. [Ahn06] and Yoon et al. [Yoo07] proposed new methods for the efficient encoding of color data. Ahn et al. [Ahn06] used a mapping table based on the vertex layer traversal algorithm. Instead of encoding color coordinates of each vertex, they encode the index of the vertex color in the mapping table. A color value in the mapping table is encoded when it appears for the first time during the traversal. In other words, they have to encode the index of each vertex and the corresponding color coordinates in the mapping table. To further improve the efficiency, they also used a delta coding for color index encoding. Yoon et al. [Yoo07] introduces a prediction method using connectivity and geometry information of neighboring vertices. They consider different weights for the neighboring vertices using angle analysis. Then the color value of the current

vertex is predicted from weighted averaged color values.

Geometry images [Gu02] [Yao08] permit to represent compactly the colored geometric models using 2D images. There exist also some algorithms which allow the simplifying the mesh taking the color information into account [Hop99] [Gar98] [Roy05]. However, these algorithms do not provide a way to reconstruct the original mesh. To our knowledge, there is no progressive mesh coder allowing the encoding of color information.

2. DESCRIPTION OF BASE ALGORITHM

Our color compression scheme is based on the valence-driven progressive approach proposed by Alliez and Desbrun [All01a]. This algorithm uses the good statistical property of the native distribution of vertex valences for the mesh connectivity encoding. This approach iteratively decimates a set of vertices by combining decimation and cleansing conquests to get different levels of details (LOD). Decimation conquest consists in traversing the mesh patch by patch using a gate-based traversal; the front vertex of the current gate is removed only when its valence is below 7, in order to preserve compactly the vertex valence distribution. The hole left is then re-triangulated. The boundary edges of the actual patch are pushed into a FIFO list. The decimation conquest continues with the next available gate in the FIFO list, performing a breadth first traversal. Similarly, cleansing conquest removes only vertex of valence 3.

Fig.1 illustrates this mechanism: a regular input mesh (Fig.1.a) is simplified by decimation conquest (Fig.1.b). A set of independent vertices (red vertices) is removed and patches are re-triangulated. After performing cleansing conquest (Fig.1.c), vertices of valence 3 (blue vertices) are removed. We can see that as the input mesh is regular, the simplified mesh is also regular. Even for irregular meshes, this algorithm delivers better triangulation at coarse levels than the work of Cohen-Or et al. [Coh99]. During the compression stage, valences of removed vertices and additional null codes (in case of irregular mesh) are encoded for the connectivity.

For the geometry coding, Alliez and Desbrun first applied a global and uniform quantization to the coordinates of the mesh vertices. Then, they used both the barycentric prediction and the approximate Frenet coordinate frame, separating normal and tangential components to further optimize the bit rate. The base vectors of the local frame are built from the current gate (one of the boundary edges of the patch)

and the approximated patch normal. The barycenter is obtained by averaging positions of neighboring vertices. The difference between the position of the vertex to be removed and the barycenter is then encoded in the local frame.

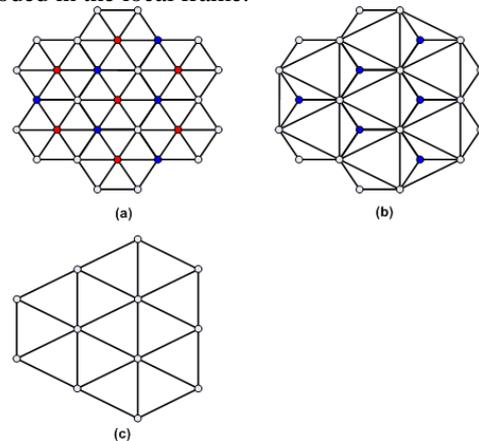


Figure 1. An example of decimation (b) and cleansing conquests (c) applied on a regular mesh (a).

Recently, Lee et al. [Lee09] proposed an improved geometric coder using a discrete bijection. They adopted the bijection method of Cartens et al. [Car99] and optimized the coding efficiency by providing an angle minimization. They also proposed a framework to improve the rate-distortion (R-D) trade-off by using adaptive quantization during the mesh simplification process.

In the following of this paper, we use the mesh traversal and the connectivity encoding techniques of [All01a] and the geometry coder of [Lee09].

3. COLOR COMPRESSION

The amount of color data associated to the mesh can be as large as or even larger than connectivity and geometry without an adaptive compression method. Therefore, a specific technique is required to reduce efficiently these data.

We propose in this section two methods which permit to encode the color data associated with mesh vertices, in a progressive manner.

Color space transform

Before to compress any color data, all colors expressed in the RGB space are transformed into the Lab space. The Lab space is the luminance-chrominance representation which describes more closely the human perception system. Moreover, this representation is more decorrelated than the RGB space. Thus, the Lab space is more appropriate to the

data compression. After this transformation, each color is represented using 8 bits for L, a and b color components as in the initial RGB space.

Prediction-based method

Since we consider the connectivity reduction of Alliez and Desbrun [All01a], the simplest method to predict the color value of the current vertex to decode is to use the average color of neighbors, like the prediction used for geometry encoding as illustrated in Fig. 2.

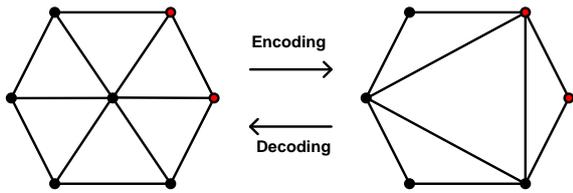


Figure 2. A vertex is removed (resp. inserted) during the encoding (resp. decoding) process. Its position is predicted from the averaged position of the neighboring vertices.

However, this prediction is not very efficient because the color data own a different behavior than geometry. In the case of quite regular meshes, the difference of positions (geometrical distance) between two vertices connected by an edge is relatively small, hence the barycentric prediction, explained in Section 2, can be performed efficiently. However, the color difference between two adjacent vertices can be very important, especially in the case of a vertex located in a color boundary, resulting that the averaging prediction is quite ineffective.

We can observe that the color value of a vertex is generally very close to at least one of its neighboring vertices' colors. Based on this observation, we propose a method which selects the proper color among the colors of the neighboring vertices so as to predict more efficiently. To perform this color selection, we first calculate the average values, L_{mean} , a_{mean} and b_{mean} of the neighbor colors. Then, for each component, we select the one which is the closest to the corresponding average component among the neighboring vertices' colors. The difference between the original and the selected color component values is then entropy coded to allow the decoder to reconstruct the exact color value. During the decompression process, after an insertion of new vertex, the corresponding color data is added to the vertex, allowing the progressive reconstruction.

Mapping table method

As each vertex color is represented using 24 bits, there exist 2^{24} possible colors. Yet, the human visual perception system cannot distinguish relatively small change of colors. Hence, we propose a method to reduce the bit rate needed for color encoding by reducing the number of colors to encode.

Our method first applies a clustering algorithm to the input mesh in order to reduce the number of possible colors without seriously affecting the visual distortion. Then, we use a mapping table method as in [Ahn06], based on the observation that this method is particularly useful when there is small number of colors. Fig.3 illustrates the diagram of our method in the case of the compression process.

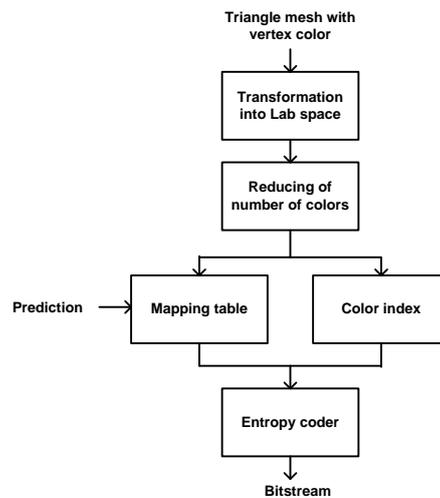


Figure 3. Diagram of the encoding process of our second algorithm.

The clustering method is widely used for 2D image compression [Sal98]. It consists in finding a set of representatives (Look Up Table) and in mapping each vertex color to its nearest representative. To generate a correct mapping table by minimizing the color distortion as much as possible, we use the well-known K-means clustering algorithm.

1. K initial seeds colors are selected from the mesh color data set.
2. K clusters are created by associating each color to the nearest seed.
3. The centroids of each cluster are used as new seeds and the new clusters are created.

The algorithm repeats step 2 and 3 until the all seeds are unchanged. Since the efficiency of the clustering algorithm depends on the initial condition of the seeds, we use as initial seeds the K more frequent colors of the input mesh in order to strengthen the approximation. After finding K representatives, each vertex color is replaced by its closest representative.

A result of this clustering algorithm is illustrated in Fig.4 with the Globe model containing initially 5030 colors. Although the number of possible colors is reduced to 256 colors, one can hardly distinguish the color distortion.

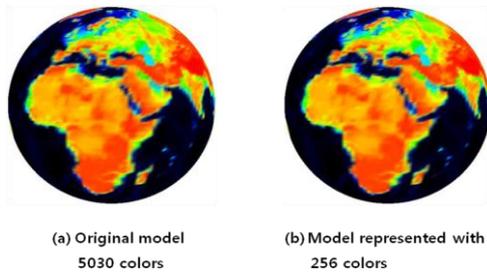


Figure 4. Color reduction based on clustering for the Globe model.

To encode the color data, we use the mapping table containing the final representatives obtained by the clustering algorithm. At the compression stage, when removing a vertex, the color index corresponding to its color in the mapping table is encoded.

To further enhance the rate-distortion performance and also to reduce additionally the coding cost, all color values contained in the mapping table are encoded in a progressive way. When the resolution level is augmented (when the mesh is refined to one higher level), the information of new colors are sent, enlarging the size of the mapping table. Fig. 5 illustrates an example of the progressive decoding of the mapping table. For a given resolution level, the mapping table contains 4 colors (C0 to C3). When a new vertex is inserted, and if the decoder identifies that the associated color is not present in the current mapping table then the new color value is added to it.

Furthermore, we try to reduce the coding cost needed for the encoding of the mapping table. In Ahn et al.'s work [Ahn06], they encode each color values in the mapping table using 24 bits. We reduce this coding cost by using our prediction-based method. During the compression process, we use our prediction method when removing each vertex. And we store only the difference between the original color value and the predicted color of the last encountered vertex for each color of the mapping table. So, during the mesh reconstruction, when a vertex is inserted and its color is revealed for the first time, we use information of the neighbors to acquire the correct color value of the corresponding color in the mapping table.

Even when the full resolution of the geometry has been reached, there still exist some differences of colors between the reconstructed color mesh and the original one, due to the color number reduction step

(i.e. the clustering). However, depending on the needs, the original vertex colors can be restored, by encoding the difference of color between the initial color value of each vertex and its representative during the clustering phase. These differences are sent at the end of the decompression process.

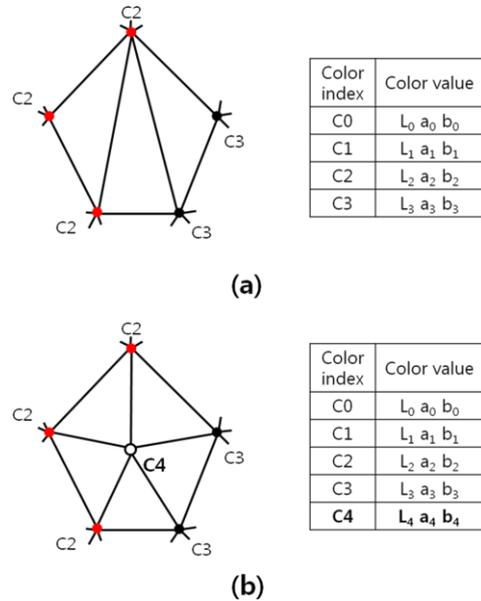


Figure 5. An example of progressive decoding of the mapping table. Initial mapping table (a) is enlarged when a new color, C4, appears (b).

4. EXPERIMENTAL RESULTS

Fig. 6 shows the 3D models used in our experiments. Each coordinate of vertices of these models is quantized using 10 bits.

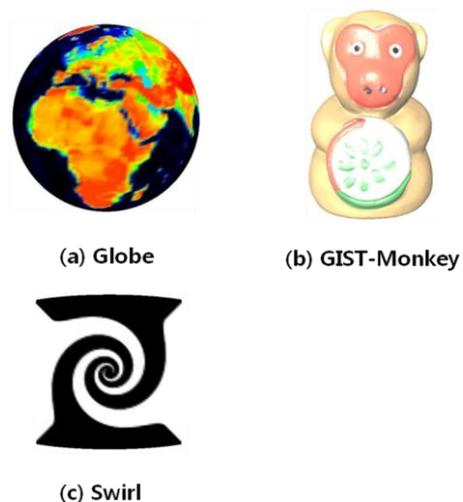


Figure 6 : Models used for compression.

Lossless compression

Table 1 shows lossless compression results for the test models using our methods. The bit rates needed for compression of the color information and those of the mesh connectivity and the geometry (C+G) are given in bits-per-vertex (bpv). As most of the well-known state-of-the-art progressive algorithms do not consider color data, the efficiency of our prediction method is compared with the prediction scheme used in Yoon et al.'s work [Yoo07] and the averaging prediction. The method of Yoon et al. was originally applied in a single-rate way in their work. We have adapted their prediction method based on angle analysis for the mesh traversal technique of [All01a]. We can observe that the performance of these prediction schemes is similar for each model and better compression rates are obtained for the models containing large surface of smooth color variation, such as GIST-Monkey and Swirl models. For all test models, our method outperforms those of [Yoo07] and the averaging prediction method, especially for the Swirl model which contains many color boundary vertices and for those the color difference on the boundary is important.

Results of lossless compression of our mapping table method are also given. Different numbers of seeds, K , are used during the color number reduction step. We can see that the more the number of initial seeds increases, the more the coding rates decreases. This is because the cost of the original color restitution applied after reaching the finest geometry resolution level increases rapidly when the value of K becomes smaller. As a consequence, the result of the mapping table is better than our prediction method when the value of K is superior to 256.

Progressive compression

Fig. 7 illustrates some intermediates meshes with respective coding rates. All the rates presented in this figure include the amount of connectivity, geometry and color data. Our two methods produce intermediates results with a quite good visual quality both for the geometry and the color even for low bit rates (< 5 bpv).

In this figure, the GIST-Monkey model is used to compare the efficiency of our two methods: prediction method (d-f) and mapping table method (g-i). As expected, the mapping table method produces intermediate meshes of similar visual quality with less bit rates. Even though the number of colors has been severely reduced, from 6669 to 32, one can hardly sense the discrepancy comparing to the results of the prediction method.

5. CONCLUSION

In this paper, we have presented two methods for progressive encoding of colored meshes. To our knowledge the proposed methods are the first ones which consider the effective color coding in the field of 3D progressive compression. Our first algorithm based on the prediction is easily implementable and produces quite good results even for low bit rates. The second algorithm combining the mapping table with the clustering delivers intermediate meshes of almost equal visual quality with fewer bits, enhancing the rate-distortion trade-off.

As future work, we will investigate a reliable metric permitting to measure the global distortion between two meshes taking mesh geometry and also color into account, in order to evaluate the rate-distortion performance.

ACKNOWLEDGMENTS

We would like to thank Hyun Soo Kim for sending us the color mesh models. This work has been supported by French National Research Agency (ANR) through COSINUS program (project COLLAVIZ n°ANR-08-COSI-003).

Models	# V	# Color	C + G	Prediction			Mapping table		
				Average	Yoon	Our	K = 64	K = 256	K = 1024
Globe	36866	5030	4.61	16.43	16.17	15.37	15.81	13.81	12.65
GIST-Monkey	50503	6669	13.5	6.49	6.49	5.95	8.52	8.33	7.23
Swirl	9216	138	4.12	9.97	10.16	6.62	3.04	-	-

Table 1. Compression rates of test models in bits-per-vertex.

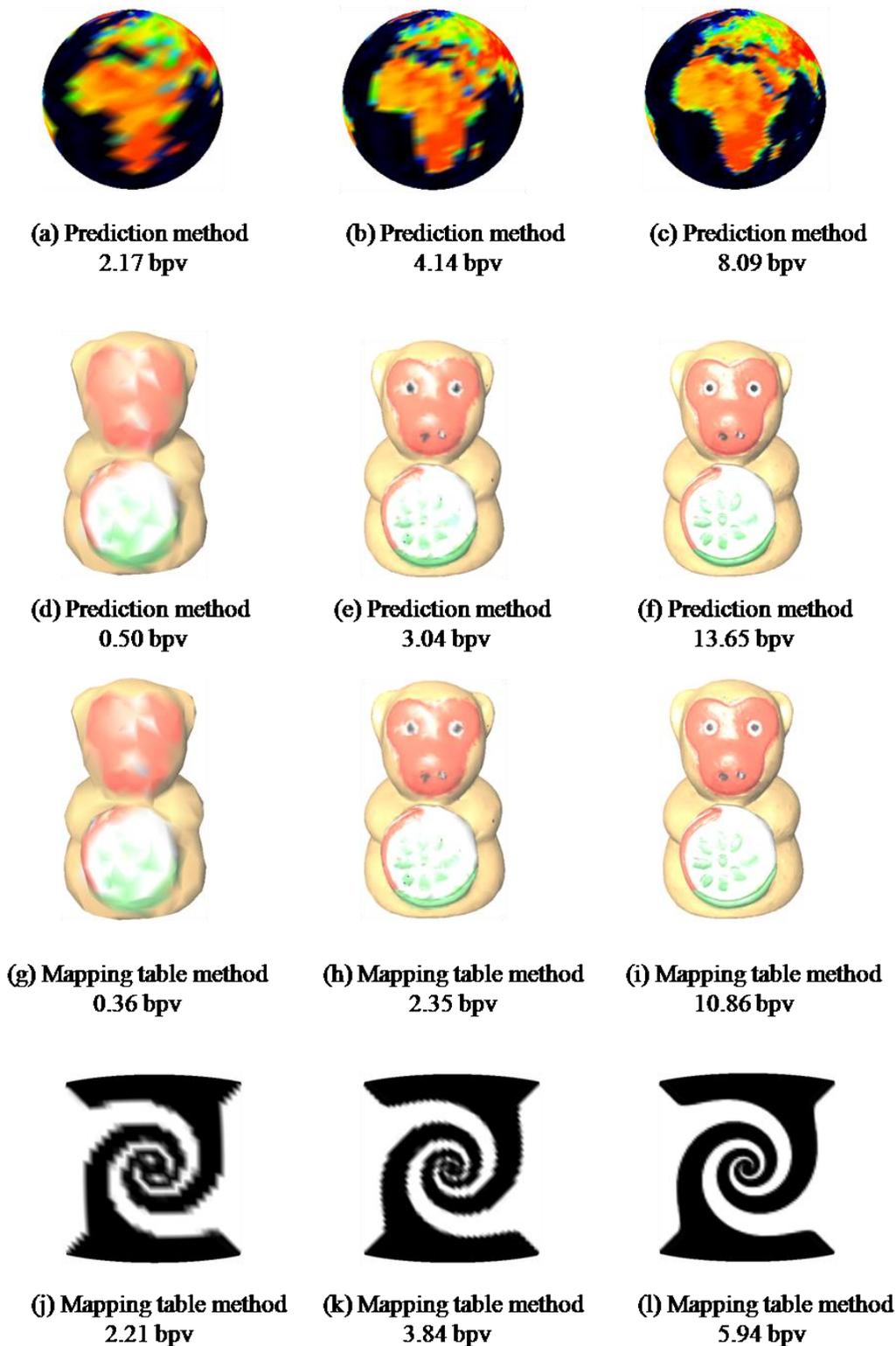


Figure 7. Result of progressive decoding of the test models. The model Globe (a – c) and the model GIST-Monkey (d – f) are progressively reconstructed using our prediction method. Intermediates meshes of the models GIST-Monkey (g – i) and Swirl (j – l) are given by our mapping table method. For both models, the number of possible colors are reduced, using $K = 32$ seeds in the clustering step. The bit rates include the connectivity, the geometry and the color information.

REFERENCES

- [Ahn06] J. Ahn, C. Kim, Y. Ho. Predictive compression of geometry, color and normal data of 3-D mesh models. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(2):291-299, 2006.
- [All01a] P. Alliez and M. Desbrun. Progressive compression for lossless transmission of triangle meshes. In *ACM SIGGRAPH*, 198-205, 2001.
- [All01b] P. Alliez and M. Desbrun. Valence-Driven connectivity encoding for 3D meshes. In *Eurographics*, 480-489, 2001.
- [Baj99] C. L. Bajaj, V. Pascucci, and G. Zhuang. Single resolution compression of arbitrary triangular meshes with properties. In *IEEE Visualization*, 1999, 307-316.
- [Car99] H.-G. Cartens, W.A. Deuber, W. Thumser, and E. Koppenrade. Geometrical bijections in discrete lattices. *Combinatorics, Probability and Computing*. 8:109-129, 1999.
- [Coh99] D. Cohen-Or, D. Levin, and O. Remez. Progressive compression of arbitrary triangular meshes. In *IEEE Visualization Conference Proceedings*, 67-72, 1999.
- [Dee95] M. Deering. Geometry compression. In *ACM SIGGRAPH*, 13-20, 1995.
- [Gan02] P.-M. Gandoin and O. Devillers. Progressive lossless compression of arbitrary simplicial complexes. *ACM Transactions on Graphics*, 21(3):372-379, 2002.
- [Gar98] M. Garland and P. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *IEEE Visualization*, 263-269, 1998.
- [Gu02] X. Gu, S. Gortler, and H. Hoppe. Geometry Images. *ACM Transactions on Graphics*, 21(3):355-361, 2002.
- [Gum98] S. Gumhold and W. Strasser. Real time compression of triangle mesh connectivity. In *ACM SIGGRAPH*, 133-140, 1998.
- [Hop96] H. Hoppes. Progressive meshes. In *ACM SIGGRAPH*, 99-108, 1996.
- [Hop99] H. Hoppes. New quadric metric for simplifying meshes with appearance attributes, In *IEEE Visualization*, 59-66, 1999.
- [Kar02] Z. Karni, A. Bogomjakov, and C. Gotsman. Efficient compression and rendering of multi-resolution meshes. In *IEEE Visualization Conference Proceedings*, 347-354, 2002.
- [Lee09] H. Lee, G. Lavoué, and F. Dupont. Adaptive coarse-to-fine quantization for optimizing rate-distortion of progressive mesh compression. In *VMV*, 73-81, 2009.
- [Lee08] T. Lee, Y. Wang, and T. Chen. Animation key-frame extraction and simplification using deformable analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(4):478-486, 2008.
- [Paj00] R. Pajarola and J. Rossignac. Compressed progressive meshes. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):79-93, 2000.
- [Pen05] J. Peng and C.-C.J. Kuo. Geometry-guided progressive lossless 3D mesh coding with octree (OT) decomposition. In *ACM SIGGRAPH*, 609-616, 2005.
- [Ros99] J. Rossignac. Edgebreaker : Connectivity compression for triangle meshes. *IEEE Transaction on Visualization and Computer Graphics*, 5(1):57-61, 1999.
- [Roy05] M. Roy, S. Foutou, A. Koschan, F. Truchetet, and M. Abidi. Multiresolution analysis for meshes with appearance attributes, In *ICIP*, 816-819, 2005
- [Sal98] D. Salomon. Data compression: The complete reference. Springer Verlag, 1998.
- [Tau98a] G. Taubin, A. Guéziec, W. Horn, and F. Lazarus. Progressive forest split compression. In *ACM SIGGRAPH*, 123-132, 1998.
- [Tau98b] G. Taubin and J. Rossignac. Geometric compression through topological surgery. *ACM Transaction on Graphics*, 17(2):84-115, 1998.
- [Tou98] C. Touma and C. Gotsman. Triangle mesh compression, In *Proceedings of Graphics Interface*, 26-34, 1998.
- [Yao08] Z. Yao and T. Lee. Adaptive Geometry Image. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):948-960, 2008.
- [Yoo07] Y. Yoon, S. Kim, and Y. Ho. Color data coding for three-dimensional mesh models considering connectivity and geometry information. In *ICME*, 253-256, 2007.

High-Quality Wavelet Compressed Textures for Real-time Rendering

Nico Grund
Philipps-Universität Marburg
Hans-Meerwein-Str.
35032 Marburg, Germany
ngrund@informatik.uni-marburg.de

Nicolas Menzel
Philipps-Universität Marburg
Hans-Meerwein-Str.
35032 Marburg, Germany
menzel@informatik.uni-marburg.de

Michael Guthe
Philipps-Universität Marburg
Hans-Meerwein-Str.
35032 Marburg, Germany
guthe@informatik.uni-marburg.de

ABSTRACT

Although modern graphics hardware provides up to 1.5 gigabytes of memory, methods for effective texture compression are still required since there is always demand for more detailed and realistic images. In this paper, we present a method for the effective compression of large images and textures based on a quadratic B-Spline wavelet. The transformation is followed by a tree-compaction algorithm, which achieves high compression ratio at good image quality.

Keywords

Texture Compression, Wavelets.

1 INTRODUCTION

Compression of large textures and images is of crucial interest in many fields in computer graphics. The programmability of modern GPU allows texture and image compression and decompression algorithms to exploit the full parallel processing power and streaming capability. However, one considerable obstacle is yet the limited support and capacity for general purpose data storage on the graphics card: Though provided with up to 1.5 gigabytes of memory, modern GPU's texture size is still limited to currently 8192x8192 pixels.

Wavelet encoding has proven to be an appropriate tool for image compression, as in JPEG2000 [15]. Advantages are that it is easily implemented in software and can be adapted to hardware for improved performance [16]. There are several benefits arising from wavelet compression. First, the encoding itself leads to a straightforward lossy compression scheme by quantizing the coefficients. By encoding the wavelet coefficients into a quadtree, some memory can be saved by removing subtrees containing only zero coefficients after quantization. This means that textures will require less memory for storage, allowing them to fit into the limited texture size of the graphics card without the use of tiling. This way a shader program can be used for decompression and filtering.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Based on these observations, we present a compact tree-coding algorithm for the efficient high-quality compression of two-dimensional image data and a real-time random access decompression algorithm running on the GPU. Our approach is easily extensible to multidimensional data and to non-linear HDR data.

2 RELATED WORK

The S3 Inc. introduced five simple lossy block-decomposition-based compression schemes with compression rates of 4:1 and 8:1 [10] for 8-Bit RGBA images, which have been adopted by the Microsoft DirectX framework. Based on the observation that large textures, as required for terrain rendering, are not supported by graphics cards, Tanner et al. [14] proposed the clipmapping algorithm, which subdivides a huge texture into small tiles which fit into the texture memory.

For the compression of images the JPEG2000 standard [15, 2] supports the use of the LeGall and the Cohen-Daubechies-Feauveau 7/5 wavelet, superseding the discrete cosine transform used in regular JPEG compression. Wavelet-based compression schemes have proven to be more flexible, providing higher compression rates while yielding higher quality. Compared to other algorithms, they demand a higher decompression complexity. Therefore, recent work aimed at the use of modern graphics-hardware to yield interactive frame rates.

Beers et al. [1] introduced a vector-quantization-based technique that uses a precomputed codebook and stores a smaller texture of indices into this codebook. The size of the codebook determines the level of compression. More recently, Fenney [4] described a way to store a compressed texture so that decompress-

sion needs one lookup per sample only. Schneider et al. [12] introduced a compression scheme for static and time-varying volumetric datasets. This algorithm is based on a vector quantization with a fixed bit-rate. To initialize the compression they use a codebook, which is obtained using a splitting technique. The number of generated entries is confined to the bit-rate of the quantization. The compression rate is nearly 20:1.

Shapiro [13] presents the embedded zerotree wavelet algorithm (EZW), which is based on a discrete wavelet transform and a zerotree coding to store a compact multiresolution representation of significance maps, which contains the positions of the significant wavelet coefficients. This method can provide good performance with very low complexity. The disadvantage of the EZW procedure is, that all values are classified by an certain threshold. Coefficients below this threshold are simply omitted. As an result of this it will remove noise in uniform regions but also it generates blurry artifacts in the reconstructed image. DiVerdi et al. [3], proposed a method to implement the EZW algorithm for decoding on graphics hardware using the Haar wavelet. The wavelet coefficients are arranged in a tree with a zero node, where all child pointers of the leaves and nodes, which contain coefficients equal to zero, point to the zero node. While they achieve good compression rates, noisy images are problematic since too few wavelet coefficients are sufficiently close to zero for an imperceivable difference.

3 WAVELET-TRANSFORMATION

Wavelet transformation in general has been well studied in literature so we will not discuss it in detail. The most important property of the wavelet transformation is that it decomposes the image into perceptually meaningful subbands that can afterwards be compressed more efficiently than the original image.

Before the wavelet transformation the gamma correction is applied for linearization of the intensity values. This step needs to be replaced by a log-mapping in the case of HDR data. In both cases, the RGB color space is converted to the $Y'P_bP_r$ color space, where a luminance value and two differential color values are stored to consider the human visual system, which is more sensitive to changes in luminance than in color.

The choice of the wavelet basis is crucial for the wavelet compression. The two major characteristics of a basis are the width of support and the compression it can provide. A wider support yields better compression results, but is computational more expensive. We implemented three different wavelet bases in order to compare their benefits and disadvantages.

Our first implementation is the Haar wavelet, which has the most compact support. This simplicity makes it optimal for decoding performance. Disadvantages are, however, that it is neither continuous nor differentiable.

This results in highly visible block artifacts in the compressed image.

The LeGall biorthogonal wavelet, which is also described in the JPEG2000 specification [15, 2], is continuous, but not differentiable. Compared to the Haar wavelet, it represents local changes in frequency smoother and thus produces more appealing compression results. As shown in Figure 1, its support is three times as wide as the Haar wavelet.

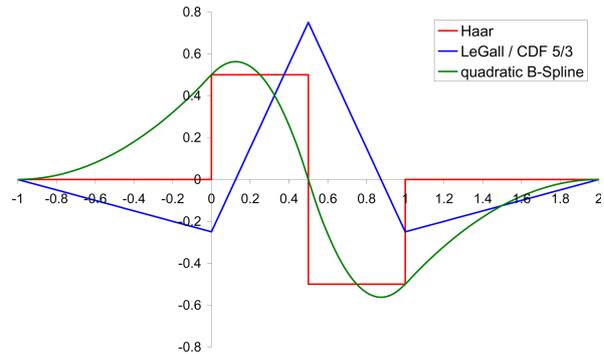


Figure 1: Haar, LeGall and quadratic B-Spline mother wavelet.

The quadratic B-Spline wavelet [11] is both continuous and differentiable. It therefore should achieve the best compression results compared to the two previous bases. It has the same support width as the Le Gall wavelet so the decompression performance is equivalent. The associated coefficients of the analysis and synthesis filter are shown in Table 1.

4 TREE-BASED COMPRESSION

Unfortunately, an entropy-based coding of the quantized wavelet coefficients as in image compressions algorithms like JPEG2000 is not suitable for real-time decompression on the GPU. Instead we first build a tree data structure from the wavelet decomposed image and then exploit redundancy in this tree by converting it into a general directed graph. In this procedure, identical or similar nodes are iteratively combined into a single node until a desired compression ratio is achieved.

i	Analysis Filter Coefficients		Synthesis Filter Coefficients	
	Lowpass Filter	Highpass Filter	Lowpass Filter	Highpass Filter
-1	1/4	1/4	-1/4	-1/4
0	3/4	3/4	3/4	3/4
1	3/4	-3/4	3/4	-3/4
2	1/4	-1/4	-1/4	1/4

Table 1: Coefficients of the quadratic B-Spline analysis and synthesis filter.

4.1 Wavelet tree

Based on the dyadic decomposition a natural tree structure for the wavelet coefficients is to store the LH, HL, and HH coefficients of a single pixel in the current level together with four pointers to the next finer level. The LL coefficient for the root level then needs to be stored outside the tree. This way the coefficients required to reconstruct a single pixel can be collected by traversing the tree from the root node to the leaf containing the highest resolution coefficients for that pixel. The major drawback is that for storing the quantized coefficients only 9 bytes are required, while the pointers require 12 bytes, when using up to 24 Bits which allows up to 16 MB for the compressed representation.

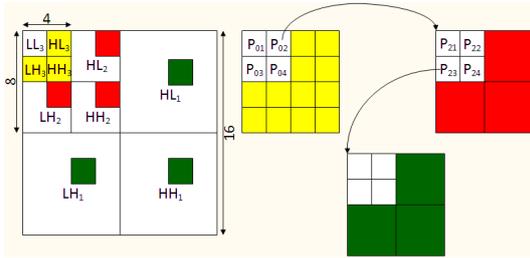


Figure 2: Dyadic decomposition and derived tree data structure.

In our approach we reduce the pointer overhead by grouping the wavelet coefficients of a two by two pixel block on each level. This way, only four pointers are required per 12 quantized YCC coefficients. Thus, the overhead is only 12 bytes per 36 bytes of data or in other words roughly 33%. Since the lowest resolution level only contains one coefficient of each type, four coefficients need to be stored outside the tree instead of only one. As these must be considered separately anyways, we stop the wavelet decomposition at two by two pixels and store all of them as LL coefficients. Figure 2 shows the dyadic decomposition and the resulting tree data structure.

4.2 Tree compression

After the tree data structure is generated, redundant nodes are iteratively removed. Since combining two nodes also joins their subtrees, only nodes with the same children are candidates for such a collapse operation. The final data structure now is a general directed graph with the addition of a specifically marked root node (see Figure 3). As a collapse operation might introduce an approximation error the ordering of collapses as well as the choice which two nodes are collapsed at each step determine the quality of the decompressed result.

We use a priority queue to perform the node collapse operations in an optimal order on the directed graph. To minimize the total mean square error (MSE), the key

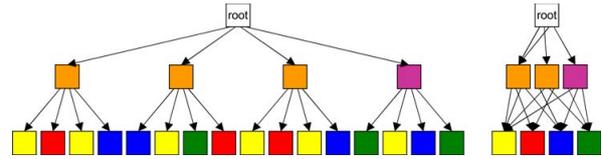


Figure 3: Uncompressed (left) and compressed (right) directed graph data structure. The colors depict identical coefficients stored in the tree nodes.

by which the operations are sorted needs to be proportional to the sum of squared differences (SSD) of all nodes collapsed together by this operation, i.e. all original nodes collapsed into the two candidates i and j . As the coefficients of each node are the average coefficients of all contained original nodes, this error $\epsilon(i, j)$ can be computed by summing up the SSD of both nodes (ϵ_i and ϵ_j) with the appropriately weighted SSD when collapsing the coefficients of the two candidates:

$$\epsilon(i, j) = \epsilon_i + \epsilon_j + d^2(i, j) \frac{w_i w_j}{w_i + w_j},$$

where $d^2(i, j)$ is the sum of squared differences of the coefficients of node i and j and $w_{i/j}$ is the sum of the weights of all original nodes collapsed to i and j , respectively. For the computation of the new coefficients, those of node i and j are simply multiplied by the weight stored in each of these nodes and divided by the new weight which is the sum w_i and w_j .

Since each node is always collapsed with the one for which the collapse has the lowest cost, we only need to find the closest node c_i for each node i and store this pair in the priority queue. This problem is similar to the nearest neighbor search in high dimensional spaces as our coefficient vector has a dimensionality of 36. The only exception is that the distance between two nodes with different child nodes must be set to infinite to prevent collapsing them. Section 4.2 discusses the nearest neighbor search algorithm we use in more detail.

When a collapse is performed, some of the queue entries become invalid and need to be recomputed. Assuming that the new nearest neighbor of those nodes introduces a higher SSD we can postpone the recomputation until that collapse is fetched from the priority queue. The only nodes for which we need to immediately find the nearest neighbor are the newly constructed node and all nodes that had one of the two collapsed nodes as immediate children. The latter is necessary as these nodes might now have a closer neighbor than the one that was previously found. Another property we used to speed up the initial filling of the priority queue is that inner nodes cannot be collapsed before the first few leaf nodes were removed since they cannot initially have the same child nodes.

Zerotree coding In addition to the optimizations described above we can also remove all leaf nodes for

which their coefficients are all quantized to zero by introducing a zero node similar to [3]. Since the zerotree coding does not need a nearest neighbor search or a priority queue, those parts of the wavelet tree that do not contain any information can be quickly removed. In contrast to [3] we do, however, not collapse nodes containing near-zero coefficients although these might be collapsed with the zero node at a later time if the introduced SSD is the lowest one.

As this step reduces the total number of nodes before the first neighbor search and the maximum number of collapse operations in the priority queue it can significantly reduce the total runtime. This is especially important for images that required a padding before the wavelet transformation.

Nearest neighbor search As mentioned above the coefficient vector for which we need to find the nearest collapse candidate is 36-dimensional. Thus we require an efficient method to search the nearest neighbor in this 36-dimensional space. Since each collapse implies removing two and adding one point to the candidate set, a spatial acceleration data structure like the r-tree [6] cannot be used and we need to restrict ourselves to a linear ordering based on some sort of key value.

Fortunately, we can exploit the fact that most coefficient vectors will be centered around the origin with a more or less gaussian distribution. Therefore, we chose our hash function to be the distance to the origin and only need to search those node with a similar distance. As soon as we find the first candidate, we can thus efficiently stop searching in one of the two directions if points farther away or closer to zero cannot introduce a lower error.

5 IMPLEMENTATION

To achieve real-time decompression, we had to meet some constraints that are given by the graphics hardware. First the coefficient values have to be quantized to the range 0 – 255 using a global scaling to the range $[0, 1]$ when storing them in a 24 bit RGB texture. After this, all values are simply scaled by the factor 255. The quantization also restricts us to textures of size 256 in each dimension since the color value is to be directly used as texture coordinate. Therefore, we use a three-dimensional texture to encode the tree. The size of this texture is $256^2 \times 2^n$, where $0 \leq n \leq 8$ and each pixel uses 24 bit in the regular RGB format.

As shown in figure 5 we encode blocks in pairs of 4×4 pixels. In each block, the upper left four pixels contain pointers to the children of the current node. In each pointer pixel, the color values contain the texture coordinates of the child nodes upper left pixel. With this scheme, we can encode $64 \times 64 = 4096$ nodes in each layer of the texture so we can store up to one million nodes or 36 million unique coefficients.

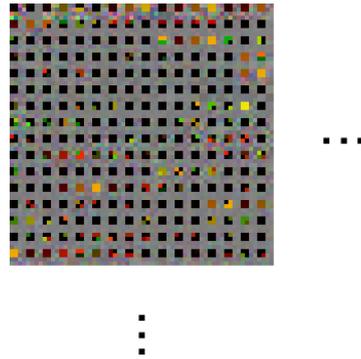


Figure 4: Part of the compressed wavelet data stored in the 3D-texture. Since the image is taken from depth 0, the root node is visible in the upper left corner.

5.1 Parallel decompression

For parallel decompression on the GPU, all wavelet functions contributing to the current pixel need to be evaluated and multiplied with the corresponding coefficients. The number of coefficients per pixel depends on the width of the mother wavelet and is one for Haar and three for LeGall und quadratic B-Spline in each dimension. This yields a total of 3 or 27 coefficients per level for Haar and LeGall/quadratic B-Spline, respectively. To extract these coefficients, one (Haar) or four nodes (both others) have to be visited per level. This sums up to 5 texture lookups per level for the Haar wavelet and 31 for the other two wavelets. Note, that since the child nodes are not queried at the leaf level, the total number of lookups is $4l - 1$ for the Haar wavelet and $31l - 4$ for both others, where l is the number of levels in the coefficient tree.

Although the number of lookups for the more complex wavelets might seem rather high, the Haar wavelet only allows nearest neighbor interpolation and thus produces inferior quality when zooming. To achieve bilinear interpolation the number of lookups for the Haar wavelet is quadrupled. This yields a total of $16l - 4$ which is approximately half than that of the other two wavelets. Due to the smoother wavelet functions however, these produce better quality images at the same compression rate and thus the higher number of texture lookups is tolerable.

6 RESULTS

To evaluate our proposed algorithm and compare it to existing approaches, we mainly used images from the image compression benchmark [5] (Figure 6 and 7).

A quality comparison of the three implemented wavelet transformations is shown in Figure 5. The Haar wavelet shows significant block artifacts, which neither appear using the LeGall nor the quadratic B-Spline wavelet. Since the LeGall scaling function is a linear filter, it tends to produce star-shaped artifacts. The quadratic B-Spline wavelet reproduces slightly



Figure 5: From left to right: Haar-, LeGall- and B-Spline wavelet.

more detail than the LeGall wavelet. In addition, the biquadratic interpolation that comes for free with the B-Spline wavelet generates smoother results when magnifying the image. The PSNR is similar for all three wavelets, where the Haar wavelet has the lowest (41.7 dB) and LeGall (45.4 dB) and quadratic B-Spline (42.9 dB) are slightly better.



Figure 6: Compression results with embedded zero tree coding (left) and with our approach (right).

