The 15th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2007

in co-operation with

EUROGRAPHICS

W S C G ' 2007

University of West Bohemia
Plzen
Czech Republic

January 29 – February 1, 2007

Full Papers Proceedings

Co-Chairs

Jarek Rossignac, Georgia Tech, USA
Vaclav Skala, University of West Bohemia, Plzen, Czech Republic

Edited by
Jarek Rossignac, Vaclav Skala

WSCG'2007 Full Papers Proceedings

Editor-in-Chief: Vaclav Skala
University of West Bohemia, Univerzitni 8, Box 314
CZ 306 14 Plzen
Czech Republic
skala@kiv.zcu.cz
Managing Editor: Vaclav Skala

Author Service Department & Distribution:
Vaclav Skala - UNION Agency
Na Mazinách 9
322 00 Plzen
Czech Republic

Printed at the University of West Bohemia

Hardcopy: *ISBN 978-80-86943-98-5*

WSCG 2007

International Programme Committee

Alexa, Marc (Germany)
Bartz, Dirk (Germany)
Bekaert, Philippe (Belgium)
Benes, Bedrich (U.S.A.)
Bengtsson, Ewert (Sweden)
Bieri, Hanspeter (Switzerland)
Biri, Venceslas (France)
Bouatouch, Kadi (France)
Chen, Min (U.K.)
Chrysanthou, Yiorgos (Cyprus)
Coquillart, Sabine (France)
Davis, Larry (U.S.A.)
Deussen, Oliver (Germany)
du Buf, Hans (Portugal)
Ferguson, Stuart (U.K.)
Ferko, Andrej (Slovakia)
Goebel, Martin (Germany)
Groeller, Eduard (Austria)
Hauser, Helwig (Austria)
Hege, Hans-Christian (Germany)
Jansen, Frederik (The Netherlands)
Klosowski, James T. (U.S.A.)
Kobbelt, Leif (Germany)
Kruijff, Ernst (Austria)
Magnor, Marcus (Germany)
Midkiff, Sam (U.S.A.)
Myszkowski, Karol (Germany)
Pasko, Alexander (Japan)
Peroche, Bernard (France)
Puppo, Enrico (Italy)
Purgathofer, Werner (Austria)
Rauterberg, Matthias (The Netherlands)
Rokita, Przemyslaw (Poland)
Rossignac, Jarek (U.S.A.)
Rudomin, Isaac (Mexico)
Sbert, Mateu (Spain)
Schilling, Andreas (Germany)
Schumann, Heidrun (Germany)
Shamir, Ariel (Israel)
Skala, Vaclav (Czech Republic)
Sochor, Jiri (Czech Republic)
Teschner, Matthias (Germany)
Theoharis, Theoharis (Greece)
Veltkamp, Remco (The Netherlands)
Weiskopf, Daniel (Canada)
Wu, Shin-Ting (Brazil)
Wyvill, Brian (Canada)
Zemcik, Pavel (Czech Republic)

WSCG 2007 Board of Reviewers

Adzhiev,Valery (U.K.)
Alessandro, Piva (Italy)
Ammon, Lorenz (Switzerland)
Ancuti, Cosmin (Belgium)
Andreadis, Ioannis (Greece)
Andujar, Carlos (Spain)
Aspragathos, Nikos (Greece)
Aveneau, Lilian (France)
Bamidis, Panagiotis (Greece)
Bargouli, Maria (Greece)
Bartz, Dirk (Germany)
Battiato, Sebastiano (Italy)
Bekaert, Philippe (Belgium)
Bellon, Olga (Brazil)
Benes, Bedrich (USA)
Bengtsson, Ewert (Sweden)
Beyer, Johanna (Austria)
Bieri, Hanspeter (Switzerland)
Bilbao, Javier J. (Spain)
Biri, Venceslas (France)
Bottino, Andrea (Italy)
Bouatouch, Kadi (France)
Bourdin, Jean-Jacques (France)
Brunet, Pere (Spain)
Bruni, Vittoria (Italy)
Brunnet, Guido (Germany)
Buehler, Katja (Austria)
Butz, Andreas (Germany)
Callieri, Marco (Italy)
Clasen, Malte (Germany)
Coleman, Sonya (U.K.)
Coquillart, Sabine (France)
Crosnier, Andre (France)
Daniel, Marc (France)
Davis, Larry (USA)
de Geus, Klaus (Brazil)
Deussen, Oliver (Germany)
Dingliana, John (Ireland)
du Buf, Hans (Portugal)
Duce, David (U.K.)
Durikovic, Roman (Slovakia)
Egges, Arjan (The The Netherlands)
Eisemann, Martin (Germany)
Erbacher, Robert (USA)
Faudot, Dominique (France)
Feito, Francisco (Spain)
Felkel, Petr (Czech Republic)
Ferguson, Stuart (U.K.)
Ferko, Andrej (Slovakia)
Fernandes, Antonio (Portugal)
Fischer, Jan (Germany)
Flaquer, Juan (Spain)
Galin, Eric (France)
Galo, Mauricio (Brazil)
Ganovelli, Fabio (Italy)
Garcia-Alonso, Alejandro (Spain)
Gautron, Pascal (France)
Giannini, Franca (Italy)
Goebel, Martin (Germany)
Gonzalez, Pascual (Spain)
Grammalidis, Nicolaos (Greece)
Gresh, Donna (USA)
Groeller, Eduard (Austria)
Gutierrez, Diego (Spain)
Habbecke, Martin (Germany)
Haber, Tom (Belgium)
Hadwiger, Markus (Austria)
Harding, Bruce (USA)
Haro, Antonio (USA)
Hast, Anders (Sweden)
Hauser, Helwig (Austria)
Havran, Vlastimil (Czech Republic)
He, Xiangjian (Australia)
Hege, Hans-Christian (Germany)
Horain, Patrick (France)
Hornung, Alexander (Germany)
House, Donald (USA)
Chen, Min (U.K.)
Chin, Seongah (Korea)
Chover, Miguel (Spain)
Chrysanthou, Yiorgos (Cyprus)
Iglesias, Andres (Spain)
Ihrke, Ivo (Germany)
Jaillet, Fabrice (France)
Jansen, Frederik (The The Netherlands)
Joan-Arinyo, Robert (Spain)
Kähler, Ralf (Germany)
Kalantari, Leila (Canada)
Karabassi, Evaggelia-Aggeliki (Greece)
Kersten, Marta (Germany)
Kinuwaki, Shinichi (Germany)
Kjelldahl, Lars (Sweden)
Klajnsek, Gregor (Slovenia)
Klosowski, James T. (USA)
Knight, Michael (U.K.)
Kobbelt, Leif (Germany)
Kohlmann, Peter (Austria)
Kolcun, Alexej (Czech Republic)
Kruijff, Ernst (Austria)
Lamecker, Hans (Germany)
Lanquetin, Sandrine (France)
Larboulette, Caroline (Austria)
Lee, Seungyong (Korea)
Leitao, Miguel (Portugal)
Lensch, Heindrik (Germany)
Lewis, J.P. (USA)
Linz, Christian (Germany)
Magillo, Paola (Italy)
Magnor, Marcus (Germany)
Malik, Muddassir (Austria)
Mandl, Thomas (Germany)
Mantler, Stephan (Austria)
Matey, Luis (Spain)
Mattausch, Oliver (Austria)
McMenemy, Karen (U.K.)
Megali, Giuseppe (Italy)
Meneveaux, Daniel (France)
Moccozet, Laurent (Switzerland)
Mokhtari, Marielle (Canada)
Mollá Vayá, Ramón (Spain)
Montrucchio, Bartolomeo (Italy)
Mould, David (Canada)
Mudur, Sudhir (Canada)
Muller, Heinrich (Germany)
Muntean, Adrian (Germany)
Myszkowski, Karol (Germany)

Nedel, Luciana P. (Brazil)
Neveu, Marc (France)
Nielsen, Frank (Japan)
Ogayar, Carlos,J. (Spain)
Oliveira, Manuel (U.K.)
Pasko, Alexander (Japan)
Patel, Daniel (Austria)
Patow, Gustavo (Spain)
Paulin, Mathias (France)
Pavic, Darko (Germany)
Pedrini, Helio (Brazil)
Peroche, Bernard (France)
Pettifer, Steve (U.K.)
Platis, Nikos (Greece)
Plemenos, Dimitri (France)
Podorelec, David (Slovenia)
Ponchio, Federico (Italy)
Popov, Stefan (Germany)
Prakash, Edmond (U.K.)
Pratikakis, Ioannis (Greece)
Přikryl, Jan (Czech Republic)
Puppo, Enrico (Italy)
Purgathofer, Werner (Austria)
Rauterberg, Matthias (The The Netherlands)
Redon, Stephane (France)
Remolar, Inmaculada (Spain)
Renaud, Christophe (France)
Revelles, Jorge (Spain)
Ribelles, Jose (Spain)
Robert, Philippe (Switzerland)
Rodeiro, Javier (Spain)
Rodrigues, Marcos (U.K.)
Rojas-Sola, José Ignacio (Spain)
Rokita, Przemyslaw (Poland)
Rossignac, Jarek (USA)
Roth, Peter,M. (Austria)
Rudomin, Isaac (Mexico)
Sanna, Andrea (Italy)
Sbert, Mateu (Spain)
Scateni, Riccardo (Italy)
Segura, Rafael (Spain)
Sellent, Anita (Germany)
Semwal, Sudhanshu (USA)
Shamir, Ariel (Israel)
Scheiblauer, Claus (Austria)
Schilling, Andreas (Germany)
Schmidt, Thomas (Germany)
Schneider, Bengt-Olaf (USA)
Schumann, Heidrun (Germany)
Sips, Mike (USA)
Sirakov, Nikolay Metodiev (USA)
Sochor, Jiri (Czech Republic)
Solis, Ana Luisa (Mexico)
Sousa, A.Augusto (Portugal)
Stroud, Ian (Switzerland)
Suarez Rivero, Jose (Spain)
Suescun, Angel (Spain)
Szekely, Gabor (Switzerland)
Štulić, Radovan (Serbia and Montenegro)
Tang, Wen (U.K.)
Tawara, Takehiro (Japan)
Tecchia, Franco (Italy)
Teschner, Matthias (Germany)
Theußl, Thomas (Austria)
Tobler, Robert (Austria)
Tonet, Oliver (Italy)
Turini, Giuseppe (Italy)
Tytkowski, Krzysztof (Poland)
Vanecek, Petr (Czech Republic)
Vasa, Libor (Czech Republic)
Veiga, Luis (Portugal)
Veltkamp, Remco (The The Netherlands)
Vergeest, Joris (The The Netherlands)
Vitulano, Domenico (Italy)
Wan, Taoruan (U.K.)
Weigel, Christian (Germany)
Weiskopf, Daniel (Canada)
Wenger, Thomas (Switzerland)
Wu, Shin-Ting (Brazil)
Wyvill, Brian (Canada)
Zach, Christopher (Austria)
Zachmann, Gabriel (Germany)
Zalik, Borut (Slovenia)
Zemcik, Pavel (Czech Republic)
Zhu, Ying (USA)
Zimeras, Stelios (Greece)