Figure 6 shows the differences between embedded zero tree coding [13] (35.4 dB) and our method (37.6 dB). Both were compressed at a rate of 23:1. One disadvantage of the zero tree coding is, that all values below a certain threshold are simply omitted. While this removes noise in uniform regions, it cannot compress data in images containing high frequencies. In these cases the threshold needs to be significantly increased to achieve a desired compression ratio and thus the quality of the reconstructed image is degraded. In contrast to this, our clustering approach can also exploit similarities in high frequency regions and thus much fewer nodes need to be collapsed with the zero node. This greatly improves the visual quality when compressing this type of images.

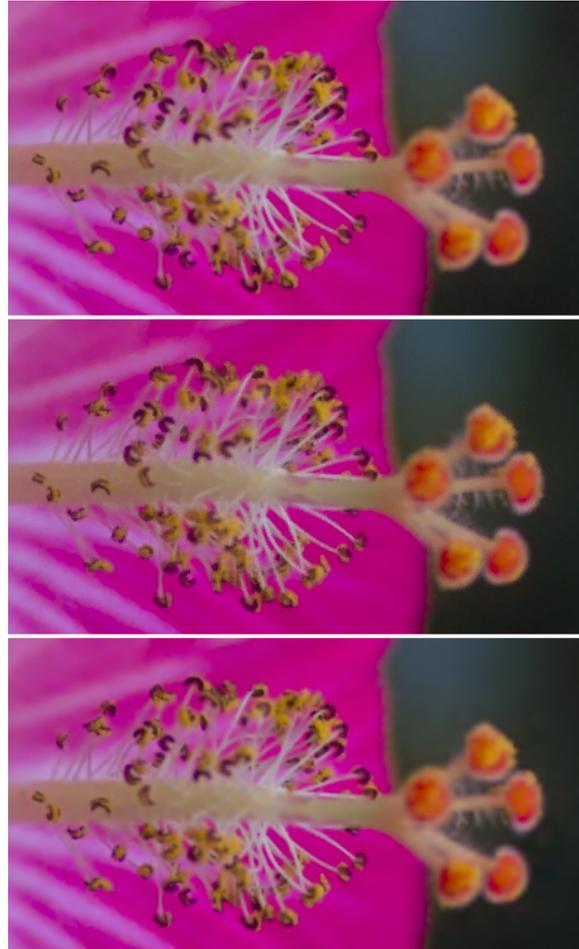


Figure 7: Comparison between S3TC (middle) and our approach with same compression ratio (top, no visual difference to original) and same quality (bottom).

In Figure 7 a comparison between S3TC (DXT1) and our approach is shown. The upper two images are both encoded with a compression ratio of 6:1 (8:1 for RGBA images) at 55.1 dB. Note, that our approach yields a significant higher quality (60.7 dB) at slightly smaller texture size (1.5 MB compared to 1.6 for DXT1). With four times the compression rate (26:1) the visual quality of our method (still 57.1 dB) is equivalent to S3TC, as shown in the lower image. Figure 8 shows an aerial image with a resolution of 3000×3000 compressed at a rate of 34:1 with 36.4 dB. Despite the high compression rate, important features are still preserved.

The decoding was performed in a pixel shader running on an nVidia GeForce GTX 295 in real-time. The performance for a 4096×4096 texture is approximately 400 Mpixels per second using the Haar wavelet and nearest neighbor filtering (100 Mpixels with bilinear filtering) and roughly 55 Mpixels per second with the LeGall and quadratic B-Spline wavelet. For smaller or larger images, the runtime is almost linear in the number of levels of the wavelet decomposition. E.g. for a



Figure 8: Drastic compression (34:1) of a 3000×3000 pixel aerial image. The marked area is magnified below the full image.

16k \times 16k texture we still achieve 46 Mpixels per second.

7 CONCLUSION AND LIMITATIONS

We presented an effective method for the compression of large textures and images based on the linear LeGall and quadratic B-Spline wavelet. With our tree-compaction algorithm, we achieve high compression ratios while still preserving high visual quality. The decompression is implemented as a pixel shader on a GPU and runs in real-time on current graphics hardware. Our approach has shown to be superior to simple zero-tree removal.

In the future we want to improve the compression time, which is currently 30 minutes for a 67 Mpixel image (8192×8192 pixel) and thus still rather slow. In ad-

dition, we want to extend our method to high dynamic range images and multi-dimensional datasets.

REFERENCES

- [1] Andrew C. Beers, M.Agrawala, and N.Chaddha, Rendering from compressed textures In *SIGGRAPH '96: Proceedings of the 23rd annual conference on computer graphics and interactive techniques*, pp.373-378, 1996.
- [2] C.Christopoulos, A.Skodras, T.Ebrahimi, The JPEG2000 Still Image Coding System: An Overview In *IEEE Transactions on Consumer Electronics, Vol.46, No.4*, pp.1103-1127, 2000.
- [3] S.DiVerdi, N.Candussi, T.Höllerer, Real-time Rendering with Wavelet-Compressed Multi-Dimensional Datasets on the GPU In *Technical Report UCSB/CSD-05-05*, 2005.
- [4] S.Fenney, Texture compression using low-frequency signal modulation In *HWWS'03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pp.84-91, 2003.
- [5] S.Garg, Image Compression Benchmark. http://www.imagecompression.infotest_images.
- [6] A.Guttman, R-trees: a dynamic index structure for spatial searching In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pp.47-57, 1984.
- [7] P.Lalonde, A.Fournier, A wavelet representation of reflectance functions *IEEE Transactions on Visualization and Computer Graphics*, pp.329-336, 1997.
- [8] P.Lalonde, A.Fournier, Interactive rendering of wavelet projected light fields In *Proceedings of the 1999 conference on Graphics interface '99*, pp.107-114, 1999.
- [9] D.Le Gall, A.Tabatabai, Subband Coding of Digital Images Using Symmetric Short Kernel Filters and Arithmetic Coding Techniques In *Proceedings of the ICASSP 1988*, pp. 761-765.
- [10] S3TC DirectX 6.0 Standard Texture Compression *S3 Inc*, 1998.
- [11] F.F.Samavati, R.H.Bartels, Local Filters of B-spline Wavelets In *Proceedings of International Workshop on Biometric Technologies 2004*, pp.105-110.
- [12] J.Schneider, R.Westermann, Compression Domain Volume Rendering In *Proceedings of the 14th IEEE Visualization 2003*, pp.39-47.
- [13] J.M.Shapiro, Embedded Image Coding Using Zerotrees of Wavelet Coefficients In *IEEE Transactions on Signal Processing, Vol.41 No.12*, 1993, pp.3445-3462
- [14] C.C.Tanner, C.J.Migdal, M.T.Jones, The Clipmap: A Virtual Mipmap In *Proceedings of SIGGRAPH 98*, pp.151-158.
- [15] D.S.Taubman, M.W.Marcellin, JPEG2000: Image Compression Fundamentals, Standards and Practice *Kluwer Academic Publishers*, 2001.
- [16] J.Wang, T.-T.Wong, P.-A.Heng, and C.-S.Leung, Discrete wavelet transform on GPU In *Proceedings of ACM Workshop on General Purpose Computing on Graphic Processors*, pp. C-41, 2004.

A system for panoramic navigation inside a 3D environment

Polceanu Mihai
"Ovidius" University, Faculty of
Mathematics and Computer
Science
Bd. Mamaia nr.124
900527, Constanta, Romania
polceanum@gmail.com

Popovici Alexandru
"Mircea cel Batran" National
College
Bd. Stefan cel Mare nr.6
900726, Constanta, Romania
popovici.alexandru@gmail.com

Popovici Dorin-Mircea
"Ovidius" University, Faculty of
Mathematics and Computer
Science
Bd. Mamaia nr.124
900527, Constanta, Romania
dmpopovici@gmail.com

ABSTRACT

This paper presents a physical user interface intended to help the user (or multiple simultaneous users) to achieve an intuitive movement inside a 3D environment without using common interaction devices such as mouse or keyboard, while stressing the aspect of reducing financial investments. After exposing an analysis of current solutions and implementations of related topics, we argue our implementation and give detailed aspects of hardware and software architecture of the system, as well as a comprehensive efficiency study and explore the use cases with people with motor impairment. As future work, we intend to extend the usability of the system and release it under the GNU General Public License (GPL) for free use and further development by other parties.

Keywords

Virtual Reality, 3D navigation, user interface for physically disabled individuals.

1. INTRODUCTION

Regardless of the quality of simulated 3D worlds, people are still conscious of the barrier between them and what they see; this is because they only benefit from a keyboard, mouse or other common input devices. This paper presents a system through which we try to whittle this barrier and give users a natural interaction tool which they can intuitively use to navigate at will with natural body movements. The concept was also designed to be easily configurable "at home" and to be a low-cost solution. Structure, efficiency and the possible uses as an enhancement for physically disabled individuals are explored in this paper.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright UNION Agency – Science Press, Plzen, Czech Republic.

2. MOTIVATION

The system discussed in this paper is an improvement of the IIUBAR setup (acronym for Interactive Informative Unit Based on Augmented Reality) described in [Pop08]. It refines the hardware setup used by its previous version, and uses Virtual Reality instead of Augmented Reality; this makes its uses slightly different: user immersion instead of fixed-point informational unit.

The goal of this paper is to extend the possible use cases of this system architecture and to exhibit its advantages and drawbacks compared to specialized hardware. This system has been developed with respect to product quality and reduced financial investments.

3. RELATED WORK

Throughout the history of user interfaces there have been many metaphors for addressing visualization and navigation inside three dimensional virtual worlds. Because of the impossibility of fully recreating a three dimensional space on a two dimensional display system, the ideas that were developed in this domain can be split into two main categories (Figure 1): fixed display metaphors with interaction devices such as mouse, keyboard, space mouse, etc, and mobile spatially aware systems.

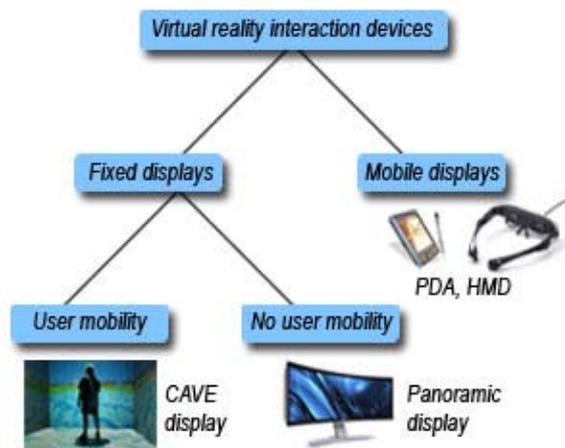


Figure 1. Interaction devices categories

The oldest member of the fixed displays is represented by the classic desktop environment on which 3D applications can run and the user can “move” inside them by pressing keys or by dragging the mouse. Display sizes and image quality have grown in direct dependence with technology, and today a wide variety is available for purchase; this technological advancement came to enhance the experience of 3D navigation by enlarging the user’s field of view and therefore contributing to a higher level of immersion inside the virtual environment. Famous examples of this technique include connecting multiple screens to form a panoramic display around the user, a similar metaphor that uses projections which materialized as the CAVE system [Cru93], wide panoramic screens [Bau05], spherical display systems, and other means of extending the field of view as much as possible. Unfortunately, as marvelous as they prove to be, extra technology comes with extra costs, and may not be easily accessible for the usual user and even for educational institutions because of the lack of funds. The category of fixed display devices can be further split into two scenarios depending on the user’s mobility and the lack of it; the CAVE system allows users to move freely in a designated space and reacts to his/her movements. The latter, although it allows the implication of multiple users, cannot give each user a personalized viewpoint but only a group-oriented interaction. So far we’ve identified some key advantages of this category: extended field of view, multiple users and high level of immersion. The inversion of the fixed display concept takes us to explore devices that use their own spatial position to transmit visual data to the user. These devices range from personal digital assistants (PDA) to head mounted displays (HMD) which come in a wide variety of designs. This category also consists of hardware that is specialized for performing precise

tasks which, through the prism of virtual reality, consist of binding the navigation to the user’s view point; this way the user can specify the desired focus in the environment either by pointing the device or using a pen also known as the peephole display concept [Yee03], or in the case of the HMD by tilting the head in direction. An example from this category is represented by the use of a palm computer for interacting with a virtual environment [Pig08]. This feature gives the great advantage of user mobility, being only constrained by the physical space available to move into. There are several drawbacks to these methods: hand-held devices can only display a small portion of the visualized virtual world and the HMD type devices should allow the user to be aware of the surrounding real environment to prevent accidental collisions while moving around. The latter is achievable through augmented reality but another impediment arises due to the low video resolution relative to natural sight; this can be overcome by using see-through lens technology, but again we stumble into cost issues. From this category, we can derive two new advantages: mobility and user-bound viewpoint.

After this analysis we propose the following question: is it possible to achieve similar performances with relatively basic cost-wise accessible hardware? One of the most used input devices used today is video. In 2004, Microsoft reckoned more than 18.5 million webcam users only with instant messaging applications [Web09a], and the number is ever growing. Webcams have become an accessible and necessary possession for internet users, and they can be used with a wide range of applications and operating systems. Using webcams as input devices is cost-efficient, but require a greater effort to create software capable of interpreting the input data; fortunately, open-source frameworks are freely available which do most of the work. Navigating inside a three dimensional environment requires linear view-point movement and the ability to rotate. In this paper we propose a model through which we try to absorb the mentioned advantages using low-cost equipment. Our solution consists of a fixed but rotatable display that interprets the user’s turning movements and level of approach for navigation. To navigate inside a three dimensional environment the basic requirements are speed and rotation, which can be achieved by determining the angle of rotation relative to a point of reference, and using the distance to the face of the user as directional speed input. In the following sections we describe the architecture of this system, its use cases and efficiency evaluation results.

4. SYSTEM ARCHITECTURE AND USE CASES

Considering the available technology previously analyzed, we decided that the best method of reducing the cost of the system is to recreate the functionalities of the specialized hardware through the means of software. This way the equipment requirements can be reduced, but we must emphasize the used software solution. Another aspect of the system is that any user with minimum knowledge on software installation and basic experience with material carving (for the special support table) can create a replica of the system at home without significant investments; we like to believe this can be a motivational factor through the satisfaction of building it. In the following subsections we present the aspects of the hardware and software used, followed by the exhibition of an official use of the system within an ongoing project, and finally we describe the multiuser support and the possibility of creating networks with multiple implementations of the system.

Hardware architecture

The system is composed of two simple webcams connected to a laptop placed on a rotatable support like illustrated in Figure 2. The purpose of the tripod table is to allow the hidden camera to look down on a cardboard marker which is needed by the software to extract rotation coordinates. When in use, the top camera is always directed toward the user(s). Each component is adjusted to fit the others, thus making the system stable for user interaction.

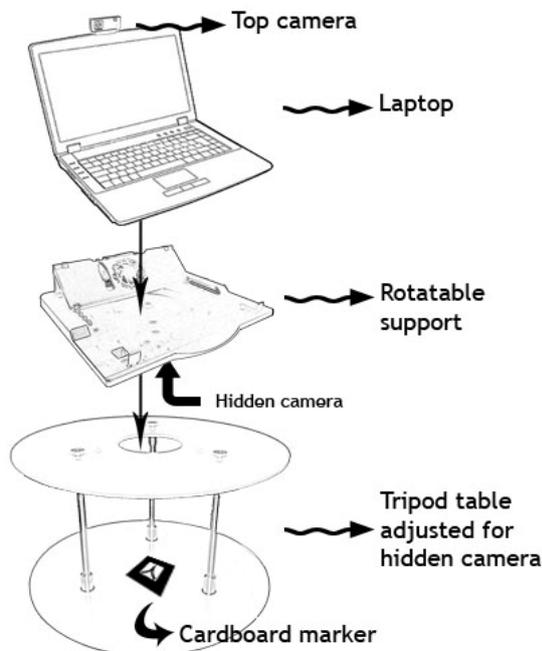


Figure 2. System schematic

Our implementation uses a laptop, a rotating laptop stand, two low-cost webcams and a hand-made tripod table, although any similar hardware can be properly used for which drivers, if required, are compatible with the utilized operating system.

Software

The application is developed in C++ and runs on UNIX (our implementation uses Ubuntu [Web09b]). It combines three open-source frameworks as follows:

- AReVi (Atelier de Realite Virtuelle) [Web09c] for virtual environment representation. AReVi is a powerful agent-based library that provides services for multi agent systems and 3D graphics.
- OpenCV (Open Computer Vision library) [Web09d] for face detection support. Visual algorithms play an important role in deciding how far away a user is from a camera. To advance or retreat in the virtual environment we chose to use the distance of the user relative to the top camera. Achieving this effect resides in detecting the user's face; the difference in face size from different positions give away the distance, i.e. if the image of the face appears larger implies that the user is closer to the camera and vice-versa. A basic threshold based noise reduction algorithm is used to prevent the navigation speed from trebling.
- ARToolkit (Augmented Reality Toolkit) [Web09e] for viewpoint orientation. Similarly to OpenCV, ARToolkit uses image processing algorithms to extract position and rotation information from a physical cardboard marker and returns a rotation matrix. For rotation in the horizontal plane, we only need one rotation angle (around the z axis); we can calculate this angle with Equation 1, where $R = R_x * R_y * R_z$ (a 3 by 3 matrix), and R_x , R_y and R_z are defined in [Fol93]. This way, the resulting angle is applied to the viewpoint inside the 3D environment and the effect of rotation is achieved.

$$|\gamma| = \text{acos}\left(\frac{R(1,1)}{\cos(\text{asin}(R(1,3)))}\right)$$

Equation 1. Rotation angle calculation

The main application works by reading data from OpenCV and ARToolkit through a local shared memory mechanism (Figure 3); this concept is not new, but makes individual builds independent from each other, and enhances simplicity of the code and extensibility of the software.

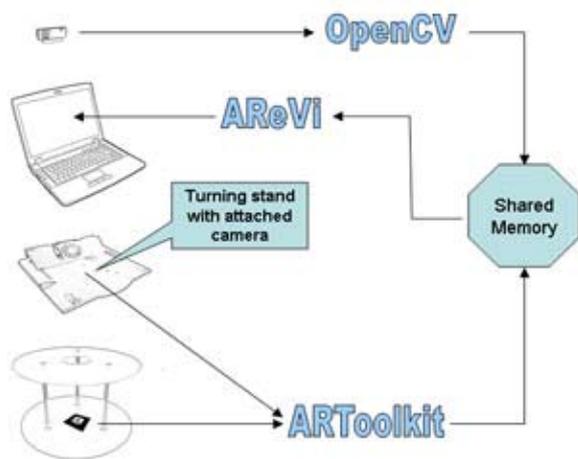


Figure 3. Communication between components

Demonstrating the system's features

For testing the system we developed two applications: a game entitled “Crystal Island” in which players enroll to solve quests by collecting crystals of different colors and bringing them to the totems which required them, and an interactive lesson about the solar system entitled “UFO Driver” in which the user controls a flying saucer and navigates through our solar system to discover the planets, our sun, and the asteroid belt (Figure 4).

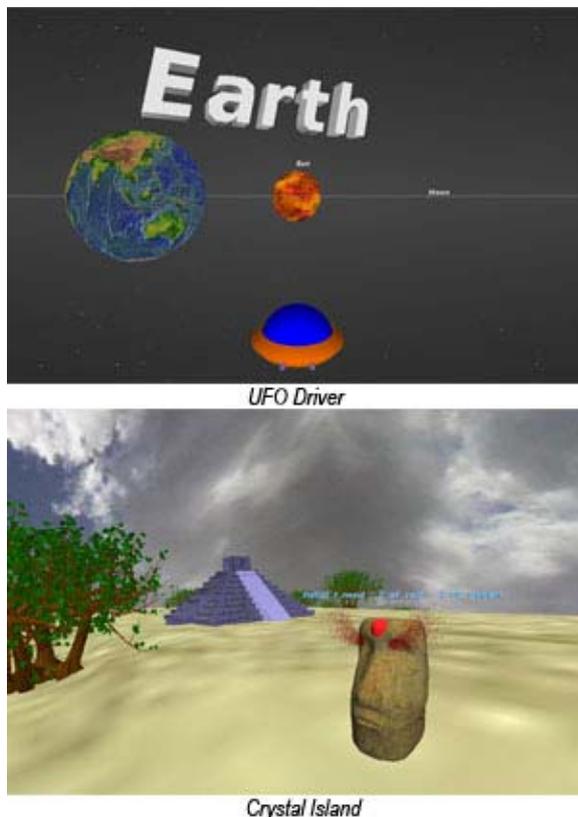


Figure 4. Applications of the system

This lesson about the solar system was created with respect to the real dimensions of the planets, to give scholars a feel of the great distances between celestial bodies and to help them grasp this information in an entertaining and intuitive way.

Virtual tour of archeological sites

Today, a lot of emphasis is put on virtually reconstructing lost cultures from different times in the history of mankind and even before. History lessons have evolved into interactive game-like experiences in which users can explore 3D replicas of ancient artifacts, buildings and even people. The immersion of the user in these environments can give visual, auditory or haptic feedback which helps to better grasp the details that were specific to a certain time in the past.

In this sense, apart from the games that we developed to evaluate the system, we also integrated the system in the TOMIS project which aims to virtually reconstruct the ancient Roman Edifice with Mosaic from Constanta, Romania, through designing, implementing, experimenting and demonstrating an interactive and collaborative multi-sensorial system based on VR/AR technologies. Although not yet complete, the reconstruction of the site has been integrated with the system and allows users to walk through the edifice like it was between the years 46 AD and 610 AD (Figure 5). The starting point of the virtual tour corresponds to the system's location so users can grasp the feel of orientation, and presence in the Tomis colony during the Roman period. The reconstruction has a hypothetical approach as the archeological information from the colony is not entirely complete. This enhancement to the project aims at promoting culture and tourism in the region.



Figure 5. Using the system in the Roman Edifice with Mosaic from Constanta, Romania

Multiuser interactivity

Having more than one system can be used to create a network of interactive “3D browsers” through which users can compete in games or explore virtual sites. The networks can be either local, using a wireless router for increased mobility, or distributed over the internet, or both local and wide area networks connected through a server located on one of the machines.

Along with the network possibility, the system supports multiuser on the same machine (Figure 6). In this situation, the control of the unit or avatar is distributed to each user participating in the interaction. To allow other people to observe without disturbing the users, adjustments have been made to this feature so that people who are more than approximately 1.5 meters away from the system (the limit of physically maneuvering the device) cannot influence the acceleration.

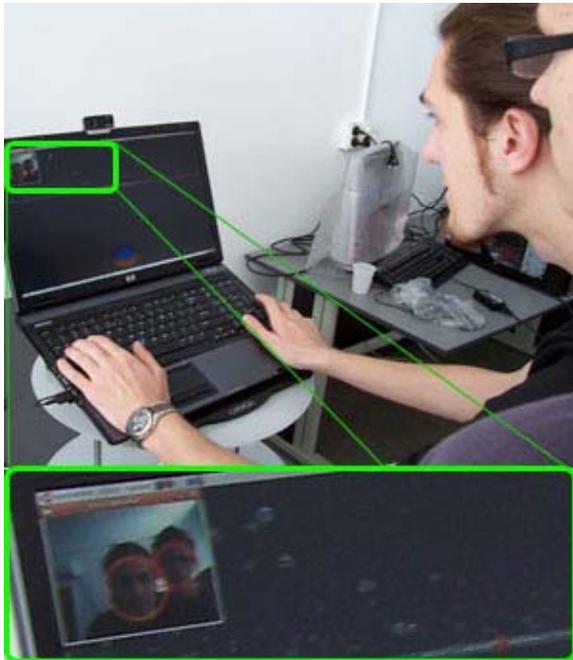


Figure 6. Illustration of multi-user support: average of the face positions is computed (magnified at bottom of picture)

When more users engage in interaction with the system, an average of the users' positions is made and requires them to work in a team to achieve the desired result. Hence to move forward in the 3D environment all users must lean forward and if one of them leans backwards the average acceleration would decrease and cause inefficient movement.

5. EFFICIENCY STUDY

With the occasion of the “Laval Virtual” contest that took place in Laval, France in 2009 [Web10] where we participated with the system under the name of

“Navoramique” (which stands for Navigation Panoramique), we took the opportunity to test its efficiency with the help of the people who tried it (Figure 7). In this section we discuss the results of the survey, and some of the suggestions received from our users.



Figure 7. An user testing the system at Laval Virtual

A survey was prepared which contained six questions about the system, as follows:

- Q1: “I think Navoramique is intuitive and easy to use.”
- Q2: “Using Navoramique is more appealing than using a keyboard and a mouse to navigate.”
- Q3: “I easily learnt how to control my movements with Navoramique.”
- Q4: “I think ‘UFO Driver’ is an interesting lesson about the Solar System.”
- Q5: “I found ‘Crystal Island’ to be an attractive quest game.”
- Q6: “I would like to have a version of this system at home.”

The possible answer choices were: “Totally disagree”, “Disagree”, “Neutral”, “Agree” and “Totally agree”. We made the surveys available both on paper and online within an application, but most users preferred the paper forms because they were faster to fill in, and more persons could submit them simultaneously (we only had one computer available for this task). Figure 8 shows the comparative results for this survey on a number of 112 users of all ages, and different nationalities.

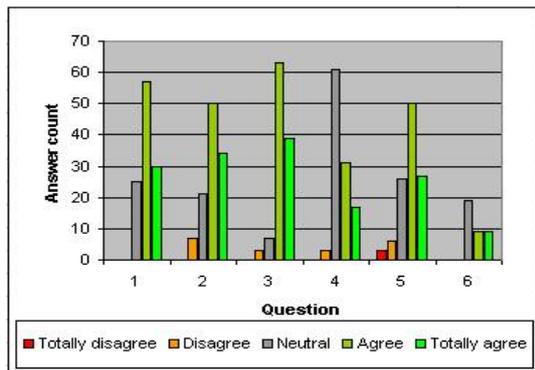


Figure 8. Answers for the evaluation survey

From the results we learnt that an average of 81% of the users gave a positive answer for the first three form items while 18 users expressed the wish for their own version of the system. As not all users played the two available games, the positive answer average was 89.91% of the people who did try the games (Q4, Q5). There were 59.5% people who tried the games from the total who submitted a survey.

As an additional note, in the case of “Crystal Island”, the player’s performance was measured by keeping the score; the score was calculated with respect to the time that the user needed to complete the tasks of the game. The users also had visual feedback of their current score and about their position in the top scores and after completing the game. The observed results were that while some users managed to achieve a top score from the first try, the others learnt quickly and were able to improve their scores after playing several times.

Some of the users also submitted comments about the system which helped us identify some weak and strong points.

One user pointed out that the table should be adjustable for each person’s height. Unfortunately we underestimated the possibility of the difference in height; most tall users had problems with the system because the laptop monitor did not permit a very wide angle of inclination, and therefore they could not adopt a comfortable position while testing. The best results appeared when the line between the user’s face and the top camera was close to the horizontal.

Another drawback would be the energy necessary to physically rotate around the table. We also encountered issues with the lighting in the room which caused problems with the face detection when the user stood between the camera and the light source.

The idea was better accepted by children who enjoyed the games and the fact that they had to move around to navigate, and by people who were not comfortable with navigating with the mouse and

keyboard. One user mentioned that “as a learning tool it would be enhanced if students have to search for information”; this underlines the factor of motivation in learning.

To conclude the statistics, the system can benefit from small comfort-related improvements, and can serve as an efficient interactive learning tool for primary school pupils and for students. It can also be used as a navigation tool, complementary to the standard input devices.

6. AIDING PERSONS WITH MOTOR DISABILITY

While at the Laval exposition, we were most moved when two persons in wheelchairs asked us if they can try the system (Figure 9). We discussed the possibility of adapting the mechanism to minimize the effort needed for navigation, and one of them suggested that the facial recognition could be also used for turning, so they can only use the head movements. Another solution which we discussed was to allow left-right navigation without having to make a whole turn, but only to slightly rotate the stand; this would give similar feedback effect and would be a lot more convenient in this case. As noted in [Hol06], the weaknesses of one modality are offset by the strengths of another, and by modality the means of interaction is implied. Slight changes to the system like the ones previously mentioned can substitute for the impairment of lower body movements.



Figure 9. Users with motor disability

Although we had not foreseen this use case, we were deeply moved by the fact that their impression was a positive one, and we hope we will collaborate with the asylum in Laval to share the technology.

7. RESULTS

The main aspect which we tried to demonstrate is that using innovative ideas together with common and accessible hardware and software resources, one can achieve efficient low-cost solutions to enhance human-computer interaction either for gaming, learning or aiding persons with disabilities. The presented system has been used for educational, entertainment and aiding purposes. It also represents a method for museums to exhibit a new, modern point of view to the visitors.

8. ACKNOWLEDGMENTS

The present system has been implemented in the Laboratory of Virtual and Augmented Reality Research (<http://www.univ-ovidius.ro/cerva/>) within the Faculty of Mathematics and Informatics of the OVIDIUS University in Constanta, Romania.

This work is supported by the TOMIS project, no: 11-041/2007, by the National Centre of Programs Management, PNCDI-2 - Partnerships program.

Thanks to everyone who submitted the system evaluation surveys and shared their impressions and suggestions with us.

9. REFERENCES

- [Bau05] Baudisch, P., Tan, D., Steedly, D., Rudolph, E., Uyttendaele, M., Pal, C., Szeliski, R., Panoramic viewfinder: providing a real-time preview to help users avoid flaws in panoramic pictures, Proceedings of the 19th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction: citizens online: considerations for today and the future, Canberra, Australia, November 21-25, ISBN 1-59593-222-4, 2005.
- [Cru93] Cruz-Neira, C., Sandin, D.J., DeFanti, T.A., Surround-screen projection-based virtual reality: the design and implementation of the CAVE, Proceedings of the 20th annual conference on Computer graphics and interactive techniques, ISBN 0-89791-601-8, 1993.
- [Fol93] J.Foley, A.van Dam, S.Feiner, J.Hughes. Computer Graphics: Principles and Practice. Addison-Wesley Co., ISBN 0-201-12110-7, 1993.
- [Hol06] Holzinger, A., Nischelwitzer, A.K., People with Motor and Mobility Impairment: Innovative Multimodal Interfaces to Wheelchairs, Computers Helping People with Special Needs, Springer Berlin / Heidelberg, ISBN 978-3-540-36020-9, 2006.
- [Pig08] Pignatelli, A., Farella, E., Brevi, F., Benini, L., Gaiani, M., On the use of a palm computer for design review interaction in a virtual room, The 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2008), Full Paper Proceedings, ISBN 978-80-86943-16-9, 2008
- [Pop08] Popovici, D.M., Polceanu, M., Interactive Informative Unit Based on Augmented Reality Technology, In Proceedings of The 3rd International Conference on Virtual learning (ICVL2008), Bucharest Univ. Press, ISSN 1844 - 8933, 2008.
- [Web09a] Website of BBC (article from 2004): <http://news.bbc.co.uk/2/hi/technology/3833831.stm> (checked Oct 2009).
- [Web09b] Website of Ubuntu operating system: <http://www.ubuntu.com/> (checked Oct 2009).
- [Web09c] Website of AReVi framework: <http://svn.cerv.fr/trac/AReVi> (checked Oct 2009).
- [Web09d] Website for OpenCV library containing links to documentation and source: <http://opencv.willowgarage.com/wiki/> (checked Oct 2009).
- [Web09e] Website of ARToolkit library: <http://www.hitl.washington.edu/artoolkit/> (checked Oct 2009).
- [Web10] Official website of the Laval Virtual event: <http://www.laval-virtual.org> (checked Jan 2010).
- [Yee03] Yee, K.-P, Peephole displays: Pen interaction on spatially aware handheld computers, in Proceedings of CHI 2003, ACM Press, 2003.

Active Shape Models on adaptively refined mouth emphasizing color images

Axel Panning
University of Magdeburg,
Germany
Axel.Panning@ovgu.de

Ayoub Al-Hamadi
University of Magdeburg,
Germany
Ayoub.Al-Hamadi@ovgu.de

Bernd Michaelis
University of Magdeburg,
Germany
Bernd.Michaelis@ovgu.de

Abstract

In this paper, we propose a hybrid method for lip segmentation based on normalized green-color histogram splitting and Active Shape Models (ASM). A new adaptive method for histogram splitting is applied in two steps. First, after defining a region of interest for mouth segmentation, a rough adaptive threshold selects a histogram region assuring that all pixels in that region are skin pixels. Second, based on these pixels, we build a Gaussian model which represents the skin pixels distribution and is used to obtain a refined optimal threshold for lip pixel classification. This process is used to refine the normalized green channel image for the elimination of inner distortions and gradients inside the lip region, which can misguide active contours (i.e. ASM) in the last step of the hybrid segmentation process. In the results, we present that the proposed method performed better than conventional ASM on unrefined color enhanced images or pure color-histogram based mouth segmentation.

Keywords: Feature extraction, Segmentation, Image processing, Application.

1 INTRODUCTION

The segmentation of mouth and lips is a fundamental problem in facial image analysis and is important for various applications. It can be utilized for lip reading, supporting speech recognition or expression analysis (i.e. facial expression, estimation of emotional state, pain recognition). Each application has its own limitation concerning speed, accuracy and robustness. The requirements for facial expression recognition can be very different depending on application context.

Often initially a color transformation is performed to exploit the different chromaticity of lips from skin. Basically, the segmentation approaches can be classified into two groups. The first group, Histogram based approaches, is a consequent continuation of the initial color transformation. The mouth region of interest (ROI) is binarized into lip and non-lip pixels, where non-lip pixels are mainly skin pixels. The crucial point in histogram based algorithms is the estimation of that particular threshold. A very easy approach, mostly used for first rough mouth segmentation is a fixed threshold, found by statistical average of numerous samples [8]. A more adaptive approach sets up a watershed like rule, which defines 15 percent of the darkest pixels in their color transformed mouth ROI as lip pixels[9]. Other

works [5] assume a certain topology in the histogram. Following this idea they seek for a local minimum between a lip and a skin heap in the histogram and define the threshold here.

The second group, is focusing on detection of lip edges in the mouth ROI [4, 2]. They apply Active Contour Models (e.g. [4]) or deformable templates [2] to the mouth's ROI. Some approaches [7, 1] stabilize their Active Contours using support tracking points. The general assumption of edge based algorithm is, that the lips generate prominent edges at the skin-lip crossing. In monochrome images only a simple shadow casting can already cause serious problems. A hybrid of color and edge information is the usage of color images and their mouth-highlighting transformed representation (e.g as used in [4, 2]). This can suppress some issues like shadow casting. But still there is no guarantee the edges of the lips create significant edges here. This might happen for many cases. For people having Asian skin tone for example this rule holds true. However, for European/Caucasian this rule does not hold for all cases anymore, since the transition from skin to lip pixels does not form rough edges here for all subjects and conditions. Another usage of color and edge information is to align deformable templates or active contours using an energy minimization function, which refers to edge information and average color intensity inside of the template (or contour) as proposed in [2, 3].

In the proposed approach the advantages of both classes of algorithms (pure color based, and shape/edge based) shall be combined in another way. We chose *Active Shape Models (ASM)*, introduced by Tim Cootes [6], as representative for the edge/shape based algorithms. The idea is, that a color based approach can contribute to an

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

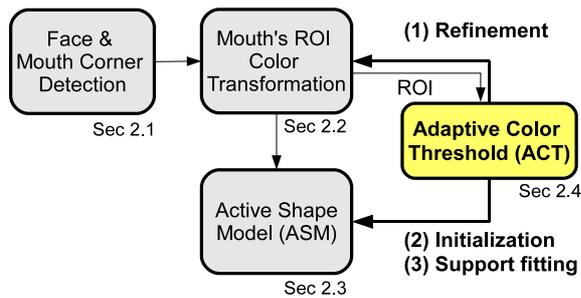


Figure 1: Flow chart of the proposed algorithm

edge and model based approach (ASM here) to improve its performance more than a simple prior transformation of color space, as most hybrid approaches do so far. In this context we propose a novel adaptive method for color based mouth segmentation. The rest of the paper is organized as follows. Section 2 describes the idea of combining histogram based thresholding and shape based extraction for mouth segmentation. The results of the proposed methods are presented in section 3. Section 4 gives a short summary and outlook.

2 MOUTH SEGMENTATION

The process chain is shown in Figure 1. All successive steps will be described in the following sub sections.

2.1 Locating Face and Mouth ROI

Object detection in image processing is always the search for a delimited area in which the targeted pattern is fitting. A general solution for this task has been developed by Viola and Jones [13]. They developed an algorithm, where a cascade of weak Haar-like features (see Fig. 2) is utilized to model image objects appearance. A Haar-like feature describes the difference of pixel intensities within similar sized sub regions of one rectangular region in an image. The most advantage, compared to other feature descriptors, is the fact, that they can be computed very fast using integral images. Once calculated, an integral image can provide the average intensity of any rectangular region of any size by one addition and two subtraction operations. This property is very important in context of applications, where speed issues are relevant. Another acceleration is provided by the cascaded structure of the classifier. During the search process not the whole classifier needs to be used at each potential position. Once one cascade step fails all successive cascade steps can be discarded, the current target region can be rejected and the search continues in the next potential region.

An implementation of the algorithm as well as face detection models can be found in the OpenCV *c/c++* library, which are widely used. Also in this work, the available face models were used for face detection. Fur-

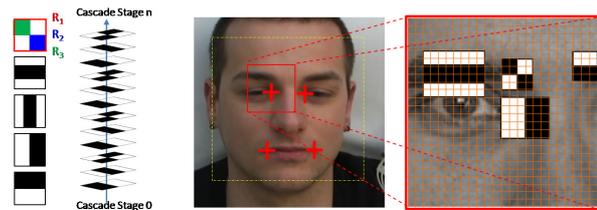


Figure 2: Left: base features for the cascade classifier and their cascading. Middle: the face region, a result of the face detector is the search area for single facial features. Right: single weak features in their local arrangement forming a strong classifier.

ther two models for detection of mouth corners in facial regions were trained in order to define a region of interest (ROI) for further mouth segmentation processing. Database for the training was the *FGnet Database* from the Technical University of Munich [15]. To train the classifier 400 positive and negative samples were chosen. Positive samples were sub images where mouth corners were directly in center of sub images. Negative samples were chosen from randomly selected sub images where the mouth corner were not centred. [12].

2.2 Color Transformation

In common a color transformation is chosen converting the RGB from \mathbb{R}^3 to \mathbb{R}^1 exploiting the difference of lip and skin pixel colorness. Using the ground truth of our database, a comparative statistic was made to analyze their ability to separate lip from non-lip pixels, based on color information only. In result the green channel from normalized *rg* was superior to all others, which is defined by $nG = R/(R + G + B)$. We will refer to this in further context as *nG* color channel. The worst results were achieved by the *YCbCr* based color transformations. Qualitative results of this prior study are given in table 1. The percentage is relative to the histogram of the complete ROI and outlines the false classified pixels using an optimal, FPR minimizing threshold found by the ground truth. Under advantageous conditions lips and skin pixel form two well noticeable bell curves in the histogram with a noticeable local minimum in between (Fig. 3 left). This can motivate approaches like [5], searching for this minor local minimum. However, these optimal cases cannot be assumed in general. The general structure of the histogram can vary in different scenarios (Fig. 3). More complex situations can create numerous minor local minima instead of only one major minimum. In other cases the smaller bell curve related to the lip pixels can be directly attached to the larger bell which represents the skin pixels without producing any local minimum (Fig. 3 middle). This multiple behavior can be observed independently from the chosen color transformation. Intersection of skin and lip color in the mouth ROI with respect to differen

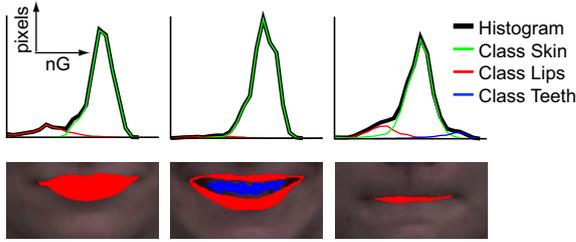


Figure 3: (Top Row): Histograms. The real histogram is the fat black line. The colored lines represent skin/teeth (no-lip) and lips. These information are only gathered by ground truth here and are a-priori unknown in application case, (Bottom Row): ground truth. The three samples show three states: normal mouth state (left), open mouth with appearing teeth (middle), pressed lips with almost none lip pixels left (bottom).

Used in	Transf.	With teeth	No Teeth
[2]	$u(Luv)$	4.61 %	3.16%
[10]	G/B	5.97 %	2.59%
[3]	G/R	2.30 %	1.45%
[4]	Cr^2	11.75 %	10.97%
[4]	Cr/Cb	13.08 %	11.16%
[9]	$R/(R+G)$	2.36 %	1.48%
not found	nG	0.09%	0.38%

Table 1: Intersection for different color transformations

color transformations was analyzed with and without teeth appearance. However, the appearance of the teeth had just a minor impact to the separability (see Table 1) using a histogram threshold.

2.3 Active Shape Model

Active Shape Models (ASM) combine assumptions about specific shape behaviour and image signal response at the model points of the shape. Base of the ASM is a set of model points forming one or more contours, which are stored in the mean shape \bar{m} . The modeled shape variance is stored in a vector matrix S . A weighting vector \vec{w} applies the different shape variations to the mean shape. The fitting process alternates two steps until convergence:

```

(0) initialize mean shape
    near the object.
do
{
  (1) search for special image
      signal near model points
      (gradients, pattern)
  (2) find a shape, based on  $S$ ,
      fitting best to the (image
      signal based) model points,
      found in step 1.
}
until (convergence)

```

Step (2) in the algorithm results in the final shape m , by applying the following equation

$$m = T(\bar{m} + S\vec{w}) \quad (1)$$

where \bar{m} is the mean shape of the mouth (a vector containing all x - and y -coordinates of the shape points one below the other), S is the matrix of column wise aligned shape variation vectors, \vec{w} is the vector, containing the weights for each shape variation of S , and T is a affine transformation including x - and y -translation, scaling and rotation. The unknown \vec{w} and T are estimated by solving

$$\delta = S\vec{w} \quad (2)$$

with

$$\delta = T^{-1}(m^*) - \bar{m} \quad (3)$$

where m^* are the associated landmark points based on any measurement in the image data. The estimation of T is described in [6].

The used shape model for the mouth consists of 22 contour points (see Fig. 4). Only the outline of the mouth will be addressed here. The mean shape \bar{m} was found by average of 20 samples. Classical ASM as introduced in [6] define the shape variation matrix S by calculating eigenvectors from the covariance matrix based on size normalized samples. This method has some drawbacks. It demands very exactly and equidistantly picked landmarks for all samples. Further a few number of samples with less variations can cause wrong mutual dependencies. To resolve semantically and technically clean modes, the shape modifier vectors for S were created manually with expert knowledge. Five different modes were defined (see Fig. 4).

The edge fitting has two main parameters. a) the method of edge detection and b) the range of edge detection. Cootes [6] suggests statistical patterns here. In case of mouth shape this results, more or less, in a kind of gradient detection. The manifold of profile structures is considerable. Only the lineup of all 57

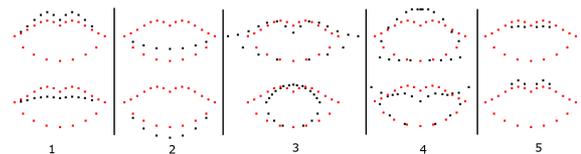


Figure 4: The five mouth modi and their behaviour. The red dots show the mean shape \bar{m} . The two stacked images of a mode show the impact of negative (lower row) or positive (upper row) weighting. Each mode represents one column of S .

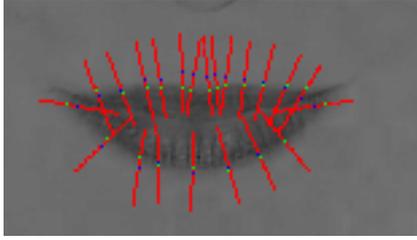


Figure 5: Red lines are the profiles taken during the edge fitting process. Blue dots mark the starting points from the last model-aligned instance respective the initial mouth model. The green dots represent that point in the profile, where the largest gradient is found.

samples does not show some special pattern, different from gradients, which could be fitted in a definite pattern. For edge fitting a profile P_i of the normal to the shape boundary is collected for each model point. The normals are defined by the neighbour points in the contour (Fig. 5). The profile contains the information of interpolated sub-pixels along the profile line. A simple concatenation of full pixels along a Bresenham based line was distorting. Profiles always are collected with a width of three pixels, where outer pixels got lower weight than inner pixels. As feature for model point detection gradient function was used, which is defined as follows:

$$p_i^* = \operatorname{argmax}_t \left(\sum_{j=0}^t p_{i,j} - \sum_{j=t+1}^k p_{i,j} \right) \quad (4)$$

where p_i^* is the found point to profile P_i with maximum gradient. The length of the profiles is an important parameter here. In the current version a length of 30% of mouth width is used (in average 30 pixels for the used samples). All Points p_i^* build the next instance of m^* .

2.4 Adaptive Lip Pixel Enhancement

Color enhancement for application of Active Contours in general (e.g. Snakes, Active Shape Models etc.) has been introduced already in previous works. But the lips of the subjects not always have uniform color. So inside of the lip itself distortions (causing gradients) can occur, which are more prominent than the outer (targeted) edge. These gradients can attract the contours falsely and thus misguide the whole active contour. To avoid this an in-between-step is suggested, which is performed after color transformation but before the application of ASM. The idea of the Lip-Pixel-Refinement is to flatten the lip pixels, in order to weaken the edges inside the lips. Therefore an adaptive histogram based algorithm (which is a considerable segmentation method itself already) will determine a threshold to define the lip pixels class in the ROI. This

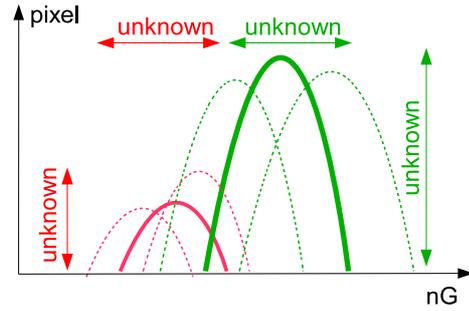


Figure 6: Model Assumption. There are more skin than lip pixels. Lip pixels have lower intensity than skin pixels. Unknown is their exact centering, scattering, distribution and intersection.

pre-segmentation is used to equalize and flatten distortions inside the (so far known) lip segment.

2.4.1 ACT: Adaptive Color Threshold

Basically the ROI contains two classes of pixels: *Lip-Pixels* and *Non-Lip-Pixels*. A general description of a statistically based model for lip or skin pixel distributions is not reliable, due large variance among subjects and illumination conditions. Thus an adaptive histogram based approach was developed to separate skin from lip pixels. As color transformation for lip pixel enhancement the green channel of the rg has been chosen (in further context referred as nG). The approach makes following assumptions:

- Skin pixels are Gaussian distributed in the histogram
- Lip pixels have lower intensities than Skin pixels (in nG)

Skin pixels and lip pixels can be mixed in the histogram (see Fig. 6), thus there is not always a perfect threshold to separate skin pixels by color information only. The algorithm prefers wrong positive skin pixels rather than wrong positive lip pixels. Latter case produces a kind of flow out which causes more damage to the segmentation than lip pixels which are classified as skin pixels. Wrong positive lip pixels are caused by a threshold greater than the optimum, with respect to the chosen nG transformation. With increasing intensity of nG also the probability of adding a high amount of pixels to the lip pixel class in one single step is increasing (see Fig. 7). Knowledge about the skin pixel distribution can provide a threshold that most likely avoids wrong positive lip pixels (see Fig. 6). The target threshold should be the foot-point of the skin pixel distribution, in order to avoid wrong positive lip pixels.

Basically a Gaussian distribution is estimated by a set of samples, calculating σ and μ , which are represented by single skin pixels here. A-Priori it's unknow which

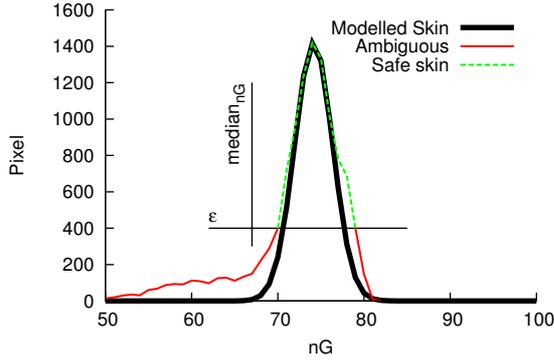


Figure 7: Based on initial guess the upper part of the histogram is defined as *safe*. This part is base for the approximation of skin pixel distribution

pixels belong to skin and to lip class. The idea is to select a part of the skin pixels, which can be assumed to be 'safely' part of skin pixel class. The condition is made simply by its number of occurrence in histogram. In other words, it is an 'initial guess' avoiding participation of lip pixels.

Let $h(x)$ be the value to the x^{th} slot of the and h_{max} be the global maximum of the histogram. To calculate σ_s and μ_s in first step σ^* is calculated for all pixels satisfying following condition:

$$h(x) > \varepsilon \quad (5)$$

$$x > median_{nG} \quad (6)$$

Condition in Eq.(5) represents an expected maximum ratio of mouth size to ROI, where ε is adjusted using a parameter α (¹) with $\varepsilon = h_{max}/\alpha$. Additionally a median constraint, relative to all occurring intensity values in nG , was introduced to avoid disturbances from peaks of very low intensities in the histogram. Both very conservative conditions formulate a reasonable 'initial guess'. However, only the median constraint itself is a decent classification, which can compete with classic watershed method (See 3.2, Fig. 12).

The Gaussian distribution has scatters less than the original skin pixel distribution. However there is a correlation between α and the ratio of σ^*/σ_s . The unknown σ_s can be approximated following equation

$$\sigma_s = \sigma^* \cdot \left(1 + \frac{1}{\alpha}\right) \quad (7)$$

The larger α , the smaller the part left out from the histogram. This will raise the quality of approximation. If

¹ In the current experiments a value of 3 was chosen. Basically values between 2 and 4 gave good results.

$\alpha \rightarrow \infty$ the whole histogram is used. But in case of application the lower intensity edge parts of skin pixel distribution is mixed with lip pixels. Therefore the choice of α is a trade off between approximation accuracy and risk to include lip pixels to the initial guess. Estimation of the threshold is done using the cumulative distribution function of $\mathcal{N}(\sigma_s, \mu_s)$ applying a low border ($\lambda = 0.01$).

$$th = \underset{x \in \mathbb{R}}{\operatorname{argmax}} (\lambda < F(x)) \quad (8)$$

with

$$F(x) = P(X \leq x) \quad (9)$$

where x is the intensity value of possible thresholds in nG .

2.4.2 Combining ACT and Active Shape Models

The result from section 2.4 contribute in three ways to the problem of ASM fitting:

1. more accurate initialization
2. fixing corner points of the model
3. better gradient fitting due refined base image (distortion reduction)

As seen in the ASM algorithm in section 2.3 the ASM need to be initialized near by the image object. The quality of initialization can effect the result enormously. For initialization the mean shape \bar{m} needs only a affine transformation $T_{tx,ty,scaling,rotation}$. The result BLOB obtained in section 2.4.1 provides such corner points, which are more accurate than the points provided by the method outlined in section 2.1. Furthermore the BLOB can be used to derive the weighting of the first shape mode (mouth opening-closing). Thus the ASM can be initialized very close and in appropriate scaling and shape to the image.

Naturally ASM suffer problems, when model points correspond to object corners with acute angles. Mouth corners represent such special case. Once the analyzed profile does not hit the narrow object, the gradient operator will not find any reasonable gradient. An additional problem appears since this weak model points represent in case of the mouth model the only forces drawing or pushing the whole model in horizontal direction. Replacing the sensor function for this model points can counteract this issue. Instead applying gradient operators the model points for mouth corners are set to the corner points found by BLOB using the ACT algorithm in section 2.4.1. These points will not change anymore during the fitting process, so they can be seen as *fixed*.

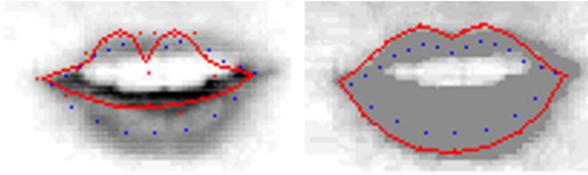


Figure 8: Sample28, impact of Adaptive Color Threshold. Left: the original nG converted image. Right: The ACT fixed image. Blue Pixels represent the initialization of the ASM. The red dots are the edge fitting points. The red line is the resulting ASM. (REMARK: BILDER LIEGEN IN ORDNER 'asmExtended/asm/')

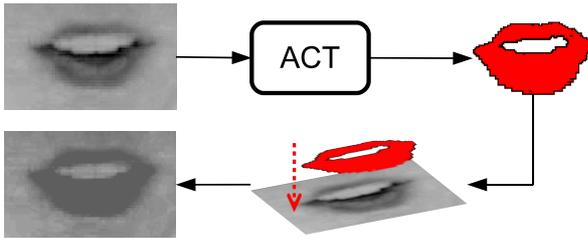


Figure 9: Scheme of fusing ACT result and nG transformed image

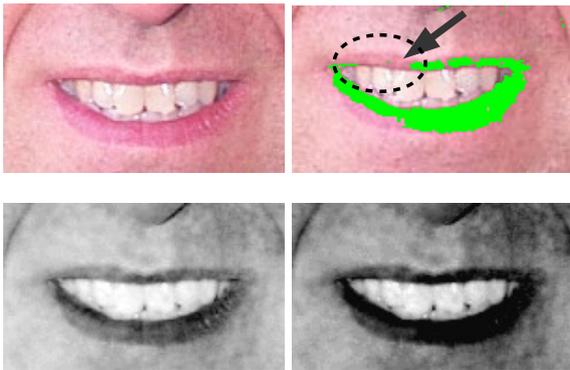


Figure 10: Combining Binary image information and ASM. Top Left: RGB, Top Right: binary result from ACT, Bottom Left: nG , Bottom Right: Fusion

One more application of the ACT is the elimination of inner gradients and distortions in the lip segment, in order to optimize the ASM algorithm. The threshold obtained in Eq. 8 which is used to refine the nG ASM work channel (See Fig.9) by applying following equation

$$I_{x,y}^* = \begin{cases} th & , I_{x,y} < th \\ I_{x,y} & , I_{x,y} \geq th \end{cases} \quad (10)$$

The fusion of binary image and original transformed nG channel has some advantage. As outlined in section 2.4.1 the method of ACT prefers a minimization of false positive lip pixels. To it is most likely it is missing a part of the mouth. Therefore in case of incompletely allocated lip pixels the soft edges at lip borders still re-

main, where in the binary image no border could be found there (see Fig. 10). This of course can also cause inner gradients, but have significantly smaller impact.

3 EXPERIMENTAL RESULTS

In the experiments the algorithm was tested on 57 images partially from the Faces Database from CIT [11] and partially from own recored data covering a wide range of illumination and saturation (see Table 2). The resulting mouth ROI had a size range of approximately 160x80 pixels. The mouth sizes varied in width between 120 and 150 pixels. The mouth height varied between 20 and 70 pixels. The higher variance is due to the opening of the mouth as greater impact to the height than the e.g. smiling has impact to the width. For each of the 57 images a ground truth was created consisting of a binary blob for lip pixels and a contour (which is equal to the outline of the binary blob).

The following sub sections will present the results and quality of the single process steps (Mouth corner point detection, lip pixel classification using ACT and Mouth contour detection using ASM).

Channel[Range]	H[0,360]	S[0,1]	I[0,1]
Mean	129.6	0.37	0.53
Variance	3666.5	0.02	0.04
Max	70.9	0.19	0.24
Min	247.6	0.72	0.91

Table 2: Image Conditions (in ROI), H=Hue, S= Saturation, I=Intensity

3.1 Detection Rate of Feature Points

Deviation for measuring detection quality of single feature points inside the face (mouth corners in this work) is given in relation to inter-ocular distance of the person (distance of both eye centers). Though this value, in relation to the face size, suffers inter-individual variations it is commonly used in shortage of better alternatives. We provide the results relative and additionally as pixel error (normal and squared) in table 3. The results with an accuracy of less than 10% relative error are good compared to other works [14]). Since the detection points are primary used to determine the ROI it was important to detect the mouth corners at all with sufficient accuracy.

Error Type	Left	Right	Overall
Relative	7.0%	6.5%	6.8%
Pixel Error	7.05px	6.41px	6.73px
Sqr Pixel Error	100.04px ²	86.32px ²	93.17px ²

Table 3: Error of mouth corner detection using method described in section 2.1

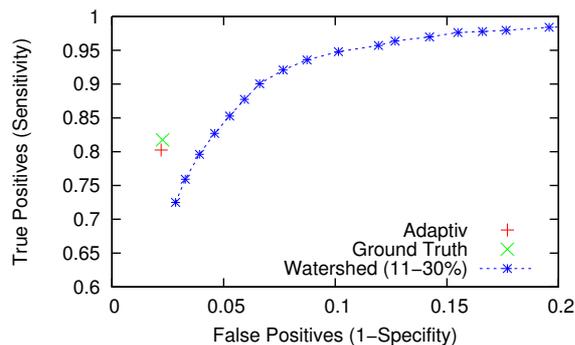


Figure 11: The cross on the left marks the quality of our proposed algorithm. The curves represent the ROC plots for different watershed percentages ranging from 10 to 30% in different color spaces. It's obvious here, that our adaptive threshold is superior to thresholds selected by a general watershed percentage. This is independent from the chosen color space or the chosen percentages for the watershed

3.2 Quality of ACT

The accuracy of the lip pixel classification is crucial for the idea of the proposed method. To show the good performance the results were put in context to an Watershed classification method (e.g. used in [9]), which is adaptive to the coloring but not to the ratio of mouth size to ROI size. Referring to the binary blob of the ground truth the adaptive classification of lip pixels, using the method proposed in in section 2.4.1, reaches a hitrate of 80.24% (True Positives - TP). However, this is no perfect result. When selecting a threshold from ground truth the rate is only insignificantly better (81.75%) (see Table 4). This is the best available hitrate aiming on low False Acceptance Rate (a maximum FAR of 5% was defined when selecting thresholds using ground truth) and False Rejection Rate. As mentioned in section 2.4.1 the primary aim is to avoid falsely accepted lip pixels (FAR). The *optimal* results in Table 4 represent results for thresholds which were chosen with respect to the ground truth and a ROC-Plot.

3.3 Improvement of ASM

In section 2.4.2 several improvements of the classical ASM algorithm were introduced. This subsection describes the impact of the different improvements. In first stage each of the three improvements were applied independent from each other, to analyze their individual impact to the algorithm. The error is calculated as average of all model-points. Ground truth was the outline of the ground truth blobs (See Fig. 3). So to each point of the ASM the distance to the closest point of the outline was calculated. The results are listed as normal and square error in Table 5. The algorithm is parametrized as outline in section 2.3. However, the second improve-



Figure 12: Results of different illumination and mouth poses.

ment of height fixing before ASM initialization is based on the information of the ACT generated blob. To measure the impact of this improvement the ASM was initialized using ACT but applied than to the unfixed nG image. The best refinement of the results is achieved by the initial height fixing. This finding should be considered in context of the chosen profile length in the ASM algorithm. Longer profiles could supersede the height initialization. On the other hand the ASM could get attracted by far objects like nose or eventually even by the chin (longer profiles of course would demand larger regions of interest). In non frontal views too long profiles also could touch regions outside the face, with unpredictable behavior. To avoid this, the algorithm would need a (likely on skin color based) good face segmentation. The ACT refinement and fixing of the ASM at the initialization points have only little but noticeable effect to the results.

When the ASM is initialized there are two options to chose the initialization points: a) the corner points which were the base for the ROI, found by the method outlined in section 2.3; b) the corner points, based on the blobs found by the adaptive threshold defined in 2.4.1. To measure the impact of different sources for initialization of ASM both available options a) and b) were exploited and additionally c) a run utilizing the ground truth points for mouth corners. These runs were done using all optimization methods listed above (ACT Refinement, Anchored Corners and ACT based height refinement). Apparently the start points taken by ACT result in a similar quality as the points chosen by ground truth in average. The facial feature points found by the AdaBoost trained Haar-Like features are sufficient to define a ROI. For the further steps if ASM initialization they lead to less accuracy.

4 SUMMARY AND CONCLUSION

In this paper new modifications for Active Shape Models based on an adaptive color based method for lip

Result Base	TP- μ	TP- σ^2	FAR- μ	FAR- σ^2
Optimal (by ground truth)	81.75%	3.88%	2.262%	0.002%
Proposed	80.24%	2.55%	2.200%	0.031%

Table 4: Results for Lip Pixel Classification (Histogram based using ACT)

Method	Error	Square Error
Classic ASM	3.21px	18.24px ²
(1) ACT Refinement	2.88px	19.40px ²
(2) Height Fix	2.08px	10.19px ²
(3) Anchored	2.89px	19.86px ²

Table 5: Impact of separately introduced improvements to ASM Algorithm

Init Method	Error	Square Error
Haar-Like	2.75px	18.24px ²
ACT	2.10px	10.17px ²
Groundtruth	2.08px	9.06px ²

Table 6: Impact of different ASM Initialization

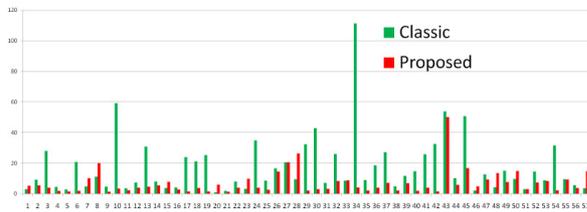


Figure 13: Image wise squared error of the proposed algorithm compared to classical method.

pixel classification were introduced. In contrast to other methods using color emphasizing of lip pixels, this method incorporates a refinement step. This refinement step eliminates edges inside the lip pixel segments, which can mislead the borders during the edge fitting step. The refining of nG image for edge fitting mainly helps to detect the lower mouth border. In this areas the crossover from skin to lip pixels often does not create a significant edge (for women this occurs more rarely due usage of lipsticks). Further the lip pixel classification creates a rough mouth blob. Based on this blob the shape model can be initialized better and closer to the real shape. The lip pixel classification performs good and is adaptive to various image conditions and skin tones. This skin vs lip color model assumption is designed and limited for Caucasian, European and Asian skin types. Further this method will suffer problems for very dark colored subjects respectively less illuminated scenes. Also the problem of bearded people was not addressed here. The more the beard color is different from general skin tone(light-gray, black) the greater the chance that this method fails. But this problem remains to all so far known methods and need further investigations and other solutions. Compared to classical shape models the presented method performs more accurate. In future works we will try to incorporate more shape modes and add a inner contour for opened mouth.

ACKNOWLEDGEMENTS

This work was supported by DFG-Schmerzzerkennung 473 (FKZ: BR3705/1-1), Innovationsfond der Universität Magdeburg, CBBS C4 M: (FKZ: UC4 -3704M) and DFG-Transregional Collaborative Research Centre SFB/TRR 62

REFERENCES

- [1] A. Al-Hamadi, A. Panning, R. Niese, and B. Michaelis. A model-based image analysis method for extraction and tracking of facial features in video sequence. In *ICSIT 2006, Amman, Vol.3*, pages 499–509, 2006.
- [2] S. Arca, P. Campadelli, and R. Lanzarotti. A face recognition system based on local feature analysis. In *Audio- and Video-Based Biometric Person Authentication*, pages 182–189, 2003.
- [3] C. Bouvier, P.Y. Coulon, and X. Maldague. Unsupervised lips segmentation based on roi optimisation and parametric model. In *IEEE International Conference on Image Processing*, pages IV: 301–304, 2007.
- [4] Jingying Chen, Bernard Tiddeman, and Gang Zhao. *Advances in Visual Computing*, volume 5359/2008 of *LNCIS*, chapter Real-Time Lip Contour Extraction and Tracking Using an Improved Active Contour Model, pages 236–245. Springer, 2008.
- [5] P. Cisar and Zelezny M. Using of lip-reading for speech recognition in noisy environments. In *Speech Processing*, pages 137–142, Prague, 2004. Academy of Sciences of Czech Republic.
- [6] T.F. Cootes, D. Cooper, C.J. Taylor, and J. Graham. Active shape models - their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, January 1995.
- [7] N. Eveno, A. Caplier, and P.Y. Coulon. Accurate and quasi-automatic lip tracking. *Circuits and Systems for Video Technology*, 14(5):706–715, May 2004.
- [8] E. A. Ince and S. A. Ali. An adept segmentation algorithm and its application to the extraction of local regions containing fiducial points. In *ISCIS*, pages 553–562, 2006.
- [9] J.Y. Kim, S.Y. Na, and R. Cole. Lip detection using confidence-based adaptive thresholding. In *International Symposium on Visual Computing*, pages I: 731–740, 2006.
- [10] Trent W. Lewis and David M.W. Powers. Lip feature extraction using red exclusion. In Peter Eades and Jesse Jin, editors, *Workshop on Visual Information Processing*, volume 2 of *CRPIT*, pages 61–67, Sydney, Australia, 2001. ACS.
- [11] California Institute of Technology. Faces 1999 (front). <http://www.vision.caltech.edu/archive.html>, 1999.
- [12] A. Panning, A. Al-Hamadi, R. Niese, and B. Michaelis. Facial expression recognition based on haar-like feature detection. *Pattern Recognition and Image Analysis*, 18(3):447–452, 2008.
- [13] Paul Viola and Michael Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.
- [14] Danijela Vukadinovic and Maja Pantic. Fully automatic facial feature point detection using gabor feature based boosted classifiers. In *International Conference on Systems, Man and Cybernetics*, volume 2, pages 1692–1698, Hawaii, 2005.
- [15] Frank Wallhoff. Facial expressions and emotion database. <http://www.mmk.ei.tum.de/waf/fgnet/feedtum.html>, Technical University of Munich, 2006.

A User-Adaptive Image Browsing System with Summarization Layout for the Personal Photo Collections

Dong-Sung Ryu, Chul-Jin Jang and Hwan-Gue Cho

Dept. of Computer Science and Engineering

Pusan National University

Keumjeong-gu, BUSAN, KOREA

{dsryu99,jin,hgcho}@pusan.ac.kr

ABSTRACT

Users spend much time organizing photos into small groups as part of photo management. Selecting good quality photos and organizing them is burdensome, as photographers amass large number of photos. This paper presents a new photo layout system with representative photos considering multiple features. Our approach consists of three steps to deal with hundreds of photos. First, we construct photo clusters by user-adapted criteria: temporal context, the number of faces, blur and luminance metrics. Then, we construct a bipartite graph that consists of photo nodes in a partite set and the constructed cluster nodes in other partite set. The representative photos of each cluster are selected by a maximal matching algorithm based on user-controlled multiple criteria. Finally, our system places the selected representative photos on a 2D grid using the placement algorithm of PHOTOLAND. Other photos in each cluster are displayed in an upper layer of a screen when the user clicks the representative photo. We conducted an experiment based on a user study; it used nine photo sets taken on a trip. The experiment showed that our system conveniently managed hundreds of photos, summarizing and visualizing them.

Keywords: digital photo, photo layout, maximal matching.

1 INTRODUCTION

The digital camera has become an indispensable commodity for people. The low price of memory encourages people to take a large number of photos. Since a digital camera is convenient and does not need extra cost to take photos, except for memory space, which is getting cheaper, people tend to take more photos than when using an analogue camera [3, 9]. Therefore, it is usual for users to take hundreds of pictures. Moreover, several users can take photos concurrently at the same event. These digital photo files can be easily exchanged by various means, such as flash memory, e-mail, ftp, and messenger. The number of photos is more increasing. People have to spend much more time organizing and browsing them.

We face several issues in managing digital photo collections due to the acquisition of large number of photos. These include:

- Poor accessibility - Low efficiency in selecting a photo in the current layout scheme. It is hard to find a specific photo amongst massive data.

- Classification of photos - We need to classify input photos based on user preference, e.g., date, event or persons in a photo.
- Preference of clustering criteria - Photos are increasing in volume and variety, since memory is cheap. Photos can be clustered using various criteria.

Photo browsing and clustering are crucial features to manage and organize many photos. Most users find what they want through a browser interface, and they spend most of their time classifying the photos into meaningful sets. In this sense, the interface to manage a large number of photos has been emphasized in recent studies. Most photo browsing systems present the images as a grid of thumbnails that the user can scroll through with a scroll bar; they can see the original version of the selected photo [8].

Meanwhile, many redundant or low quality photos occupy much space in the display area. This makes it difficult to understand the overall content of the photos. These low-priority photos do not need to be preserved in the original form. We introduce a method to select representative photos from the user's unrefined input photos based on customizable categories and visualize classified photos in a smart layout.

2 PREVIOUS WORK

Many studies related to photo management have been undertaken recently. Many useful applications have been developed to manage a large number of photos.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WSCG 2010 conference proceedings,
WSCG'2010, February 1 - 4, 2010
Plzen, Czech Republic.
Copyright UNION Agency - Science Press

Table 1: Previous work and systems for digital photo management and visualization

Method (reference)	Layout	Main features	Extra info.	Spatial info.
ACDSee [1]	Grid	Viewing only	EXIF	None
Agrafo [2]	Grid	Grouping and browsing	EXIF	Use (hybrid)
PhotoMesa [3]	Grid	Viewing (quantum treemap)	Directory info.	None
Kang [11]	Grid	Viewing (simple search)	Annotation	None
Picasa [17]	Grid	Viewing only	EXIF	None
Incremental board [18]	Grid	Viewing (similarity-based)	None	External input
Rodden [21]	Grid	Similarity-based arrangement	Annotation	Use
PhotoTOC [9]	Hierarchical	Clustering by temporal info.	timestamp	None
Kustanowitz [12, 13]	Hierarchical	Layout scheme	User input	None
Chen & Chu [4, 5]	Slide	Slideshow with layout [13]	EXIF	Use
Photo Navigator [10]	Slide	Slideshow for tracing scenes	Creation time	Use (3D)
Moghaddam [14]	Non-Grid	Layout for image retrieval	Annotation	Use
MediaGLOW [8]	Graph	Zoomable interface	EXIF	Use (graph)
Naaman [15, 16]	Geometric	Clustering based on place	GPS	Use
Quack [20]	Geometric	Community photo mining	GPS+annotation	Use

The most popular layout scheme of visualization systems is the grid layout to visualize a massive number of photos.

Many image application including ACDSee, Picasa and others use thumbnails of photos on grid layout [1, 17]. Generally, a user selects a specific photo on a grid, and then the original size photo is shown on the full screen. It is a very simple but useful method to show photos when there are less than several hundred photos. However, grid view has problems when there are too many photos. Redundant photos may occupy much of the display area. A long scroll bar is needed to explore the entire photo set.

Some enhanced grid layout schemes were proposed to overcome defects. Bederson introduced the section based grid view, PhotoMesa. It can show each directory as a section of layout [3]. PhotoMesa displays hierarchically organized photo clusters based on a file system using treemaps. It uses a simple layout for image clusters called bubblemaps. PhotoMesa emphasizes presenting large numbers of photos on a limited screen. At the top level view, photos of a specific directory are shown as tiny thumbnails. Zoomed photos with larger space are shown by selecting a section. Pinho proposed grid-based incremental board [18]. This uses an infinite grid by attaching tiny image thumbnails. Using a pre-processed photo similarity, an identical photo is located in close position to similar existing photos. It can visualize abundant photos on the screen at low cost by attaching many tiny photos to grid view. However, it is too small to see each photo, so this is not an efficient way to understand the content of input photos.

A hierarchical layout using uniform thumbnails was proposed for convenient recognition in visualizing photos. Kustanowitz proposed an organized layout scheme

with different image sizes [12, 13]. The most important image, which shows the concept of the photo set, is located at the center of display area. The other photos surround the center photo aligned according to the classification. However, the method requires identifying photos by users. In addition, it is practically limited to one sheet of display screen due to the center image. Thus, it is not suitable to use the scroll bar or to show a massive number of images. Chen & Chu applied the method on their slideshow method [4, 5]. Photos in each slide are arranged using a hierarchical layout.

Graham exploits Calendar and Hierarchical image browsers to allocate the time-intensive annotation for the photo groups [9]. He exploits the timing information to construct the collections and to automatically generate meaningful summaries. These studies help the user give a more practical structure to the photos, but they cannot provide implicit browsing regarding temporal and spatial information simultaneously. A graph-based photo layout system, MediaGLOW, uses the spring model to determine a layout in which the spring system is in a state of minimal energy [8]. This graph-based interface determines the distance between each photo node according to a variety of distance measures, such as temporal, geographic, and visual distance (tagged data). It can also deal with lots of user interaction. This interface is very useful to organize photos. Table 1 summarizes representative studies.

3 CLUSTERING WITH MULTIPLE FEATURES

In digital life, people want to cluster photos using several features; they also want to browse the corresponding summarized view with each feature. For example, let us assume the following case. Users grouped pho-



Figure 1: Result of clustering by multiple features. For convenience, we randomly pick 33 photos from the photo set. The total number of clusters with multiple features is 23. (a) Temporal clustering by time [6]. $|C_T| = 10$. (b) Clustering by the number of faces (We use *cvHaarDetectObjects* function of OpenCV library to detect the face in a photo). $|C_F| = 4$. (c) Clustering by blur metric[7]. $|C_B| = 4$. (d) Clustering by luminance in Lab color space. $|C_L| = 5$.

tos by time taken. Then, after some time has passed, they wish to find the corresponding photos that satisfy the following conditions: 1) Photos taken with his two friends, 2) A good quality photo without blur, 3) The light atmosphere of photos. In this case, he spends much time to find the corresponding photos having these conditions (Users compare the selected photo with most of the photos in each cluster). Besides, when the photos to be arranged are getting numerous, these tasks become burdensome. A user-adaptive photo browser that can provide a summarized view by multiple user clustering criteria would be very useful in this case.

We deal with a variety of similarity measures to overcome these problems. These include time photo taken, the number of faces, blur and luminance metrics. In this section, we discuss with how to cluster each photo. For discussion, let us define the following notation:

- $U : U = \langle P_0, P_1, \dots, P_n \rangle$ denotes a sequence of photos taken, where P_i is each photo image.
- $face(P_i)$: the number of faces in P_i .
- $blur(P_i)$: a perceptual blur metric of P_i [7]. ($0 \leq blur(P_i) \leq 1.0$)

- $lumi(P_i)$: a luminance metric in Lab color space for P_i . ($0 \leq lumi(P_i) \leq 1.0$)
- $time(P_i)$: a timestamp extracted from EXIF of P_i .

The first criterion is temporal context. We use Cooper's clustering method to evaluate the similarity of each photo, as below [6] :

If K increases, we can get a coarser clustering result of the photos' timestamps. For smaller K , finer dissimilarities between groups of timestamps become apparent.

The second criterion of content based clustering is the number of faces in a photo. In the photo, we can grasp the number of faces using the OpenCV face detection algorithm based on a Harr transform. Our system simply classifies photos into small groups based on the number of faces. We use the similarity of face feature as below:

$$Sim_F(P_i, P_j) = 1 - \frac{|face(P_i) - face(P_j)|}{\max_{P_k \in U} \{face(P_k)\}} \quad (1)$$

We construct a classified photo group considering some visual features such as blur and luminance metrics. The similarity of blur metrics is determined by

Frederique's method [7]. The key idea of his method is to blur the initial image and to analyze the behavior of the neighboring pixels variation. We also consider the luminance features, which are calculated from the average of L values in Lab color space. These two metrics are normalized in a defined range from 0 to 1, their similarity measures are given below:

$$Sim_B(P_i, P_j) = 1 - |blur(P_i) - blur(P_j)| \quad (2)$$

$$Sim_L(P_i, P_j) = 1 - |lumi(P_i) - lumi(P_j)| \quad (3)$$

Figure 1 shows the result of clustering by four features. The input photos are selected from our past photos taken on a trip without any special intent. For convenience, we randomly select 33 photos from the photo set in this study, since most photo sets have hundreds of photos. As a result of this clustering, we can get several small groups, $C_x^{(k)}$, where $x \in T, F, B, L$ classify four features (Temporal, Number of Faces, Blur metric and Luminance in Lab color space):

1. $C_T^{(k)}$ denotes the k -th photo cluster using Cooper's algorithm [6].
2. $C_F^{(k)} = \{P_j | face(P_j) = k\}$
3. $C_B^{(k)}$ denotes the k -th photo cluster in terms of the blur metric.
4. $C_L^{(k)}$ denotes the k -th photo cluster in terms of luminance value.

4 SELECTING REPRESENTATIVE PHOTOS

Now, we have many small groups clustered by multiple features. We select each representative photo to summarize each photo clusters. In this paper, we present a selection method of representative photos using a maximal matching graph algorithm. First, we construct a bipartite graph, whose node consists of the photos in a partite set, and the created photo clusters of section 3, in another partite set, as shown in Figure 2.

The cluster nodes on the right hand side of this graph can have multiple edges, since the photos are assigned into clusters through multiple features. However, since each cluster has just one representative photo, we have to determine which photos are assigned into which clusters in this graph. We use the maximal matching algorithm to select the representative photos of each cluster to satisfy user's clustering preference as much as possible.

Let us consider a bipartite graph $G(V, E)$, as shown in Figure 2. Placing weight $w(P_i, C_x^{(j)})$ on edge $e(P_i, C_x^{(j)})$, ($P_i \in V$, $e \in E$, V and E are the set of all vertices and edges in this graph, respectively) gives us a weighted bipartite graph with partite sets $Photos = \{P_0, P_1, \dots,$

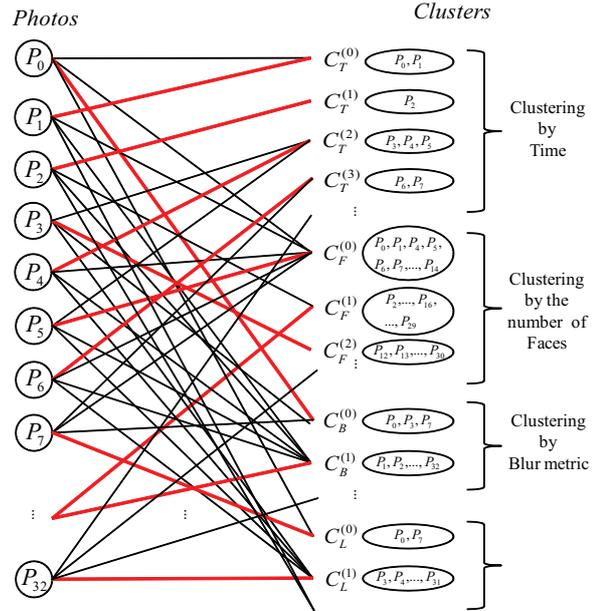


Figure 2: Maximal matching process for a selection of representative photos.

$P_n\}$ and $Clusters = \{C_T^{(0)}, C_T^{(1)}, \dots, C_F^{(0)}, C_F^{(1)}, \dots, C_B^{(0)}, C_B^{(1)}, \dots, C_L^{(0)}, C_L^{(1)}, \dots\}$. The weights of each edge are given below:

$$w(P_i, C_F^{(j)}) = \frac{1}{|E(C_F^j)|} \cdot \sum_{P_x \in C_F^j} (u_x \cdot Sim_x(P_i, P_x)) \quad (4)$$

, where $Sim_x(P_i, P_x)$ is the similarity function for each clustering feature, $\{Sim_T(P_i, P_x), Sim_F(P_i, P_x), Sim_B(P_i, P_x), Sim_L(P_i, P_x)\}$, defined as Section 3. $u_x = \{u_t, u_f, u_b, u_l\}$ is one of the user-defined parameters to control each clustering feature.

A maximal matching M of a graph G is maximal, if every edge in G has a non-empty intersection with at least one edge in M . Our system selects each representative photo of clustered groups based on the relationships of these matching M . The maximal matching of this graph means the most similar relations globally between clusters and photos, when we consider the user's intent.

Figure 3 shows a portion of the relationships between several representative photos (P_{15}, P_{23}, P_{31}) and their corresponding clusters in Figure 1. In this figure, if we consider the number of faces, three photos are respectively clustered into different clusters (bold red edges in the Figure). At the same time, they are also clustered into different clusters considering the luminance of photos (bold blue edges in the Figure). In this case, the user can control which features are used to select the representative photos using u_x in Equation 4. If the user sets $u_f = 1.0$ and other features are less than 0.1, then our system selects the red edges for the representative photos of each cluster in this Figure. If the user

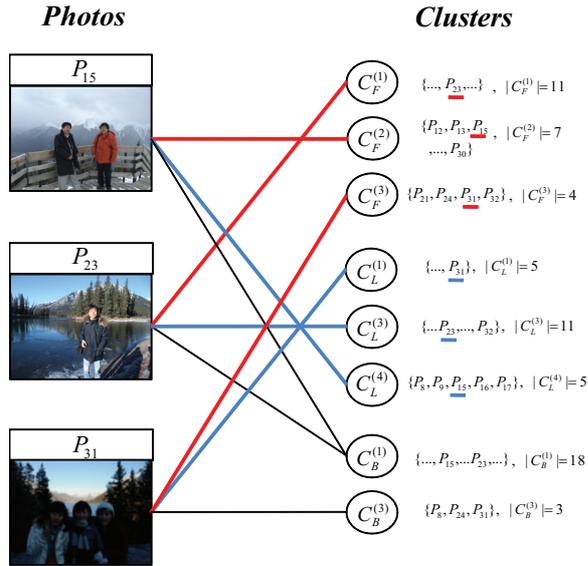


Figure 3: A portion of the graph constructed from the clusters in Figure 1. The user can control which features are used to select the representative photos using u_x in Equation 4. Bold red edges depict maximal matchings when we consider the number of faces. Bold blue edges depict maximal matchings when we consider the luminance value of each photo.