Contents

Please, find the ”journal” papers in the Journal of WSCG

Paper Code		Title	Page
		Session A **Chair:Vergeest,J.S.M.**	
A43	A full	Alsweis,M.: Extended Competition Rules for Interacting Plants (Germany)	1
D67	A full	Chourasia,A., Schulze,J.: Data Centric High Dynamic Range Transfer Functions for Volume Rendering (United States)	9
F19	A full	Kuss,A., Hege,H.-C., Krofczik,S., Borner,J.: Pipeline for the Creation of Surface-based Averaged Brain Atlases (Germany)	17
G31	A journal	Lee,J.K., Newman,T.S.: New Method for Opacity Correction in Oversampled Volume Ray Casting (United States)	1
H31	A full	Desroche.S.,Jolivet.V.,Plemenos.D.: Towards Plan-based Automatic Exploration of Virtual Worlds (France)	25
		Session B **Chair: Alexa,M.**	
B03	B journal	Holst,M., Schumann,H.: Normal Mapping for Surfel-Based Rendering (Germany)	9
B67	B full	Turchyn,P.: Memory-efficient Sliding Window Progressive Meshes (Finland) Presentation	33
C97	B full	Tejada,E., Schafhitzel,T., Ertl,T.: Hardware-accelerated Rendering of Point-based Surfaces and Volumes (Germany)	44
G03	B full	Heymann,S., Müller,K., Smolic,A., Fröhlich,B., Wiegand,T.: SIFT Implementation and Optimization for General-Purpose GPU (Germany)	317
G19	B journal	Kim,S.-S., Nam,S.-W., Kim,D.-H., Lee,I.-H.: Hardware-Accelerated Ray-Triangle Intersection Testing for High-Performance Collision Detection (Korea)	17
		Session C **Chair: Smolic,A.**	
B13	C full	Tomaszewska,A., Mantiuk,R.: Image Registration for Multi-exposure High Dynamic Range Image Acquisition (Poland)	49
C23	C full	Vucini,E., Gokmen,M., Groller, E.: Face Recognition under Varying Illumination (Austria)	57
C79	C full	Ivanovska,T., Linsen,L.: Converting RGB Volume Data to Scalar Fields for Segmentation Purposes (Germany)	65
G23	C full	Steinmetz,S., Pellenz,J., Paulus,D.: Estimation of Planar Surfaces in Noisy Range Images for the RoboCup Rescue Competition (Germany)	73
D73	C full	Schwartz,W.R., Pedrini,H.: Color Textured Image Segmentation Based on Spatial Dependence Using 3D Co-occurrence Matrices and Markov Random Fields (Brazil)	81

Session D **Chair: Kolingerova,I.**

D11	D journal	Hutter,M., Fuhrmann,A.: Optimized Continuous Collision Detection for Deformable Triangle Meshes (Germany)	25
E83	D full	Juckel,T., Beckhaus,S.: Rendering Diffuse Objects using Particle Systems inside Voxelized Surface Geometry (Germany)	89
F07	D journal	Spillmann,J., Becker,J.M., Teschner,M.: Non-iterative Computation of Contact Forces for Deformable Objects (Germany)	33
F17	D full	Carvalho,S.R., Boulic,R.,Thalmann,D.: Motion Pattern Preserving IK Operating in the Motion Principal Coefficients Space (Switzerland)	97
G97	D full	De Martino,J.M., Violaro,F.: Benchmarking Speech Synchronized Facial Animation Based on Context-Dependent Visemes (Brazil)	105

Session E **Chair:Bottino,A.**

B59	E full	Sanjurjo,J.R., Amor,M., Bóo,M., Doallo,R.: Improving Locality for Progressive Radiosity Algorithm: A Study based on the Blocking Transformation of the Scene. (Spain)	113
B71	E journal	Vivanloc,V., Hoelt,J.Ch., Hong,C.-B., Paulin,M.: Light Octree: Global Illumination Fast Reconstruction and Realtime Navigation (France)	41
C89	E journal	Linsen,L., Mueller,K., Rosenthal,P.: Splat-based Ray Tracing of Point Clouds (Germany)	51
E79	E full	Umenhoffer,T., Szirmay-Kalos,L.: Robust Diffuse Final Gathering on the GPU (Hungary)	121
G13	E full	Cadik,M.: Perception Motivated Hybrid Approach to Tone Mapping (Czech Republic)	129

Session F **Chair: Vanecek,P.**

D41	F full	Wang,H., Vergeest,J.S.M, Song,Y., Wiegers,T.: Automated 3D Scan Multi-view Registration Based on Rotation Estimation (Netherlands)	137
E97	F full	Langerak,T.R., Vergeest,J.S.M., Wang,H., Song,Y.: An Evolutionary Strategy for Free Form Feature Identification in CAD Models (Netherlands)	145
F31	F full	Domonkos,B., Csébfalvi,B.: Interactive Distributed Translucent Volume Rendering (Hungary)	153
H53	F full	Musialski,P., Tobler,R.F., Maierhofer,S.: Smooth Subdivision Surfaces over Multiple Meshes (Germany)	161
E43	F full	Gupta,R.B., Lee,B.-G., Lee,J.-J.: A new Image Interpolation Technique Using Exponential B-spline (Korea)	169

Session G **Chair: Kohout,J.**

A89	G full	Seoane,A., Taibo,J., Hernández,L., López,R., Jaspe,A.: Hardware-Independent Clipmapping (Spain)	177
C11	G full	Dimitrov,D., Knauer,Ch., Kriegel,K., Rote,G.: Upper and Lower Bounds on the Quality of the PCA Bounding Boxes (Germany)	185
G11	G full	Fairén,M., Trueba,R.: Octree-based View-dependent Triangle Meshes (Spain)	193
C43	G full	Livny,Y., Kogan,Z., El-Sana,J.: Seamless Patches for GPU-Based Terrain Rendering (Israel)	201
G07	G full	Pina,A., Bentes,C., Farias,R.: Memory Efficient and Robust Software Implementations of the Raycast Algorithm (Brazil)	209

Session H **Chair: Kolingerova,I.**

B17	H journal	Scheider,M., Klein,R.: Efficient and Accurate Rendering of Vector Data on Virtual Landscapes (Germany)	59
B29	H journal	Bade,R., Konrad,O., Preim,B.: Reducing Artifacts in Surface Meshes Extracted from Binary Volumes (Germany)	67
C71	H journal	Shi,K., Theisel,H., Weinkauf,T., Hauser, H., Hege,H.-C., Seidel,H.-P.: Extracting Separation Surfaces of Path Line Oriented Topology in Periodic 2D Time-Dependent Vector Fields (Germany)	75
E37	H journal	Kohlmann,P.,Bruckner,S.,Kanitsar,A.,Gröller,E.: Evaluation of a Bricked Volume Layout for a Medical Workstation based on Java (Austria)	83
E61	H journal	Schafhitzel,T., Falk,M., Ertl,T.: Real-Time Rendering of Planets with Atmospheres (Germany)	91

Session K **Chair: Vasa,L.**

C31	K journal	Amjoun,R.: Efficient Compression of 3D Dynamic Mesh Sequences (Germany)	99
C67	K full	Rudnick,S., Linsen,L., McPherson,E.G.: Inverse Modeling and Animation of Growing Single Stemmed Trees (Germany)	217
D03	K full	Knight,J.K., Semwal,S.K.: Fast Skeleton Estimation from Motion Captured Data using Generalized Delogne-Kasa method (United States)	225
E05	K full	Nakajima,H., Sugisaki,E., Morishima,S.: Tweakable Shadows for Cartoon Animation (Japan)	233
E23	K full	Iglesias,R., Prada,E., Uribe,A., Garcia-Alonso,A., Casado,S., Gutierez,T.: Assembly simulation on collaborative haptic virtual environments (Spain)	241

Session L **Chair: Rossignac,J.**

A19	L full	Castelló,P., Sbert,M., Chover,M., Feixas,M.: Viewpoint Entropy-Driven Simplification (Spain)	249
A73	L full	Porcu,M., Scateni,R.: Dimensional Induced Clustering for Mesh Recognition (Italy)	257
F53	L journal	Medek,P., Benes,P., Sochor,J.: Computation of Tunnels in Protein Molecules using Delaunay Triangulation (Czech Republic)	107
F71	L full	Trenkel,S., Weller,R., Zachmann,G.: A Benchmarking Suite for Static Collision Detection Algorithms (Germany)	265
G29	L full	Levet,F., Granier,X., Schlick,Ch.: Triangulation of Uniform Particle Systems: Its Application to the Texturation of Implicit Surfaces (France)	271
H07	L full	Sun,S.,Huang,D.W.: Efficient Region-Based Pencil Drawing (China)	279

Session M **Chair: Zemcik,P.**

A41	M full	Darles,E., Crespin,B., Ghazanfarpour,D.: Accelerating and Enhancing Rendering of Realistic Ocean Scenes (France)	287
A53	M full	Kuehl,B., Blom,K., Beckhaus,S.: Generation of Shadows in Scene Graphs based VR (Germany)	295
B07	M journal	Götzelmann,T., Götze,M., Ali,K., Hartmann,K., Strothotte,Th.: Annotating Images through Adaptation: An Integrated Text Authoring and Illustration Framework (Germany)	115
G67	M journal	Habel,R., Wimmer,M., Jeschke,S.: Instant Animated Grass (Austria)	123
A83	M full	Hasic,J., Chalmers,A.: Visual Attention for Significantly Influencing the Perception of Virtual (Bosnia and Herzegovina)	303
A31	M full	Li,Y., Lizhuang,M., Di,W.: Fast Colorization Using Edge and Gradient Constrains (China)	309

Session N **Chair: Hast,A.**

F47	N journal	Lourakis,M., Argyros,A.: Refining Single View Calibration With the Aid of Metric Scene Properties (Greece)	129
G59	N journal	Bottino,A., De Simone,M., Laurentini,A.: Recognizing Human Motion using Eigensequences (Italy)	135
H61	N journal	Ancuti,C., Bekaert,P.: A Scale Invariant Detector Based on Local Energy Model for Matching Images (Belgium)	143
A47	N journal	Chu,Ch., Shih,Z.-Ch.: Painterly Rendering Framework from Composition (Taiwan)	151

FOREWORD

This book contains the proceedings of WSCG'2007, the 15th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2007. The Winter School of Computer Graphics started in 1992 with about 20 participants from the former Czechoslovakia. It has become an important conference attended by the international Computer Graphics community.