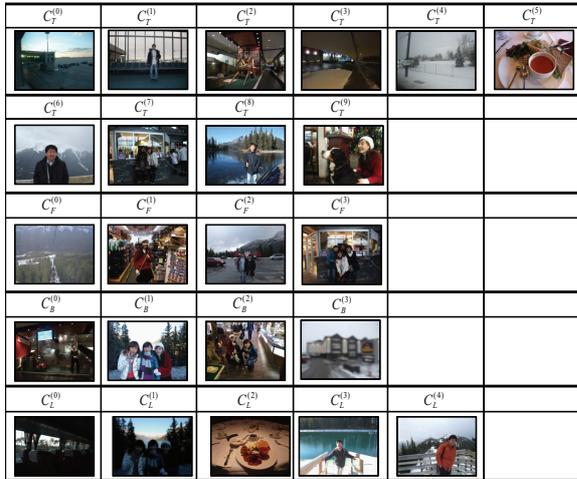


Figure 4: Corresponding result with photos selected in Figure 1. We set parameters as $u_t = 0.9$, $u_f = 0.7$, $u_b = 0.6$, $u_l = 0.6$.

also sets $u_l = 1.0$ and other features are less than 0.1, likewise our system selects the blue edges for their representative photos.

Figure 4 shows the result of representative photo selection based on each cluster in Figure 1. We implements this maximal matching algorithm using the LEDA library.

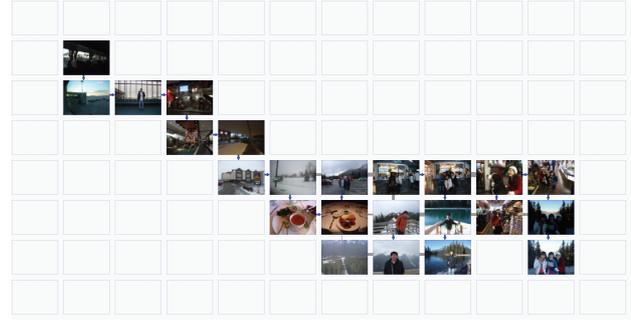


Figure 5: Result of placement for representative photos in Figure 4. We consider only the temporal context of selected representative photos in placing them.

5 LAYOUT FOR PHOTO VISUALIZATION

Our earlier paper on PHOTOLAND outlined a system that visualizes hundreds of photos on a 2D grid space to help users manage their photos [22]. This system considers spatial and temporal context simultaneously when photos are placed on a grid. We used a similar placement algorithm to visualize photos. This paper summarized the placement algorithm as below:

1. PHOTOLAND places the first photo in the center of a 2D grid.
2. It places the next photo considering temporal information and spatial context :
$$S(P_i, P_j) = (t_\alpha \cdot S_T(P_i, P_j) + (1 - t_\alpha) \cdot S_C(P_i, P_j)) \quad (5)$$
, where t_α is a user-defined parameter to control spatial and temporal weight, it ranges from 0 to 1.0. S_T and S_C denote the temporal and spatial similarity, respectively.
3. It also considers global geometric constraints, such as center of weight for placed photos and aspect ratio for a screen.
4. The temporal similarity is calculated by the logistic function of the time gap between two photos.

We use two hierarchical layers that display the representative photos and the clustered photos related to them in order to display photos. First, we consider only temporal context to place the representative photos. As mentioned before, since it is related to the user's event, the temporal context has to be considered as being most important. Figure 5 shows the result of placement for representative photos in Figure 4. Then, the user can click on a representative photo; our system displays other photos related to it in an upper layer, as shown in Figure 6. When the clustered detail photos are displayed, we rendered a semi-transparent gray background on the lower layer for representative photos.



Figure 6: Other photos related to the selected representative photos in an upper layer. We rendered a semi-transparent gray background on the lower layer for representative photos.

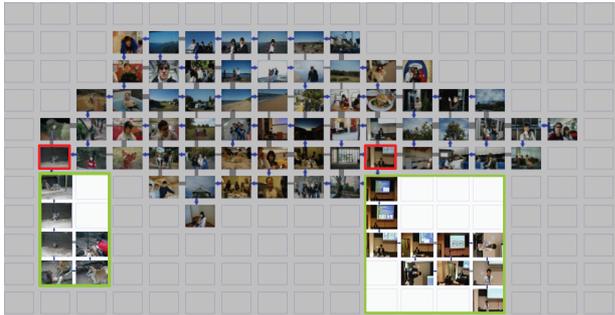


Figure 7: Photo placement result of another representative photo. Input photo set is one of the type ‘C’ sets. There are 66 clusters. They consist of 457 photos taken in Banff, Canada. We set parameters as $u_t = 0.5$, $u_f = 0.7$, $u_b = 0.7$, $u_l = 0.5$.

Figure 7 shows the result of placement for representative photos selected from one of the type ‘C’ photos taken in Banff. There are 66 clusters. The blue arrows near the grid cell depict their temporal sequence. Spatial similarity between their neighboring photos is presented by the gray line border. The thicker line depicts that the photos have colors that are more similar in 25 perceptual colors [19].

6 EXPERIMENT

We conducted three consecutive experiments to evaluate the usability of our system. These user studies were designed to understand the user’s subjective reaction to our system. Our user studies deal with the following three perspectives:

1. How much time can we save using our system in photo clustering?
2. How nice is the representative photo selection algorithm compared to random selection?
3. How quickly can users find the desired photos in each photo sets?

Sixteen people participated in our experimental sessions. The participants were six beginners, seven experts and three evaluators. We define a beginner as a user whose major is not related to computer engineering. The beginner group does not deal with computers in everyday life (Ages ranged from 25 to 36). In contrast to the beginner, the expert group consists of users whose major is related to computer engineering. The final group of participants (evaluators) consists of people who take each photo set directly.

The input data consisted of three levels of photo sets, A, B, and C based on the number of photos, described in Table 2. Each photo sets consists of evaluator’s photos taken during a trip without any special intent. We classified photos into several categories with the person who took each photo set before the experiment to compare the result of clustering. Then, these categories are embedded in the custom field of their EXIF (“On the mountain” and “Number of Face 3”). The clustering features considered were temporal context, the number of faces and luminance in Lab color space.

Table 2: Description of the input photos, A, B and C. depicts the evaluator for each photo set

Type	# of Photos	# of Evaluator	# of photo sets
A	80 ~ 100	3	4
B	150 ~ 180	2	3
C	420 ~ 460	3	2

Experiment 1. We investigated the clustering task completion time. We compared our system to a traditional scrolling interface based on a 2D grid, ACDSSee Photo manager, as a benchmark [1]. The photo sets were classified by evaluators in advance to construct the true sets for this experiment. We determined the true cluster information to be the number of clusters, the number of photos in each cluster, the categories of each cluster. We term this as “cluster information”.

We organized the new tester group for experiment 1 from the sixteen participants in the experiment. Since clustering is very subjective, we want to pick out the person who shares the memory of each photo set with the evaluator as testers, to investigate the satisfaction with the clustering results impartially. In this experiment, they consisted of the photographer’s traveling companions. We computed the satisfaction level of clustering results comparing the file names of photos in clustering folder to the cluster category label.

The precision indicates the proportion of true positives clustered as below:

$$E_p = \frac{|\{\text{true photo sets}\} \cap \{\text{user-clustered photo sets}\}|}{|\{\text{user-clustered photo sets}\}|} \quad (6)$$

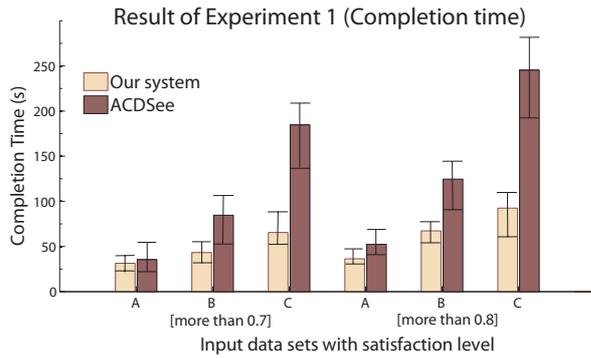


Figure 8: Average completion time to classify each cluster. The input photo sets are A, B and C as described in Table 2.

The recall measure is the number of correct results divided by the number of all relevant results. It measures the proportion of true clustered photos :

$$E_r = \frac{|\{\text{true photo sets}\} \cap \{\text{user-clustered photo sets}\}|}{|\{\text{all of relevant truth photo sets}\}|} \quad (7)$$

We use an average of precision and recall that was respectively measured as more than 0.4 in this experiment to decide if it is sufficient to satisfy the clustering result.

Now we wrapped up preparation for experiment 1. First, each tester selects one photo set from every type of photo set described in Table 2. Then, we gave the testers the cluster information of the selected set with a simple program that can divide photos into groups by a constant time gap. They were asked to divide each photo set already by the evaluator. During the experiment, the testers can know the corresponding satisfaction level of their clustering results by clicking the 'evaluation' button on the program we presented them. This simple program can report how much the current photo clustering satisfies the evaluator's clustering results, considering precision and recall. We iterate the above steps until the clustering results can be recognized as reaching the satisfaction level to compare the completion time.

Figure 8 shows the average completion time for clustering satisfaction, the satisfaction levels are 0.7 and 0.8. It shows that the layout of our system is useful to classify hundreds of photos compared to ACDSee Photo manager.

Experiment 2. We compare the representative photos selected by our system to randomly selected photos from each cluster. These selected photos are given to the evaluators. Then, evaluators were asked to score the satisfaction of each selection. Each experiment was iterated ten times per the photo set randomly selected from sets (A, B and C, respectively), for generality. The scores ranged from 4 to 10. The average score for Experiment 2 is shown in Figure 9. We excluded the

blurred features, since it is difficult to identify with the unaided eyes on a document.

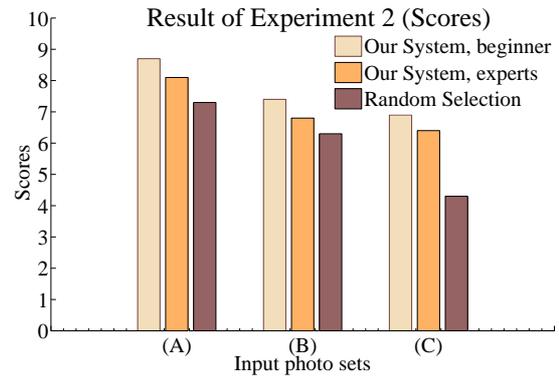


Figure 9: Result of experiment 2. Average scores of participants evaluation.

Experiment 3. The participants were asked to find each corresponding similar photo to the given images when four images were given. We had already selected photos for the correct answer based on its similar images, as a true set. We investigate the number of trials in which that they select all correct answers. Figure 10 shows the average number of trials to find the objective photos. Since the gap of trial results between beginners and experts in Experiment 3 is small, our system can be easily used by Beginners.

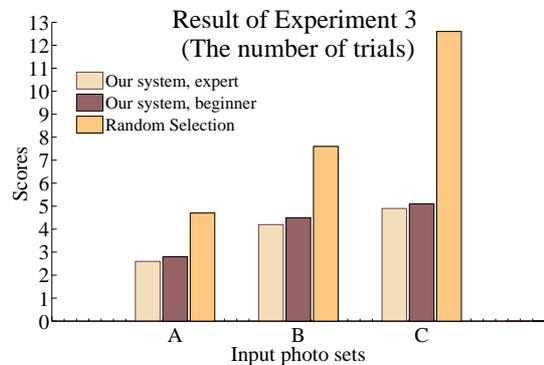


Figure 10: Result of experiment 3. Average number of trials to find all desired photos.

7 CONCLUSION

The digital camera has become an indispensable commodity for people. Tasks related to photo management, such as classification, filtering of a bad quality of photos and their construction, are increasingly part of daily life. The low price of memory allows people to take more and a greater variety of photos. The task of organizing these becomes boring and burdensome. Thus, we propose a representative photo layout system

that provides a clustering function for photo collections based on user preference.

Our clustering process used four criteria. First, our system clusters photos into small groups using multiple criteria. Then, we select the representative photo, from each classified photo groups, using a maximal matching graph algorithm. The selected photos are placed on a 2D grid using a similar placement algorithm to PHOTOLAND. The other photos corresponding to the representative photos in the same group are displayed on the upper layer when the user clicks the placed photos in a lower layer. Conclusively, let us summarize the notable contributions of this paper:

1. Our maximal matching algorithm is very useful and efficient in selecting the representative photo.
2. We apply four criteria, such as temporal context, the number of face, blur metric and luminance value in Lab color space, to cluster photos into meaningful groups. Other clustering features can be adopted if that feature is normalized between 0.0 and 1.0.
3. Our system uses two hierarchical layer structure to visualize photo groups based on its representative photos using a method similar to PHOTOLAND's placement algorithm.

The system proposed in this paper was positively received by the participants. They evaluated our system as being an intuitive photo clustering interface. However, the clustered group may at times not be able to find its representative photo. If the edge weight between the group node and its photo node is weak, the pairs are not selected in the process of maximal matching. In this case, we can not display the other photos without the representative photo. We have to develop the solution to this problem.

ACKNOWLEDGEMENTS

This work was supported by the IT R&D program of MKE/MCST/IITA (2008-F-031-01, Development of Computational Photography Technologies for Image and Video Contents).

REFERENCES

- [1] ACDSSee Photo Software Homepage. <http://www.acdsee.com/>.
- [2] João. Mota, Manuel J. Fonseca, Daniel Gonçalves, and Joaquim A. Jorge. Agrafo: a visual interface for grouping and browsing digital photos. In *Proc. of AVI '08*, pages 494–495, New York, NY, USA, 2008. ACM.
- [3] Benjamin B. Bederson. Photomesa: a zoomable image browser using quantum treemaps and bubblemaps. In *Proc. of UIST '01*, pages 71–80, New York, NY, USA, 2001. ACM.
- [4] Jun-Cheng Chen, Wei-Ta Chu, Jin-Hau Kuo, Chung-Yi Weng, and Ja-Ling Wu. Tiling slideshow. In *Proc. of MULTIMEDIA '06*, pages 25–34, New York, NY, USA, 2006. ACM.
- [5] Wei-Ta Chu, Jun-Cheng Chen, and Ja-Ling Wu. Tiling slideshow: An audiovisual presentation method for consumer photos. *IEEE Multimedia*, 14(3):36–45, July 2007.
- [6] Matthew Cooper, Jonathan Foote, Andreas Girgensohn, and Lynn Wilcox. Temporal event clustering for digital photo collections. *ACM Transactions on Multimedia Computing, Communications and Applications*, 1(3):269–288, 2005.
- [7] Frederique Crete, Thierry Dolmiere, Patricia Ladret, and Marina Nicolas. The blur effect: perception and estimation with a new no-reference perceptual blur metric. In *Proc. of the SPIEHuman*, volume 6492, page 64920I. SPIE, 2007.
- [8] Andreas Girgensohn, Frank Shipman, Lynn Wilcox, Thea Turner, and Matthew Cooper. MediaGLOW: organizing photos in a graph-based workspace. In *Proc. of IUI '09*, pages 419–424, New York, NY, USA, 2009. ACM.
- [9] Adrian Graham, Hector Garcia-Molina, Andreas Paepcke, and Terry Winograd. Time as essence for photo browsing through personal digital libraries. In *Proc. of JCDL '02*, pages 326–335, New York, NY, USA, 2002. ACM Press.
- [10] Chi-Chang Hsieh, Wen-Huang Cheng, Chia-Hu Chang, Yung-Yu Chuang, and Ja-Ling Wu. Photo navigator. In *Proc. of MULTIMEDIA '08*, pages 419–428, New York, NY, USA, 2008. ACM.
- [11] Hyunmo Kang and B. Shneiderman. Visualization methods for personal photo collections: browsing and searching in the photofinder. In *Proc. of ICME '00*, volume 3, pages 1539–1542, 2000.
- [12] J. Kustanowitz and B. Shneiderman. Meaningful presentations of photo libraries: rationale and applications of bi-level radial quantum layouts. In *Proc. of JCDL '05*, pages 188–196, 2005.
- [13] J. Kustanowitz and B. Shneiderman. Hierarchical layouts for photo libraries. *IEEE Multimedia*, 13(4):62–72, Oct. 2006.
- [14] Baback Moghaddam, Qi Tian, Neal Lesh, Chia Shen, and Thomas S. Huang. Visualization and user-modeling for browsing personal photo libraries. *International Journal of Computer Vision*, 56(1/2):109–130, 2004.
- [15] Mor Naaman, Susumu Harada, QianYing Wang, Hector Garcia-Molina, and Andreas Paepcke. Context data in geo-referenced digital photo collections. In *Proc. of MULTIMEDIA '04*, pages 196–203, New York, NY, USA, 2004. ACM.
- [16] Mor Naaman, Yee Jiun Song, Andreas Paepcke, and Hector Garcia-Molina. Automatic organization for digital photographs with geographic coordinates. *Proc. of JCDL '00*, 00:53–62, 2004.
- [17] Picasa Homepage. <http://picasa.google.com/>.
- [18] Roberto Pinho, Maria Cristina F. de Oliveira, and Alneu de A. Lopes. Incremental board: a grid-based space for visualizing dynamic data sets. In *Proc. of SAC '09*, pages 1757–1764, New York, NY, USA, 2009. ACM.
- [19] B.G. Prasad, K.K. Biswas, and S.K.Gupta. Region-based image retrieval using integrated color, shape, and location index. *Computer Vision and Image Understanding*, 94:193–223, 2004.
- [20] Till Quack, Bastian Leibe, and Luc Van Gool. World-scale mining of objects and events from community photo collections. In *Proc. of CIVR '08*, pages 47–56, New York, NY, USA, 2008. ACM.
- [21] Kerry Rodden, Wojciech Basalaj, David Sinclair, and Kenneth Wood. Does organisation by similarity assist image browsing? In *Proc. of CHI '01*, pages 190–197, New York, NY, USA, 2001. ACM.
- [22] Dong-Sung Ryu, Woo-Keun Chung, and Hwan-Gue Cho. PHOTOLAND : A new image layout system using spatio-temporal information in digital photos. In *Proc. of SAC (to appear)*, 2010.

Automatic Generation of Character Behavior by the Placement of Objects with Motion Data

Kenichi Kobori

Osaka Institute of Technology
1-79-1 Kitayama
573-0196 Hirakata, Osaka, Japan
kobori@is.oit.ac.jp

Kenji Hirose

Mitsubishi Electric Corporation
2-3-33 Miwa
669-1513 Sanda, Hyogo, Japan

Koji Nishio

Osaka Institute of Technology
1-79-1 Kitayama
573-0196 Hirakata, Osaka, Japan
nishio@is.oit.ac.jp

ABSTRACT

Recently, virtual space design with high quality three-dimensional CG has become possible due to a rapid improvement in computer performance. In order to give a CG character the movement of its own, it is necessary to apply the motion data to the character after getting them with a motion capture device or by hand work. Production costs have increased because this is complicated work for an animator. In general, the character movement depends on the objects which it holds. Our idea is that the character does not have motion data of its own, but we give the character motion to the objects that it holds. We have developed a method for automatically generating the character motion data by giving motion information to the objects. Thus, each object includes the motion data that cause the character to act. In addition, we give multiple motion data appropriate for the situations to the objects and define the relationship between objects in a virtual space. By using this relation, we can generate a variety of motion data according to the object which the character holds. The proposed method reduces the number of character motion data which should be prepared beforehand.

Keywords

Motion data, animation, movement, virtual reality

1. INTRODUCTION

Recently, virtual environment design with high quality three-dimensional CG has become possible by a rapid development of the computing power. CG characters have existed in three-dimensional virtual space to realize the space where looks just like reality. In general, we make use of motion capture data to let the CG characters act [Alb00]. We have to prepare many scenes to construct various kinds of virtual space. In addition, it is necessary to prepare many motion data of CG characters. It forces animators to enormous work. Whenever the scene is changed, they have to prepare many motion data. It is necessary that they acquire motion data again or revises it. Therefore, a method to automatically generate motion data has been required and several methods have been proposed [Kan06] [Mar99].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

However, these approaches have the restriction that only limited motion data can be generated or one CG character maintains only one motion data. Kan et al. showed that automatic generation can be achieved by using building blocks, called motion patches with motion data. Each patches is annotated with simple movements such as “walk ” or “stand up ” .

However, we need to prepare a lot of motion patches when we make complicated motion data.

We propose a method to solve these problems. In our method, the character does not have the motion data of its own. Instead, the objects that the character operates preserve movement information of the character. We can give several movement data to the object when it is necessary. In addition, the movement to the object varies depending on the movement to other object by giving the relation between two objects. The data that the object maintains include motion information of the character and start position to be carried out for the object and its geometry data. This kind of approach is known in the field of robotics, but has not been used generally in character animation .

2. PROPOSED METHOD

The object has three kinds of information to give the character some sort of action. We generate entire

motion data of the character by synthesizing the following information and walking motion data.

2.1 Information that object maintains

The information includes display information, movement information and relation information.

1) Display information

This is shape geometry data to display an object in three-dimensional virtual space. We can change the appearance of the object easily by changing these data.

2) Movement information

This data is the information to generate the movement of the character. The information includes the movement to be applied to the character, the start position of the movement and occupation area of the object in the virtual space.

Put another way, the movement information is the motion data to be carried out for the object.

For instance, a chair preserves the movement of “sit” and a ball preserves the movement of “throw”.

This movement information is basic data that we acquire with a motion capture device in advance.

The character recognizes the movement that it should perform without distinguishing objects because each object has the movement for the character.

The start position of the movement is the position where the character starts the movement for the object. The occupation area is a convex polygon surrounding the object.

When the character advances toward a target object, an avoidance path is generated utilizing the convex polygon so that the character does not collide to other objects. We describe the method in detail in section 2.2.

3) Relation information

The character sometimes operates several objects, as follows:

- a) Hammer + Nail → Drive a nail with a hammer
- b) Postcard + Mailbox → Drop a postcard into a mailbox
- c) Ball + Locker → Put a ball into a locker

In these cases, the character operates an object holding with other object.

For example, the character will do the action that it “reads postcard” when it holds the postcard. However, it is expected that the character drops the postcard into a mailbox when it holds the postcard in front of the mailbox.

In order to achieve above-mentioned mechanism, we need to define the relation between the postcard and the mailbox and add these two objects the information that the character drops the postcard into the mailbox. The object holds three kinds of identifiers, as shown in Figure 1.

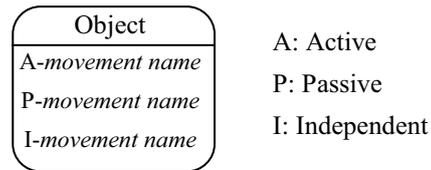


Figure 1. Relation information

Identifier A (Active) means the movement that the character performs. Identifier P (Passive) means the movement that the character performs in the state of holding an object with identifier A.

Identifier I (Independent) means the movement that the character performs independently.

Figure 2 shows the relations among three objects, a mailbox, a postcard and a trash bin.

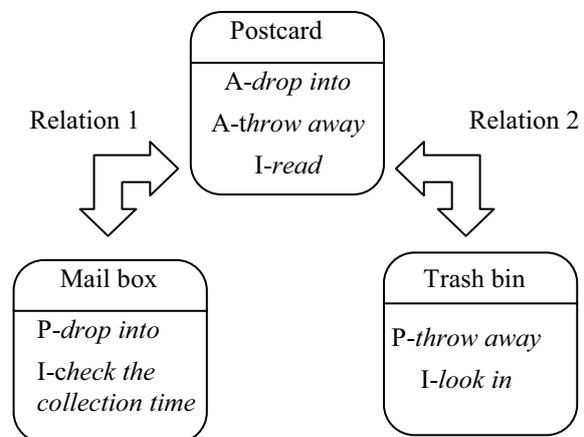


Figure 2. Relations among three objects

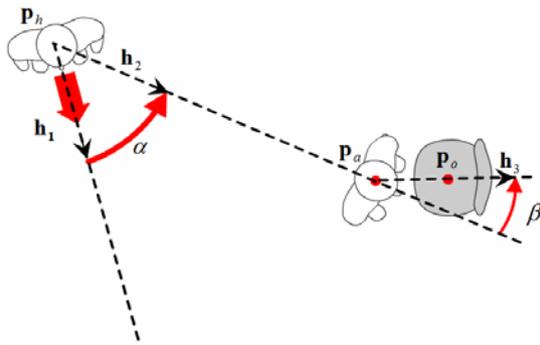
The postcard and the mailbox have the relation 1 each other and the trash bin and the postcard have the relation 2 each other. The character can drop the postcard into the mailbox when it holds the postcard by relation 1. On the other hand, the character does not drop into the mailbox without the postcard, but checks the collection time of mails. Furthermore, the character can throw the postcard away into the trash bin when it holds the postcard by relation 2. If the character does not hold the postcard, it will look in the trash bin.

2.2 Motion data generation

The motion data of the character is determined by using the movement information and the relation information. The final motion data are generated by synthesizing the movement to be applied and walking movement.

1) Outline of generation of movement

Figure 3 shows the situation that a character and a chair are placed in the virtual space.



- P_h : Start position of a character
- P_a : Position of the movement to be applied
- P_o : Position where an object is put on
- h_1 : Initial direction vector of a character
- h_2 : Vector from P_h to P_a
- h_3 : Vector from P_a to P_o
- α : Angle between h_1 and h_2
- β : Angle between h_2 and h_3

Figure 3. Motion data generation

First, the character turns α degrees to the direction h_2 and walking motion is allocated for the character. When the character arrives at the position P_a , it turns β degrees.

Next, the avoidance path is generated by using walking motion in the case that obstacles exist in the direction which the character is moving in.

The movement which the chair has is applied to the character. As a result, the motion data are generated from walking motion to the motion which the character performs for the object. In the case that plural objects are placed in the virtual space, the process mentioned above is repeated until the character completes the movement that all objects have.

2) Generation of avoidance pass

When the character goes straight on toward a target object, there is the case that it collides with an object which blocks the character's way, as shown in Figure 4. We generate the avoidance pass by using the occupation area of the object, as shown in Figure 5.

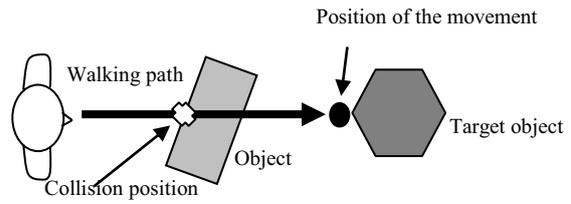


Figure 4. Collision of object with character

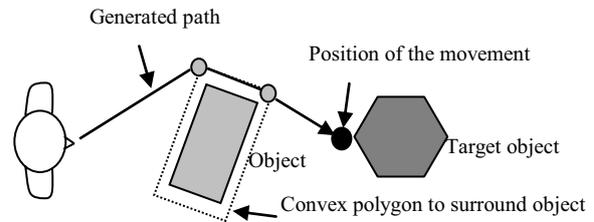


Figure 5. Avoidance path generation

However, the movement of the character becomes unnatural at those vertices of the convex polygon to surround the object because the generated pass is a set of lines through those vertices. Therefore, we interpolate the generated pass with a curve, as shown in Figure 6. P_s is a start point and P_e is an end point of the path. P_1 and P_2 are the vertices of the convex polygon. First, three points P_{s1} , P_{12} and P_{2e} are inserted in the middle point of these three lines. Next, two quadratic Bezier curves are generated by using P_{s1} , P_1 , P_{12} and P_{12} , P_2 , P_{2e} . The final path consists of the line P_s - P_{s1} , the quadratic Bezier curve P_{s1} - P_{12} , the quadratic Bezier curve P_{12} - P_{2e} and the line P_{2e} - P_e .

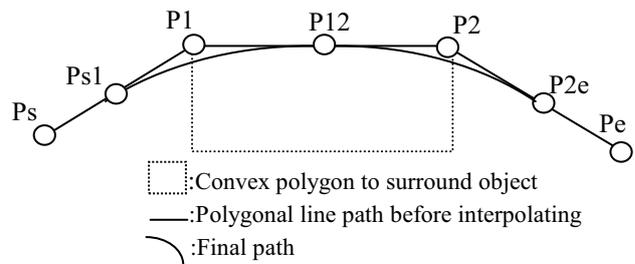


Figure 6. Curve interpolation of avoidance path

3) Generation of the movement to walk

The movement to be applied starts at the position of the movement in the state that the character stands upright. When the character arrives at the position of the movement to be applied, the posture must be standing straight. In order to meet this requirement, we prepare seven kinds of walk movement for the half period in Figure 7 in order to join the walk movement and the movement to be applied smoothly.

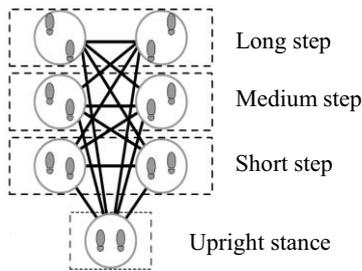
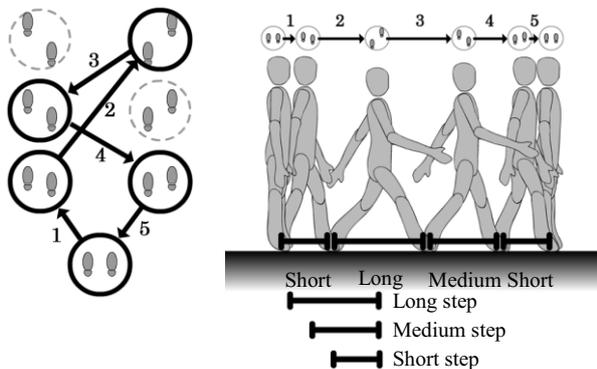


Figure 7. Walk movement for half period

We choose several walk movements among seven kinds of walk movement and connect those movements so that the final posture becomes standing straight. For instance, the walk movement is generated in Figure 8(b) when five walk movements are connected in numerical order, as shown in Figure 8(a).

4) Motion transition

In general, the posture becomes unnatural at the portion where we just connect motion data A to B, as shown in Figure 9.



(a) An example of connection (b) Generated walk movement

Figure 8. An example of walk movement generation

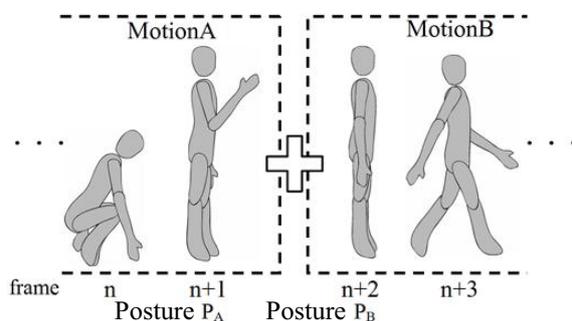


Figure 9. Unnatural motion to be connected

Therefore, we insert transition frames so that the posture P_A gradually resembles posture P_B , as shown in Figure 10. The motion data is modified so that the

posture changes from motion A to B naturally by inserting intermediate motion data between those two motions. The posture of the character is expressed by Euler angle of each joint. Interpolation between two postures becomes unnatural if we use the interpolation of Euler angle [Jam90]. We use spherical linear interpolation to avoid Gimbal Lock problem. Two Euler angles to be interpolated are transformed to two quaternions and those are interpolated by using spherical linear interpolation [Tom06].

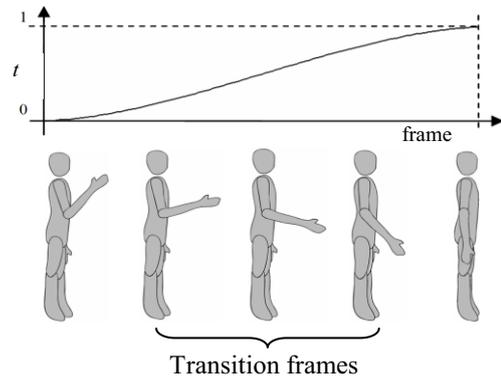


Figure 10. Interpolated postures

3. EXPERIMENTS AND RESULTS

1) Placement of two objects with the relation, as shown in Figure 11.

Table 1 shows two objects and the movement to be applied.

No.	Objects	Movement to be applied
1	Hammer	Pick up
2	Wood	Drive nail

Table 1. Allocated objects and the movements

First, we put a character and two objects which have the relation data in a virtual space.

Figure 12 shows the result of the movement. The number under each figure shows the frame number.

The wood has no movements for the character directly after placement of the character and the objects, as shown in Figure 13(a). After we select the movement “pick up” in Figure 13(b), the movement “drive nail” can be selected, as shown in Figure 13(c) because the hammer and the wood have the relation each other.

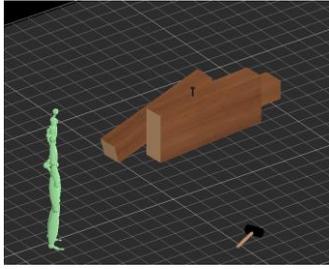


Figure 11. Placement of two objects and a character

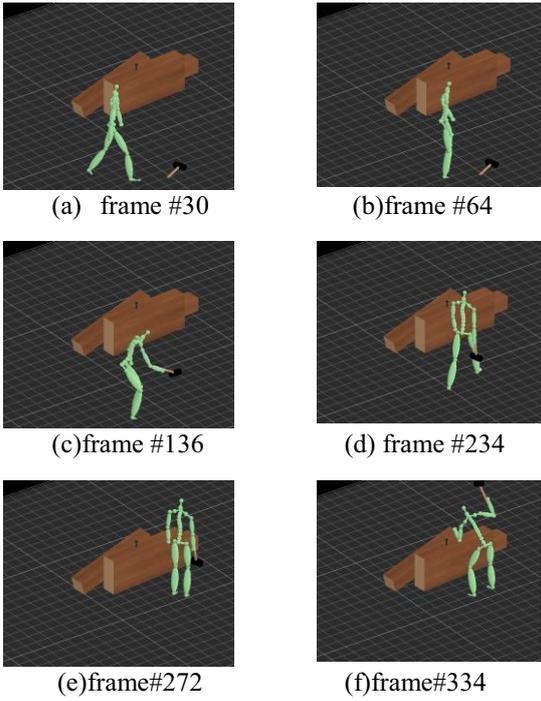
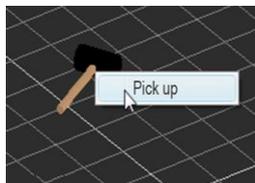


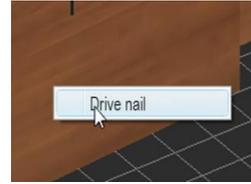
Figure 12. Result of motion data generation to two objects with relation



(a) Before picking hammer



(b) Selection of "Pick up"



(c) Appearance of the movement "Drive nail"

Figure 13. Selection of the movement to be applied

2) Placement of five objects (illuminator, television, post card, door and mailbox) and a character in a virtual space in Figure 14.

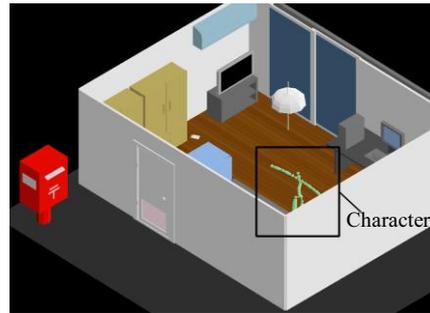


Figure 14. Three dimensional virtual space

Table 2 shows the movement to be applied to five objects. First, we set the character and five objects in the virtual space. Next, we select the movement to be applied to five objects in the movement order of the character.

No.	Objects	Movement to be applied
1	Illuminator	Turn on
2	Television	Switch on
3	Post card	Pick up
4	Door	Open and pass through
5	Mailbox	Drop into

Table 2. Allocated objects and movements

Figure 15 shows the result of the movement. The result shows that the motion data of the character is generated by the order in Table 2.

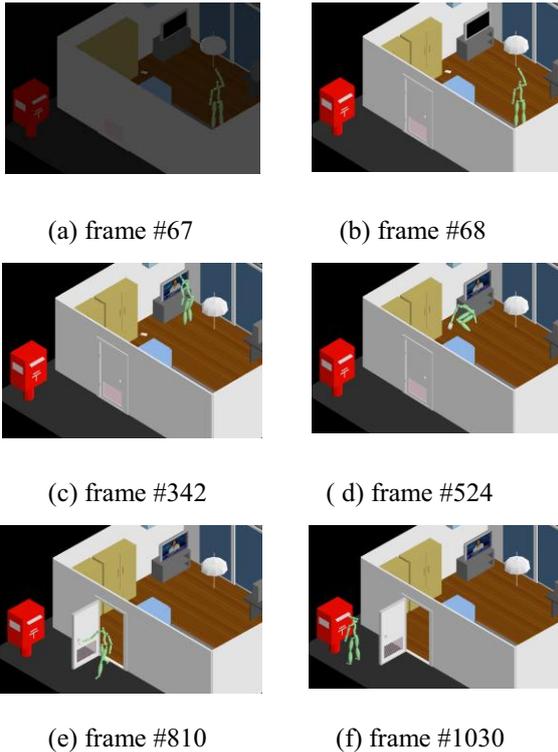


Figure 15. Result of motion data generation

The last movement of “Drop into mailbox” is generated by using the relation between the post card and the mailbox. The character doesn’t drop the card into the mailbox in case that it does not hold the post card. The relation information enables generation of variety of motions. When the television is replaced by an electric fan, as shown in Figure 16, we can also generate another objective movement by only replacing those objects. The movement of the character in Figure 16(b) is different from the movement in Figure 16(a) obviously.

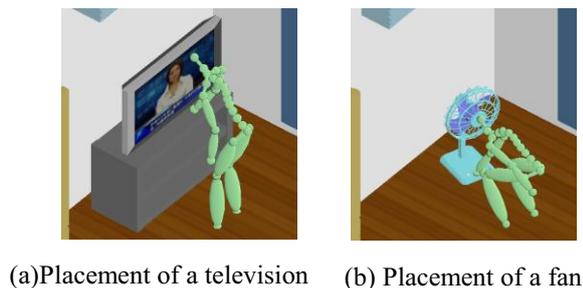


Figure 16. Scene modification

Even if the modification of the scene results from the change of objects, it is unnecessary to capture motion data afresh. This method produces reduction of motion generation cost.

4. CONCLUSION

In this paper, we proposed an automatic generation method of character behavior by only objects placement. We achieved this motion generation by giving the movement of the character to objects. The relation information between the objects enables generation of various motions depending on the object that the character operates. Experimental results showed that our method is effective.

There are some directions for future works. For instance, unnatural walking motion may be generated when the character changes a direction to walk. This problem is caused by using only seven kinds of walking movement, as shown in Figure 7. In order to solve this problem, we could also improve walking motion by introducing Motion Graphs [Kov02] that is widely used to generate natural walking motion. In addition, another future work will focus on adding the function that a character acts in corporation with other characters when several characters exist in three dimensional virtual space.

5. REFERENCES

- [Alb00] Alberto Menache: “Understanding Motion Capture for Computer Animation and Video Games”, MORGAN KAUFMANN PUBLISHERS, pp.14-36, pp.121-142,2000.
- [Kan06] Kang Hoon Lee, Myung Geol Choi, Jehce Lee: “Motion Patches: Building Blocks for Virtual Environments Annotated with Motion Data”, ACM Transactions on Graphics, Vol.25 No.3, pp.898-906,2006.
- [Mar99] Marcelo Kallmann, Daniel Thalmann: “Direct 3D Interaction with Smart Objects”, Proc. of the ACM symposium on Virtual reality software and technology, pp.124-130, 1999.
- [Jam90] James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, *Computer Graphics 2nd edition*, ADDISON-WESLEY, 1990.
- [Tom06] Tomas Akenine-Möller, Eric Haines: “REAL-TIME RENDERING”, Born digital Inc., pp.36-42,2
- [Kov02] Kover L., Gleicher M. and Pighin F.: “Motion Graphs”, ACM Transactions on Graphics, Vol.21, No.3, pp. 473-482, 2002.

RenderMan's Power to Visualization's Rescue

Julio Espinal
jae3161@rit.edu

Virginia Allen
vla8446@rit.edu

Kwesi Amable
kxa9006@rit.edu

Reynold Bailey
rjb@cs.rit.edu

Hans-Peter Bischof
hpb@cs.rit.edu

Department of Computer Science, Rochester Institute of Technology

ABSTRACT

Most visualization systems employ a data flow approach in order to create visual representations of data. The data flows along a directed graph through the different components, gets filtered, extracted, analyzed, and finally converted into an image. Most visualization systems use one graphic toolkit or library to create the image. These toolkits and libraries are not created equally; some are better suited than others to solve given problems. Being able to pick and choose would often generate a better result. Within the Spiegel framework any toolkit, which can be used in a Java environment, can be employed to create the image. In this paper, we explain the Spiegel framework and how Pixar's PhotoRealistic RenderMan® can be used to visualize scientific data.

Keywords

Visualization Framework, RenderMan®, Data Flow Languages.

1. INTRODUCTION

Most visualization systems employ a data flow approach along a directed graph to filter, extract, analyze, and finally convert data into an image. They generally use one specific, unchangeable graphic toolkit or library to create images. The features of these toolkits and libraries vary significantly; some are better suited than others to solve given problems. Simply changing from one toolkit or library to another often produces strikingly different results.

Additionally, visualization systems run on different hardware platforms, which use different drivers to access the graphics card. For example, OpenGL running on two different platforms, Mac OS X and Windows, using the same NVIDIA GeForce 9400 graphics processor will not execute all shaders in the same manner. As a result, the images generated from the same program may differ in quality.

Within the Spiegel framework [Bis05], any toolkit that can be used in a Java environment can be employed in order to create the best possible image. In this paper, we explain the Spiegel framework and how Pixar's PhotoRealistic RenderMan® can be used to visualize scientific data.

The rest of the paper is structured as follows: section 2 discusses general visualization principles; section 3 presents an overview of how a data flow architecture can be used for creating visualizations; a brief survey of related work is presented in section 4; Our approach for incorporating RenderMan® into the

existing Spiegel visualization framework is described in sections 5, 6, 7, and 8. Finally, results and future work are presented in sections 9 and 10 respectively.

2. VISUALIZATIONS

Spiegel was designed as a visualization tool for the Center for Computational Relativity and Gravitation (CCRG) at Rochester Institute of Technology. Spiegel has been used mainly to visualize simulations of galactic events like black hole mergers, gravitational waves, and galaxy mergers. However, it can be used to visualize any type of data. For CCRG, the visualizations created by Spiegel are used to help debug and understand the simulations from which they are generated, as well as explain the science to the general public.

Certain galactic events like black hole mergers cannot be observed in practice. Therefore, a visualization of a black hole merger cannot be compared to a photograph. This makes it relatively easy to generate visualizations because it is not bound to a specific pre-conceived image. On the other hand the Hubble Space Telescope took images of nebulae like the one shown in Figure 2. It is difficult to accurately generate this scene in 3D on a computer.

A typical simulation writes the current state of the model into a file at discrete moments in time. The visualization of scientific data always follows the same rules. The state at successive time steps of the

simulation data is read and subsequently converted to a visual representation.

This can be done for all time steps in parallel if the data of each time step is complete and independent of other time steps. If this is not the case, the data can always be pre-processed. Because of this, visualization systems are excellent candidates for execution on a cluster. The Spiegel framework is no exception. As shown in Figure 1, individual frames of the visualization can be generated in parallel thus reducing execution time.

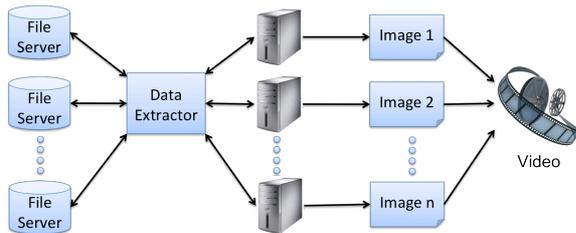


Figure 1: Overview of the Spiegel framework. Data is extracted from one or more file servers and distributed to a cluster of computers. Each processing unit in the cluster generates one (or more) frames of the complete visualization sequence in parallel. These frames are then combined to create a video.

Prior to this work, the Spiegel framework utilized only Java3D or JOGL to create 3D images. However, these libraries were limited to rendering simplistic models, which in some cases, results in unattractive images for a general audience. Java3D and JOGL cannot be pushed to render extremely difficult visual scenes. For example, it is impossible with either library to generate an impressive looking nebula, like the Cat's Eye Nebula shown in Figure 2. The authors do not argue that an image like the Cat's Eye Nebula cannot be generated on a computer, but they argue that Java3D or JOGL are not the right tools with which to solve this problem.



Figure 2: Cat's Eye Nebula. Courtesy of NASA and the European Space Agency. Image generated by the Hubble Space Telescope.

Pixar's PhotoRealistic RenderMan® has been widely used in the computer graphics community for over two decades to create stunning computer generated imagery. RenderMan's reputation has grown over the years and it is still used today to render scenes in many big-budget Hollywood films. Because of its power, huge benefits can be gained by incorporating RenderMan® into existing visualization systems.

3. DATA FLOW ARCHITECTURE

Most current visualization systems utilize a data flow architecture [Bis09]. Components have communication endpoints, which can be connected to form a visualization program. When the program is executed, data is passed from one component to another. Each component performs specific operations that contribute to the final result.



Figure 3: Example of a program created in Spiegel that illustrates the data flow architecture.

Figure 3 shows an example of a program created in Spiegel using its graphical interface. The node Stars reads the file named sim.dat specified as an argument and sends the data to the node Stars3D, and from there the data is sent to the last node in the graph, Camera3D. The Stars, Stars3D, and Camera3D components are simply small programs, which perform specific operations on the data.

Most visualization frameworks, like Iris Explorer, the grandfather of all visualization systems [Fou95], do not expose this functionality to the user. Vish [Ben07] and Spiegel [Bis09] are frameworks that expose this functionality to the user; consequently, they are very easy to extend.

The Unix operating system [Rit74] allows one to create a data flow architecture using pipes. This allows for the connection of multiple simple programs to create powerful systems. But more than this, it fosters the reuse of existing components. This increases the productivity of a developer. A Unix program like:

```
sort file | uniq | sort -n | head -5
```

will print out the 5 most often occurrences of the same line in file.

Vish and Spiegel follow the same philosophy as Unix. In the end, this allows for the use of any tool/library that can convert data into an image.

The authors explored OpenGL, JOGL, and PhotoRealistic RenderMan® within the Spiegel visualization framework.

4. RELATED WORK

Other options besides RenderMan® exist for rendering realistic images. These include OpenGL with GLSL shaders and DirectX. Because Spiegel was designed to be platform independent, DirectX was an impractical choice because it is not fully supported on all platforms. Additionally, not every extension in OpenGL is supported on all graphics cards. These factors led us to consider RenderMan®. Due to the intuitive shading language and film-quality rendering, RenderMan® is superior to OpenGL. Even though it takes more time to render an image, the quality is significantly better and appeals to a general audience. The RenderMan® Interface is well documented and its reputation has been proven in the field for over twenty years. RenderMan® automatically performs many calculations that need to be performed manually in OpenGL. For example, with lighting enabled, the normal and view vectors are automatically calculated. Furthermore, setting up the camera and the scene is easier compared to OpenGL. The RenderMan® standard defines five types of shaders: surface, light, volume, imager and displacement; on the other hand, GLSL only supports vertex and fragment shaders. RenderMan's shaders have a very modular design; therefore, it is possible to edit certain parts of the pipeline without affecting other aspects. It is also possible to have multiple variations of a base shader, which facilitates the evaluation of the effects. Scene setup is also easier in RenderMan® as parameters can be added to a RIB (RenderMan® Interface Bytestream) file to guide scene generation as opposed to explicitly defining the scene in OpenGL.

5. RENDERMAN®

As stated previously, part of RenderMan's appeal is its modular design and multiple shader types. They can also be layered together to create unique textures. Once a shader is compiled, it can be used in any RenderMan® Interface Bytestream (RIB) file. A RIB file describes the environment and the various objects within a scene. RIB files can reference other RIB files in order to add existing objects to other scenes.

In many cases, the data set requires much processing time to produce a movie. The processing time increases drastically when rendering high-quality, photorealistic scenes. The Spiegel framework allows for distribution over a cluster, as shown in Figure 1, to generate the images in parallel, which reduces execution time.

Spiegel splits the RenderMan® interface into several components. These components include lighting, shader extractor, RIB generator, and camera settings. Because RenderMan® is very flexible, it is possible

to have multiple instances of most of the components. As data flows through each component, a RIB file depicting the scene is generated and ultimately processed by the PhotoRealistic RenderMan® renderer to produce the desired image.

6. ARCHITECTURE

A modeling application is used to create and compile the RIB file. During compilation, the modeling application will parse each line of the RIB file and call the corresponding RenderMan® Interface (RI) routine. Once all of the information is gathered, RenderMan® will then bound and split each primitive. Figure 4 illustrates all the phases involved in the architecture.

During the bound and split phase, each primitive is checked whether or not it is within the bounding box. The bounding box is the viewing area in which the scene will be depicted. It is based on the current location of the camera and the size of the screen. If an entire primitive is not within the bounding box, then it is discarded; however, if a primitive is partially in the bounding box, then it is split. When a primitive is split this means that it is made into smaller polygons until a single one can fit into the bounding area. This can be seen in Figure 5 when a sphere is split into smaller polygons that create the whole sphere. Once the smaller polygons of the primitive fit into the bounding box, the polygons that are still not within the bounding box are discarded.

Once each primitive is bound and split, they are diced into a grid of micro-polygons. These micro-polygons will be small enough to approximately represent a pixel on the screen. As seen in Figure 5, these grids will allow for the shaders to manipulate the primitives. The first shader applied, if one is specified, is the displacement shader. These shaders need to be applied first because they manipulate the vertices' data, such as the position or normals, and this information is a basis for other shaders. Once the displacement shader is applied, the surface shaders are used next to manipulate the surface of the primitive. In order to apply the surface shaders, the lighting also needs to be taken into account to produce appropriate shadows. The location of the lights also needs to be considered, because if a light is directed towards a primitive, then the surface shader needs to adjust the color according to the type of surface and make that area brighter than the rest of the object's surface. Last, the atmosphere shader is applied in order to make changes to the primitive's color along with its opacity. After the objects are bounded and split, diced, and shaded, the image is rendered and displayed onto the screen.

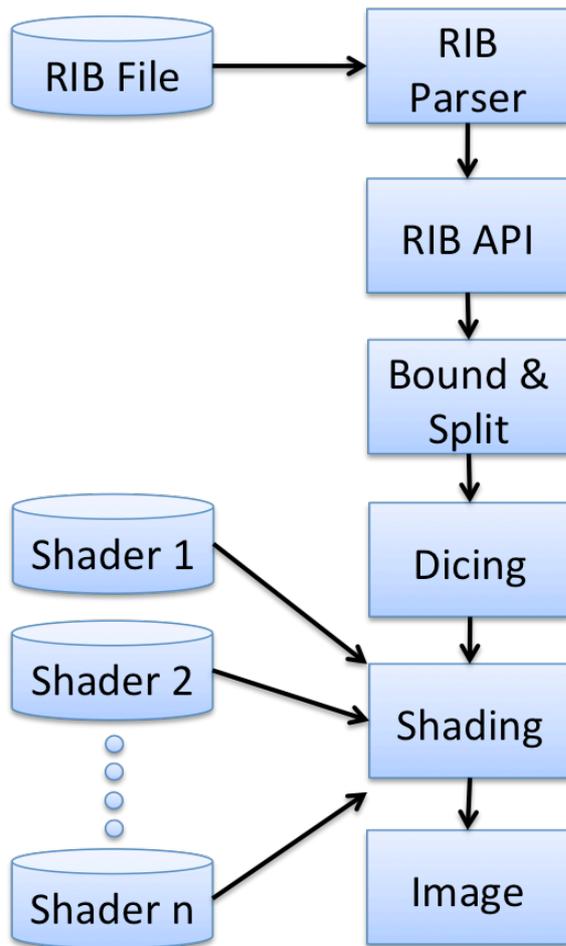


Figure 4: The stages involved in the RenderMan® architecture.

7. RENDERMAN® PROGRAM

Consider the example of trying to render stars in a galaxy. The following snippet of code shows part of the RIB file that is generated:

```

...
TransformBegin
  Translate -0.1 0.6 -0.3
  Scale 0.1 0.1 0.1
  Color [0.46 0.46 0.4]
  Surface "glow"
  "attenuation" "2"
  Sphere 1 -1 1 360
TransformEnd
TransformBegin
  Translate 0.1 -0.3 0.4
  Scale 0.2 0.2 0.2
  Color [0.0, 0.0, 0.0]
  Sphere 1 -1 1 360
TransformEnd
...

```

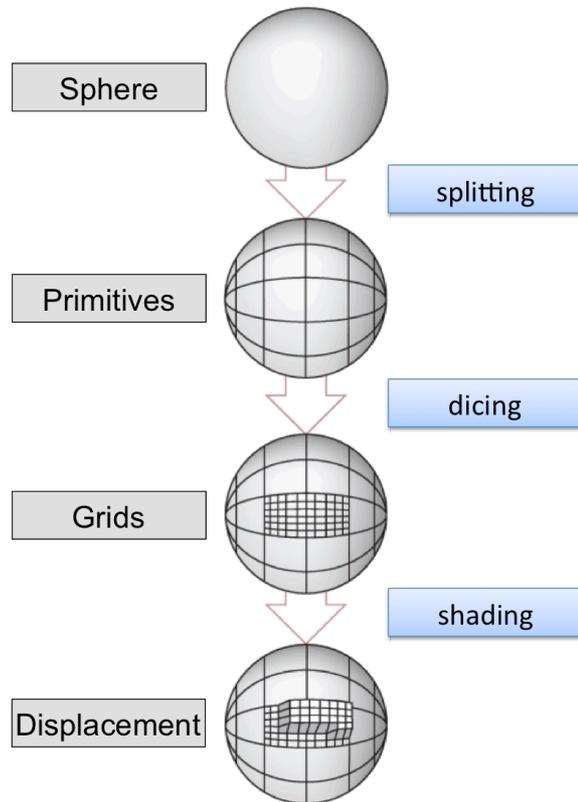


Figure 5: Illustration of a sphere being split, diced, and shaded. Image adapted from renderman.pixar.com [Pix09].

The first piece of information, between the first TransformBegin/End, describes the characteristics of one star within the image. The surface of a star is described by using a shader called “glow”; this can be seen on the “Surface” line. The second piece of information is for a black hole. The scale of a black hole is slightly larger than the stars and the color is black.

8. SHADERS

The key to generating realistic images from RenderMan® is shaders. A shader is a function written in the RenderMan® shading language that calculates the color and position of a point on the surface of the object. The RenderMan® plug-in for Spiegel allows the user to select the shader from a file. The program will parse the header of the shader file to determine the parameters it takes. It will then dynamically add an input to Spiegel’s shader module for each of these parameters. This module contains the name of the shader and a list of variables with their values. The camera module generates the main RIB file. It imports the previously generated RIB file that contains the models. After generating the RIB file, the RenderMan® renderer (*prman*) is invoked to produce an image. This process is illustrated in

Figure 6. The module gets the camera position, image size, and the render quality in as parameters. It also has parameters for information about the interpolation and the motion blur. To create an interpolated movie, the program reads each time step until it has four time steps, it will then render all of the frames that should go in between these four time steps.

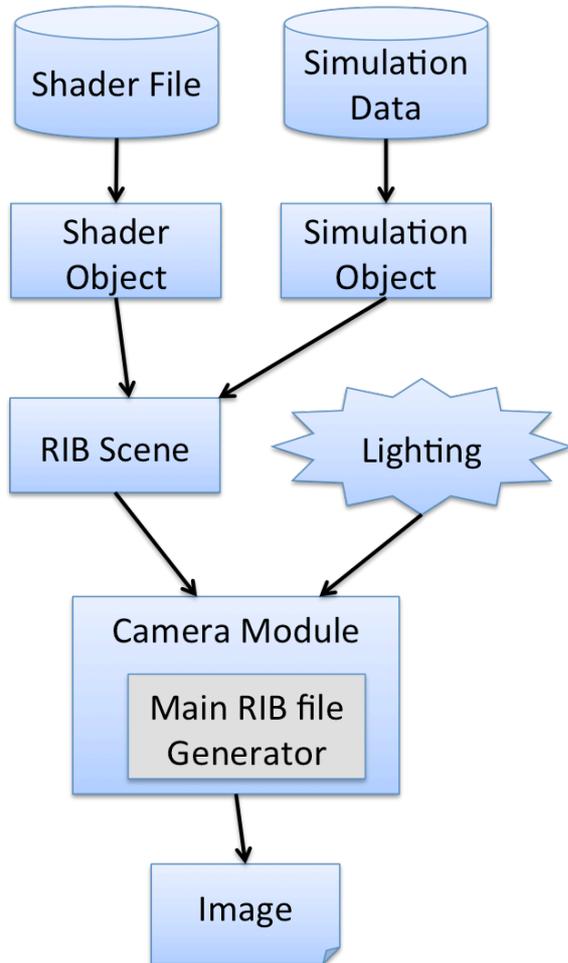


Figure 6: Illustration of how the main RIB file is built and used.

There are several Spiegel modules for RenderMan® lighting. These modules add support for ambient, distant, spot, and point lights. To add a light, connect the light module to the RenderMan® camera module. The “lights” input supports the connection of multiple lights at the same time. The parameters of the lights can be changed via Spiegel's interface. These parameters include light intensity and color along with others depending on the type of light. It is important to note that some shaders, do not use lighting to determine how to render the objects. This means that, when using these shaders, adding lights will have no effect on the final image.

9. RESULTS

The Spiegel framework was used to create video clips of black hole mergers for the show “The Universe: Cosmic Holes” which aired on the History Channel in 2008. The videos were rendered using OpenGL and depicted black holes as simple Gouraud shaded spheres against a static texture mapped background. Figure 7 (left) shows a single frame of a three black hole merger that was rendered using the old Spiegel/OpenGL approach. Figure 7 (right) shows an image that was rendered using the new Spiegel/RenderMan framework. In this case, the individual stars surrounding the central black hole are rendered using a shader which gives a more realistic glowing effect.

We generated images based on a simulation of a three-galaxy merger. Figure 8 shows one frame of the merger viewed from the side and Figure 9 shows the merger viewed from the top. For these images, a different shader which emphasizes the appearance of the back holes was used.

10. FUTURE WORK

RenderMan has been successfully incorporated into the Spiegel visualization framework and has been used to create visualizations of galactic events such as black hole mergers. The new framework allows for distribution over a cluster. This was successfully verified for a small cluster. In the future, we will have access to Blue Waters [NCS09]. Blue Waters will consist of 100,000 nodes and the peak performance will be in the Peta-flop range. The Spiegel framework will be ported to this cluster and its scalability will be analyzed.

Sonification [Her05], the art of representing data by using sound, is a rapidly evolving area of research. We plan to explore various approaches for using sonification models to further enhance our visualizations.

Many visualization algorithms are designed to visualize a very specialized problem. Unfortunately these algorithms cannot be used outside the tool in which they are implemented. A language named *Sprache* is used to describe a visualization program in Spiegel [Bis05]. However, it is not well suited for working with data that is distributed over multiple servers. We plan to redesign this language to handle distributed data and distributed rendering for the new Spiegel/RenderMan framework.

Finally, one of the major limitations of our project was the time it took to render images. The use of a cluster to render individual frames in parallel helps to reduce the overall rendering time for a video sequence, however each individual frame could potentially take a long time. Although PhotoRealistic

RenderMan is an efficient software renderer, it is still subject to long processing times for complex scenes. We plan to explore the use of multi-core GPUs to speed up the rendering time.

11. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Award No. CCF-0851743. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and not necessarily reflect the views of the National Science Foundation.

The authors would also like to thank to Manuela Campanelli, Carlos Lousto and Yosef Zlowocher from the Center for Computational Relativity and Gravitation for their help and fruitful discussions. This team provided the data used and the interpretation of the visual representation.

12. REFERENCES

- [Ant00] A. A., & Larry, G. (2000). Basic Geometric Pipeline. In *Advanced RenderMan: creating CGI for motion pictures* (pp. 136-143). San Diego, CA: Academic Press.
- [Ben07] Werner Bengler and Georg Ritter and René Heinzl, *The Concepts of VISH*, 4th High-End Visualization Workshop, Obergurgl, Tyrol, Austria, June 18-21, 2007, 978-3-86541-216-4.
- [Bis05] Hans-Peter Bischof, Jonathan Coles: *A Movie Is Worth More Than a Million Data Points*, *Lecture Notes in Computer Science*

Publisher: Springer-Verlag GmbH, ISSN: 0302-9743 Subject: Computer Science Volume 3514/2005, Title: Computational Science ICCS 2005: 5th International Conference, Atlanta, GA, USA, May 22-25, 2005

- [Bis09] Hans-Peter Bischof, Swathi Annamala: *The KISS Principle Applied to Dataflow Languages Paradigms for Visualization Frameworks*, *Proceedings of the 2009 Conference on Modeling Simulations and Visualization Methods*, p. 48-55, ISBN: 1-601320-120-1.
- [Fou95] David Foulser. Iris explorer: a framework for investigation. *SIGGRAPH Computer Graphic*, 29(2):13{16, 1995.
- [Her05] Thomas Hermann, Andy Hunt, "Guest Editors' Introduction: An Introduction to Interactive Sonification," *IEEE MultiMedia*, vol. 12, no. 2, pp. 20-24, Apr. 2005, doi:10.1109/MMUL.2005.26.
- [NCS09] Blue Waters Announcement. Retrieved October 20, 2009, from NCSA's website <http://www.ncsa.illinois.edu/BlueWaters>.
- [Pix09] Pixar's RenderMan Performance. (2009). Retrieved November 30, 2009, from Pixar website: https://renderman.pixar.com/products/whats_renderman/2.html
- [Rit74] D. M. Ritchie and K. Thompson. The Unix time-sharing system. *Communications of the ACM*, 17:365-375, 1974.

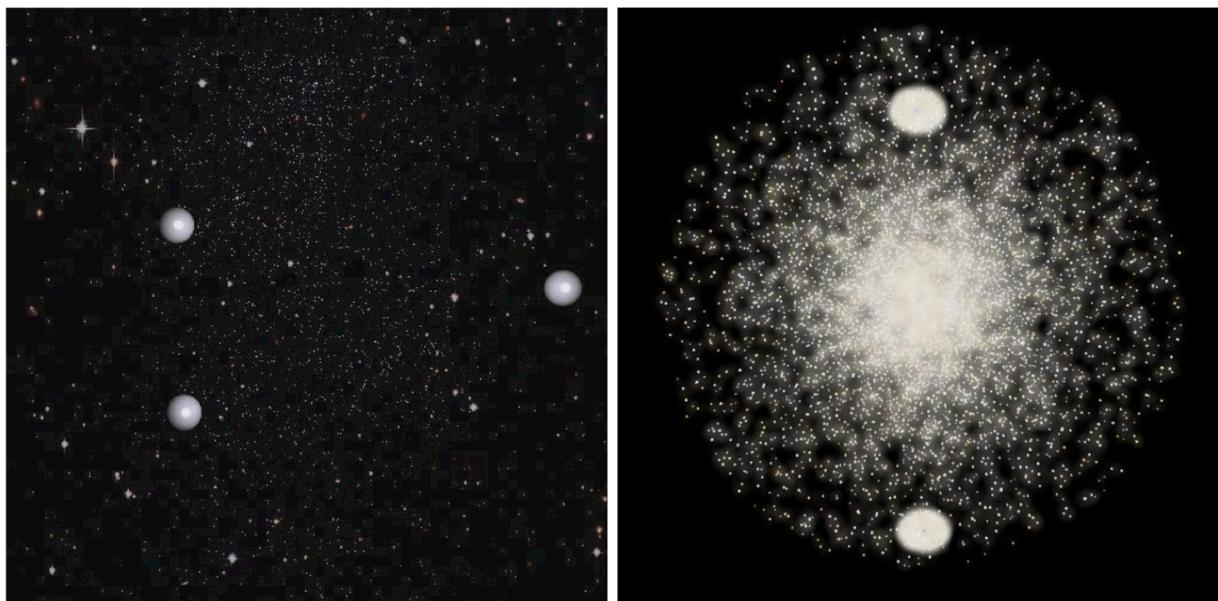


Figure 7: Image rendered using old Spiegel/OpenGL framework (left). Image rendered using new Spiegel/RenderMan® framework using shaders.

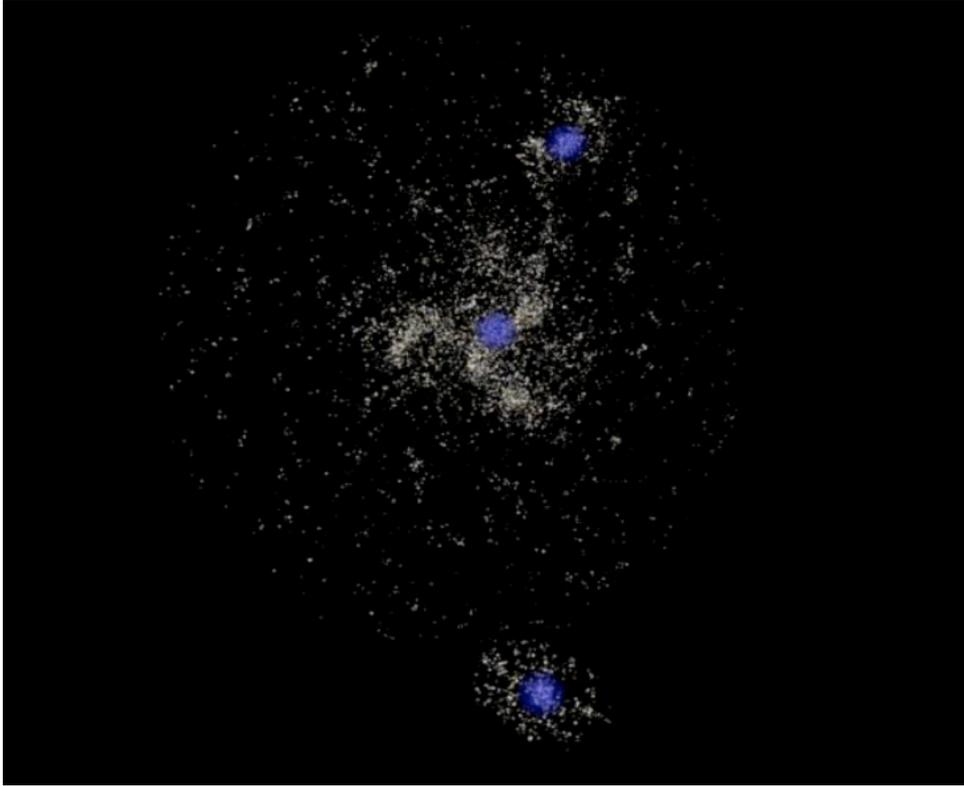


Figure 8: One frame from a three-galaxy merger viewed from the side. Image created by the Spiegel Visualization System using RenderMan®.

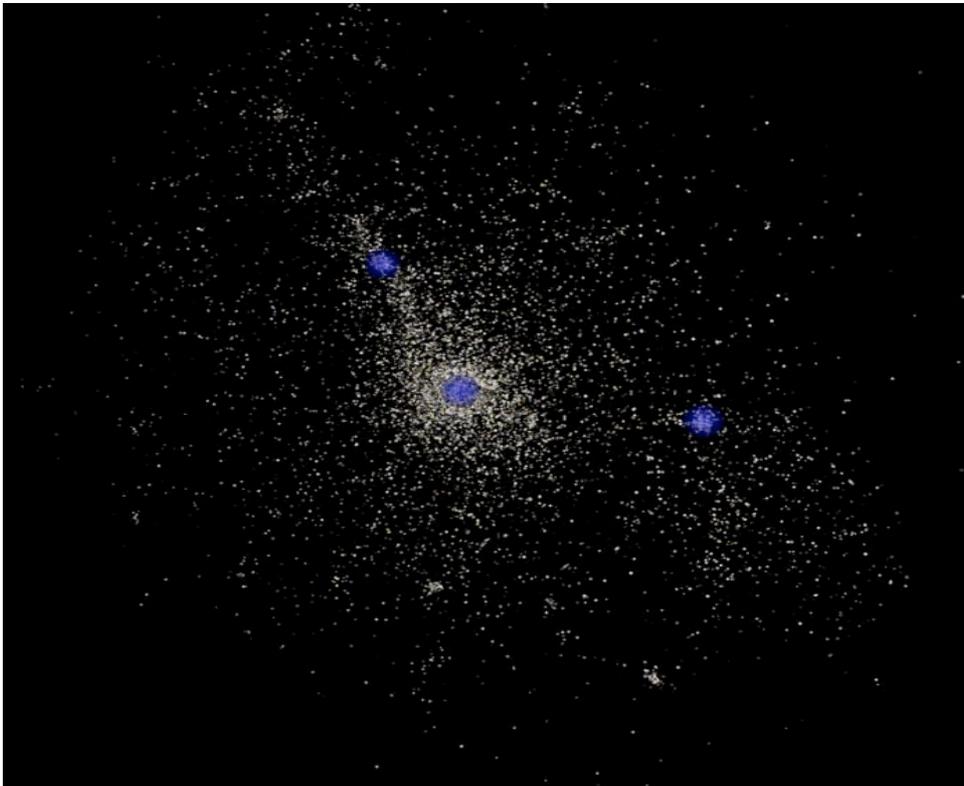


Figure 9: One frame from a three-galaxy merger viewed from the top. Image created by the Spiegel Visualization System using RenderMan®.

Image Authentication Using Robust Image Hashing with Localization and Self-Recovery

Ammar M. Hassan,
Ayoub Al-Hamadi, Bernd Michaelis
IESK
Otto-von-Guericke-University
Magdeburg, Germany
{Ammar, Al-Hamadi}@ovgu.de

Yassin M. Y. Hasan
Computer Sc. and Info. Dept.
Taibah University
Madinah, KSA
ymyhasan@aun.edu.eg

Mohamed A. A. Wahab
Electrical Engineering Dept.
Minia University
Minia, Egypt

ABSTRACT

The rapid growth of efficient tools, which generate and edit digital images demands effective methods for assuring integrity of images. A semi-fragile block-based image authentication technique is proposed which can not only localize the alteration detections but also recover the missing contents. The proposed technique distinguishes content-preserving manipulations from the content alterations using secure image hashing instead of cryptographic hashing. The original image is divided into large blocks (sub-images) which are also divided into 8×8 blocks. Secure image hashing is utilized to generate the sub-image hash (signature) which may slightly change when the content-preserving manipulations are applied. Furthermore, the sub-image code is generated using the JPEG compression scheme. Then, two sub-image hash copies and the sub-image code are embedded into relatively-distant sub-images using a doubly linked chain which prevents the vector quantization attack. The hash and code bits are robustly embedded in chosen discrete cosine transform (DCT) coefficients exploiting a property of DCT coefficients which is invariant before and after JPEG compression. The experimental results show that the proposed technique can successfully both localize and compensate the content alterations. Furthermore, it can effectively thwart many attacks such as vector quantization attacks.

Keywords

Cryptographic hashing, image authentication, image hashing, watermarking.

1. INTRODUCTION

The current advances in information technology, the widespread multimedia applications and wireless services require efficient methods for guaranteeing privacy, security, protection and integrity of the assorted multimedia data categories. Since many recently developed devices and efficient software products offer consumers worldwide capabilities of flexibly creating, manipulating, and exchanging multimedia data, considerable efforts and contributions have been lately made on digital watermarking that inserts a piece of information (the watermark) into multimedia (host/cover) data for many purposes such as [Has04, Has07, Won01]:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

image authentication, copyright protection, fingerprinting, broadcast monitoring and data hiding.

For example, in medical archiving and e-commerce, we strongly desire to be sure that the images are genuine and in the news reporting, it is important that the image truthfully reflects the real view at the time of capture [Lan99, Won01]. For image authentication purposes, it is required that the watermarking algorithm is blind, secure and so sensitive that slight modifications to the image content are detected and precisely localized [Lin99, Yeu97]. Fragile [Won98, Bar02, Cel02], semi-fragile [Eki04, Lin00, Lin01a, Lin07, Mae06], self-recovery/embedding [Fri99b, Lin01b, Lue08, Wan08] watermarking schemes have recently been presented for image authentication.

Fragile image authentication schemes are so sensitive to pixel changes where their watermarks are easily damaged even in case of harmless changes in the image data due to content-preserving manipulations that do not affect the content [Lin99]. Hence, fragile image authentication is applicable and of interest only in case of lossless environment, i.e., coding, storage, transmission (of the watermarked image). The fundamental objective of the attacker facing such fragile watermark is to keep a watermark that makes

his/her altered or completely forged image, “pass” the verification test as authentic [Has04, Has07, Lue08]. Block-based fragile/semi-fragile image authentication schemes provide attack localization but they are vulnerable to vector quantization (VQ) attacks [Hol00], relying on that the watermark embedding/verification processes are run on independent blocks. Once an attacker has a table of authenticated blocks (with the same security parameters), he/she can use the best-authenticated approximation of an un-authenticated block without having the verification process detecting his/her alterations. This type of attack principally differs from the attacks against copyright protection and information hiding where the attacker may mainly want to significantly distort or remove the watermark with imperceptible alterations in the image [Kut00, Kir02].

Various global and block-based (sized down to pixel-wise) fragile image authentication methods have been developed. A simple fragile scheme simply replaces the least significant bits (LSBs) of the image of interest with the checksum (i.e., modulo-2 addition) bits of a long word of some most significant bits (MSBs) [Lin99]. In [Yeu97], the use of a user-defined color look-up tables (LUTs) guided pixel-wise adjustment to embed the watermark is proposed. Wong’s block-based method [Won97] and its public-key modified versions [Won98, Won01] replace the LSBs of each block with a signature of its MSBs, with the image size, image index and/or block index, xor-ed with its corresponding watermark block.

On the other hand, semi-fragile image authentication techniques embed watermarks so robustly to survive (to some, application dependent, extend) various kinds of typical image processing manipulations such as lossy compression as long as the image contents are preserved. At the same time, embedded watermarks must detect malicious alterations such as deleting or adding an object. In many semi-fragile schemes, the relations between pairs of discrete cosine transform (DCT) coefficients in a block are used as the block signature. Then, the signatures (watermarks) are robustly embedded in low frequency coefficients [Lin00, Lin01a]. In [Mae06], the authors introduce two methods to generate the signatures using the discrete wavelet transform (DWT). In the first method, random values are added to the difference between two coefficients before the difference is encoded to generate the signature bit. The second method proposes the use of a multiple nonuniform quantizer to encode the coefficient difference in each pair. Lin et. al. use the differences of DCT coefficients as signatures and modify other DCT coefficients to match the signatures [Lin07].

Furthermore, to not only localize altered regions but also compensate for the damage, self-recovery/embedding image authentication techniques have

been presented that embed an image approximation into the image itself in a fragile [Lue08, Wan08] or semi-fragile [Has07, Fri99a] way using various techniques.

An original self-recovery/embedding image authentication technique based on JPEG compression has been introduced in [Fri99b]. A JPEG compressed version of each block B is inserted into the LSBs of the block $B + \vec{P}$, where \vec{P} is a vector of length approximately 1/3 of the image size, with a randomly chosen direction. The algorithm limitations and possible attacks are addressed in [Fri99c, Lue08]. In [Lin01b], Lin and Chang have proposed an algorithm using quantized coefficients of the DCT of the image blocks as a watermark and modifying the coefficients differences to match the quantized coefficients (watermark). The attacker can easily defeat the verification process applying the same algorithm into a fake image. Instead of using a JPEG compression version as an image approximation, Wang and Tsai have used fractal codes of a ROI (region of interest), which is chosen as the important object in the host image [Wan08]. On the other hand, Lue et al. proposed a technique that uses a halftone version of the host image as an approximation image [Lue08].

In this paper, image hashing technology, which will be described in the next sections in details is utilized to generate the sub-image signature. A code of the approximated sub-image is computed using the principals of JPEG compression. Then, the sub-image signature copies and the sub-image code are robustly embedded into DCT coefficients of two relatively-distant sub-images making a doubly linked chain.

The remainder of this paper is organized as follows: cryptographic hashing, which is mostly used to generate image/block signature in fragile algorithms, and image hashing, which we adopt to generate the proposed signatures, are described in Section 2. In Section 3, existing image hashing schemes are presented. The proposed technique is introduced in Section 4. Experimental results are shown in Section 5. In Section 6, the conclusion is presented.

2. CRYPTOGRAPHIC HASHING AND IMAGE HASHING

The cryptographic hash functions such as MD4, MD5, and SHA [Men01, Sch96] map the input data to a short fixed length string. For the hash function H_c and the input data d , it should be easy to compute the hash $h_c = H_c(d)$. For this type of functions, called one-way-functions, it is too hard to estimate the input data d from the hash h_c . Hash functions have, at least, the following additional properties [Men01, Sch96]:

- Given the hash h_c , it is computationally infeasible to find an input which hashes to that output, i.e. it is hard to find d such that $H_c(d)=h_c$.
- Given the data d , it is hard to find another input data d_0 which hashes to the same output, i.e. it is hard to find $H_c(d)=H_c(d_0)$.
- It is computationally infeasible to find any two inputs d_0 and d_1 which have the same output (i.e., satisfying collision resistance).

It is clear that the cryptographic hash is so sensitive to changes in the input data where small changes, even a single bit, dramatically change (~50%) the output. To secure the hash, it may be encrypted by an encryption algorithm. The cryptographic hash is mostly used for digital signatures and fragile image authentication.

On the other hand, the image (visual) hash function H maps the input image (or sub-image) to an output $h=H(I)$ that is invariant under perceptually insignificant image changes with the following main properties[Fri00, Mih01, Swa06, Ven00, Tan08]:

- It is hard to find two different images having the same or very close hash value(s) (collision resistance).
- Given h , perceptual changes to an image I lead to a different hash $H(I') \neq H(I)$.
- The hash is key dependent, for security reasons, so that different keys give significantly different hash values.

The main difference between image (visual) and cryptographic hashing is that image hashing accepts perceptually insignificant changes in the input image with small hash changes; but small changes in the input data lead to very significant changes in the cryptographic hash.

3. IMAGE HASHING SCHEMES

In [Fri00], The image hash is generated by projecting the input image onto patterns which are generated using a zero-mean uniform distributed key random generator. The resulting hash is resilient to many normal operations but it is not collision free [Swa06]. Venkatesan et al. have introduced an image hashing algorithm that uses the discrete wavelet transform (DWT) of an image. Statistics of each subband block are calculated, randomly quantized and encoded to generate the final hash value [Ven00]. Unfortunately, the algorithm does not work well for object insertion. In [Mih01], the DWT is employed to capture the image hash based on thresholding and iterative filtering. Swaminathan et al. [Swa06] have exploited the Fourier-Mellin transform to generate image features. In the polar coordinate, the summation of image values along angle axis at equal distant points for a specific radius is an image feature. The image features for radii are represented as the image hash.

In [Tan08], a robust image hash algorithm uses a non-negative matrix factorization (NMF) scheme for generating the image hash. First, the image undergoes preprocessing as a sequence of image resizing, color space conversion and low-pass filtering. The preprocessed image is then divided into unequal blocks. Next, each block is rescaled to a fixed size and put as a vector in a matrix that is undergone NMF. The elements of the NMF coefficient matrix are quantized and encoded to generate the image hash. We use this algorithm to generate the sub-image signature in our proposed image authentication technique. So, we describe it in more details in the rest of this section. The scheme is composed of the following four main steps:

First step: Image preprocessing

- The image is resized to $q \times q$ using bi-linear interpolation.
- The color space of $q \times q$ image is converted to $YCbCr$.
- The Y plane is passed through a low-pass filter.

Second step: Building the secondary image

- The preprocessed image U is randomly divided into t strips, and each strip is again divided into t blocks with varied sizes, resulting in $t^2=N_b$ blocks in total.
- Each block is resized to $k \times k$ using bi-linear interpolation.
- Each $k \times k$ block is stacked to construct a $k^2 \times 1$ vector v .
- Each vector v is used as a column in a pseudo-random order to form the $m \times n$ matrix V , where $m=k^2$. V is called the secondary image.

Third step: Data reduction

- V undergoes NMV giving the coefficient matrix C (see the appendix).
- C entries are quantized to generate a binary matrix C^b as follow:

$$c_{l,j}^b = \begin{cases} 0, & c_{l,j} \leq c_{l,j+1} \\ 1, & c_{l,j} > c_{l,j+1} \end{cases} \quad (1)$$

where $c_{l,j}$ denotes the entry of C in the l^{th} row and the j^{th} column, and $c_{l,n+1} = c_{l,1}$.

Final step: Hash security

- C^b entries are concatenated to form a binary string.
- The binary string is interleaved using a key to produce a key-dependent image hash h .