Submissions to WSCG'2007 came from numerous countries on different continents. From these submissions, 186 papers have been sent to reviewers. Each paper was reviewed by at least three experts selected among the members of the large International Program Committee. 61 papers have been accepted to be presented at WSCG'07, representing a 33% acceptance ratio. The best 20 of these papers have been selected to be published in a special issue of the Journal of WSCG and will be available through the conference Digital Library at **http://wscg.zcu.cz/DL/wscg_DL.htm**.

Furthermore, the published papers are listed on the major international scientific indexes such as Thomson ISI.

WSCG'2007 features three key-note lectures:

- Shape Comparison, Simplification, and Compression by J. Rossignac from Georgia Tech, USA
- Intuitive Editing of Surface Meshes by M. Alexa from T.U. Berlin, Germany
- 3D Video and Free Viewpoint Video – Technologies, Applications and MPEG Standards, by A. Smolic, from the Heinrich-Hertz-Institut, Germany

Finally, we would like to express our thanks to all the authors who submitted technical papers to WSCG'2007, and to the reviewers whose work and dedication made it possible to put together a very exciting program of high technical quality.

The Program Co-Chairs

Jarek Rossignac, Georgia Institute of Technology, Atlanta, USA

Vaclav Skala, University of West Bohemia, Plzen, Czech Republic, Plzen

Extended Competition Rules for Interacting Plants

Monssef Alsweis
University of Konstanz
Department of Computer and Information Science
78457 Konstanz, Germany

Alsweis@inf.uni-konstanz.de

ABSTRACT

The goal of this paper is the interactive and realistic rendering of 3D competing plant covering a landscape. We introduce the so-called FON-Model as a new method for the visualization and simulation of competing plant root systems in which the overlapping area of the depleted-zone between competing root systems is quantatively found. We visualize different conditions for the resource competition between plant root systems. Additionally, we present an improved model for the competition for light that is dependant upon the translucency of the plant foliage and the density of the plant skeleton. In connection to it we introduce a new competition simulations for nurse plants. Finally, we simulate and visualize competition between a healthy plant and a plant that at one point in time becomes sick. As results, our method greatly increases the reality of the rendered landscape and allows rendering of large landscapes in interactive environments.

1 INTRODUCTION

Modeling and visualization of natural phenomena has taken an important position in the world of computer graphics. Plant modeling and ecosystem simulation is considered a main field in the area of rendering of natural phenomena. In our approach we investigated competition between different plant species in the under ground and above ground zone. The results of our findings are used to simulate and visualize ecosystems more realistically.

This paper is an extension of our previous work in [1] and [2]. In [1] a new method for the simulation and visualization of the development of plant populations is described in that the competition of natural resources between plants is simulated in two major areas: symmetric and asymmetric competition. The symmetric competition represents the root competition for resources, and the asymmetric competition represents the trunk, branch, and leaf competition for light. In [2] we presented an extension to [1], which illustrates the visualization of competition based on nine conditions of interaction between plants.

Our approach addresses four major areas of ecosystem visualizations. In the first part, we develop a new method to compute the root competition between neighboring plants by producing different types of root systems, which are utilized in the geometric simulation model of the root architecture.

In the second part of our approach, we simulate the competition for the light that is caused by the tree geometry and foliage characteristics. Here the competition is not always considered as an asymmetric competition. Since, the competition for light degrades slowly during the symmetric and asymmetric interval.

Some positive neighbor effects are protecting neighbors from excessive solar radiation and resultant water loss and providing mechanical support and protection from herbivores [3]. The classic example of positive neighbor effects is that of "‘nurse plants"’ in arid systems [18]. In this main part we will simulate and will visualize the competition between the nurse plants and other species of plants.

In the last part of our approach, a new type of competition is simulated and visualized. This type of competition takes place between a healthy plant and a plant that becomes sick at a certain time during the simulation process. To render a large scene of different types of plants in real-time, we use the Wang Tile method [11] for the visualization of the 3D plant architecture.

2 PREVIOUS WORKS

A number of approaches and procedures were already applied and implemented to simulate and visualize large plant ecosystems. The most prominent are classified in three major categories: modelling a single plant, composing and modeling a large plant population, and visualizing a scene in real-time.

Modelling of a Single Plant: In 1968, Aristid Lindenmayer proposed a method for the simulation of the development of multi-cell organisms, the so-called L-System [20]. Przemyslaw Prusinkiewicz advanced 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.
Copyright UNION Agency – Science Press, Czech Republic

 ISBN 978-80-86943-98-5

idea of L-Systems and made considerable progress in modeling, simulation and visualization of the development of plants. Other approaches followed [22, 21] and in [14] Deussen und Lintermann improved the modelling process of a single plant with their X-frog software. In order to simulate a slowly dying plant, we used the X-frog software with different leaf textures.

Modelling Large Complex Plant Populations: Simulation and visualization of a natural scene with thousands of plants was introduced to computergraphics in [13], where also the terrain was modelled, and to give the scene a realistic appearance different types of single plants were distributed according to the nature of their environment. In [19] a multilevel model was used to simulate and visualize a natural scene. In the present work Wang-Tiles were used [11] that allow to render even quite complex scenes efficiently.

Efficient Rendering: To render a complex natural scene in real-time, in [11] a non-periodic tiling method was applied. The plants are distributed to each tile with a 2D Poisson distribution. In [12] a new method was proposed to cover an area with realistic appearing forests for real-time rendering. Here, dense forests were simulated that corresponded to continuous non-repetitive areas, which were filled with thousands of trees. Behrendt et al. in [5] introduced techniques for the realistic real-time rendering of complex landscapes that consist of highly detailed plant models. Benes et al. [6, 8, 7]

Light interaction: There have been several methods for computing the light interaction among plant, cf. [16] however, for an efficient simulation of larger areas a simpler model is needed. We restrict ourselves to such a model that uses an approximation of the plant shape to determine the received light.

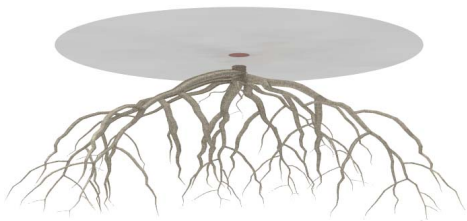


Figure 1: The FON Model for the under ground area.

3 THE FON MODEL

To represent a plant in our work, we applied the FON (Field-of-Neighborhood) Model, which was defined by Berger [9]. In the FON model, each plant has a circular

Figure 2: The FON Model. The zone of influence (R_{FON}) depends on the diameter of the trunk (R_{basal}).

zone of influence (ZOI). The radius of the zone determines the distance, in which the neighboring plants influence each other. Each plant is represented by its position, size, and age. Additionally, each plant has two zones of influence, see Figs. 1 and 2. The first zone represents the under ground competition (root competition), and the second represents the above ground competition (the competition for light.) When computing the two zones, the amount of resources for the under ground and the amount of light for the above ground are taken into consideration. To determine the zone of influence, we use a non-linear function of the basal radius [9, 2].

$$R_{FON} = a\,(R_{basal})^b \tag{1}$$

R_{FON} is the radius of the influence zone of the single plant. R_{basal} is the radius of the single plant, see Fig. 2. a and b are constants and depend on the intensity of the resources in the under ground and on the intensity of the light in the above ground. If the field in which the plants are cultivated is fertile, a and b will be small and the under ground zone and competition will be small as well [4]. In the other case, a and b will be large and consequently the under ground zone will be large as well, which will cause the competition among ressources.

4 MODEL AND METHOD

The growth of a single plant depends on its effective size, maximum size, and maximum growth rate. The possible growth rates of the single plant are denoted by GR as done in [4, 2]. They are defined as follows:

 ISBN 978-80-86943-98-5

$$GR = \frac{MGR.R_{Basal}}{R_{Max}}(1 - \frac{R_{basal}}{R_{Max}}) \quad (2)$$

MGR is the maximum growth rate of the plant. R_{Max} is the maximum size of the plant. R_{basal} is the effective radius of the trunk see Fig.2. GR is the growth rate of the plant without considering the competition with its neighbor. In order to define the growth rate $\triangle R_{Basal}$ for different competition systems, and to show the results we apply a method presented in [4]:

$$\triangle R_{Basal} = GR.C \quad (3)$$

Whereas C equals the competition factor. It is determined as follows:

$$C = \begin{cases} 1 - 2\varphi_i & : \quad \varphi_i <= 0.5 \\ 0 & : \quad \varphi_i > 0.5 \end{cases} \quad (4)$$

Whereas φ_i is the competition factor in interval $[0,1]$. It can be determined for different systems of competition as described in the following paragraphs.

4.1 Competition for Light

The competition between neighbouring plants for light occurs, if the plant shoots are overlapping. This overlapping causes a deficiency for the light that is needed by the small plants. In Figs. 3 and 4 we see that the small plant under the tree in Fig. 3 receives only a part of the light (symmetric competition for light). In contrast, the small plants under the tree in Fig. 4 do not receive any part of the light (asymmetric competition for light).

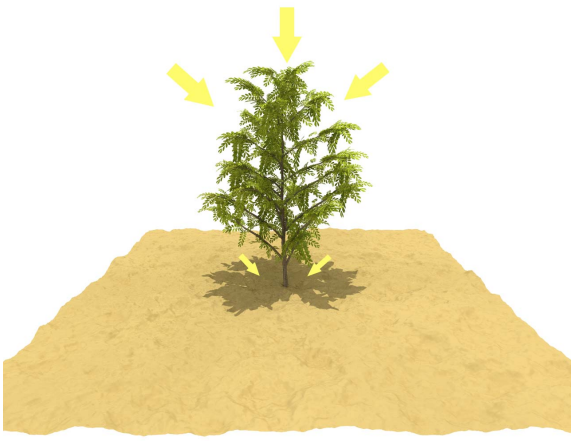


Figure 3: A non-dense tree lets the small plants receive light, which is passing through the foliage

Consequently, the competition occurs either symmetric or asymmetric or it is a mixture of both. This depends on the mean factor L_{fac} of the large plant. The

Figure 4: A dense tree creates a dark shadow, therefore the small plants do not receive light.