4. PROPOSED TECHNIQUE

Image hashing is employed to generate the sub-images' hashes (signatures) which are used to check the authenticity of an image. Two signature copies of each sub-image are robustly embedded into two

$G_{kl}(j)$								$G_{k2}(i_1)$								$G_{k3}(j)$								$G_{k4}(i_2)$									
1	2	3	4	5	6	7	8	25	34	19	36	21	30	39	24	2	30	39	24	25	34	19	36	21	3	29	38	47	32	33	42	27	44
9	10	11	12	13	14	15	16	33	42	27	44	29	38	47	32	3	29	38	47	32	33	42	27	44	4	52	37	46	55	40	41	50	35
17	18	19	20	21	22	23	24	41	50	35	52	37	46	55	40	4	52	37	46	55	40	41	50	35	2	54	7	48	49	2	43	4	45
25	26	27	28	29	30	31	32	49	2	43	4	45	54	7	48	2	54	7	48	49	2	43	4	45	3	12	53	6	15	56	1	10	51
33	34	35	36	37	38	39	40	1	10	51	12	53	6	15	56	3	12	53	6	15	56	1	10	51	4	5	14	23	8	9	18	3	20
41	42	43	44	45	46	47	48	9	18	3	20	5	14	23	8	4	5	14	23	8	9	18	3	20	2	22	31	16	17	26	11	28	13
49	50	51	52	53	54	55	56	17	26	11	28	13	22	31	16	2	22	31	16	17	26	11	28	13									

Figure 1. Example of the proposed scheme for choosing relatively-distant sub-images.

(a) Original sub-images. (b) Sub-images after $G_{kl}(j)$ column-wise circular shifts. (c) Sub-images after $G_{k2}(i_1)$ row-wise circular shifts.

relatively distant sub-images which are pseudo-randomly chosen using a doubly linked chain in low frequency DCT coefficients. In the proposed technique, the image of interest is divided into sub-images. The sub-image hash (signature) is computed using the secure image hashing algorithm [Tan08] and the sub-image code, which represents the approximated sub-image is generated using the JPEG compression principles. Then, the sub-image hash copies and the sub-image code are robustly inserted into relatively distant sub-images. In the next subsections, the embedding and the verification processes are described in details.

Embedding Process

The original $M \times N$ image I is divided into $m' \times n'$ sub-images as follows:

$$I = \{SI_{1,1}, SI_{1,2}, \dots, SI_{2,1}, SI_{2,2}, \dots, SI_{\lceil M/m' \rceil, \lceil N/n' \rceil}\} \quad (2)$$

where $m' \bmod 8=0$ and $n' \bmod 8=0$, $\lceil x \rceil$ is the floor of x . For each sub-image $SI_{i,j}$, the sub-image hash $h_{i,j}$ is computed using the secure image hashing algorithm [Tan08] such that:

$$h_{i,j} = H(SI_{i,j}) \quad (3)$$

To compute the sub-image code $C_{i,j}^s$, the sub-image $SI_{i,j}$ is resized to 8×8 . Then, the resized sub-image is undergone the DCT. The DCT coefficients are quantized using the quantization table which corresponds to 50% quality JPEG compression. Then, the quantized coefficients are encoded using a fixed bit allocation table to generate the sub-image code.

Each sub-image is divided into 8×8 blocks as follow:

$$SI_{i,j} = \{b_{1,1}^{i,j}, b_{1,2}^{i,j}, \dots, b_{m'/8, n'/8}^{i,j}\} \quad (4)$$

Two hash copies and the code of each sub-image are robustly inserted and spread into two relatively distant sub-images generating a doubly linked chain. In the color images, we use the Y channel of the $YCbCr$ color space for embedding the hash copies and codes. The choice of the two relatively distant sub-images depends on the sub-image index and it is controlled by secret keys as follows:

$$\begin{aligned} i_1 &= (i + G_{k1}(j)) \bmod M_s + 1, G_{k1}(j) \in \llbracket M_s/3 \rrbracket, \llbracket 2M_s/3 \rrbracket \\ j_1 &= (j + G_{k2}(i_1)) \bmod N_s + 1, G_{k2}(i_1) \in \llbracket N_s/3 \rrbracket, \llbracket 2N_s/3 \rrbracket \\ i_2 &= (i + G_{k3}(j)) \bmod M_s + 1, G_{k3}(j) \in \llbracket M_s/3 \rrbracket, \llbracket 2M_s/3 \rrbracket \\ j_2 &= (j + G_{k4}(i_2)) \bmod N_s + 1, G_{k4}(i_2) \in \llbracket N_s/3 \rrbracket, \llbracket 2N_s/3 \rrbracket \end{aligned} \quad (5)$$

where G_{k1} , G_{k2} , G_{k3} and G_{k4} are key seed random generators with keys $k1$, $k2$, $k3$ and $k4$. M_s and N_s are the number of sub-images per column and row, respectively.

Fig. 1 illustrates an example of the indices of the first relatively distant sub-image for each sub-image after $G_{kl}(j)$ column-wise circular shifts followed by $G_{k2}(i_1)$ row-wise circular shifts.

Then, each block of distant sub-images is transformed to the frequency domain using the DCT. We robustly embed two hash copies of the sub-image and the sub-image code (sub-image approximation) into the two relatively distant sub-images. One copy is embedded into the first distant sub-image blocks and the other copy into the second distant sub-image blocks. Furthermore, we divide the sub-image code into two groups which are embedded into the two distance sub-images. For embedding a bit of a sub-image hash copy or a bit of a sub-image code, we use a proved theorem given in [Lin00]. The theorem explains that if a DCT coefficient is quantized by $Q_{qf}(v)$ (qf refers to the compression quality factor), this coefficient can be reconstructed after JPEG compression with $qf_l > qf$. Depending on this theorem, we can embed a bit into a DCT coefficient using an arbitrary quantization step and we can also recover this bit even if JPEG compression is applied with a quality factor greater than the quality factor, which is used in the embedding operation. Therefore, if we arrange the DCT coefficients of a block in zigzag order, the chosen coefficient of the block $b_{u,v}^{i,j}$, which has an index (u,v) in the sub-image $SI_{i,j}$, is modified as follows:

$$mb_{u,v}^{i,j}(l) = \begin{cases} \left[\frac{b_{u,v}^{i,j}(l)}{Q_m(l)} \right] Q_m(l), & \left[\frac{b_{u,v}^{i,j}(l)}{Q_m(l)} \right] \bmod 2 = h_{i,j}(t) \\ \left[\frac{b_{u,v}^{i,j}(l)}{Q_m(l)} + \text{sign} \left(\frac{b_{u,v}^{i,j}(l)}{Q_m(l)} - \left[\frac{b_{u,v}^{i,j}(l)}{Q_m(l)} \right] \right) \right] Q_m(l), & \text{else} \end{cases} \quad (6)$$

where $mb_{u,v}^{i,j}$ is the modified block, Q_m is the specific quantization table, $\lceil x \rceil$ is the round of x , l is the chosen middle frequency coefficient index, t is the hash index, $\text{sign}(x)$ is equal 1 if x is a positive value and it is -1 if x is a negative value. Using (6), we can embed the bits of the hash and also the bits of the code into low frequency coefficients, which have pre-specific indices. A sub-image hash copy and the first group of the code are embedded into chosen coefficients of the

first distant sub image blocks. This operation is repeated for embedding another copy of the sub-image hash and the second group of the code into other chosen coefficients of the second distant sub-image blocks. After embedding the hashes and codes of all sub-images, the DCT coefficients are converted back to pixel integer domain. There is a possibility for losing some embedded bits by the rounding and truncation which are used for converting to the pixel domain. Therefore, we use an iteration procedure to assure the embedded bits are exactly extracted from the authenticated image.

Verification Process

In the verification process, the alterations that may occur on an authenticated image are not only detected and localized but also repaired. In the verification process, the test image I' is divided into sub-images and each sub-image hash $h'_{i,j}$ is computed. For each sub-image $SI'_{i,j}$, the corresponding distant sub-images indices are computed using (5). The embedded hash copy $he^1_{i,j}$ and the first group of the sub-image code are extracted from the first distant sub-image $SI'_{i1,j1}$. A bit is extracted as follows:

$$he^1_{i,j}(t) = \left[\frac{b_{u,v}^{i1,j1}(l)}{Q_m(l)} \right] \bmod 2 \quad (7)$$

where $b_{u,v}^{i1,j1}$ is the block, which has an index (u,v) , in the sub-image $SI'_{i1,j1}$, and $Q_m(l)$ is the quantization step. The other hash copy $he^2_{i,j}$ and the second group of the code are extracted by the same method from the second distant sub-image $SI'_{i2,j2}$. The two groups of the code are combined together to be the extracted code $C^e_{i,j}$ of the sub-image $SI'_{i,j}$. To evaluate the match of hashes, the normalized Hamming distance is used which is defined as:

$$d(h1, h2) = \frac{1}{L} \sum_{t=1}^L |h1(t) - h2(t)| \quad (8)$$

where L is the length of the hash string. For each sub-image, we compute the normalized Hamming distance between the computed and extracted hashes. The status of the sub-image $ST_{i,j}$ (altered sub-image or not) is evaluated as follows:

$$ST_{i,j} = \begin{cases} 0, & d(h'_{i,j}, he^1_{i,j}) > T \text{ and } d(h'_{i,j}, he^2_{i,j}) > T \\ 1, & \text{else} \end{cases} \quad (9)$$

where T is a threshold, $ST_{i,j}=0$ if the sub-image $SI'_{i,j}$ is considered as an altered sub-image, otherwise $ST_{i,j}=1$. For each altered sub-image, the approximated original sub-image can be recovered if the two distance sub-images of the concerned sub-image are not altered. Therefore, the reconstructed sub-image $SI^c_{i,j}$ is rebuilt as follows:

$$SI^c_{i,j} = \begin{cases} dec(C^e_{i,j}), & ST_{i,j} = 0 \text{ and } ST_{i1,j1} = ST_{i2,j2} = 1 \\ LOST, & ST_{i,j} = 0 \text{ and } (ST_{i1,j1} = 0 \text{ or } ST_{i2,j2} = 0) \\ SI'_{i,j}, & ST_{i,j} = 1 \end{cases} \quad (10)$$

where $LOST$ is a sub-image that is marked as a lost sub-image, dec is a sub-image decoding method.

5. EXPERIMENTAL RESULTS

To examine the robustness of the proposed technique, we consider the performance of it to JPEG compression and additive noise. The proposed system has been tested using 50 512×512 images. We firstly study the effects of JPEG compression with a range of quality factors. Then, the additive Gaussian noise effects are addressed. The size of the used sub-image is 32×32. To calculate the sub-image hash, the parameters are $r=8$, $t=2$ and $k=16$. Thus, the hash length is 32 bits. In the robustness tests of the proposed technique, the quantization table of 50% quality JPEG compression is used as a predefined quantization table Q_m . The chosen coefficients' indices of the first distant sub-image blocks are $\{(1,4),(4,1)\}$ for embedding the hash copy and $\{(2,3),(3,2)\}$ for embedding the first group of the code. For the second distant sub-image blocks, the chosen coefficients' indices are $\{(2,4),(4,2)\}$ for the second hash copy and $\{(1,3),(3,1)\}$ for the second group of the code.

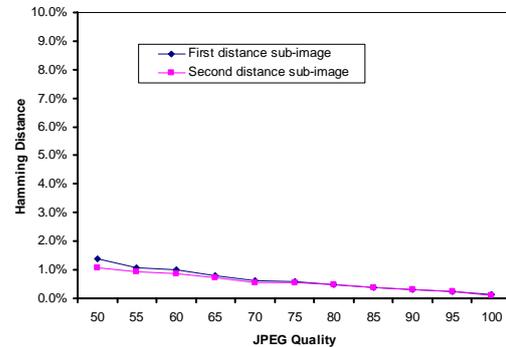


Figure 2. Average Hamming distance between the computed and extracted hashes for various JPEG compression quality factors.

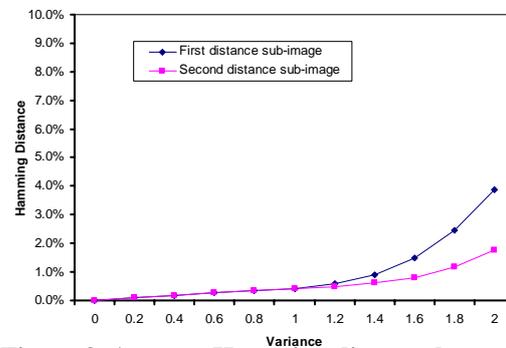


Figure 3. Average Hamming distance between the computed and extracted hashes for various Gaussian noise variances.

Fig. 2 illustrates that the average Hamming distance between the computed sub-image hash and extracted hashes recovered from the first distant sub-image and the second distant sub-image, respectively for various JPEG compression quality factors.

The effects of the additive zero-mean Gaussian noise have also been tested. Fig. 3 shows the average Hamming distance between the computed sub-image hash and extracted hashes which are extracted from the first distant sub-image and the second distant sub-image, respectively for various noise variances. From these figures, we observe that the normalized Hamming distance values are less than 9%. Thus, we can use this value as a threshold T .

To validate the proposed technique, we test it to check its capability of detecting local malicious manipulations mixed with JPEG compression. Fig. 4 is the original image and the approximated image, which represents the codes of all sub-images is shown in Fig. 5. The correlation coefficient between the original (grayscale version) and approximated images is 0.9198 and the peak signal to noise ratio PSNR of the approximated image relative to original image is 26.07 dB. The original image is authenticated using the proposed technique with the used quantization table Q_m of 70% quality JPEG compression to yield the image of Fig. 6. The correlation coefficient between the original and authenticated images is 0.9982 and the PSNR of the authenticated image relative to the original image is 42.47 dB. The authenticated image is altered by a local malicious attack. Then, it is undergone 80% quality JPEG compression to yield the image of Fig. 7. In Fig. 8, the proposed technique efficiently detects and localizes the content alterations. The proposed technique can not only localize the alteration detection but also successfully recover the missing contents as shown in Fig. 9.

6. CONCLUSION

A self-recovery semi-fragile image authentication technique is proposed which uses secure image

hashing with improved localization. Using image hashing in the proposed technique to generate the signatures gives the proposed technique the capability to be robust against the normal operations such as JPEG compression and additive noise. To thwart the vector quantization attack, two sub-image hash copies and the sub-image code are securely embedded into two relatively distant sub-images. The experiment results explain



Figure 5. Approximated image, correlation coefficient=0.9198, PSNR=26.07dB.



Figure 6. Authenticated image, correlation coefficient=0.9982, PSNR=42.47dB.



Figure 4. Original image.



Figure 7. Altered version of the authenticated image.

that the proposed technique successfully distinguishes the normal manipulations such as JPEG compression from malicious operations and precisely localizes the alteration detections. Moreover, the proposed technique can successfully compensate the missing contents.



Figure 8. Verification result marking the altered regions.



Figure 9. Reconstructed image.

7. ACKNOWLEDGMENTS

This work is supported by Forschungspraemie (BMBF-Förderung, FKZ: 03FPB00213) and Transregional Collaborative Research Centre SFB/TRR 62 "Companion-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

8. APPENDIX

Non-negative matrix factorization NMF

In NMF, a non-negative matrix V is factorized into two matrices, B and C :

$$V \approx B C \quad (11)$$

where B and C are called the base matrix and the coefficient matrix respectively. The factors C and B must be non-negative.

If the size of V is $m \times n$, the sizes of B and C are $m \times r$ and $r \times n$, respectively. If r is chosen as less than m and n , NMF may be used for dimensionality reduction.

To compute B and C , the following updating rules are applied [Tan08]:

$$B_{i,l} \leftarrow B_{i,l} \frac{\sum_{j=1}^n C_{l,j} V_{i,j} / (BC)_{i,j}}{\sum_{j=1}^n C_{l,j}} \quad (12)$$

$$C_{l,j} \leftarrow C_{l,j} \frac{\sum_{i=1}^m B_{i,l} V_{i,j} / (BC)_{i,j}}{\sum_{i=1}^m B_{i,l}} \quad (13)$$

where $i=1,2,\dots,m$; $j=1,2,\dots,n$; $l=1,2,\dots,r$.

9. REFERENCES

- [Bar02] Barreto, P. S. L. Kim, M. H. Y. and Rijmen, V. : Toward a secure public-key blockwise fragile authentication watermarking, IEE Proc. Vision, Image & signal Proc., vol.149, no.2, pp.57-62, 2002.
- [Cel02] Celik, M. U. Sharma, G. Saber, E. and Tekalp, A. M. : Hierarchical watermarking for secure image authentication with localization, IEEE Trans. on Image Proc., vol.11, no.6, June 2002.
- [Eki04] Ekici, O. Sankur, B. Coskun, B. Naci, U. and Akcay, M. : Comparative evaluation of semi-fragile watermarking algorithms, Journal of Electronic Imaging, vol.13, no.1, pp.209-216, Jan. 2004.
- [Fri99a] Fridrich, J. : Methods for tamper detection in digital images, in Proc. ACM Workshop on Multimedia and Security, Orlando, 1999, pp.19-23.
- [Fri99b] Fridrich, J. and Goljan, M. : Images with self-correction capabilities, in Proc. ICIP'99, Kobe, Japan, 1999.
- [Fri99c] Fridrich, J. and Goljan, M. : Protection of digital images using self embedding, Symp. Content Security and Data Hiding in Digital Media, New Jersey Institute of Technology, May 14, 1999.
- [Fri00] Fridrich, J. and Goljan M. : Robust hash functions for digital watermarking, in Proc. IEEE Int. Conf. Information Technology: Coding Computing, Mar. 2000, pp. 178–183.
- [Has04] Hasan, Y. M. Y. and Hassan, A. M. : Fragile blockwise image authentication thwarting

- vector quantization attack, in Proc. IEEE ISSPIT'04, Rome, Italy, 2004.
- [Has07] Hasan, Y. M. and Hassan, A. M. : Tamper detection with self-correction hybrid spatial-dct domains image authentication technique, in Proc. IEEE ISSPIT'07, Cairo, Egypt, 2007.
- [Hol00] Holliman, M. and Memon, N. : Counterfeiting attacks on oblivious block-wise independent invisible watermarking schemes, IEEE Trans. on Image Processing, vol. 9, no. 3, pp.432-441, March 2000.
- [Kir02] Kirovski, D. and Petitcolas, F. A. P. : Blind pattern matching attack on watermarking systems, IEEE Trans. on Signal Processing, pp.1-9, 2002.
- [Kut00] Kutter, M. Voloshynovskiy, S. and Herrigl, A. : The watermark copy attack, in Proc. SPIE Elect. Imaging, San Jose, USA, Jan. 23-28, 2000.
- [Lan99] Lan, T. and Tewfik, A. H. : Fraud detection and self embedding, in Proc. ACM Multimedia'99, Orlando, FA, 1999.
- [Lin99] Lin, E. and Delp, E. : A review of fragile image watermarks, in Proc. Of the ACM Multimedia and Security Workshop, 1999, pp. 25-29.
- [lin00] Lin, C. and Chang, S. : Semi-fragile watermarking for authenticating JPEG visual content, SPIE Security and Watermarking of Multimedia Contents II EI '00, SanJose, CA, Jan. 2000.
- [Lin01a] Lin, C. and Chang, S. : A robust image authentication method distinguishing JPEG compression from malicious manipulation, IEEE Trans. On Circuits and Systems of Video Technology, vol. 11, no. 2, Feb. 2001.
- [Lin01b] Lin, C. and Chang, S. F. : SARI: Self-authentication and recovery image watermarking system, Proceedings of the ninth ACM Conference on Multimedia, Ottawa, Canada ,2001.
- [Lin07] Lin, C. Su, T. and Hsieh, W. : Semi-fragile watermarking scheme for authentication of JPEG images, Tamkang Journal of Science and Engineering, vol. 10, no. 1, pp. 57-66, 2007.
- [Lue08] Lue, H. Lu, Z. Chu, S. and Pan, J. : Self embedding watermarking scheme using halftone image, IEICE Trans. Inf.&Syst., vol.E91-D, no.1, Jan.2008.
- [Mae06] Maeno, K. Sun, Q. Chang, S. and Suto, M. : New semi-fragile authentication watermarking techniques using random bias and nonuniform quantization, IEEE Trans. On Multimedia, vol. 8, no. 1, Feb. 2006.
- [Men01] Menezes, A. J. Orschot, P. C. and Vanstone, S. A. : Handbook of Applied Cryptography, CRC Press, 2001.
- [Mih01] Mihçak, M. K. and Venkatesan, R. : New iterative geometric methods for robust perceptual image hashing, in Proc. ACM Workshop Security and Privacy in Digital Rights Management, Philadelphia, PA, Nov. 2001.
- [Sch96] Schneier, B. : Applied Cryptography: Protocols, Algorithms, and Source Code in C, John Wiley & Sons, USA, 1996.
- [Swa06] Swaminathan, A. Mao, Y. and Wu, M. : Robust and secure image hashing, IEEE Trans. On Information Forensics and Security, vol. 1, no. 2, June 2006.
- [Tan08] Tang, Z. Wang, S. Zhang, X. Wei, W. and Su, S. : Robust image hashing for tamper detection using non-negative matrix factorization, Journal of Ubiquitous Convergence and Technology, vol. 2, no. 1, may 2008.
- [Ven00] Venkatesan, R. Koon, S. M. Jakubowski, M. H. and Moulin, P. : Robust image hashing, in Proc. IEEE Int. Conf. Image Processing, Vancouver, BC, Canada, Sep. 2000, vol. 3, pp. 664-666.
- [Wan08] Wang, S. and Tsai, S. : Automatic image authentication and recovery using fractal code embedding and image inpainting, Journal of the Pattern Recognition Society, vol. 41, pp. 701 - 712, 2008.
- [Won97] Wong, P. W. : A watermark for image integrity and ownership verification, in Proc. IS & TPIC, Portland, OR, USA, May 1997.
- [Won98] Wong, P. W. : A public key watermark for image verification and authentication, in Proc. ICIP, NY, USA, Oct.4-7, 1998, pp.425-429.
- [Won01] Wong, P.W. and Memon, N. : Secret and public key image watermarking schemes for image authentication and ownership verification, IEEE Trans. on Image Processing, vol. 10, pp. 1593-1601. Oct. 2001.
- [Yeu97] Yeung, M. and Mintzer, F. : An invisible watermarking technique for image verification, in Proc. ICIP'97, Santa Barbara, CA, USA, 1997.

Multi-Threaded Real-Time Video Grabber

Zdeněk Trávníček

DCGI, FEE

Czech Technical University in Prague

Czech Republic

zdenek.travnicek@fel.cvut.cz

Roman Berka

Institute of Intermedia, FEE

Czech Technical University in Prague

Czech Republic

berka@iim.cz

ABSTRACT

Communication in general incorporates technologies with increasing number of communication modes. Special applications are developed in the area of virtual reality, multimedia communications and others where combinations of audio, video, 3D data are sent between two (or more) distant users which can commonly interact with these data. A form of so exchanged information usually requires, among others, special forms of presentation. Thus stereoscopic and virtual reality visualization devices are used to present intricately structured information in multi-modal form.

There are situations where the presented information is to be rendered in real-time and transmitted to the remote user in form of a video-stream. In this case, the content is presented on a local visualization device (e.g. CAVE) being simultaneously sent to a remote device. Thus a method how to obtain rendered data from graphics hardware in real-time is necessary.

The problem is, how to obtain the rendered data for transmission with minimal impact on the rendering and visualization process. In this paper, we present a method how to retrieve video stream from an arbitrary running OpenGL application, capturing every frame with minimal impact on performance.

Keywords: OpenGL, real-time video grabber, streaming video, streamcast

1 INTRODUCTION

With the rise of 3D digital media, stereoscopic movies and upcoming 3D television, the need for a new sources of stereoscopic signal emerges. The usual sources of such a signal are cameras in stereoscopic setups or pre-rendered video sequences. There are many applications rendering 3D images, some of them even stereoscopic ones. Those could be great source for such a stream, but they usually does not support producing an video that could be directly used as a source of video signal for stream nor support saving video to a file.

In order to use such an application we need to be able to retrieve output of the running application in real-time (see fig. 1). From other point of view, we may simply want to record output of running application and store it locally for later, offline use. In order to get those, we could alter the application itself to produce such a video stream or file. We can also use some screen grabbing application (streamcast) or have a hardware solution.

As the graphics hardware and software technologies changes over the time, the problem is still actual and new approaches appear. The main problem is related to the cost of the grabbing process because the data source (typically a graphical subsystem) produces content in

real time. Thus the grabber should obtain pictures with minimal impact on the rendering process.

We first describe known methods of the video grabbing which appeared during a period of the last decade. These methods are evaluated according to our criteria based on modification that needs to be done to the application itself, impact on performance of the application and possibility of grabbing stereoscopic images from quad buffer. We evaluate the performance loss for multi-core/multi-CPU systems. Next, our own asynchronous wrapper is described and compared with the already implemented solutions. Finally, some applications of the described wrapper are presented.

2 STATE OF THE ART

There exist several approaches to the solution of how to acquire a stream of graphical data from an application running on the system. These approaches are then often implemented for various purposes. We can classify them into four basic groups:

- alteration of the application which is the source of the data
- screen grabbing
- combination of previous two methods
- capturing output of the graphics hardware

These methods are know explained and compared in the next paragraphs.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

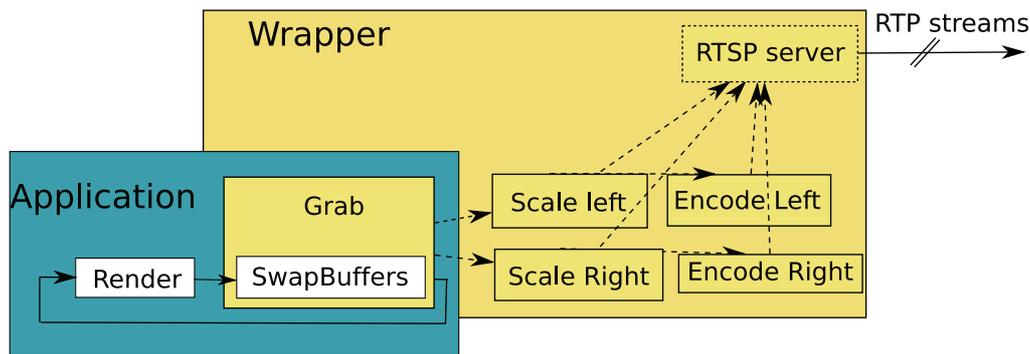


Figure 1: A general scheme of grabber.

2.1 Altering an Application

The method of altering an existing application has an obvious drawback in a need to have source codes for the application and also alteration of every application we use. This basically limits the usability of it to applications where we have source code (typically open-source). The solution is also complicated when we use many different applications.

Aside from that, this method has an advantage in knowing everything about the application to have full control over the grabbing process. Thus it can grab the images synchronously with the rendering speed. Also, for an application rendering stereoscopic images into quad-buffer, this method can grab images for both eyes. The implementation is specific to every application as well as the performance loss. This solution can be therefore seen in special applications (e.g., applications working as real-time video content generators for network projects or art performances).

2.2 Screen Grabber

The screen grabbing represent a next approach where the graphical information is obtained independently on the application code. Using a standalone screen grabber does not require any alteration of the application, but on many systems it has problems on accelerated windows. It won't be synchronized with the speed of an application as it does not have any information about architecture of the application. Asynchronous grabbing can introduce image distortions when the frame buffer is changed during read, it can miss frames when the application renders faster than the grabber grabs and can unnecessarily grab the same image multiple times, when the application stalls or is just slower than the grabber. Furthermore, this method would fail for quad-buffer stereo.

As an example of such an approach, there are applications like *scrot* and *xsnap* realized in GNU/Linux environment. The code of the grabber runs outside the

context of the application, so the impact on the rendering speed should be quite small.

2.3 Combined Solution

Another solution would be combination of above mentioned two methods. Here, a separate grabber without modifying the application is used. This can be done using an wrapper to rendering library, i.e. OpenGL, which would inject some code to proper place of the rendering process and execute it there. Provided our code could get enough information about rendering window, we can grab the exact window, adjust the area being grabbed when the application window changes and we can start the grabbing exactly once per frame.

There is an opensource project *captury* using this solution. In this project, the code is executed in context of rendering thread of the application, effectively slowing down the rendering of every frame by grabbing, compressing and saving every frame, before it the buffers gets swapped.

2.4 Hardware Solution

A hardware solution means plugging some device into output of graphics card and process it on other computer or in the device itself. This solution needs separate hardware, it is quite expensive, and is not synchronized with the application's speed. The output signal needs to be cropped when rendering only into a window. In addition, the captured signal has given parameters, like resolution, which are not easily controllable during the grabbing process. On the other hand, it has absolutely no impact on the application itself, as there's no processing on the rendering machine.

As the acquisition of the video from graphics hardware in real-time is an interesting problem new solutions implemented directly in the graphics boards rises. In August 2009, nVIDIA released solution to record/output SDI uncompressed video directly to/from Quadro GPU's memory. As this information is too

much new, we had no chance to test it before submission of this paper.

3 MULTI-THREADED REAL-TIME VIDEO GRABBER

The solution we propose is a modified approach to wrapping rendering library's calls and injecting our code there.

The key is in using a wrapper, that "hooks" onto few library calls in order to retrieve information about application's window and to grab the window in a right time.

The grabbing itself is done in the context of the rendering thread using standard methods to retrieve the content of framebuffer. This directly implies that, when rendering in quad-buffer mode for active stereoscopy, we can easily get both images as we can control the flow of the code. After getting the frame we send it to an other thread to next process. This ensures that the impact will be as small as possible, provided the machine has multi-core CPU or multiple CPUs. The processing itself can be done in multiple threads also, to use more available cores more effectively. In the processing threads, we can save the video to the local storage or stream it over network and optionally compress it.

The implementation we present was done under GNU/Linux environment, using an OpenGL applications and nVIDIA QUADRO FX cards to render active stereoscopic images in quad-buffered mode.

3.1 Wrapping

The wrapping is done by utilizing linux dynamic loader, which takes care of loading libraries and resolving symbols. Using `LD_PRELOAD` environmental variable recognized by the loader, we tell it to preload a shared object before an application and use it for symbol resolving with higher priority. In the shared object we provide hooks on few function that inject our code before the real call to the library function.

Namely we "hook" onto `glViewport` in order to get information about the window size and it's changes. We also use this as a point to initialize the processing threads. We also hook onto framework specific functions in order to swap buffers (`glXSwapBuffers`, `SDL_GL_SwapBuffers`). When the application calls swap buffers, it signalizes it has finished rendering the frame, so it's the right place for us to grab it and send it to the next process. It is also the place where we can drop frames if the application is rendering too fast. Our implementation also wraps `dlsym` call to catch symbol resolving done in real-time and not by dynamic loader.

3.2 Grabbing

During a rendering process the rendered images are stored in two (or four in case of stereoscopic output) frame buffers which are periodically swapped. On principle, there are two types of frame buffer reading:

- asynchronous – based on so called *Pixel Buffer Objects* [Biermann et al., 2004]
- synchronous – direct buffer reading

First, retrieving the image is done by calling `glReadPixel` with correctly set read buffer in OpenGL context, optionally on initialized Pixel Buffer Object (PBO). PBO approach moves the reading into background so it does not block the rendering thread. But it introduces a delay of 1 frame, because we get the data on the next buffer swap.

The direct approach introduces delay into the rendering thread, which means a slowdown of the application, but we get the data sooner. We support both methods. By changing actual buffer and repeating the read, we can retrieve data for the other eye, if we have quad-buffer stereo.

3.3 Processing

The processing threads are doing color space conversions and re-sampling. Other threads can take care of possible video compression and others can stream it or save it locally. Processing of stereoscopic signals is done by pairs of threads to improve multi-threaded performance.

3.4 Summary

A scheme of the process is shown on figure 2. Original application is wrapped in it's call to `Swap Buffers` (usually `glXSwapBuffers`) is intercepted and instead of it, our code is executed. Content of the framebuffer is then grabbed as described in 3.2 and sent for processing to other threads. Then original `SwapBuffers` method is called and control is returned to the application. Meanwhile the data from framebuffer are being processed in other threads and eventually streamed out (or recorded).

The whole grabbing process is done in the context of the rendering thread, but the rest of the processing is done in other threads, not directly affecting the application's performance. So the impact to application is mostly defined by the slowdown that takes place in the grabbing functions. Of course, in case the application would do some CPU intensive operation the video (i.e. compression), it may place load to the CPU and indirectly slowing down the application.

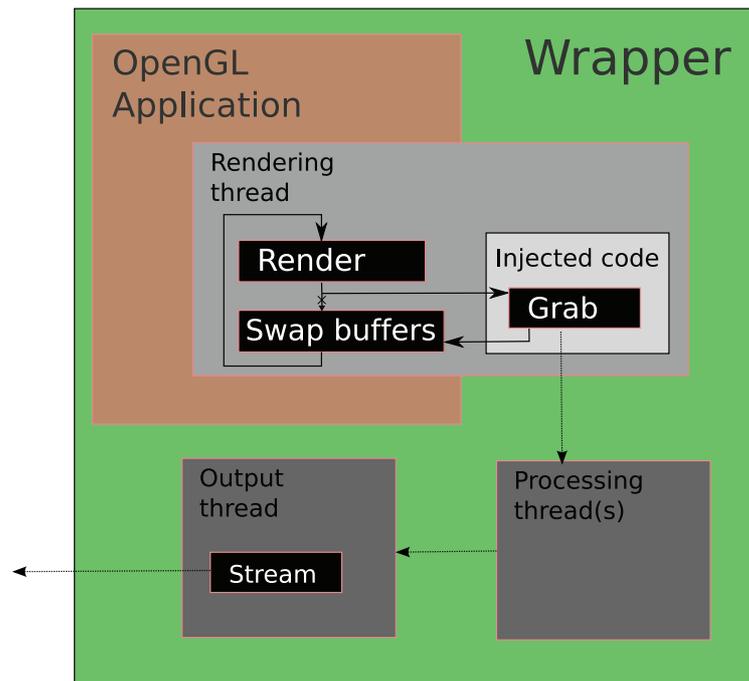


Figure 2: Scheme of the wrapped grabber

4 APPLICATIONS

The possibility to capture rendered video in real-time has lot of applications in wide area. As the problem described in this paper is part of another project, we can mention some applications which already use our grabber.

4.1 Project C2C

Described method is successfully used in project Cave2Cave (C2C) [Berka et al., 2009] to stream a stereoscopic video signal from applications running in CAVE-like system [Cruz-Neira et al., 1992] and to present it on remote site (see fig 3).

We use the the multi-threaded grabber to get video of the application, scale it, optionally compress it and stream it using standard protocol RTP [Schulzrinne et al., 1996]. The grabber also creates RTSP [Schulzrinne et al., 1998] server to provide SDP descriptions [Arkko et al., 2006] of the streams. This way we can (and we do) present applications from our CAVE system to distant viewers. The use of standard streaming protocols allows us to partially preserve possibility of receiving data by standard players used by remote user.

4.2 Prerendering

Another use of the method is to allow prerendering with applications that does not support it natively. For example, application rendering complex model which can not be rendered in real-time could be used to render it as fast as it could while having it's whole run recorded. Then we simply playback the recorded video at the

requested speed. This allows us to present output of any application even in cases, when the application itself can not do it in real-time. We successfully used this method for presenting walks through very complex VRML models to public.

4.3 Industrial Applications

As the grabber can wrap theoretically any OpenGL application (it depend on correctness of application implementation in relation to OpenGL library), it offers itself in such situations where some industrial product (like an architectural model or model of a car) is to be, probably interactively, presented to a remote user without necessity to send these data to his/her computer. It is important when there is not possible to move real data or software, e.g. due to license limitations. Using systems like CAVE, running our grabber on each wall, as a source of content, an application then allows to mediate immersive environment remotely using standards described in already referenced RFC documents.

5 CONCLUSION

The proposed method allows real-time retrieval of rendered stereoscopic images from arbitrary OpenGL application without a need to modify the application itself. It can be used as base for a system to record an output of an application to local storage for offline use or to stream the content over network in real-time.

The solution has potentially lot of applications in wide area of remote visualizations also on immersive devices or in the area of collaborative environments. As it has been already mentioned above the problem

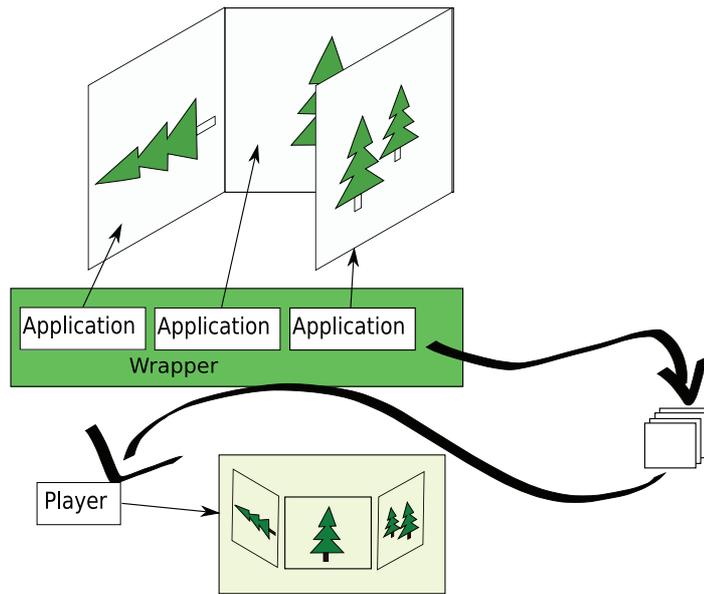


Figure 3: Scheme of the multi-projection screen based configuration. A scene rendered in the resource device with 3 projection walls is grabbed and the resulting video is transmitted to the remote device where it is presented on remote projection wall.

with grabbing methods is in continuous development and follows possibilities of contemporary technologies. For now, we can expect that the support of hardware solutions will be probably accessible for wider area of applications.

6 ACKNOWLEDGMENTS

This work has been partially supported by:

CESNET, association of legal entities,
Prague, Czech Republic
under the research program MSM 6383917201

Czech Technical University in Prague
Institute of Intermedia

Center for Computer Graphics
under the research program LC-06008

REFERENCES

- [Arkko et al., 2006] Arkko, J., Lindholm, F., Naslund, M., Norrman, K., and Carrara, E. (2006). Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP). RFC 4567 (Proposed Standard).
- [Berka et al., 2009] Berka, R., Trávníček, Z., Havran, V., Bittner, J., Žára, J., Slavík, P., and Navrátil, J. (2009). *Networking studies III, Selected Technical Reports*, chapter CAVE to CAVE: Communication in a Distributed Virtual Environment, pages 161–174. CESNET, 1st edition. ISBN: 978-80-904173-4-2.
- [Biermann et al., 2004] Biermann, R., Carter, N., Cornish, D., Craighead, M., Kilgard, M., Kirkland, D., Leech, J., Paul, B., Roell, T., Romanick, I., and Sandmel, J. (2004). ARB_pixel_buffer_object specification http://www.opengl.org/registry/specs/arb/pixel_buffer_object.txt. Web page. Downloaded in October 2009.
- [Cruz-Neira et al., 1992] Cruz-Neira, C., Sandin, D., Defanti, T., Kenyon, R., and Hart, J. (1992). The cave: Audio Visual Experience Automatic Virtual Environment. *Communications of the ACM*, 35(6):65–72.
- [Schulzrinne et al., 1996] Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V. (1996). RTP: A Transport Protocol for Real-Time Applications. RFC 1889 (Proposed Standard). Obsoleted by RFC 3550.
- [Schulzrinne et al., 1998] Schulzrinne, H., Rao, A., and Lanphier, R. (1998). Real Time Streaming Protocol (RTSP). RFC 2326 (Proposed Standard).

Markov Random Fields on Triangle Meshes

Vedrana Andersen
DTU Informatics
R. Petersens Plads
2800 Kgs. Lyngby
Denmark
va@imm.dtu.dk

Henrik Aanæs
DTU Informatics
R. Petersens Plads
2800 Kgs. Lyngby
Denmark
haa@imm.dtu.dk

Andreas Bærentzen
DTU Informatics
R. Petersens Plads
2800 Kgs. Lyngby
Denmark
jab@imm.dtu.dk

Mads Nielsen
DIKU
Universitetsparken 1
2100 Copenhagen
Denmark
madsn@diku.dk

ABSTRACT

In this paper we propose a novel anisotropic smoothing scheme based on Markov Random Fields (MRF). Our scheme is formulated as two coupled processes. A vertex process is used to smooth the mesh by displacing the vertices according to a MRF smoothness prior, while an independent edge process labels mesh edges according to a feature detecting prior. Since we should not smooth across a sharp feature, we use edge labels to control the vertex process. In a Bayesian framework, MRF priors are combined with the likelihood function related to the mesh formation method. The output of our algorithm is a piecewise smooth mesh with explicit labelling of edges belonging to the sharp features.

Keywords: Mesh, smoothing, Markov Random Fields.

1 INTRODUCTION

Markov Random Fields (MRF) have been used extensively for solving Image Analysis problems at all levels. The local property of MRF makes them very convenient for modeling dependencies of image pixels, and the MRF-Gibbs equivalence theorem provides a joint probability in a simple form, making MRF theory useful for statistical Image Analysis. While some examples are mentioned below, MRF have rarely been used for mesh processing. One reason could be that MRF are usually defined on regular grids, but this is by no means required.