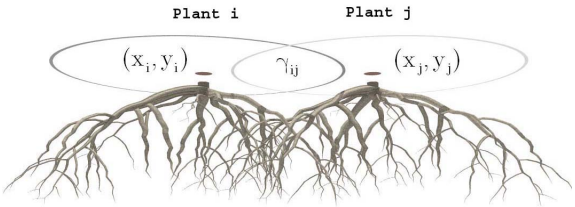


Figure 5: The geometry of the spatial competition between the individual plants i and j with position (x_i, y_i) and (x_j, y_j) is a function of the overlapping area γ_{ij}.

competition for the light α_i [9, 1] is determined by accounting for the density of the overlapping $IN(i,j)$ between the plants i and j as follows [2]:

$$\alpha_i = \frac{\sum_{j=1}^{n} \frac{3\beta_{ij}}{IN(i,j)}}{A_i} \quad (5)$$

Whereas A_i is the size of the plant i in Fig.5. n is the number of plants that overlap with the i th plant. β_{ij} is determined as follows:

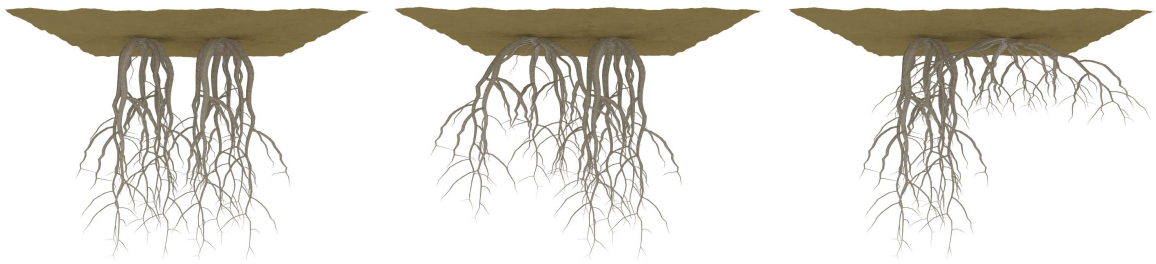
$$\beta_{ij} = (5 + L_{fac})\frac{\gamma_{ij}}{10} \quad (6)$$

Whereas γ_{ij} is the size of the overlapping area between plant i and plant j (see Fig. 5). $L_{fac} \in [0,5]$ is the translucency factor. If $L_{fac} = 0$, the competition for light is considered symmetric. If $L_{fac} = 5$, the competition is considered asymmetric.

4.2 Competition for Resources

The competition for resources depends on different factors such as the root density, the extend of the root area, and the plasticity either during the root growth or in the characteristics of the enzymes [10]. In order to render

 ISBN 978-80-86943-98-5



(a) (b) (c)

Figure 6: Three different root competition systems. In (a) the competition is between deep and deep root systems. In (b) the competition is between non-deep and deep root systems. In (c) the competition is between deep and flat root systems.

and visualize this competition, the root density and the root area are computed. The competition between deep root systems and flat root systems is only half of the competition with other root systems [15] [17].

For the rendering and visualization of the competition for the resources, the competition is considered as the symmetric competition. In [23] it is limited to nine conditions, see Tab.1. Fig.6 illustrates the three conditions of this table (deep with deep, deep with medium deep and deep with flat roots).

	F	C	T
F	FF	FC	FT
C	CF	CC	CT
T	TF	TC	TT

Table 1: This table shows the nine conditions that presents the competition of three different root systems. Whereas F, C, and T present flat, non-deep, and deep root systems.

The competition for the resource σ_i with consideration to the root size and the root density factor T_f is defined in the following equation [1, 2]:

$$\sigma_i = \frac{\gamma_{ij}}{2.A_i}(1+\frac{T_f}{10}) \tag{7}$$

Whereas $T_f \in [0,10]$ is the root size and root density factor. Here the symmetric competition is an interval of $[\frac{\gamma_{ij}}{2.A_i}\ \frac{\gamma_{ij}}{A_i}]$.

4.3 Nurse Plants

Some desert plant can only establish themselves in close proximity to a larger plant, usually a shrub, because the shade of the larger plant provides protection from the intense solar radiation and resultant heat and transpiration that a seedling otherwise would experience. Therefore the small plants developed their system, in order to position their seeds under the shrub. Plant establishment in deserts is largely determined by negative effects of a superabundant plant resource "‘solar radiation"’ for which plants in other environments compete (see the Fig.10).

4.4 The Competition Between Healthy and Sick Plants

In order to simulate and visualize the competition between a healthy and a sick plant or other plants that suddenly started deteriorating at a certain time during the competition process, the competition factors for both healthy and sick plants are taken into account. The influence of the sick plant onto the healthy plant lessens gradually. The competition of resources between sick and healthy plants is ascertained as follows:

$$\sigma_i = \frac{\gamma_{ij}}{2A_i} + \gamma_{ij}T_r \tag{8}$$

Whereas $T_r = T_k + T_g, T_r \in [0..1]$ combines sickness factors and health factors. T_k gradually diminishes with time; consequently, the healthy plants grow without competition. T_g does not change over time.

4.5 General Competition

The general competition between plants is a weighted combination of symmetric and asymmetric competition:

$$\varphi_i = p\alpha_i + (1-p)\sigma_i \tag{9}$$

Whereas φ_i is the general competition of the i plant from its neighbor. $p \in [0\ 1]$ is the part of the competition for the resources in the general competition.

5 RESULTS

Fig. 7 illustrates the competition for light between non-dense trees and different small plants. It is shown that the small plants can grow under that tree, because they receive a part of the light. In Fig. 8 we visualize the competition for light between a dense tree and different small plants. In this picture we see that the small plants cannot grow, since they do not receive sufficient light.

 ISBN 978-80-86943-98-5

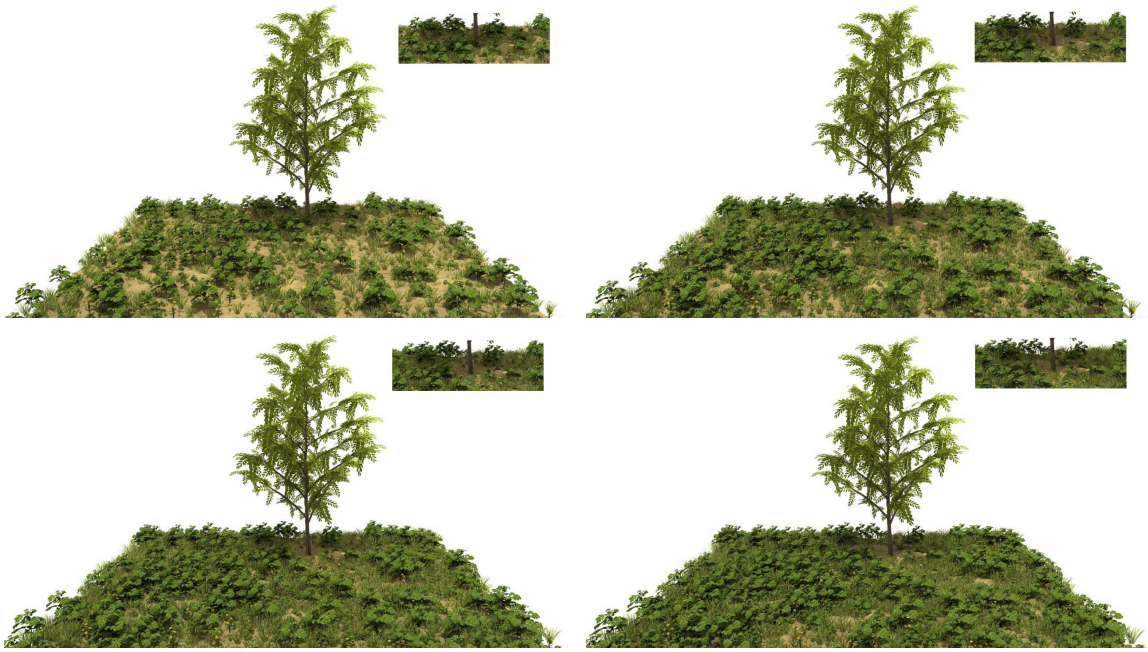


Figure 7: Four stages illustrate the competition for light between non-dense trees and different small plants.

In Fig. 9 the competition for resources between dense tree roots and flat tree roots is illustrated. We note that the flat roots grow slow as a cause of the overlapping with the deep roots, since the deep roots are stronger than the flat roots.

In Fig. 10 the competition for the nutrients is simulated under the nurse plants. Here the Nurse plant has positives influence on the small plants. The small plants affect negativ on others. Therefore the equation 9 is modified slightly by now subtracting $(1-p)\sigma_i$.

In Figure 11 we show that the sicker plant allows the other plant to grow quicker, because the competition degrades with time.

Applying our method, we produced natural scenes, in which the competition between different plants and resources is simulated. We have illustrated these scenes in the accompanying video in order to visualize the implementation of our system in the following steps:

- The competition for light between a non-dense tree and different small plants.
- The competition for light between a dense tree and different small plants.
- The competition for resources for different root systems.
- The competition between sick and healthy plants.
- The competition between nurse plants and small plants.

The accompanying video shows that thousands of small plants were rendered with large trees for different conditions of resource competition. To render growth of thousands of plants interactively in real-time, we used the Wang Tile method.

REFERENCES

[1] M. Alsweis and O. Deussen. Modeling and visualization of symmetric and asymmetric plant competition. *Natural Phenomena 2005. Eurographics Association in coorperation with ACM/SIGGRAPH.*, pages 83–88, 2005.

[2] M. Alsweis and O. Deussen. Efficient simulation of vegetation using light and nutrition competition. *Simulation und Visualisierung 2006, Magdeburg.*, pages 83–88, 2006.

[3] A.R.Berkowitz, C.D.Canham, and V.R.Kelly. Competition vs. facilitation of tree seedling growth and survival in early successional communities. *Ecology*, pages 1156–1168, 1995.

[4] S. Bauer, U. Berger, H. Hildenbrandt, and V. Grimm. Cyclic dynamics in simulated plant populations. *Proceedings. Biological sciences / The Royal Society.*, 269(1508):2443–2450, 2002.

[5] Stephan Behrendt, Carsten Colditz, Oliver Franzke, Johannes Kopf, and Oliver Deussen. Realistic real-time rendering of landscapes using billboard clouds. *Computer Graphics Forum*, 24:507–516, 2005.

 ISBN 978-80-86943-98-5

[6] B. Benes. An Efficient Estimation of Light in Simulation of Plant Development. In *Computer Animation and Simulation'96*, Springer Computer Science, pages 153–165. Springer–Verlag Wien New York, 1996.

[7] B. Benes. A Stable Modeling of Large Plant Ecosystems. In *Proceedings of the International Conference on Computer Vision and Graphics*, pages 94–101. Associoation for Image Processing, 2002.

[8] B. Benes and E. Mill'an. Virtual Climbing Plants Competing for Space. In Nadia Magnenat-Thalmann, editor, *IEEE Proceedings of the Computer Animation 2002*, pages 33–42. IEEE Computer Society, 2002.

[9] U. Berger and H. Hildenbrandt. A new approach to spatially explicit modelling of forest dynamics: Spacing, ageing and neighbourhood competition of mangrove trees. *Ecological Modelling*, 132:287–302, 2000.

[10] Brenda B. Casper and Robert B. Jackson. Plant competition underground. *Annual Review of Ecology and Systematics*, 28:545–570, 1997.

[11] Michael F. Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. Wang tiles for image and texture generation. *Proceedings of ACM SIGGRAPH.*, pages 287 – 294, July 2003.

[12] Philippe Decaudin and Fabrice Neyret. Rendering forest scenes in real-time. In *Rendering Techniques '04 (Eurographics Symposium on Rendering)*, pages 93–102, June 2004.

[13] O. Deussen, P. Hanrahan, B. Lintermann, R. Mech, M. Pharr, and P. Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. *SIGGRAPH, Orlando, Florida, Proceedings of ACM*, pages 275 – 286, 1998.

[14] O. Deussen and B. Lintermann. A modelling method and user interface for creating plants. In *Proc. Graphics Interface 97*, pages 189–198, May 1997.

[15] Rubio Gerardo, Walk Tom, Ge Zhenyang, Yan Xiaolong, Liao Hong, and Lynch Jonathan P. Root gravitropism and below-ground competition among neighbouring plants: A modelling approach. *Annals of Botany*, 88(5):929–940, 2001.

[16] William Van Haevre, Fabian Di Fiore, Philippe Bekaert, and Frank Van Reeth. A ray density estimation approach to take into account environment illumination in plant growth simulation. In *SCCG '04: Proceedings of the 20th spring conference on Computer graphics*, pages 121–131, New York, NY, USA, 2004. ACM Press.

[17] Jr. James F. Cahill. Interaction between root and shoot competition vary among species. *Proceedings. Oikos*, 99:101–112, 2002.

[18] J.G.Franco-Pizana, T.E.Fulbright, D.T.Gardiner, and A.R.Tipton. Shrub emergence and seedling growth in microenvironments created by prosopis glandulosa. *Journal of Vegetation Science*, 7(2):257–264, 1996.

[19] Brendan Lane and Przemyslaw Prusinkiewicz. Generating spatial distributions for multilevel models of plant communities. *Proceedings of Graphics Interface*, pages 69–80, 2002.

[20] A. Lindenmayer. Mathematical models for cellular interaction in development. *Parts I and II. Journal of Theoretical Biology*, 18:280 – 315, 1968.

[21] Lars Mündermann, Yvette Erasmus, Brendan Lane, Enrico Coen, and Przemyslaw Prusinkiewicz. Quantitative modeling of arabidopsis development. *Plant Physiology*, 139:960–968, 2005.

[22] Przemyslaw Prusinkiewicz. Modeling and visualization of biological structures. *In Proceedings of Graphics Interface*, pages 128–137, 1993.

[23] Ge Zhenyang, Luo Xiwen, and Jonathan P Lynch. Computer modeling of the effect of root architecture on competition for diffusive nutrient uptake with neighboring root systems. *International Conference on Agricultural Engineering*, pages 161–167, 1999.

 ISBN 978-80-86943-98-5

Figure 8: Competition for light between dense trees and different small plants.

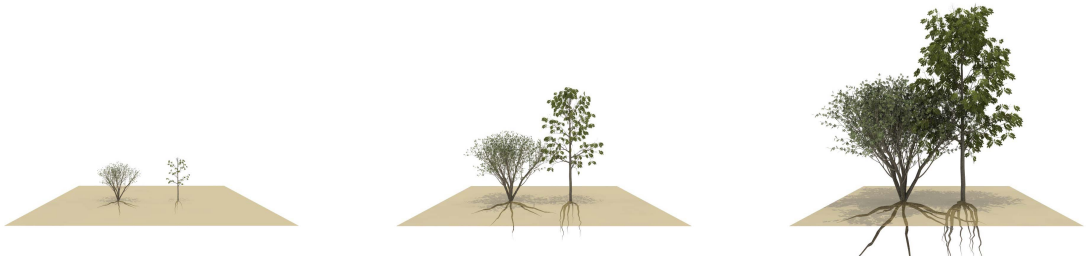


Figure 9: Competition for resources between dense tree roots and flat tree roots.

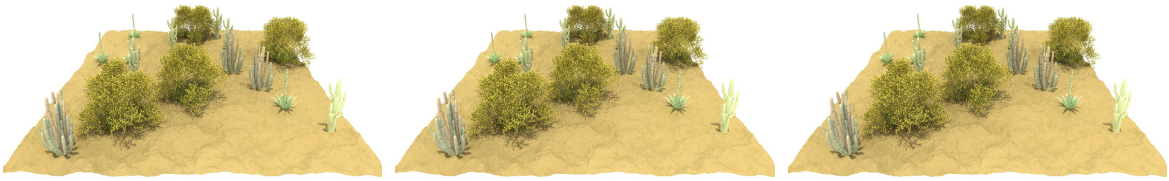


Figure 10: Competition for resources between nurse plants and small plants.

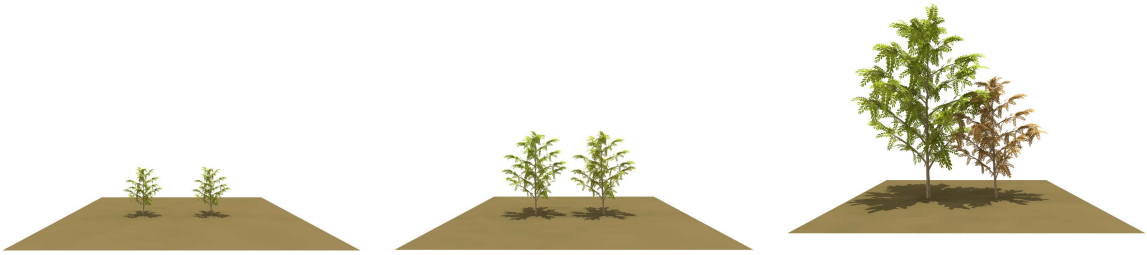


Figure 11: Competition between a sicker plant and a healthy plant.

 ISBN 978-80-86943-98-5

Data Centric Transfer Functions for High Dynamic Range Volume Data

Amit Chourasia
San Diego Supercomputer Center
University of California San Diego
amit@sdsc.edu

Jürgen P. Schulze
California Institute for Telecommunications and Information Technology
University of California San Diego
jschulze@ucsd.edu

ABSTRACT

Creating effective transfer functions for high dynamic range scalar volume data is a challenging task. For data sets with limited information about their content, deriving transfer functions using mathematical properties (gradient, curvature, etc.) is a difficult trial and error process. Traditional methods use linear binning to map data to integer space for creating the transfer functions. In current methods the transfer functions are typically stored in integer look-up tables, which do not work well when the data range is large. We show how a process of opacity guidance with simple user interface can be used as the basis for transfer function design. Our technique which uses opacity weighted histogram equalization lets users derive transfer functions for HDR floating point easily and quickly. We also present how to adopt these techniques for real-time interactive visualization with minimal pre-processing. We compare our techniques with traditional methods and show examples.

Keywords: Transfer Function, High Dynamic Range, Histogram Equalization, Floating Point Data, Volume Rendering

1 INTRODUCTION

Volumetric visualization lets users present complex, 3D data sets in visual form. An important part of this is the design of transfer functions for color and opacity, to be mapped on the data values. Using a better transfer function is analogous to increasing signal to noise ratio, where the signal is the information desired by the user. As the notion of signal is not defined for volume datasets, there is a need to do this task in a convenient and often exploratory manner. This is where the design of transfer functions plays an important role. Unfortunately, this process is often time consuming and cumbersome.

Transfer function design for medical datasets has been the focus of many studies. In general, this type of data consists of various layers of biological materials (flesh, bone, tissue, etc.). These materials can often be segmented by algorithmically identifying their boundaries.

But what about non-medical datasets, like datasets resulting from computational simulations, like computational fluid dynamics, weather, or simulations of physical processes? With faster processor speeds and the widespread adoption of clustering technology, the use of computational simulation to model physical phenomena is widespread. These simulations can generate enormous amounts of data. The characteristics of these datasets are very difficult to categorize but they often show some common traits like a high dynamic range (HDR), and subtle details in high fidelity. Our work is focused on HDR floating point data sets.

For HDR floating point data sets, transfer function design becomes an increasingly difficult task, primarily because the data range by far exceeds the usual 8 bits of color resolution on the monitor. It is unlikely that a single classifier will provide the desired results. In addition, there is often limited a priori information about the data content which compounds the problem for creating good visualizations.

Data distribution directly affects opacity and color mappings. If interesting data is clustered within small data ranges, then color and opacity need to be tuned specifically for each such range to show it at the maximum possible detail in the visualization result.

In this paper, we present a method which lets users design transfer functions without using derived and hard to comprehend mathematical properties like gradient, curvature, tensor, etc. In our method, we use an extension of histogram equalization, along with opacity guidance provided by the user. Our new algorithms allows the user to minimize the effort of tuning the transfer functions. We show how to utilize data distribution as a basis for transfer function design and how it can be done in a real-time software application. In addition, we compare our results to the popular technique of mapping a color gradient linearly to a data range. The transfer functions our algorithms generates can also be generated manually, but it would take the user considerably more time to do so from the start. Our algorithms

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.

are intended to provide a fast way to generate reasonable transfer functions which can then be refined manually.

The main contributions of this paper are:

1. Opacity Weighted Histogram Equalization (OWHE) as the basis for transfer function design.

2. Efficient utilization of the available color space, thus yielding a more detailed visualization of structures in data sets.

3. Real-time interaction with a simple and intuitive user interface to access the algorithm's parameters.

The paper is structured as follows. In Section 2 we review related work. In Section 3 we discuss the underlying theoretical background. Section 4 describes the algorithms we have developed. Our user interface is presented in Section 5. We present the application of our method to three typical high-dynamic range data sets in Section 6. Section 7 concludes the paper and Section 8 suggests areas for future work.

2 RELATED WORK

A multitude of methods have been proposed to generate transfer functions. They can be broadly classified into four categories which are described in detail by Pfister et al. [11]. The first are trial and error methods where the user tweaks parameters to achieve the desired result. Second, data centric approaches work without assuming a data model [1]: the contour spectrum which computes metrics of the data and integrates the results with the user interface used to set iso-values. Third, a data centric approach which assumes an underlying data model [5, 6]: this method semi-automatically generates the transfer function by utilizing the complex relationship between data and its optical properties, the assumption being that the features of interest are boundaries between materials; our approach is not based on material boundaries, gradient values, or multidimensional transfer functions. Fourth, an image centric method based on organized samples; this is similar to the Design Galleries concept [9].

Other approaches, related to the ones presented in this paper are volume visualization using approaches from high dynamic range imaging (HDRI) [16]. The HDR method performs volume rendering in floating-point space, thus preserving precision, and then uses tone mapping to display the result. Practically, insufficient HDR display hardware [4] and difficult interpretation of tone mapped results are remaining problems with this technique. Furthermore, the trial and error effort now moves to tone mapping [3] and exposure selection instead of transfer function design.

Potts and Moeller [12] propose to use transfer functions on a logarithmic data scale. This method provides a mechanism for the user interface for transfer function design. We think that for many users it is more intuitive to see the transfer function on a linear scale and interactively zoom in and out where necessary.