In this paper we demonstrate that feature preserving mesh smoothing may conveniently be cast in terms of MRF theory. Using this theory we can explicitly model our knowledge of properties of the surface (*prior knowledge*, e.g. how smooth the surface should be, which sharp features should it contain) and our knowledge of the noise (*likelihood*, e.g. how far do we believe the measured position of a vertex is likely to be from the true position). The central element of the MRF formulation is that we use Bayes rule to express the probability of any mesh configuration by defining

its of prior and likelihood independently. This division of responsibilities often turns out to be a benefit.

For instance, a big advantage of the MRF formulation is that we can use the likelihood to keep the mesh fairly close to the input, avoiding the shrinkage associated with many other schemes. Unlike [Hildebrandt and Polthier, 2007] we do not obtain a hard constraint, but meshes far from the input can be made arbitrarily unlikely by choosing an appropriate likelihood function.

We investigate the use of MRF for formulating priors on 3D surfaces in a number of different ways. The smoothness prior encodes the belief that a smooth surface (according to some fairness criterion) is more probable than a noisy surface. In particular, we show how we can use one MRF to perform explicit labelling of edges according to how sharp they are, and another MRF to find optimal vertex positions according to the smoothness prior. Using our edge labelling from the first MRF to control the vertex smoothing, we are able to recapture very subtle sharp features on the noisy mesh.

2 RELATED WORK

Mesh-smoothing algorithms have a long history in the field of geometry processing since the early work of [Taubin, 1995], which demonstrated the connection between various explicit linear methods using the so called umbrella operator and low pass filtering. In [Desbrun *et al.*, 1999] a discrete Laplace Beltrami operator was introduced and the connection between smoothing and mean curvature flow was explained. Both techniques are efficient, but fail to distinguish

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

between the noise and the features of the underlying object.

To address this problem, anisotropic diffusion [Desbrun *et al.*, 2000] and diffusion smoothing of the normal field [Tasdizen *et al.*, 2002] were proposed. The results are impressive, but the computation complexity puts a limit on the size of the model. More efficient methods were also developed, such as non-iterative feature-preserving smoothing [Jones *et al.*, 2003] based on robust statistics, and an adaptation of bilateral filtering to surface meshes [Fleishman *et al.*, 2003].

Another feature preserving smoothing method, fuzzy vector median smoothing [Shen and Barner, 2004], is a two-step smoothing procedure. In the first step face normals are smoothed using a robust method which employs distance to median normal as smoothing weight. In the next step vertex positions are updated accordingly. More recently, in [Diebel *et al.*, 2006] a Bayesian approach was proposed. This method uses a smoothness prior and the conjugate gradient method for optimization. It is feature-preserving, but without an explicit feature detection scheme. Similar to [Diebel *et al.*, 2006], we use a Bayesian approach, but unlike that method we obtain feature preservation by explicitly detecting the set of chosen features. Our method is also more flexible, allowing us to use a variety of priors and likelihood potentials.

The method for recovering feature edges proposed in [Attene *et al.*, 2005] is based on the dual process of sharpening and straightening feature edges. Vertex-based feature detection using an extension of the fundamental quadric is utilized in a smoothing method described by [Jiao and Alexander, 2005].

Comprehensive study on the use of MRF theory for solving Image Analysis problems can be found in books [Li, 2001; Winkler, 2003]. MRF theory is convenient for addressing the problem of piecewise smooth structures. In [Geman and Geman, 1984] a foundation for the use of MRF in Image Analysis problems is presented in an algorithm for restoration of piecewise smooth images, where gray-level process and line processes are used. Another application of MRF for problems involving reconstruction of piecewise smooth structures is [Diebel and Thrun, 2005], where high-resolution range-sensing images are reconstructed using weights obtained from a regular image.

There are some previous examples of using MRF theory to 3D meshes, but the applications are somewhat different. In [Willis *et al.*, 2004] MRF are used in the context of surface sculpting with the deformation of the surface controlled by MRF potentials mod-

elling elasticity and plasticity. MRF was also used for mesh analysis and segmentation in [Lavoué and Wolf, 2008].

Our work investigates the possibility of formulating surface priors in terms of MRF, and using those priors for reconstructing the surface from the noisy data. Unlike most other mesh smoothing algorithms, our approach does not only preserve sharp ridge features, but also explicitly detects the ridges.

The method described here is not automatic and requires an estimation of a considerable set of parameters. However, this allows a great control over the performance of the priors.

3 MESH SMOOTHING USING MRF

Markov Random Fields is a powerful framework for expressing statistical models originating in computational physics, and it has proven highly successful in Image Analysis [Li, 2001; Winkler, 2003]. A MRF is, essentially, a set of sites with associated labels and edges connecting every site to its neighbors. The labels are the values which we wish to assign (e.g. pixel color, vertex position or edge label), and it is a central idea in MRF theory that the label at a given site must only depend on the labels of its neighbors. This framework lends itself well to mesh based surfaces, where the neighborhood of a vertex can be naturally defined via its connecting edges.

Apart from a well developed mathematical framework one of the main advantages of MRF is that its Markovianity (local property) makes it quite clear what the objective function is and what a MRF based algorithm aims at achieving. Exponential distributions are often used, and the joint probability distribution function of given configuration f (e.g. combined vertex location) is given by

$$P(f) \propto e^{-\sum U(f)} ,$$

where the $U(f)$ can be seen as energy terms or potentials defined on neighborhoods. In order to find the most likely configuration f , we need to obtain

$$\min_f \sum U(f) . \quad (1)$$

In our proposed framework, we wish to smooth a given mesh. Some of the $U(f)$ in (1) are thus data (likelihood) terms penalizing the displacement of the vertices in the smoothed mesh relative to the original mesh. Other terms would be prior terms which express how likely a surface is *a priori*, i.e. without making reference to how far removed it is from the data.

3.1 Likelihood

We want the output of the smoothing to relate to the input mesh, which has an underlying true surface cor-

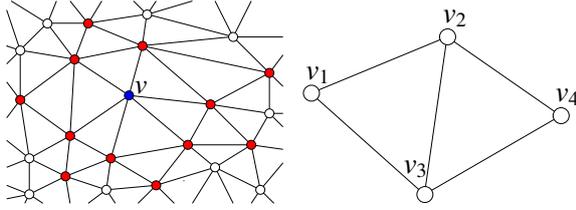


Figure 1: Left: A neighborhood structure for the smoothness prior. The neighbors of the vertex v are marked red. When we move vertex v , we only need to look at its neighboring vertices to calculate the change in the joint smoothness potential. **Right:** A collection of 4 vertices, expressing two adjacent faces.

rupted by the noise of the data-acquisition device. Assuming isotropic and Gaussian measurement noise we choose quadratic function for the likelihood energy

$$U_L(v) = \alpha \|\mathbf{v}^0 - \mathbf{v}\|^2$$

where \mathbf{v}^0 and \mathbf{v} denote the initial and the current position of the vertex v . The constant α is used as the weight determining how much faith one has in the data.

There is always a possibility of plugging in a different likelihood function in our model, e.g. a volume preserving likelihood function or likelihood utilizing some specific knowledge about data acquisition process.

3.2 Smoothing Potential

Alongside the data term we also have some a priori terms expressing our assumptions about how a smoothed mesh should look. Firstly, we have a smoothing potential, which is basically a penalty function, ρ , based on the difference between the normals of adjacent faces, see Figure 1

$$U_s(v_1, v_2, v_3, v_4) = \rho(\mathbf{n}_{123} - \mathbf{n}_{243}) , \quad (2)$$

where \mathbf{n}_{123} and \mathbf{n}_{243} are the normals of the two adjacent faces. The suitable MRF neighborhood for above formulation is defined as follows: two different vertices are neighbors if they belong to the adjacent faces. In this smoothing scheme the label of each mesh vertex is its spatial position, which is adjusted to minimize the chosen energy function.

The choice of the smoothness potential can greatly influence the feature preserving property of the smoothing. On the one side, there is a over-smoothing quadratic potential developed by [Szeliski and Tonnesen, 1992]

$$\rho(\mathbf{x}) = \|\mathbf{x}\|^2 ,$$

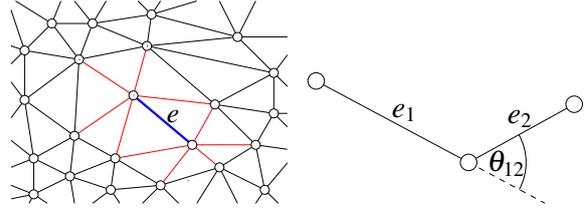


Figure 2: Left: A neighborhood structure for the edge support prior. The neighbors of the edge e are marked red. The neighboring edges support each other if they lie along the same line. **Right:** A pair of edges. The support for the edges e_1 and e_2 depends on the size of the angle θ_{12} .

on the other side, there is a feature preserving square root potential developed by [Diebel *et al.*, 2006]

$$\rho(\mathbf{x}) = \|\mathbf{x}\| .$$

In our case, feature preservation will be handled by the explicit edge labelling, which allows us to use the aggressive quadratic potential for smooth regions, without thinking about its feature preservation properties.

3.3 Edge Labelling

In many mesh smoothing tasks the presence of clear ridge features in the result is part of our a priori expectation. This is included in our MRF model where we, as an integral part of the smoothing process, label mesh edges as being ridge edges or not. Edge label ε is a number from the interval $[0, 1]$ which indicates how probable it is that the given edge is a part of a sharp ridge feature. Those labels will later be used to introduce discontinuities in the smoothing process.

Edge labelling is in itself based on a MRF model consisting of two terms, edge sharpness term U_{E1} , and the neighborhood support term U_{E2} .

The larger the dihedral angle ϕ_e , of a mesh edge is, the more probable it is that the edge lies along the surface ridge. The first term is thus given by

$$U_{E1}(e) = (\phi_0 - \phi_e)\varepsilon , \quad (3)$$

where ϕ_0 is a ridge sharpness threshold, and ε is the label assigned to the edge e .

The second term of the edge labelling is the neighborhood support, i.e. the presence of other ridge edges along the same ridge line. We assign a support energy to all pairs of edges, see Figure 2. A measure of parallelism between the edges is used in the formulation of the support potential

$$U_{E2}(e_1, e_2) = -\cos(\theta_{12})\varepsilon_1\varepsilon_2 , \quad (4)$$

where θ_{12} is the angle between the edges e_1 and e_2 , and ε_1 and ε_2 are the labels assigned to e_1 and e_2 . Fea-

ture edges lying on a straight line will have a maximum support, the orthogonal edges do not support each other, and feature edges meeting at a sharp angle are discouraged.

There are additional constrains one can use to define ridge edges, like e.g. dihedral angle changing slowly along the ridge line, or the expectation that the ridge edge itself is smooth.

3.4 The Coupled Model

The smoothing potential and the edge labelling are coupled in a feature preserving scheme, which smoothes the mesh, but not over the edges labelled as sharp. This is obtained by using edge labels as weights for the smoothing potential, which is now, for the setting as in Figure 1

$$U_s(v_1, v_2, v_3, v_4) = (1 - \epsilon_{23})\rho(\mathbf{n}_{123} - \mathbf{n}_{243}) .$$

The edges labelled as sharp with will not contribute to the smoothness potential, and the smoothed surface will be allowed to form a ridge along those edges.

In total, we are minimizing the sum of three terms: the likelihood term, (weighted) smoothing potential, and the edge labelling potential, which in turn consists of the edge sharpness term and neighborhood support term.

3.5 Optimization

At present we use the Metropolis sampler [Winkler, 2003] with simulated annealing for the optimization, i.e. computing a solution to (1). This is a somewhat cumbersome but flexible method, allowing for widespread experimentation with different objective functions. The clear advantage of this approach is that we do not make any assumptions about the potentials.

The Metropolis sampler is a random sampling algorithm, which generates a sequence of configurations from a probability distribution using a Monte Carlo procedure. The sampling scheme consists of randomly choosing a new label for a single site, and replacing the old label with the probability which is controlled by the current temperature. For an initially high temperature, the new configuration can be accepted even if it has a smaller probability than the old one. This allows the algorithm to leave local energy minima. The temperature then gradually decreases and the system converges.

In our case, a new label is either a new vertex position (randomly sampled in the vicinity of the present position), or a new edge label for the ridge detection. Instead of optimizing simultaneously over all defined potentials, we have in each iteration of the optimization process first detected the feature edges (consider-

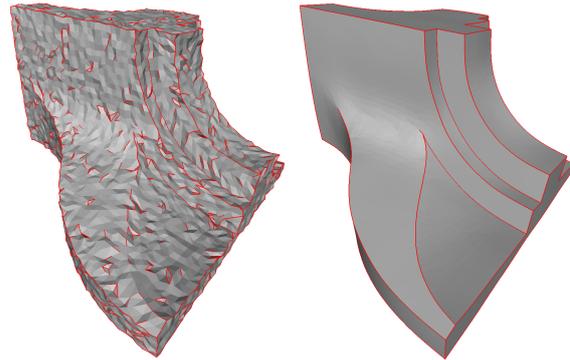


Figure 3: Smoothing fandisk model using our feature preserving method with explicit edge labelling. Left: Fandisk model corrupted with the Gaussian noise. Edges are initially labelled based only on the sharpness of the dihedral angle. Right: The resulting smooth mesh and the resulting edge labelling.

ing vertex positions to be fixed), and then displaced the vertices (considering edge labels to be fixed).

More specialized and efficient algorithms have been developed for many kind of MRF problems e.g. via filtering, belief propagation and graph cuts (in case of discrete labels). After showing that MRF is a good formulation of the mesh smoothing problem, the search for faster optimization method is part of our ongoing work. A conjugate gradient method would probably provide sufficiently good results in a more efficient way.

4 RESULTS

The results of our experiments prove the feasibility and versatility of using MRF on triangular meshes. Explicit edge labelling when smoothing models with sharp ridge features is shown in the Figure 3. In an initial noisy mesh it is impossible to detect feature edges based only on the local information. However, our algorithm converges to a configuration where all the ridges get correctly labelled and even the subtle feature edges get detected. Correct edge labelling allows us to choose aggressive smoothing prior and obtain results superior to using only a single feature preserving prior, as demonstrated in the Figure 4. Note that, unlike the fuzzy vector median smoothing (which is generally very successful in preserving edges and smooth regions), our method detects and preserves a subtle ridge in the front of the model, and is partly preserving a disappearing ridge close to models back. The most other smoothing methods will either miss those subtle ridges, or will not remove the low frequency noise.

5 DISCUSSION

There are many alternative ways of using MRF on triangle meshes. Instead of labelling vertices with spa-

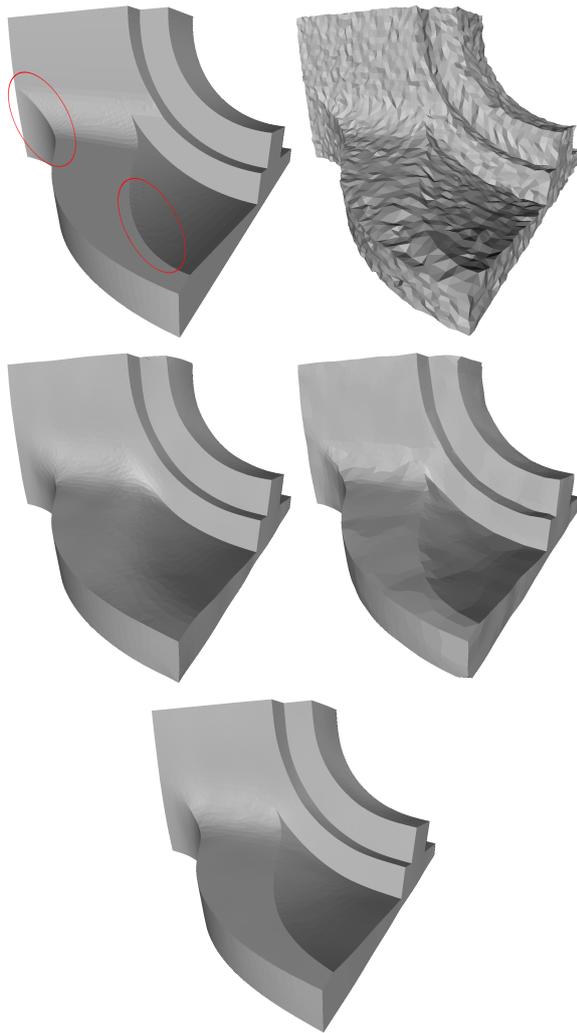


Figure 4: Smoothing fan disk model using the different feature preserving methods. Top row: Original model and the model corrupted with the Gaussian noise. The two subtle ridges are circled in the original model. Middle row: Results of fuzzy vector median smoothing and MRF smoothing using only the feature preserving square root potential. Bottom row: Results of MRF smoothing using the quadratic potential and the explicit edge labelling. Note the preserved subtle ridges.

tial positions, vertex labels can also be used to classify vertices into smooth segments. Furthermore, vertex labels could be used to detect features, classifying the vertices into those that are a part of the smooth surface, those that are on the ridge and vertices that are a corner, in a manner similar to [Lavoué and Wolf, 2008]. MRF can also be defined on mesh faces, either for segmentation or aligning face normals.

Having enough prior knowledge of the problem at hand, one can tailor the surface potentials to obtain the desired result. By including the curvature information

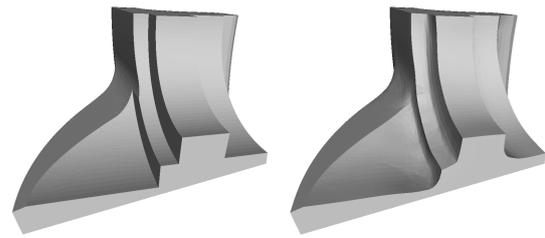


Figure 5: Obtaining curvature clamping by providing curvature information to edge detection process. Left: Initial mesh. Right: The result of clamping the curvature to discourage the concave sharp ridges.

in the edge labelling process we can detect only certain ridges, while skipping the others, obtaining curvature clamping behavior mentioned in [Botsch *et al.*, 2008] and being the focus of the recent article [Eigensatz *et al.*, 2008], see Figure 5. Extending the size of the vertex neighborhood it is possible to formulate the prior for piecewise quadratic surfaces and also model the ridge behavior more precisely.

To demonstrate the great flexibility and versatility of the MRF formulation we include another example of mesh smoothing. Inspired by a two-step smoothing method [Shen and Barner, 2004], we used MRF to obtain the smooth normal field, which is then used for reconstructing vertex positions. Now we have the mesh faces as the sites of the MRF, with the MRF labels being the normal direction of the faces. The vertex update step is taken directly from [Shen and Barner, 2004], which in turn uses a method developed by [Taubin, 2001] where the system of equations gets solved in a least squares sense to obtain the vertex positions update.

One of the important differences between the vertex based smoothing and face based smoothing is the possibility to perform smoothing of the normals without changing the geometry of the mesh, which makes this approach more effective. The disadvantage is that it is not so straightforward to include displacement-based likelihood function. The results of using this method can be seen on the Figure 6.

REFERENCES

- [Attene *et al.*, 2005] Marco Attene, Bianca Falcidieno, Jarek Rossignac, and Michela Spagnuolo. Sharpen & bend: Recovering curved sharp edges in triangle meshes produced by feature-insensitive sampling. *IEEE Trans. on Visualization and Comp. Graph.*, 11(2):181–192, 2005.
- [Botsch *et al.*, 2008] Mario Botsch, Mark Pauly, Leif Kobbelt, Pierre Alliez, Bruno Lévy, Stephan Bischoff, and Christian Rössl. Geometric model-

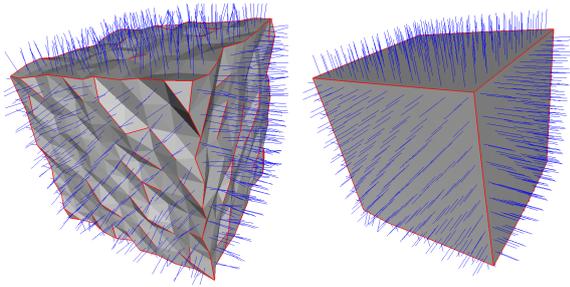


Figure 6: Smoothing a noisy cube using the face and the edge processes. Left: A synthetic cube corrupted with Gaussian noise with the initial normal field and the initial edge labelling. Right: The resulting mesh, with the smooth normal field and the resulting edge labelling.

ing based on polygonal meshes. Eurographics 2008 Full-Day Tutorial, 2008.

- [Desbrun *et al.*, 1999] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proc. of the 26th Annual Conf. on Comp. Graph. and Interactive Techniques*, pages 317–324, 1999.
- [Desbrun *et al.*, 2000] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Anisotropic feature-preserving denoising of height fields and images. In *Proc. of Graphics Interface*, pages 145–152, 2000.
- [Diebel and Thrun, 2005] James R. Diebel and Sebastian Thrun. An application of Markov random fields to range sensing. In *Proc. of Conf. on Neural Information Processing Systems*, 2005.
- [Diebel *et al.*, 2006] James Richard Diebel, Sebastian Thrun, and Michael Brünig. A Bayesian method for probable surface reconstruction and decimation. *ACM Trans. on Graphics*, 25, 2006.
- [Eigensatz *et al.*, 2008] Michael Eigensatz, Robert Walker Sumner, and Mark Pauly. Curvature-domain shape processing. *Comp. Graph. Forum*, 27(2):241–250, 2008.
- [Fleishman *et al.*, 2003] Shachar Fleishman, Iddo Drori, and Daniel Cohen-Or. Bilateral mesh denoising. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 950–953, 2003.
- [Geman and Geman, 1984] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(332):721–741, 1984.
- [Hildebrandt and Polthier, 2007] Klaus Hildebrandt and Konrad Polthier. Constraint-based fairing of surface meshes. In *SGP '07: Proc. of the 5th Eurographics Symp. on Geometry Processing*,

pages 203–212, 2007.

- [Jiao and Alexander, 2005] Xiangmin Jiao and Phillip J. Alexander. Parallel feature-preserving mesh smoothing. In *Int. Conf. on Computational Science and Its Applications (4)*, pages 1180–1189, 2005.
- [Jones *et al.*, 2003] Thouis R. Jones, Frédo Durand, and Mathieu Desbrun. Non-iterative, feature-preserving mesh smoothing. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 943–949, 2003.
- [Lavoué and Wolf, 2008] Guillaume Lavoué and Christian Wolf. Markov Random Fields for Improving 3D Mesh Analysis and Segmentation. In *Eurographics 2008 Workshop on 3D Object Retrieval*, 2008.
- [Li, 2001] Stan Z. Li. *Markov Random Field Modeling in Image Analysis*. Springer Verlag, Tokyo, second edition, 2001.
- [Shen and Barner, 2004] Yuzhong Shen and Kenneth E. Barner. Fuzzy vector median-based surface smoothing. *IEEE Trans. on Visualization and Comp. Graph.*, 10(3):252–265, 2004.
- [Szeliski and Tonnesen, 1992] Richard Szeliski and David Tonnesen. Surface modeling with oriented particle systems. In *SIGGRAPH '92: Proc. of the 19th Annual Conf. on Comp. Graph. and Interactive Techniques*, pages 185–194, 1992.
- [Tasdizen *et al.*, 2002] Tolga Tasdizen, Ross Whitaker, Paul Burchard, and Stanley Osher. Geometric surface smoothing via anisotropic diffusion of normals. In *VIS '02: Proc. of the Conf. on Visualization 2002*, pages 125–132, 2002.
- [Taubin, 1995] Gabriel Taubin. A signal processing approach to fair surface design. In *SIGGRAPH '95: Proc. of the 22nd Annual Conf. on Comp. Graph. and Interactive Techniques*, pages 351–358, 1995.
- [Taubin, 2001] Gabriel Taubin. IBM research report: Linear anisotropic mesh filtering. Technical Report RC22213, IBM Research Division T.J. Watson Research Center, 2001.
- [Willis *et al.*, 2004] Andrew Willis, Jasper Speicher, and David B. Cooper. Surface sculpting with stochastic deformable 3d surfaces. In *ICPR '04: Proc. of the 17th Int. Conf. on Pattern Recognition*, volume 2, pages 249–252, 2004.
- [Winkler, 2003] Gerhard Winkler. *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods: A Mathematical Introduction*. Springer Verlag, 2003.

Interactive Ray Tracing Client

Michal Radziszewski
AGH, Krakow, Poland
mradzisz@student.agh.edu.pl

Witold Alda
AGH, Krakow, Poland
alda@agh.edu.pl

Krzysztof Boryczko
AGH, Krakow, Poland
boryczko@agh.edu.pl

ABSTRACT

In this paper we present an interactive GPU-based, GUI client, working with rendering server employing ray tracing based global illumination. The client is designed to guarantee interactivity (namely 1/60sec response time) no matter how slow the rendering server is. The client dynamically adjusts image resolution to match the server performance and complexity of the rendered scene. When the scene is modified, the image may appear out of focus and noisy, depending on the machine computational power, but usually is readable. With no interrupt from the client, the image is progressively improved with new data from the server. The system exploits hybrid programming model – CPU for the server and GPU for the client.

Keywords: Real-time global illumination, quasi-Monte Carlo ray tracing, hybrid CPU and GPU programming.

1 INTRODUCTION

Many contemporary approaches to ray tracing based global illumination rely on computational power of graphics hardware, eg. [25]. Unfortunately, true, unrestricted, global illumination algorithms, which solve the Rendering Equation [9], are not well suited for GPU architecture. Such implementation is possible, as has been shown numerous times, but is severely restricted when compared with classic multi-core CPU solutions, since GPUs cannot process irregular data structures effectively [7].

Our renderer, based on significantly modified Bidirectional Path Tracing [22] and Photon Mapping [8] with quasi-Monte-Carlo (QMC) approach [12] is designed for flexibility of CPUs. It allows rendering, in full spectrum, of arbitrary scene primitives, arbitrary materials, textures, and more. The only restriction is, in fact, a computer memory size. Such, traditionally CPU based, algorithms are rather difficult to port to GPUs. When, despite all problems, they are ported eventually, performance benefits of GPUs over multicore CPUs are often questionable [7].

This paper presents a different approach to obtain interactivity. Pure ray tracing algorithms are based on point sampling scene primitives, not using scan line rasterization at all. This gives much freedom in the way how samples are chosen, however QMC ray tracing algorithms produce a huge number of samples, which do not fit in raster RGB grid. Converting these data to 3x8bit integer based RGB image at interactive frame

rates may be impossible even for multi-core CPUs, especially when dynamic image resolution has to be adjusted to the server rendering speed and scene complexity, with some non-trivial post-processing added. As we will show, conversion of ray tracing output to a displayable image and many post-processing effects can be expressed purely by rasterization operations, in which GPUs excel. The main idea behind the presented approach is therefore the usage of the best suitable processor for a given algorithm, instead of porting everything to GPUs.

2 RELATED WORK

The concept of ray tracing is not new [27]. Because it can produce much better image than hardware rasterization, for several years there has been a lot of research dedicated to run it in real time, despite its high computational cost [15]. Ray tracing based global illumination is even more expensive. However, for some time now real time global illumination algorithms are being developed also [23].

Just after the appearance of first programmable DirectX 9 class graphics processors there were first attempts to use it for ray tracing [17]. Nowadays, vast majority of contemporary real time global illumination algorithms are based on computational power of modern GPUs, e.g. [11, 25]. Unfortunately, they still put restrictions, often quite severe, on scene content (limited range of material and geometry representation), scene size, and illumination phenomena which are possible to capture.

However, this is not the only way to obtain interactivity – nowadays multi-CPU Intel workstations can perform interactive ray tracing [16], yet true global illumination is still unachievable. Interactivity can also be obtained using clusters of machines with CPU rendering [2].

On the other hand, approach presented here is substantially different from those above – placing abso-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

lately no restrictions on scene and illumination effects. It uses GPU just to display and postprocess image made from CPU ray traced point samples, in resolution dynamically adjusted for real time performance.

3 REQUIRED SERVER OUTPUT

In general the server may run any point sampling algorithm, but in this project we rely on QMC ray tracing. The visualization client assumes the specific format of the server's output. In the following subsections we describe in detail the conditions which should be fulfilled to make the client work properly. We also show how to convert Photon Mapping to meet these assumptions.

The server should provide stream of color values scattered uniformly at random locations in the screen space. The uniformity of sampling ensures acceptable image quality even at low sampling rates, which is typical due to high computational cost of ray tracing.

Additionally, the output stream should be generated roughly uniformly in time. Otherwise the client might fail to maintain interactive refresh rates.

3.1 Bidirectional Algorithms

Some most advanced ray tracing algorithms trace rays in both directions – from the camera towards lights (camera rays), and in the opposite one (light rays). Such approaches produce two kind of samples, which must be processed differently in order to produce displayable images [22].

The client accepts two input streams. The format of samples is identical in both streams: $([u, v], [x, y, z, w])$, where $[u, v]$ are screen space coordinates, in $[0, 1]^2$ range, or, perhaps, with slight overscan to avoid post-process filtering edge artifacts, x, y, z is sample color value in CIE standard [6], and w is sample weight.

The two streams differ only in interpretation of sample density. The pixels of image from camera rays are evaluated by averaging local samples using any suitable filter – sum of weighted samples is divided by sum of weights. On the other hand, pixels of light image are formed using a suitable density estimation technique – samples are filtered and summed, but not divided by sum of weights. Therefore, a sample density affects only quality of camera image, while it affects both quality and brightness of light image. The final, displayable, image is a sum of both camera and light images, the latter divided by a number of traced paths.

Obviously, not all ray tracing algorithms need both – camera and light – output streams. For example, Path Tracing [9] and Photon Mapping [8] produce camera samples only, while Particle Tracing [1] needs only light image. Therefore, the visualization client employs an obvious optimization – it skips processing of a stream given that no samples were generated into it.

3.2 Coherent vs. Non-Coherent Rays

For some time now it is often claimed that it is beneficial to trace camera rays in a coherent way, because it can significantly accelerate rendering [24, 2]. This is true, but only for primary rays (sent directly from camera or light source). Unfortunately, rays, which are scattered through the scene, do not follow any coherent pattern and caching does not help much. Since true global illumination algorithms typically trace paths of several rays, these algorithms do not benefit much from coherent ray tracing.

What is more, coherent ray tracing tends to provide new image data in tiles, which make progressive improvement of image quality difficult. On the other hand, we have chosen to spread even primary rays as evenly as possible, using carefully designed Niederreiter-Xing QMC sequence [13] as the source of pseudorandom numbers. Therefore, it can be expected that very few traced rays provide reasonable estimate of colour of the entire image, and subsequently traced rays improve image quality evenly.

3.3 Full Spectral Rendering

Having in mind further processing, it may be useful to output full spectral images [5, 18]. However, full spectral representation requires huge amount of memory. For example, full HD spectral image in 16bit floating precision and with 3nm wavelength sampling from 400nm to 700nm needs as much as $1920 \times 1080 \times 100 \times 2B \approx 400MB$, while RGB one requires $1920 \times 1080 \times 3 \times 2B \approx 12MB$.

The standard CIE XYZ space seems to be the best option instead, since an RGB space, which depends on a particular display hardware, is not a plausible choice. For this reason our client accepts CIE XYZ color samples. The presented server natively generates full spectral data and converts it internally from full spectrum to the three component color space.

3.4 One-pass Photon Mapping

Original Photon Mapping [8] is a two pass technique. This obviously violates the requirement of steady sample stream – during photon tracing there are no samples generated, causing high latency before image starts to appear. We have found that Photon Mapping actually can be done in one pass, with only minor loses in efficiency compared to the original approach. The new algorithm uses a linear function of number of image samples (n) to estimate minimal necessary photon count in photon map to obtain image with quality determined by n . Therefore, the photon map is no more static structure – new photons are added while new image samples are rendered.

Immediately two issues have to be solved – synchronization of read and write accesses to the photon map

structure in parallel photon mapping and balancing kd-tree. Synchronization can be performed with simple read-write locks (classic readers-writers problem).

On the other hand, kd-tree balancing requires significant algorithm modification. We have chosen to balance the scene space instead of photons. The original algorithm starts with bounding box of all photons (unknown in our approach) and in each iteration places splitting plane at a position such that half of the photons remains on the one side of the plane. Otherwise, our algorithm starts with bounding box of the entire scene, and in each iteration it splits it in half across dimension in which the box is the longest. Splitting stops when all nodes contain less photons than a certain threshold (5-6 seems to be optimal) or a maximum recursion depth is reached. Adding new photons require just splitting of some of the nodes, where there happens to be too many photons.

The idea is somehow similar to Irradiance Caching algorithm [26]. Similarly as in this method, our approach starts with empty structure and fills it through rendering. However, Irradiance Caching calculates irradiance samples when they are needed by camera rays, while our modified Photon Mapping traces photons in a view independent manner.

Strictly speaking, the new approach does not generate batches of samples in roughly uniform time. Due to kd-tree lookup computational complexity as well as linear dependence between number of photons in kd-tree and number of samples computed, the average time to calculate n th sample is the order of $\mathcal{O}(\log n)$, where n is the sample number. Logarithm, however, changes slowly, and the client is designed to adjust to slow changes of rendering speed by modifying size of batch of samples.

4 CLIENT AND SERVER ALGORITHMS

Finally, a GPU task is to convert point samples into a raster image. The conversion is done with resolution dynamically adjusted to the number and variance of point samples. In the image, a color conversion from XYZ to RGB space of current monitor, together with gamut mapping, tone mapping, gamma correction and other post-processing effects are performed.

As a target platform we have chosen a GPU compatible with OpenGL 3.x [19] and GLSL 1.5 [10]. Major part of algorithm is coded as a GLSL shader, which suits our needs very well. Recent technologies, such as Nvidia CUDA, ATI Stream, or currently being developed OpenCL are not necessary for this kind of algorithm.

The rendering task is split into two processes (or threads in one process, if a single application is used as a client and server) running in parallel: a server wrapper process and visualization process. The rendering

process may be further split into independent threads, if multicore CPUs or multiple CPU machines are used.

4.1 Server Wrapper Process

Ray tracing can produce virtually unlimited number of samples, being limited only theoretically by machine numerical precision (our implementation can generate as many as 2^{64} samples before sample locations eventually start overlap). Therefore, ray tracing process is reset only immediately after user input, which modifies the scene. Otherwise, it runs indefinitely, progressively improving image quality.

The server wrapper runs on a separate thread, processing commands. The wrapper recognizes three commands: *term*, *abort*, and *render*. The *term* command causes wrapper to exit its command loop, and is used to terminate the application. The *abort* command aborts current rendering, and is used to reset server to the new user input (for example, camera position change).

The *render* command orders server to perform rendering. The rendering is aborted when either *abort* or *term* command is issued. Maximum time to abort rendering is a time necessary to generate just one sample. Any algorithm capable of generating the specified output (see Section 3) can be used. In our server implementation, rendering is performed in parallel on multicore CPUs.

The wrapper allows registering asynchronous *finish* event. This event is generated when rendering is finished (either a prespecified number of samples was generated or *abort* was issued). The event can be used to synchronize client with server. Apart from sending asynchronous messages, the wrapper can be queried synchronically for already rendered samples. Since this query just copies the data to the provided buffer, server blocking due to synchronization takes little time.

4.2 Client Process

Client is responsible for visualizing samples generated by server, and additionally it processes GUI window system messages. Client stores its internal data in the four screen-aligned textures, in the IEEE 32bit floating point format. A 4-channel $[X, Y, Z, W]$ texture and a single component variance $[Var]$ texture are stored for camera and light input streams. Therefore, client stores 40 bytes of data per screen pixel, apart from standard integer front and back buffers. The details of client main loop are presented in Figure 1.

When all GUI messages are processed, client rasterizes new samples, generated by the server, into its internal textures. This task is performed by the render-to-texture feature of Framebuffer Object (FBO). The client sets an empty vertex program, which only passes through data, and a geometry program which is equivalent to rendering textured point sprites fixed functionality. The input is a stream of two elements – two

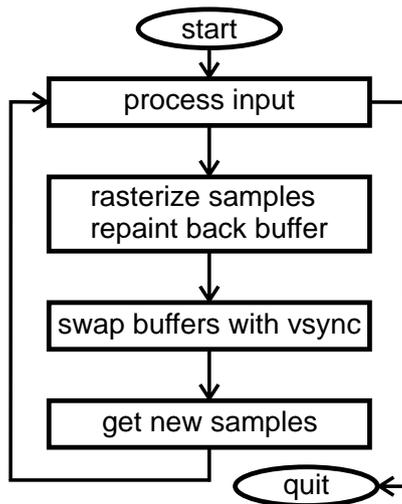


Figure 1: Main loop of visualization client process.

component screen position (u, v) and four component color (x, y, z, w) . Input is placed in Vertex Buffer Object (VBO), and is then rendered with GL 'render points' command. Points are rendered in blending mode set to perform addition, ensuring that all samples add up instead of overwriting previous texture content.

Additional input is a monochromatic HDR filter texture, used to draw point sprites. The texture is normalized (all the texel values add up to one) and the texture border value is set to zero. The filter texture is applied without rescaling and with bilinear filtering, thus preserving filter normalization, which is crucial for algorithm correctness. We have found that 5x5 texel windowed Gaussian blur gives good results.

The rendering is performed in two passes. First, color textures are updated. In the second pass, using already up-to-date color textures, variance textures are updated. In both passes, the same samples are rendered. The variance is updated using the formula $V_j = V_{j-1} + \sum_i (Y_i - \bar{Y}_j)^2$, for j th batch of i samples. The formula does not give the best theoretically possible results, since the mean \bar{Y} is approximated using only already evaluated samples. The alternative formula $V_j = Y2_j - \bar{Y}_j^2$, $Y2_j = Y2_{j-1} + \sum_i Y_i^2$, which requires storing sum of squares ($Y2$) instead of variance, should be avoided due to poor numerical stability (even negative variance results are possible). In both formulas the division by $n - 1$ factor, where n is the total number of samples in a given stream, is omitted. This division is performed when variance data is read from its texture.

The sample rasterization algorithm works as follows:

1. The content of client sample buffer (pairs $[u, v], [x, y, z, w]$) is loaded into VBO, interpreted as 2D point coordinates and 4D color. There is one buffer for both streams. Samples which come from light stream are encoded with negative weights.

Stream separation is performed further in the fragment program.

2. Monochromatic float texture with filter image is selected and point draw command is issued. The texture is used as a texture sprite for emulated point sprites. Fragment program performs multiplication of 'color' attribute by the texture value $[X, Y, Z, |W|]$. The output is saved to the color texture of camera stream if $W \geq 0$ or light stream otherwise.
3. After rasterization, textures are detached from FBO, GPU MIP-map build command is issued.
4. Texture $LoDs$ (used by 'repaint back buffer' processing) for both streams are evaluated as $LoD_i = \log_4(P/S_i)$, where P is number of pixels on the screen and S_i is the number of samples from i th stream computed so forth.
5. Second draw is issued, with variance textures as output this time. The variance is evaluated only for luminance (Y) component, since three component variance typically do not help much and substantially complicates algorithm. Variance output for each stream is $(Y_{avg} - Y)^2$, where Y_{avg} is read from previously generated color texture, and Y is luminance of currently processed sample, multiplied by filter texture.
6. Similarly to color textures, variance textures are detached from FBO, GPU MIP-map build command is issued.

In order to repaint back buffer, client draws a screen-sized quad, using the four textures as an input. The screen is filled with custom fragment program. The program accepts following control parameters: level of detail (LoD) for both streams, light image weight (Lw), image brightness (B), contrast (C), gamma (G), color profile matrix (P), and variance masking strength (Vm). Level of detail (LoD) is already evaluated during rasterization. Now, the LoD values are used by fragment program to blur texture data if not enough samples are computed. Light image weight is got from the server along with samples, and its value is equal to the number of paths traced from light sources. This parameter is used to scale light image texture appropriately, such that the texture can be summed with camera image texture.

Image brightness, contrast, gamma and color profile are set by the user, and their values adjust the image appearance. Additionally, the visualization client is able to add a glare effect as an additional post-process, implemented as a convolution with a HDR glare texture, generated according to [20]. However, sufficiently large glare filters are far beyond computational power of contemporary GPUs for real-time screen refresh rate. Since

these parameters are defined only for client, and do not affect server rendering at all, their values can be modified freely without resetting the server rendering process.

Variance of samples is estimated only for luminance (CIE Y channel), using the standard variance estimator ($V \approx \frac{1}{N-1} \sum (E(Y) - Y_i)^2$, where N is the number of samples, Y_i are luminance values, and $E(Y)$ is the luminance value estimated from samples computed so far. The client is able to increase blurriness according to the local changes in estimated variance, hence slightly masking noise produced by stochastic ray tracing. The noise to blurriness ratio can be controlled by Vm parameter.