Tzeng and Ma [13] use the floating-point data space during segmentation to identify the various materials present in a data set. However, their approach does not work well if the data set contains continuous densities which do not occur as clusters. Also, defining and searching clusters is a cumbersome task. Lundstorm et. al. [8] use spatial coherence for transfer function design for medical datasets.

For data sets without distinguishable materials and highly inhomogeneous data distribution, none of the above methods sufficiently addresses the dynamic range of the data and its implications. Our method aims to semi-automatically maximize the use of color space for regions of interest, while still allowing the visualization of the entire data set. Most existing volume rendering algorithms can seamlessly incorporate our method into their classification and rendering pipeline as an additional option for transfer function design.

3 HIGH DYNAMIC RANGE TRANSFER FUNCTIONS

This paper addresses the rendering of single-valued floating-point volume data sets. The challenge is that it is often unknown to the user which parts of the floating-point range are of interest, i.e., what data ranges colors and opacities should be mapped to. Many traditional methods allow the user to specify minimum and maximum values to constrain the data range, and then specify color and opacity mappings for the selected range. Histograms can facilitate the process of finding the range of interest to visualize, but they do not help much when the data values are spread out over a very large range. Often, the user wants a quick overview of what is in a data set before making fine adjustments to pull out part of it. A simple way that can achieve this is an opacity function of constant, low opacity. Or a transfer function based on the histogram can be created by mapping higher opacity to higher histogram values, or vice-versa. All these methods pose significant limitations like limited control over the opacity mapping, or a cumbersome trial and error process.

A simple, but frequently used approach to map floating point values in volume data sets to colors is to linearly map the values between minimum and maximum value to a user specified color gradient. This approach is referred to as Linear Binning in the upper half of the box of Existing Techniques in Figure 1. In the recent past, more sophisticated methods have been developed. Some approaches retain the high dynamic data range during the volume rendering step by using a floating-point image buffer. Then, tone mapping is used to convert the HDR image to the color space today's graphics

 ISBN 978-80-86943-98-5

hardware uses, which is usually 24 bits per pixel. Figure 1 illustrates this approach in the lower half of the Existing Techniques box.

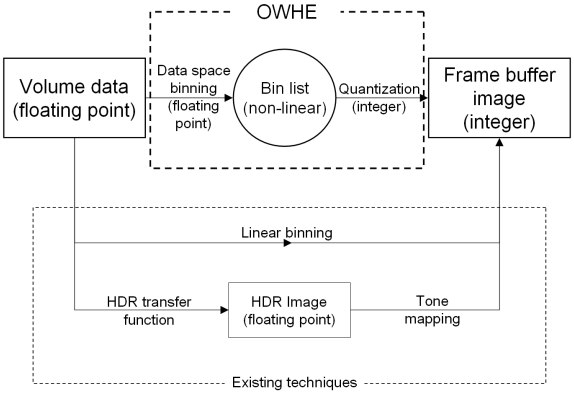


Figure 1: Flowchart showing the Opacity Weighted Histogram Equalization algorithm with the existing techniques of linear binning and HDR with tone mapping. Our approach is shown in the box labeled OWHE, the existing techniques are shown in the dotted box below.

The latest graphics cards support floating-point textures and even floating-point frame buffers. Texture hardware-based volume rendering benefits from these by allowing it to use much higher precision when volume slices are blended with the frame buffer, so that more features of the data set can be preserved during rendering. However, the problem of mapping color and opacity to data over a large data range still persists. Many floating-point volume data sets have sections of high data activity, as well as large sparsely used data ranges. This paper presents an approach to support the user in creating meaningful transfer functions for these kinds of data sets with minimal manual intervention.

3.1 Histogram Equalization

The transfer function generation method we are presenting in this paper is an extension of histogram equalization (HE). In image processing, histogram equalization is used to improve the contrast of images. The idea is that the brightness values in the image are increased or decreased such that the histogram becomes flat. The formula for histogram equalization is:

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{m} \qquad k = 0,1,2,...,L-1$$

s_k are the equalized histogram values, r_k are the original histogram values, L is the number of gray levels in the image, n_j is the number of times each gray level appears in the image, and m is the total number of pixels in the image.

3.2 Opacity Weighted Histogram Equalization

Histogram equalization has been used quite successfully in image processing to improve the image quality by improving contrast. In volume rendering, HE alone can provide a quick way to considerably improve the color mapping; we will show this in Section 6. However, HE alone does not help with the creation of the opacity transfer function. Our transfer function generation approach is a two-step approach. In the first step, the user uses HE to create a first approximation of a useful color mapping for the data set. At this point, a constant, low opacity function or maximum intensity projection is used to show the entire data set. In the second step, the user places a few simple transfer function widgets on the transfer function to pull out specific parts of the data set. Then we apply our Opacity Weighted Histogram Equalization (OWHE) algorithm to refine the color mapping, using the additional hints the software gets from user defined opacity widgets. Our approach is depicted in the upper half of Figure 1.

The basic idea of OWHE is to modulate the number of times a voxel value occurs in the data set by the opacity the user assigned to it. Thus, higher color resolution is created in areas of high opacity, whereas in low opacity regions, fewer colors are used. In our tests, we found that this approach can achieve very good visualization results with much less effort than if the user tried to create the same transfer function manually.

Our modification of the Histogram Equalization function is indicated below. op_j is the user specified opacity of the data value.

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{op_j \times n_j}{m} \qquad k = 0,1,2,...,L-1$$

Figure 2 compares linear color mapping to OWHE. In OWHE, the colors are more compressed the higher the data frequencies in the histogram, and the higher the desired opacity. In areas with fewer data points in the histogram or lower opacity, larger data ranges get mapped to the same color. We use the same color map in both sample images.

For rendering, we are using a standard light emission model for the volume, along with back-to-front alpha blending. We target standard graphics and display hardware, based on a precision of 24 bits per pixel. Our algorithm can potentially improve the visualization results even on true HDR displays which can display more than 24 bits per color.

4 IMPLEMENTATION DETAILS

We implemented the HE algorithm, as well as the OWHE algorithm and integrated them with the DeskVOX [2] volume rendering software. Both

 ISBN 978-80-86943-98-5

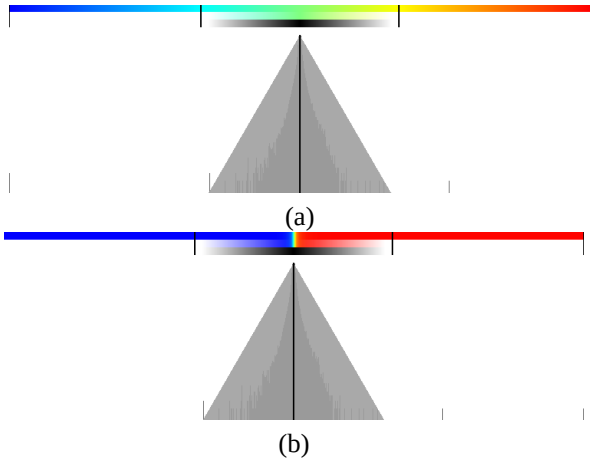


Figure 2: (a) linear binning, (b) OWHE. The data values are located along the horizontal axis. The bars at the top of each image show, from top to bottom: color only, color and opacity, opacity only, bin boundaries. Shown below is the histogram, overlaid by the opacity function.

algorithms have various parameters which can be changed by the user from within the volume rendering application. We implemented these parameters because they turned out to be useful when using the algorithms on real data sets.

Cull Skip regions: This parameter specifies whether regions with transfer function widgets which map zero opacity to a data range (Skip widgets) should be excluded from the binning process. This impacts especially the HE algorithm, because it, by default, uses the entire data range.

Cull Duplicate Values: By default the HE process uses all data values in the volume and distributes them evenly into bins. With this option enabled, the user can specify that duplicate values be removed, so as to evenly distribute different values only. This makes a difference if the data set contains disproportionately large amounts of certain values, which are not of particular interest but should not entirely be excluded from the binning process.

Trim to Range: The data range the algorithm operates on can be constrained to be within a minimum and maximum value. These minimum and maximum values can be the absolute min/max values in the data set, or they can be values in-between. If this option is disabled, all data values are used for the HE algorithm.

Fast Sampling: Using all data values of the volume in the algorithms can be slow if the volume is large. The Fast Sampling option uses only the specified number of samples for the binning process, which speeds up the algorithm, but also makes it less accurate. As Tzeng and Ma have stated in [13], sampling about 1% of the data values often achieves satisfactory results.

5 USER INTERFACE

We developed a user interface for the transfer function editor which integrates our transfer function generation methods with traditional approaches to transfer function design. Figure 3 shows a screen shot of the editor. In the upper half of the window the transfer function is displayed. At the top of it are three colored bars: the upper one shows just the color mapping, the next shows the combined mapping of color and opacity, and the third shows opacity only. The data values are located on the horizontal axis of the graph. The black, vertical lines can be grabbed with the mouse and moved to left and right, so as to place a particular color control point in the color bar. In the same manner the user can use multiple widgets to create the opacity mapping.

Below the color bars is the opacity function. The gray area is the opacity mapping. The opacity function can be created from four different elements (widgets): pyramids, Gaussians, custom functions with control points, and Skip widgets. As with the colors, each opacity widget can be grabbed and moved with the mouse by clicking on a black, vertical line. The user can composite the transfer function with multiple widgets; at overlapping areas of widgets the opacity is determined by the maximum value in one of the widget.

The horizontal axis represents linear data values. Both color and opacity transfer functions are displayed in linear data range. When our algorithms are active, it is more difficult to manipulate the color widgets because the control lines can be much closer together. It is easier to use the control lines when the user zooms in to them. To zoom, the user can use numeric range or use the mouse wheel where the mouse location acts as the center for the region to be zoomed. In pressed state the mouse wheel enters pan mode, where the user can move the transfer function to the left and right.

5.1 Parameter Window

Figure 4 shows the dialog window with the parameters available for our algorithms. When the button labeled *Distribution based data range mapping* is off, the transfer function uses a linear distribution of colors over the range specified by *Start value* and *End value*, and our algorithms are disabled. We used this setting to compare our algorithms to the linear binning approach in Section 6. When *Distribution based data range mapping* is checked, the algorithms described in this paper are applied. The user can choose between HE and OWHE, which have been described in Section 3.

6 RESULTS AND DISCUSSIONS

We have applied our technique to three different scalar floating-point data sets. The data sets have different dynamic range and properties. We chose a representative single time step from the following three temporal

 ISBN 978-80-86943-98-5

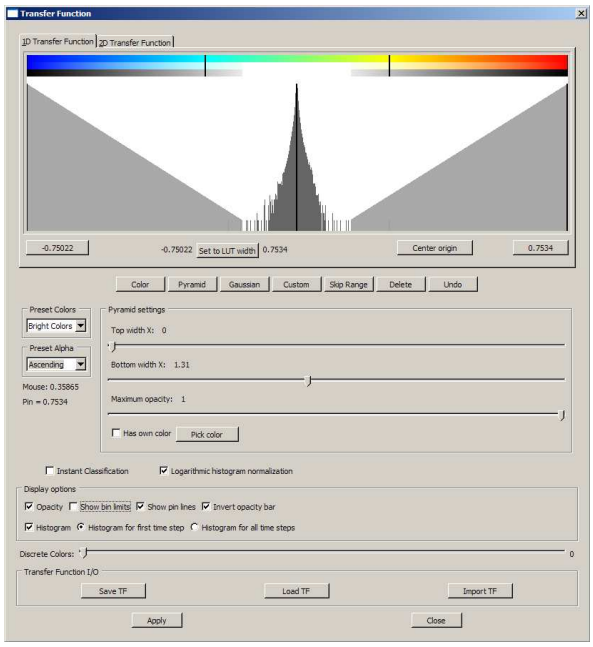


Figure 3: Our widget-based transfer function editor.

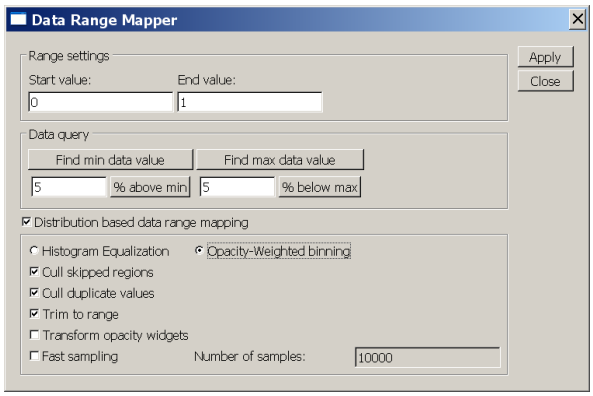


Figure 4: The dialog window to set the new data range mapping parameters.

datasets, all of which are results from simulations on supercomputers.

1) **TeraShake** is an earthquake simulation [10, 15], done to study seismic wave propagation based on kinematic and dynamic rupture models. We use the X component of velocity with a data set size of $750 \times 350 \times 100$ voxels. The data covers a small range but has subtle details in very narrow regions.

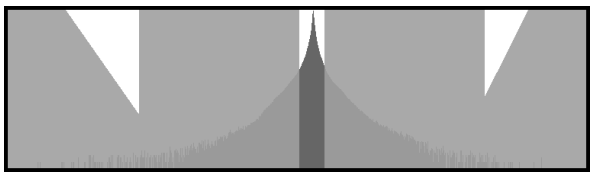


Figure 5: Histogram (logarithmic scale) and opacity function of the TeraShake earthquake simulation data set.

2) **Isabel** is the result of the simulation of a hurricane [14]. We use the snow data with a data set size of $500 \times 500 \times 100$ voxels.

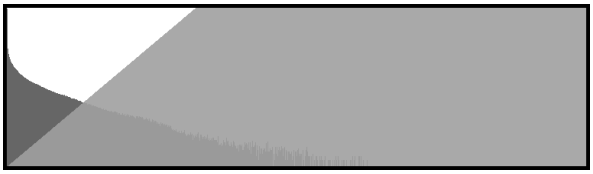


Figure 6: Histogram (logarithmic scale) and opacity function of the hurricane Isabel data set.

3) **Enzo** is a simulation [7] of supersonic compressible turbulence at Mach 6 performed with the Enzo code for cosmology and astrophysics, using 512 CPUs of the DataStar supercomputer at the San Diego Supercomputer Center. It was designed to model a sub-volume (linear size: 5 pc = $1.5 \cdot 10^{17}$ m) within a star forming a molecular cloud. We use a sub-sampled matter density data set with a size of $512 \times 512 \times 128$ voxels.

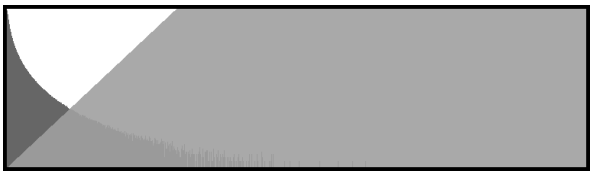


Figure 7: Histogram (logarithmic scale) and opacity function of the cosmology data set Enzo.

More detailed properties of the above data sets are listed in Table 1, and the data histograms are shown in Figures 5, 6, and 7. Figures 8a-c show rendering results of different datasets, generated with our OWHE algorithm.

6.1 Comparison of Linear Color Mapping with HE and OWHE

We compare our HE and OWHE methods to the popular linear color mapping method henceforth referred to as standard method. For the comparisons, we created appropriate opacity transfer functions for each dataset. The opacity functions are shown in figures 5, 6, 7, along with the data histogram.

In the standard method, color map and opacity are applied to a linearly data range. The renderings achieved with the standard method are shown in Figures 8(a),(d) & (g). For HE and OWHE, we use the same opacity transfer functions as for the standard method, but the color mapping gets modified by the algorithms. The processing time for HE and OWHE depends on the number of voxels in the data set and is on the order of several seconds. The corresponding renderings for HE and OWHE are shown in Figures 8(b),(e),(h) & (c),(f),(i).

In the TeraShake data set, most of the data is located around the value 0.0 (see histogram in Figure 5), which represents no ground motion. In the images, the opacity

 ISBN 978-80-86943-98-5

for the data is set such that wave features are visible. A color map has been created to show positive velocity as red-orange-yellow, and negative velocity as blue-cyan-green. Black has been chosen to show a velocity of 0.0 with no opacity. A desirable visualization for this is to show the wave propagation structure in low, mid, and high range all together. With the standard method (see Figure 8(a)) it is difficult to observe the wave propagation features. The HE method (see Figure 8(b)) shows only a minor improvement over the standard method, but our OWHE method (see Figure 8(c)) is able to bring out subtle details of the wave form and its location, which takes a great deal of effort if done manually.

The Isabel data set we chose for this comparison is snow mixing ratio. We created a visualization which lets us see the entire volume, and in addition the coastline of Florida is visible for orientation. The opacity function 6 has been chosen such that higher snow mixing ratio is more opaque. As can be seen in Figure 8(d), visualization with the standard method provides information just for a relatively small region. The HE method is of not much help in this instance but the OWHE method (see Figure 8(e)) is able to show much more detail of the data value distribution by more effectively using the available color space (see Figure 8(f)).

The Enzo volume shows "'matter density"' during a star formation process. We chose an opacity function similar to that used for the Isabel data set, where higher density is completely opaque and lowest is transparent. The standard method (see Figure 8(g)) provides little distinction in features of interest. The HE method (see Figure 8(h)) shows some detail. The OWHE method (see Figure 8(i)) brings out more details in the data, even thin filaments can be seen in high fidelity. The V- and U-shaped shocklets or "'Mach cones"', characteristic of the supersonic turbulence found in interstellar gas, can clearly be seen.

	Enzo	**Isabel**	**TeraShake**
# Voxels	$33.6 \cdot 10^6$	$25.0 \cdot 10^6$	$28.1 \cdot 10^6$
% Unique	73.2	86.5	71.5
Min	$8.52 \cdot 10^{-3}$	0.00	$-6.68 \cdot 10^{-1}$
Max	$2.23 \cdot 10^2$	$1.35 \cdot 10^{-2}$	$5.97 \cdot 10^{-1}$
Mean	$9.28 \cdot 10^{-1}$	$2.23 \cdot 10^{-6}$	$-1.43 \cdot 10^{-4}$
Std dev	1.58	$1.54 \cdot 10^{-5}$	$8.50 \cdot 10^{-3}$

Table 1: Data statistic for Enzo ($512 \times 512 \times 128$), Isabel ($500 \times 500 \times 100$) and TeraShake ($750 \times 375 \times 100$). % unique is the percentage of duplicate values, removed with our Cull Duplicates parameter. Min/Max are the minimum and maximum data values. Mean is the mean value of all data values, including duplicates. Std dev is the standard deviation over all data values, including duplicates.

6.2 Limitations

Our methods works well for HDR floating point data, but they have some limitations. The methods are not suitable to all datasets. For example in medical datasets with little variation in the data our method will not work well. Additionally, in cases where high color fidelity is not of much importance for visualization our method will not be beneficial.

7 CONCLUDING REMARKS

We presented techniques to facilitate the creation of transfer functions for high dynamic range floating point data sets by taking data distribution and user input into account. We compared our techniques to standard methods and demonstrated how our techniques can be used to create useful transfer functions faster and with less effort. Our experience with this technique indicates that time required to get meaningful images is significantly reduced but a survey would be needed to quantify and test this hypothesis. The color space utilization with our method is efficient leading to high fidelity detail in renderings.

8 FUTURE WORK

In the future, we want to improve our rendering method by supporting higher precision frame buffers (e.g., with Nvidia's GL_NV_float_buffer extension). We would also like to utilize HDRI color space for transfer functions with our technique. Other existing histogram equalization techniques could be added as comparative exploration methods. In addition, we want to extend the application of this technique to temporal data sets.

9 ACKNOWLEDGEMENTS

We would like to thank Steve Cutchin (SDSC) for helpful comments and ideas. Alexei Kritsuk provided the Enzo data and helped us to explain the visualization. The research was funded through NSF SDSC core grant and CalIT2 research grant.

REFERENCES

[1] C.L. Bajaj, V. Pascucci, and D.R. Schikore. *The Contour Spectrum*. IEEE Visualization '97 Proceedings, pp. 167–173, 1997.

[2] Deskvox. *Deskvox volume rendering software by Jurgen P. Schulze*. URL: http://www.calit2.net/ jschulze/projects/vox/, 2006.

[3] R. Fattal, D. Lischinski, and M. Werman. *Gradient domain high dynamic range compression*. In Proceedings of ACM SIGGRAPH 2002, Computer Graphics Proceedings, Annual Conference Series. ACM Press / ACM SIGGRAPH, July 2002. 10, 13, 15, 2002.

[4] A. Ghosh, M. Trentacoste, , and W. Heidrich. *Volume Rendering for High Dynamic Range Displays*. Proc. EG/IEEE VGTC Workshop Volume Graphics '05 , July 2002. 10, 13, 15, 2005.

[5] G. Kindlmann and J.W. Durkin. *Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering*. Proceedings of the IEEE Symposium on Volume Visualization '98, pp. 79–86, 1998.