The blurriness is created by low pass filter or bilateral filtering [14] guided by variance estimation, which potentially can be much better in preserving image features than a simple low pass filter. However, bilateral filtering works correctly only if noise is less intense than image features. When image is heavily undersampled, this assumption may not be satisfied, and a low pass filter remains the only viable option. For example, in Figure 3, the two leftmost images cannot be enhanced by bilateral filtering. On the other hand, this technique does a good job improving the quality of middle image from Figure 5.

Unfortunately, the noise masking feature can hide only the random error which is the result of variance. It cannot hide (in fact, it cannot even detect) other kind of error resulting from bias. The variance is the only source of error in Bidirectional Path Tracing, while Photon Mapping error is dominated by bias.

The algorithm processes its input as follows:

1. The program reads data from both variance maps, using requested $LoDs$ through hardware MIP-mapping.
2. $LoDs$ for both streams are evaluated according to initial $LoDs$, the variance and Vm , for i th stream: $LoD'_i \leftarrow LoD_i + Vm \log_4([Var])$.
3. $[X, Y, Z, W]$ textures of both streams are sampled, this time using just evaluated LoD' and custom filtering technique (hardware MIP-mapping produces very poor results, see section 4.3 for more detailed discussion).
4. Texture samples for both streams are normalized, i.e. $[X, Y, Z, W] \rightarrow [X/W, Y/W, Z/W, 1]$ (if $W = 0$, then sample is considered to be $[0, 0, 0, 1]$). Then, light texture sample, divided by Lw , is added to camera texture sample, producing single result for further processing.
5. Optionally, glare effect is applied here. Our glare texture is generated to be applied in XYZ color space instead of RGB one.

6. Tone mapping of luminance (Y) is performed, using very simple yet effective procedure: $Y' \leftarrow 1 - \exp(-(B * Y)^C)$, while X and Z components are scaled by Y/Y' ratio. If $Y = 0$ it means that image is black at that point and $X'Y'Z' \leftarrow (0, 0, 0)$ is used.
7. Resulting $X'Y'Z'$ is multiplied by matrix P , and a basic gamut mapping is performed. We do not use elaborated algorithms here – simple desaturation of out-of-gamut colors, just to keep mapped luminance unmodified, works reasonably well. Now output is in RGB format, normalized to $[0, 1]$ range.
8. Finally, gamma correction using G is performed.

Next, client swaps front and back buffers, in synchronization with screen refresh period. This guarantees constant frame rate (typically 60Hz for common LCDs).¹ Finally, client reads new samples from the server. The reading is performed with synchronization, blocking the server for a moment. However, client does not display samples immediately, blocking server just for copying this portion of data to its internal buffer for later processing.

4.3 MIP-mapping Issues

Images produced by rasterizing ray traced samples are created as screen-sized textures. Should enough samples be generated, these images could be used immediately without any resampling. Unfortunately, contemporary CPUs are far too slow to generate at least $\#screen_pixels$ of such samples in, say, $1/30sec$, which is required for real time performance. Therefore, some kind of blurring texture data, according to fraction of necessary samples generated and the local sample variance, have to be performed.

While MIP-mapping is reasonably good in filtering out texture details which would otherwise cause aliasing, it cannot be used reliably to blur the texture image. Blurring by using LoD bias parameter of texture sampling function produces extremely conspicuous and distracting square pattern, with severe bilinear filtering artifacts (see Figure 2 for details). This is not surprising, since a GPU uses box filter to generate MIP-maps and linear interpolation between texels to evaluate texture value at sampled point. Moreover, MIP-mapping with polynomial reconstruction instead of linear one fails as well. We have used custom texture sampling with Catmull-Rom spline interpolation for this purpose.

¹ GPU class must be properly selected for a monitor resolution. If GPU is too poor, interactivity is not obtained. We found that best contemporary single processor GPU (Nvidia GTX 285, at the time of testing) is enough for refresh rate of 30Hz in full HD. Such issue, however, does not slow down the server – the same number of samples is still rendered in the same amount of time, they are just displayed more rarely, in larger batches.

Visually good results can be obtained by using Gaussian blur:

$$I(u,v) = \frac{\sum_i \sum_j T_{ij} g_{ij}(u,v)}{\sum_i \sum_j g_{ij}(u,v)}.$$

The I is texture sample, u, v is the sample position, T are texel values, and $g_{ij} = \exp(-\sigma d_{ij}^2)$ is the filter kernel, with σ controlling blurriness, and d_{ij} is the distance between the u, v position and texel T_{ij} . Unfortunately, direct implementation of Gaussian blur requires sampling an entire texture for evaluation of any texture sample, which is far beyond computational capabilities of contemporary GPUs. The weight of Gaussian filter, however, quickly drops to zero with increasing distance from evaluated sample. Truncating the filter to a fixed size window containing limited number of samples is a commonly used practice.

The simple truncation is not always optimal, since quality of truncated Gaussian filter depends strongly on the σ parameter – to obtain similar quality with different sigmas, an $\mathcal{O}(\sigma^{-1})$ number of texels have to be summed. That is, if a Gaussian filter is truncated too much, it starts to resemble a box filter. In our case, σ varies substantially, and therefore more advanced technique should be used. We may notice that decreasing a resolution of the original image twice, and increasing σ four times, approximates the original filter on the original image. Eventually, the following algorithm is employed: initial MIP-map level is set to zero, and while σ is smaller than a threshold t , the σ is multiplied by four, and MIP-map level is increased by one.

The threshold t and number of summed texels have been adjusted empirically to balance the blur quality and computational cost. First we have found that truncation range R of roughly 2.5 is a maximum value which ensures reasonable performance. For such truncation, setting $t \approx 1$ is reasonable. Additionally, it is better to use a product of g and smooth windowing function w instead of original g if truncation is used. The $w = 1 - \text{smoothstep}(0, R, d)^E$, where E controls how quickly w drops to zero with distance, works quite well. The value $E = 8$ yields good results.

What is more, the transition between MIP-map levels is noticeable and decreases image quality. This is especially distracting if σ varies across the image, which is the case because blur is adjusted to the locally estimated variance. Therefore, similarly as in trilinear filtering, the Gaussian blur is performed on two most appropriate MIP-map levels, and the results are linearly interpolated, avoiding sudden pops when MIP-map level changes. Therefore, truncation to range 2.5 cause blurring to use $2[(2 \cdot 2.5)^2] = 50$ texture fetches on average, which is costly, yet acceptable on contemporary GPUs.

The sophisticated filtering scheme is used only for $[X, Y, Z, W]$ textures. Variance $[Var]$ textures, not being displayed directly, do not have to be sampled with any-

thing more complicated than basic MIP-mapping. This saves some computational power of a GPU, yet does not produce noticeable visual artifacts.

5 RESULTS

The quality of rendered images obviously mostly depends on the rendering algorithm used. We have tested the visualization client in cooperation with Path Tracing (Figure 3) and Photon Mapping (Figure 4). Both figures present initial image rendered after 1/30sec and show the speed of image quality improvement. All the tests were performed on Intel Core i7 CPU and Nvidia 9800 GT GPU, in 512x512 resolution.

The client is responsible merely for visualization and postprocessing, assuming that it is provided with stream of point samples, scattered roughly evenly through entire image. The only algorithm for image quality improvement is noise reduction based on variance analysis. The error due to variance (seen as high frequency noise) is much more prominent in results of Path Tracing than in Photon Mapping, so the noise reduction has been tested on the first algorithm. The results are presented in Figure 5.

When multiple processors are used in the same application, good load balancing is important. While it is well known how to load balance ray tracing work between multiple CPUs, in our application it is impossible to balance loads between visualization client and ray tracing server. The subtasks performed by CPUs and GPU are substantially different and suited for different architectures of these two processors, so work cannot be moved to the less busy unit as needed. In fact, on contemporary machines rendering server is always at full load, and GPU can be not fully utilized, especially when low resolution images are displayed. However, it is good to have some reserve in GPU power to ensure real time client response.

6 CONCLUSION

We have presented an interactive GUI visualization client for displaying ray traced images online, written mainly in GLSL. Apart from visualization, the client can hide noise of input data by means of variance analysis. Additionally, the client can apply glare effect as a postprocessing technique, which is performed quite efficiently on GPU.

The client is able to obtain interactivity regardless of the ray tracing speed. However, the price to pay is blurriness of images rendered at interactive rate. Nevertheless, the image quality improves quickly with time whenever rendered scene is not changed.

Our approach scales well with increasing number of CPU cores for ray-tracing, as well as with increasing number of shader processors on a GPU. Moreover, the program never reads results from the GPU, so it does not cause synchronization bottlenecks, and should

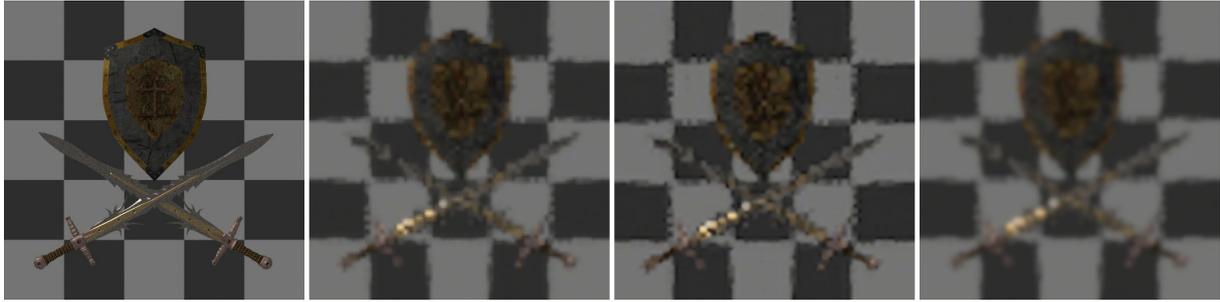


Figure 2: Comparison of MIP-mapping and custom filtering based blur quality. From left: reference image, hardware mipmapping, custom reconstruction based on Catmull-Rom polynomials, windowed Gaussian blur.

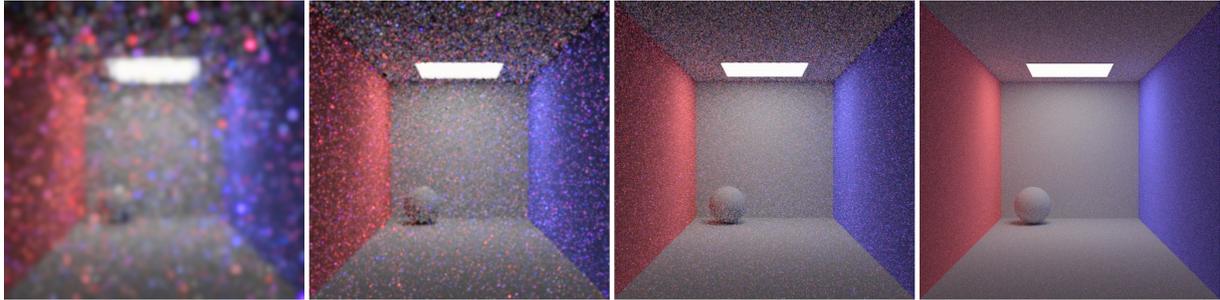


Figure 3: Results of Path Tracing (from left: after 1/30sec, 1/3sec, 3sec, 30sec). The Path Tracing error appears as noise, blur in the first two images is caused by undersampling (far less than 1 sample per pixel were evaluated).

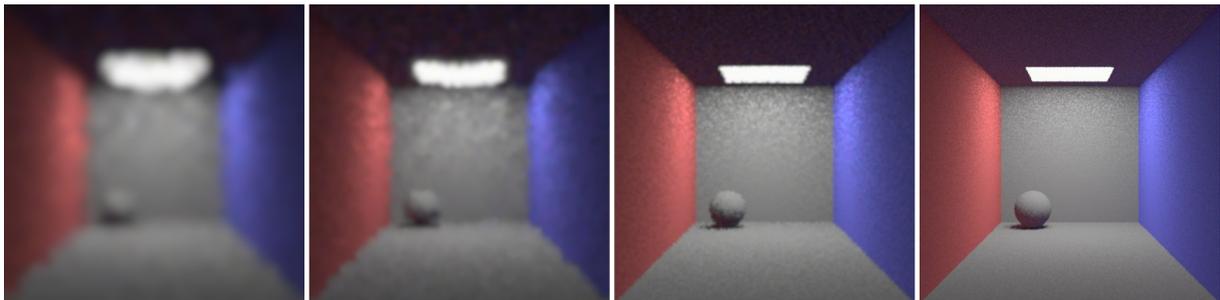


Figure 4: Results of Photon Mapping (from left: after 1/30sec, 1/3sec, 3sec, 30sec). Photon Mapping does not produce much noise, but due to overhead caused by photon tracing and final gathering, less image samples than with Path Tracing were computed, which cause some blurriness.

be friendly with multi-GPU technologies like SLI or Crossfire.

Additionally, we have modified the Photon Mapping algorithm to be a one-pass technique, with the photon map being updated interactively during the whole rendering process. This enables using Photon Mapping with the presented visualization client, which then could ensure progressive image quality improvement, without any latencies resulting from construction of photon map structure.

Our visualization client has a lot of potential for future upgrades. The adaptive filtering technique [21] seems to be good approach to significantly reduce image noise on the side of the visualization client. Moreover the client can be extended to support frameless rendering [3, 4]. This very interesting and promising technique can improve image quality substantially us-

ing samples from previous frames, provided that subsequent images do not differ too much.

In future we plan to introduce to our client stereo capability, using OpenGL quad-buffered stereo technology. Ray tracing algorithms can easily be converted to render images from two cameras at once, and a lot of them can do this even more efficiently than rendering two images sequentially (for example, Photon Mapping can employ one photon map for both cameras, and similarly, Bidirectional Path Tracing can generate one light subpath for two camera subpaths). Unfortunately, stereo rendering doubles the load on the GPU shaders, as well as on the GPU memory. However, it seems that interactive stereo can be obtained by slight decrease of custom texture filtering quality.

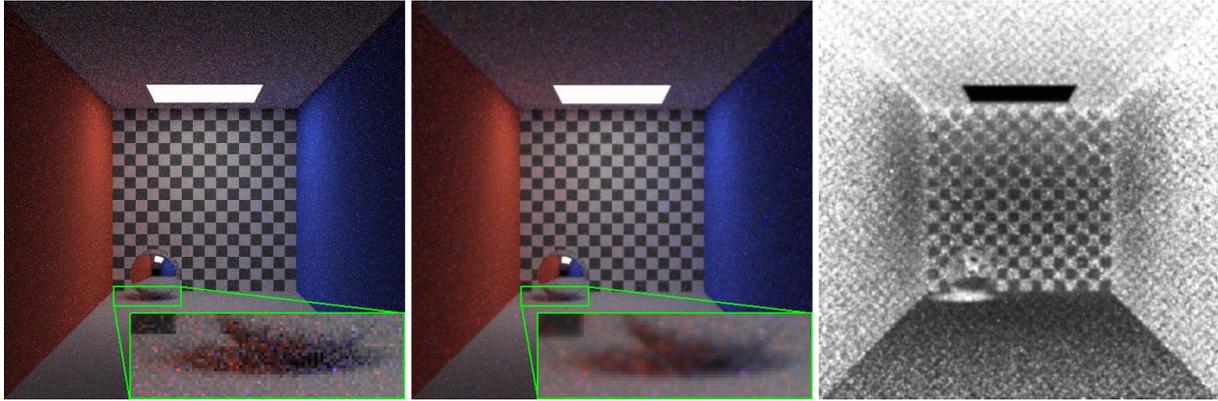


Figure 5: Noise reduction based on variance analysis of Path Tracing image (from left: no noise reduction, with noise reduction, variance image). The difference is noticeable especially in shadowed area beneath the sphere and on the indirectly illuminated ceiling.

ACKNOWLEDGEMENTS

Support of this work by AGH Grant number 11.11.120.865 is kindly acknowledged.

REFERENCES

- [1] James Arvo and David Kirk. Particle Transport and Image Synthesis. In *SIGGRAPH 1990 Proceedings*, pages 63–66, New York, NY, USA, 1990.
- [2] Carsten Benthin. *Realtime Ray Tracing on Current CPU Architectures*. PhD thesis, Saarland University, Saarbrücken, Germany, 2006.
- [3] Gary Bishop, Henry Fuchs, Leonard McMillan, and Ellen Scher Zagier. Frameless rendering: Double buffering considered harmful. In *SIGGRAPH 1994 Proceedings*, volume 28, pages 175–176, New York, NY, USA, 1994.
- [4] Abhinav Dayal, Cliff Woolley, Benjamin Watson, and David Luebke. Adaptive Frameless Rendering. In *Rendering Techniques 2005*, pages 265–275, 2005.
- [5] Kate Devlin, Alan Chalmers, Alexander Wilkie, and Werner Purgathofer. Tone reproduction and physically based spectral rendering. In *State of the Art Reports, Eurographics 2002*, pages 101–123, September 2002.
- [6] Bruce Fraser, Chris Murphy, and Fred Bunting. *Real World Color Management, second edition*. Peachpit Press, Berkeley, CA, USA, 2005.
- [7] Anwar Ghuloum. The Problem(s) with GPGPU. http://blogs.intel.com/research/2007/10/the_problem_with_gpgpu.php, 2007.
- [8] Henrik Wann Jensen. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001.
- [9] James T. Kajiya. The rendering equation. In *SIGGRAPH 1986 Proceedings*, pages 143–150, New York, NY, USA, 1986.
- [10] John Kessenich, Dave Baldwin, and Randi Rost. *The OpenGL Shading Language, version 1.50*, 2009.
- [11] Morgan McGuire and David Luebke. Hardware-accelerated global illumination by image space photon mapping. In *Proceedings of the 2009 ACM SIGGRAPH/EuroGraphics conference on High Perf. Graphics*, New York, NY, USA, 2009.
- [12] Harald Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, Philadelphia, USA, 1992.
- [13] Harald Niederreiter and Chaoping Xing. Low-discrepancy sequences and global function fields with many rational places. *Finite Fields and Their Applications*, 2(3):241–273, jul 1996.
- [14] Sylvain Paris, Pierre Kornprobst, Jack Tumblin, and Frédo Durand. A gentle introduction to bilateral filtering and its applications. *Siggraph 2008 course notes*, 2008.
- [15] Steven Parker, William Martin, Peter-Pike Sloan, Peter Shirley, Brian Smits, and Charles Hansen. Interactive Ray Tracing. In *Symposium on Interactive 3D Graphics*, pages 119–126, 1999.
- [16] Daniel Pohl. Light It Up! Quake Wars Gets Ray Traced. *Intel Visual Adrenaline*, 2:34–39, 2009.
- [17] Timothy J. Purcell, Ian Buck, William R. Mark, and Pat Hanrahan. Ray tracing on programmable graphics hardware. *ACM Transactions on Graphics*, 21(3):703–712, 2002.
- [18] Michal Radziszewski, Krzysztof Boryczko, and Witold Alda. An Improved Technique for Full Spectral Rendering. *Journal of WSCG*, 17(1):9–16, 2009.
- [19] Mark Segal and Kurt Akeley. *The OpenGL Graphics System: A Specification, version 3.2*, 2009.
- [20] Greg Spencer, Peter Shirley, Kurt Zimmerman, and Donald P. Greenberg. Physically-based glare effects for digital images. In *SIGGRAPH 1995 Proceedings*, pages 325–334, New York, NY, USA, 1995. ACM.
- [21] Frank Suykens and Yves D. Willems. Adaptive Filtering for Progressive Monte Carlo Image Rendering. In *Proceedings of the 8th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media (WSCG) 2000*, pages 220–227, 2000.
- [22] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, Stanford, CA, USA, 1997.
- [23] Ingo Wald, Carsten Benthin, and Philipp Slusallek. Interactive Global Illumination Using Fast Ray Tracing. In *Proceedings of the 13th Eurographics Workshop on Rendering*, pages 15–24, June 2002.
- [24] Ingo Wald, Philipp Slusallek, Carsten Benthin, and Markus Wagner. Interactive rendering with coherent ray tracing. In *Computer Graphics Forum*, pages 153–164, 2001.
- [25] Rui Wang, Rui Wang, Kun Zhou, Minghao Pan, and Hujun Bao. An efficient gpu-based approach for interactive global illumination. *ACM Transactions on Graphics*, 28(3):1–8, 2009.
- [26] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. In *SIGGRAPH 1988 Proceedings*, pages 85–92, New York, NY, USA, 1988.
- [27] Turner Whitted. An Improved Illumination Model for Shaded Display. *Communications of the ACM*, 23(6):343–349, 1980.

A Novel Image Transformation For Solving Complex Image Mapping Problems

Igor V. Maslov	Yulia D. Detkova	Izidor Gertner
St-Petersburg State Polytechnical University	St-Petersburg State Polytechnical University	The City College of New York
29 Polytechnicheskaya street	29 Polytechnicheskaya street	138th Street at Convent Avenue,
Russia195251, St.-Petersburg	Russia195251, St.-Petersburg	NAC 8/206
ivm3@columbia.edu	detkova@avalon.ru	USA 10031, New York, NY
		gertner@cs.ccny.cuny.edu

ABSTRACT

The paper proposes a novel image transformation called Image Local Response (ILR) that can be used for solving complex image mapping problems. The proposed transformation brings together two approaches based on the pixel value distribution and image features. Image local response is defined as the average value of the difference between the transformed and the original copies of the same image whereby the transformation is small, i.e., the components of the corresponding parameter vector have sufficiently small unit values. The response has a few interesting properties useful in image mapping. The validity of the proposed image transformation is shown on sample complex image mapping problems formulated as the multi-objective piecewise imaging optimization problem.

Keywords

Image mapping, response analysis, imaging optimization, evolutionary algorithm.

1. INTRODUCTION

Many tasks related to digital image processing deal with comparing (i.e., matching or mapping) images of different types and sizes. Examples of such tasks include e.g., image registration, object or target recognition, and pattern matching. These tasks, in turn, play a pivotal role in many important real world applications like remote sensing, security systems, robotics, computer vision, medical imaging, information fusion, and industrial control.

The approaches that can be used for comparing the images can be divided into two main groups.

1. The first group of methods compares the distributions of the pixel values in the images, either explicitly or implicitly. One of the problems associated with this approach is related to the changing light conditions between the images. In this case, the comparison of the pixel values becomes difficult since no matching pixels can be found.

Moreover, the comparison of the different types of imagery, e.g., infrared and real visual images obtained from the different types of sensors (as in multi-sensor image fusion) becomes virtually impossible using this approach.

2. Methods in the second group attempt to find a set of salient characteristics, i.e., features that are common for the compared images. Choosing the appropriate features is by no means a trivial task. It becomes even more difficult if the images are simultaneously misaligned and distorted by some sort of complex geometric transformation, e.g., affine or perspective.

The proposed in the paper image transformation uses the combination of the both abovementioned approaches; the transformation is called Image Local Response (ILR). The concept of ILR is somewhat related to image neighborhood and block operations [Seu00a], [Pit00a], [Ima06a], as well as to the node and edge functions proposed in [Muc98a], although it is based on a fundamentally different idea rooted in Green's functions [Bar89a] and response analysis [Ger02].

The paper is organized as follows. Section 2 gives the definition of Image local response and describes its useful properties. Section 3 discusses sample experimental results of object mapping in the case of geometrically distorted images. Section 4 concludes

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

the paper with the summary of the proposed approach.

2. DEFINITION AND PROPERTIES OF IMAGE LOCAL RESPONSE

In digital image processing, solving image mapping (matching) problem means finding an adequate vector V of parameters defining the unknown transformation A between the images. In its most general form, the sought transformation A can be a fairly complex one, although in many cases it can be represented or approximated with some suitable general affine transformation.

The concept of Image Local Response (ILR) is based on a fairly simple and rational idea: since the mapping problem searches for the unknown transformation A , it seems logical to explore the response of the image to this particular type of transformation. This task can be accomplished by mapping a transformed image Img' onto self (i.e., onto the original image Img), with a sufficiently small transformation vector V_u . In accordance with this idea, Image local response R_p at a point P is defined as the value of the difference F between the transformed, Img' and the original, Img copies of the same image. Here, the transformation A_u at the point P is small, i.e., the components of the parameter vector V_u have sufficiently small unit values.

The simplest way of defining the image difference F is to compute a squared difference of the pixel gray values over some area ω_R , in the following way:

$$F = \frac{\sum_{\omega_R} (g(x', y') - g(x, y))^2}{\omega_R^2}, \quad (1)$$

where $g(x, y)$ and $g(x', y')$ are the gray values of the image Img in the area ω_R before and after the transformation, correspondingly [Bro98a].

Making the area ω_R sufficiently small has two important implications.

1. The general affine transformation fairly accurately approximates other interesting and plausible image transformations (e.g., perspective) that can be found in real world applications [Ros76a]. This means that one can compute ILR once, i.e., for the affine transformation, and then use the computed values in image mapping with some other, even more complex transformations.
2. The difficulty of mapping images with different pixel value distributions can be significantly mitigated when using ILR since the later maps a particular image onto itself (i.e., onto the same pixel value distribution) within a small area.

Here, the area ω_R is called “response area”. For convenience and without loss of generality, a square box $r \times r$ can be chosen as the response area, where r is called “response radius”. In the case of the general affine transformation, image response has to be computed for the vector V_u defined by nine parameters: the translations DX and DY along the x - and y -axes; the rotation θ in the xy -plane; the non-isotropic scaling factors SX and SY along the x - and y -axes; the shear SHX and SHY along the x - and y -axes; and the reflections RX and RY about the x - and y - axes. The shaded subarea in Figure 1 shows what part of the small response area near the point P will be changing during the unit transformation, in the case of translation, rotation, and scaling.

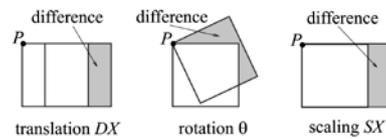


Figure 1. Computing local response at point P for translation, rotation, and scaling.

Computing Image local response according to Formula (1) with the chosen small values of ω_R and r is similar to computing Green’s functions extensively used in mathematical physics and engineering [Bar89a]. It can be easily shown that, as in the case of Green’s function, the response value R_p rapidly decreases as the distance from the point P (i.e., the value of r) increases.

```

foreach pixel P, do
  foreach component of  $V_u$ , do
    compute (1)
  endforeach component

  compute response  $R_p = \frac{\sum_{i=1}^N F_i}{N}$ 
endforeach pixel

```

Figure 2. Algorithm for computing Image local response.

The foregoing definition of the ILR suggests the algorithm shown in Figure 2. In the algorithm, the value of the difference F is computed for each of the N components of the vector V_u . In the case of the general affine transformation, $N = 9$. Finally, the response value R_p at the point P is computed as the averaged sum of all N differences F_i ($i = 1, \dots, N$). In

order to compute the response values for the border pixels, the image can be appropriately padded.

The values of image response can be represented in a graphical form - see Figure 3. As one can see, Image local response has a dual nature. On the one hand, ILR is defined in the form of a matrix computed from the pixel value distribution, as Formula 1 suggests. On the other hand, ILR represents the image feature in the form of the contours of the objects that are present in the image. The duality of ILR allows one to transfer the search for the proper image transformation A in image mapping problem from the actual image space I into the response space R . In this case, the difference between two images Img_1 and Img_2 can be evaluated as a squared difference of the response values over the area Ω of the overlap of the both images, in the following way:

$$F = \frac{\sum_{\Omega} (R_2(x', y') - R_1(x, y))^2}{\Omega^2}, \quad (2)$$

where $R_1(x, y)$ and $R_2(x', y')$ are the response values of the reference image Img_1 and the transformed image Img_2 , correspondingly.

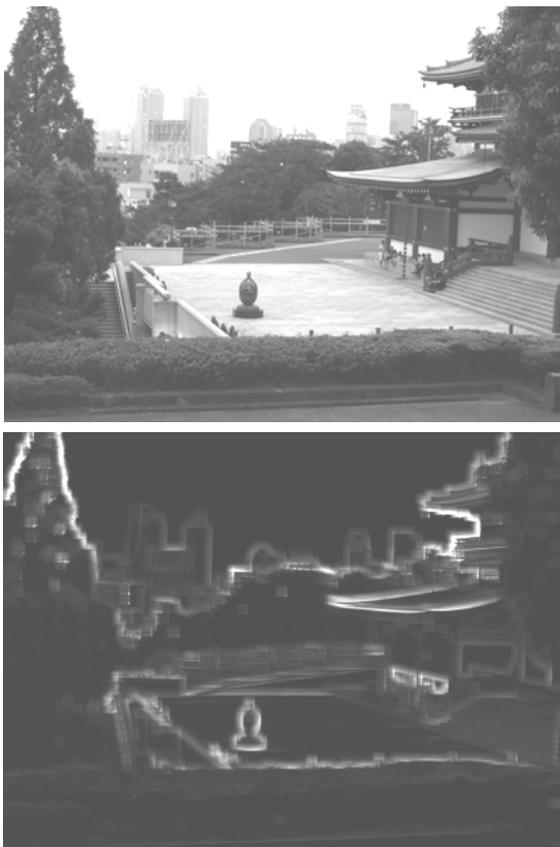


Figure 3. Original image of a scene (top) and its response representation (bottom).

The different types of imagery are shown in Figure 4 whereby a wireframe and a principal model of the same object expose different pixel values distributions. That makes the mutual mapping of the images with the direct comparison of their gray values impossible. On the other hand, the response images computed according to (1) and shown in Figure 5 exhibit clear definition of the common contours of the both objects, i.e., their main feature.



Figure 4. Original images of an object: the wireframe (left) and the principal (right) model.

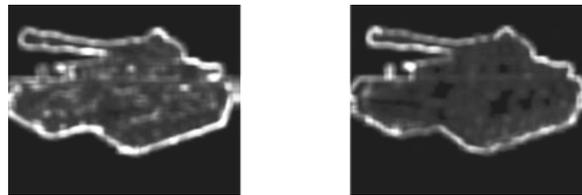


Figure 5. Image local response of the wireframe (left) and the principal (right) model.

Image local response has a few interesting and helpful properties that can be effectively used in computationally intensive image mapping problems.

1. As mentioned before, Image response preserves the main image feature, the contours of the objects in the scene.
2. Using the matrix of the response values significantly reduces the amount of information that has to be processed during the search for the proper transformation A . Only the higher response values would participate in the image mapping computations provided the sparse response matrices of the images are represented using efficient data structures.
3. The algorithm for computing ILR shown in Figure 2 can be easily parallelized, so all pixels comprising the image would be processed concurrently on a modern GPU, thus making the computational complexity of the algorithm equal to $O(1)$.
4. Image mapping can be formulated as an optimization problem whereby the image difference plays the role of the objective function that has to be minimized. In this case, ILR provides a smooth bell-shaped fitness landscape very well suited, e.g., for

the evolutionary search where the selection of the successful partial solutions drives the search towards the complete optimal solution [Ash06a].

5. In some cases, the model of Image local response can be effectively used to control local search when image mapping is formulated as an optimization problem. In particular, the value of the vector $\alpha = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ of the coefficients in the Downhill simplex method can be adjusted to the landscape of the objective function thus accelerating the search [Mas05a]. This particular property of ILR is based on the fact that in the close vicinity of the optimal solution, ILR fairly well approximates the objective function, i.e., the global difference between the images.

3. COMPUTATIONAL EXPERIMENTS WITH IMAGE MAPPING AND IMAGE LOCAL RESPONSE

The proposed image transformation in the form of Image local response was tested on a few image mapping problems [Mas08b]. A sample set of three 2D grayscale images is shown in Figure 6. The 300×300-pixel reference image Img_0 contains an object arbitrarily rotated in the 3D coordinate system. Two template images are a 178×195-pixel top view Img_1 and a 185×66-pixel left view Img_2 of the same object. The corresponding image responses computed in accordance with the algorithm given in section 2 are shown in Figure 7.

The search for the proper mapping from the template images onto the reference image is formulated here as an imaging optimization problem solved with a hybrid evolutionary algorithm [Mas08b]. The following conditions are present that complicate the problem:

- two or more template images are used to represent the different views of the same object;
- the object of the mapping undergoes significant distortion caused, e.g., by an arbitrary rotation in the 3D space; such a mapping cannot be defined with a single transformation vector;
- the difference between the images cannot be formulated as a single fitness function; consequently, the search has to deal with the multiple objectives of the optimization.

In accordance with the proposed approach, the search is conducted in the response space \mathbf{R} , as opposed to the actual image space \mathbf{I} . In order to accommodate the multiple template images, an advanced computational model is used. The model includes the multiple populations, so that every template is represented by its own independent population. Since the template objects can undergo significant

distortion, every template object is divided into k sections, so each section can have its own transformation vector \mathbf{V}_k . This approach corresponds to a piece-wise approximation of the actual image transformation $A(\mathbf{V})$.

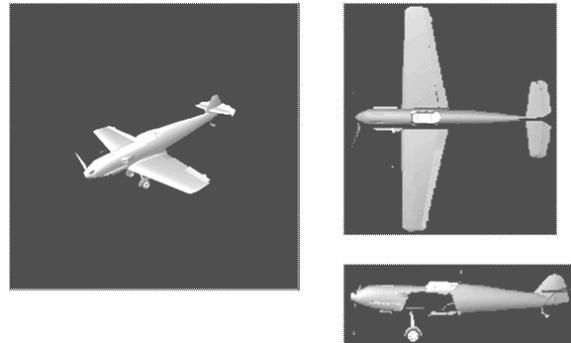


Figure 6. A sample set of three 2D grayscale images: reference image (left) and two template images (right).

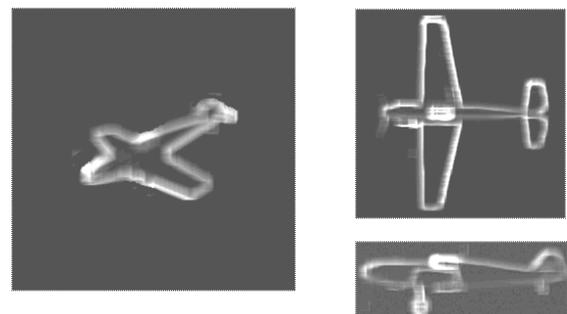


Figure 7. Response images of the sample test set.

The computational algorithm further assumes that every object in the image has some prominent basic feature in the form of a trunk to which all other parts of the object are attached. Here, such a feature is called a “hull”. The transformation of the hull can be defined by the main vector \mathbf{V}_A of the general affine transformation and a complementary vector \mathbf{V}_D of elastic deformations. The latter vector describes the deviation of the actual hull transformation from the main vector \mathbf{V}_A .

In its most general form, the entire algorithm works as two relatively independent phases implementing the global search and the local correction. The global search phase attempts to find the optimal solution for the hull transformation, i.e., the best mapping between the template hulls and the reference hull. The local corrections phase attempts to find the optimal piece-wise approximation of the actual image transformation using the hull transformation as its initial approximation. Because of the complex composite structure of the template model and a two-

phase search algorithm, one expression for fitness function is not sufficient. The search is conducted in the multi-objective space using the different expressions for the fitness function at the different stages of the algorithm.

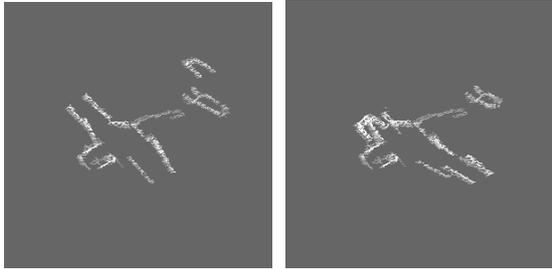


Figure 8. Intermediate results at the different stages of the piece-wise mapping.



Figure 9. Result of the piece-wise mapping of the template objects onto the reference image.

Figure 8 shows some intermediate stages of the piece-wise mapping of the different object sections onto the reference image. Figure 9 shows the final result of the image mapping. As one can see, the template images were successfully mapped onto the reference image using the piece-wise transformations of the original template objects in the response space.

Another interesting and important image mapping problem is medical image registration. Here, different slice images obtained with the CT or MR scan have to be put into the same framework by computing their mutual transformations. Figure 10 presents two sample MR images of different slices. The transformation has to be found that maps the

template image Img_1 (Figure 10, right) onto the reference image Img_0 (Figure 10, left).

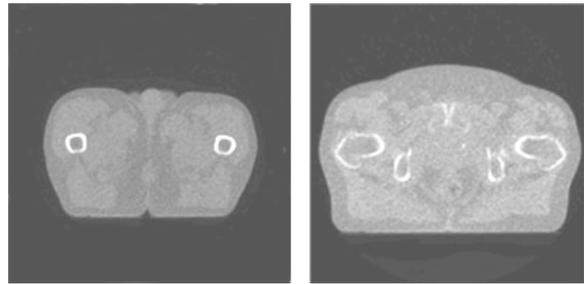


Figure 10. A sample set of MR images: reference (left) and template (right).

The search for the optimal mapping is conducted using the proposed approach, in the same manner as the search for the solution of the previous problem. Figure 11 shows the response matrices of the both images, and Figure 12 presents the final result of the mapping. As one can see, the algorithm was able to find a fairly good mapping of the template onto the reference image. Further improvement of the solution can be achieved with the usage of the adaptive division of images into sections. That would help remove certain roughness and discontinuities in the resulting image transformations.

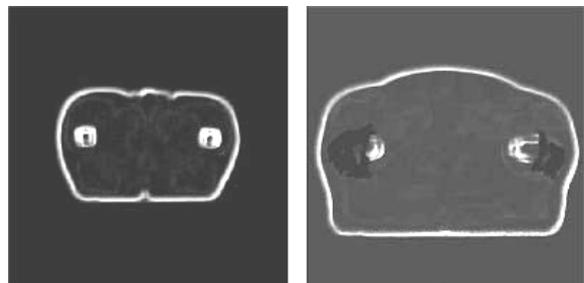


Figure 11. Responses of the MR images.

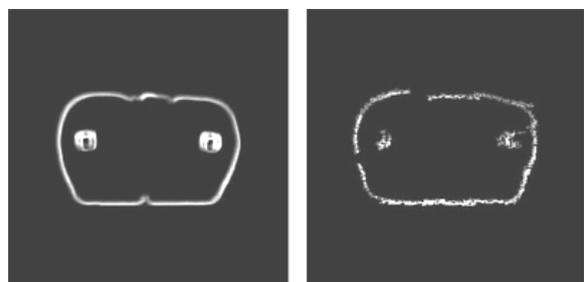


Figure 12. Result of the piece-wise mapping of the template (right) onto the reference image (left).

The results of the computational experiments presented in this section validate the proposed

approach in the form of Image local response and its applicability to solving complex image mapping problems.

4. CONCLUSION

The paper proposes a novel image transformation in the form of Image Local Response (ILR) that can be used for solving complex image mapping problems. The proposed transformation brings together two approaches based on the pixel value distribution and image features.

Image local response is defined as the average value of the difference between the transformed and the original copies of the same image. Here, the transformation is small, i.e., the components of the corresponding parameter vector have sufficiently small unit values.

The response has a few interesting properties useful in image mapping:

- it significantly reduces the amount of information that has to be processed during the search for the correct mapping parameters,
- it retains the main features of the object shape, its contour,
- the algorithm for computing response values is inherently parallel,
- response provides a bell-shaped fitness landscape very well suited for solving image mapping problem with the evolutionary search,
- the ILR model can be used to effectively control and accelerate the search for the proper mapping.

The validity of the proposed image transformation is shown on complex image mapping problems formulated as multi-objective piece-wise imaging optimization problem.

5. REFERENCES

[Ash06a] Ashlock, D. Evolutionary computation for modeling and optimization. Springer, 2006.

[Bar89a] Barton, G. Elements of Green's functions and propagation: Potentials, diffusion, and waves. Clarendon Press, Oxford, 1989.

[Bro98a] Brooks, R.R., and Iyengar, S.S. Multi-sensor fusion: Fundamentals and applications with software. Prentice Hall, New York, 1998.

[Ger02] Gere, J.M., and Timoshenko, S.P. Mechanics of materials. Nelson Thornes, 2002.

[Ima06a] Image processing toolbox user's guide. The MathWorks, 2006.

[Mas05a] Maslov, I.V., and Gertner, I.. Reducing the computational cost of local search in the hybrid evolutionary algorithm with application to electronic imaging. Engineering Optimization 37, No 1, pp. 103-119, 2005.

[Mas08b] Maslov, I.V. Improving the performance of Evolutionary algorithms in imaging optimization. Ph.D. Thesis, City University of New York, New York, 2008.

[Muc98a] Muchnik, I., and Mottl, V. Bellman functions on trees for segmentation, generalized smoothing, matching and multi-alignment in massive data sets. DIMACS Technical Report 98-15, February 1998.

[Pit00a] Pitas, I. Digital image processing algorithms and applications. John Wiley & Sons, 2000.

[Ros76a] Rosenfeld, A., and Kak, A.C. Digital picture processing. Academic Press, New York, 1976.

[Seu00a] Seul, M., O'Gorman, L., and Sammon, M.J. Practical algorithms for image analysis: Description, examples, and code. Cambridge University Press, Cambridge, UK, 2000.