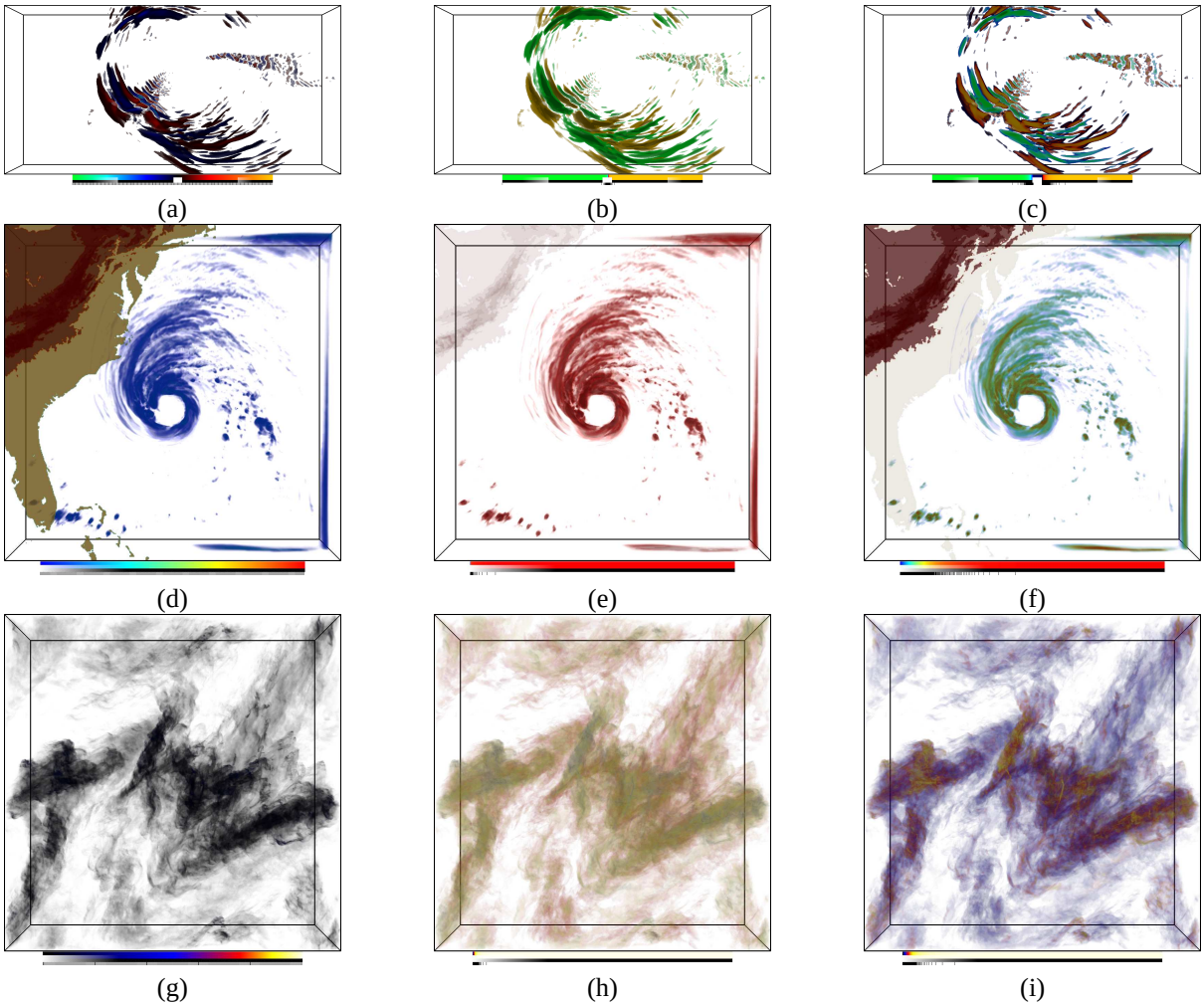


Figure 8: (a),(d),(g) Linear binning; (b),(e),(h) Histogram Equalization; and (c),(f),(i) OWHE renderings

[6] J. Kniss, G. Kindlmann, and C. Hansen. *Multi-Dimensional Transfer Functions for Interactive Volume Rendering*. IEEE Transactions on Visualization and Computer Graphics, 8(3), pp. 270–285, 2002.

[7] A. Kritsuk, R. Wagner, M. Norman, and P. Padoan. *High Resolution Simulations of Supersonic Turbulence in Molecular Clouds, in Numerical Modeling of Space Plasma Flows*. Proceedings of CalSpace-IGPP International Conference, Palm Springs, 26-30 March 2006, to appear in Publ. Astron. Soc. Pacific., 2006.

[8] Claes Lundström, Anders Ynnerman, Patric Ljung, Anders Persson, and Hans Knutsson. *The alpha -histogram: Using Spatial Coherence to Enhance Histograms and Transfer Function Design*. EUROVIS - Eurographics /IEEE VGTC Symposium on Visualization 2006, 2006.

[9] J. Marks, B. Andalman, P.A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. *Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation*. ACM SIGGRAPH '97 Proceedings, pp. 389–400, 1997.

[10] K.B. Olsen, S. Day, J.B. Minster, Y. Cui, A. Chourasia, M. Faerman, R. Moore, P. Maechling, and T. Jordan. *Strong Shaking in Los Angeles Expected From Southern San Andreas Earthquake*. Geophys. Res. Lett., 33, L07305, doi:10.1029/2005GL025472, 2006.

[11] H. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L. Sobierajski Avila, K. Martin, R. Machiraju, and J. Lee. *Visualization Viewpoints: The Transfer Function Bake-Off*. IEEE Computer Graphics and Applications, May/June 2001 (Vol. 21, No. 3), pp. 16–22, 2001.

[12] S. Potts and T. Moeller. *Transfer Functions on a Logarithmic Scale for Volume Rendering*. Proceedings of Graphics Interface '04, 2004.

[13] Fan-Yin Tzeng and Kwan-Liu Ma. *A Cluster-Space Visual Interface for Arbitrary Dimensional Classification of Volume Data*. Proceedings of the Joint Eurographics-IEEE TVCG Symposium on Visualization '04, May, 2004.

[14] VisContest. *Visualization Contest at IEEE Visualization Conference '04*. URL: http://vis.computer.org/vis2004contest/data.html, 2004.

[15] VisContest. *Visualization Contest at IEEE Visualization Conference '06*. URL: http://vis.computer.org/vis2006/contest_c.html, 2006.

[16] X. Yuan, M.X. Nguyen, B. Chen, and D.H. Porter. *High Dynamic Range Volume Visualization*. IEEE Visualization '05 Proceedings, pp. 327–334, 2005.

 ISBN 978-80-86943-98-5

Pipeline for the Creation of Surface-based Averaged Brain Atlases

Anja Kuß Hans-Christian Hege
Zuse Institut Berlin
Takustraße 7
14195 Berlin, Germany
kuss@zib.de
hege@zib.de

Sabine Krofczik Jana Borner
Freie Universität Berlin
Königin-Luise-Str. 28/30
14195 Berlin, Germany
sabine@krofczik.de
boerner@neurobiologie.fu-berlin.de

ABSTRACT

Digital atlases of the brain serve as a spatial reference frame which can be used to relate data from different image modalities and experiments. In this paper we describe a standardized pipeline for the creation of extendable surface-based anatomical insect brain atlases from 3D image data of a population of individuals. The pipeline consists of the major steps imaging and preprocessing, segmentation, averaging, surface reconstruction, and surface simplification. At first, 3D image data sets from confocal microscopy are resized, stitched, and initially displayed using standard image processing and visualization tools. Then brain structures, such as neuropils and neurons, are labeled by means of manual segmentation and line extraction algorithms. The averaging step comprises affine and elastic registration and a mean shape selection strategy. Finally non-manifold surfaces of the labeled and aligned structures are reconstructed using a generalized surface reconstruction algorithm. These surfaces are simplified and adapted to further needs by decimation and retriangulation. The chosen methods of each step are adequate for a variety of data. We propose an iterative application of the pipeline in order to build the atlas in a hierarchical fashion, integrating successively more levels of detail. The approach is applied in several different neurobiological research fields.

Keywords: Digital neuroanatomy, brain atlas, surface reconstruction.

1. INTRODUCTION

One specific aim of neuroscience is to analyze learning and behavior. Understanding such neural functions is based on knowledge about the wiring of the brain and functional properties of individual neurons as well as parts of the nervous system. Therefore anatomical and functional properties of neural networks are investigated.

Particularly suited for such analysis are insect brains, since they are easy to access and brain substructures as well as important single neurons are already known and identified. Furthermore, due to the small size of the structures one is able to examine them as a whole down to a level of single neurons. Much knowledge is extracted by comparing anatomical structures and their functions of individuals at several development stages or from individuals that equal or differ genetically.

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.

In the last decade 3D imaging methods, like confocal microscopy or transmission electron tomography, have become standard together with sophisticated staining methods. Nowadays they provide large amounts of data that has to be organized, analyzed and explored. Almost a revolution from the biological point of view are population-averaged standard atlases that for the first time in biology provide spatial references. These allow the researchers to relate data from different experiments and different data modalities as well as to accumulate information with spatial reference. Furthermore they facilitate visualization of neurobiological data and form a basis for investigating the relation between brain morphology, brain function, and genetic factors. In the long term this will help to turn neuroanatomy into a more quantitative science.

We describe a generalized pipeline for the generation of insect brain atlases from 3D image data of a population of individual brains. We consider surface-based atlases, since these are easier to deal with than volumetric atlases. The pipeline roughly comprises imaging, segmentation, averaging, and surface reconstruction. All methods required within

the pipeline were integrated in the open visualization platform Amira [SWH04].

This paper is organized as follows: In section 2 we give an overview about existing brain atlases. Section 3 describes the concept and the steps of the atlas generation pipeline. Applications of the procedure and the resulting atlas are presented in Section 4. We discuss the limits of our approach and conclude in Section 5.

2. RELATED WORK

For the human brain a variety of digital population-based atlases have been introduced and continuously enhanced. Techniques for the generation of such brain atlases, for instance, are described in [GMT00] and [TT01]. They focus on the development of registration and averaging strategies using cerebral CT and MRT scans. A processing environment that enables researchers to combine and execute established methods is presented in [RMT03]. It has been used successfully to build average intensity atlases with fully automated processing. The general steps for atlas generation of human brains are similar to the ones for the generation of insect brain atlases. Nevertheless imaging modalities and application dependent objectives require the development of more specific and appropriate techniques.

Meanwhile invertebrate brain atlases also do emerge. In [RZM*02] and [RBMM01] the generation of atlases for the fruitfly *Drosophila melanogaster* and the honeybee *Apis mellifera* are described. Besides generation methods supplemental information on neurobiological backgrounds and application fields of insect brain atlases can be found in [BRR*05] and [PH04]. Web interfaces (http://www.neurofly.de; http://www.neurobiologie.fu-berlin.de/BeeBrain) allow downloading and interactive viewing of currently existing models. The analysis of insect brains is still an explorative research field. So far little attention has been paid to the integration and sequencing of all steps needed for a complete atlas generation into one single visualization platform. The insertion of a taxonomic description of brain structures and the usage of advanced visualization and interaction techniques also remain challenging tasks.

3. ATLAS GENERATION PIPELINE

The proposed pipeline for the generation of brain atlases is composed as follows: First the image data is preprocessed and initially visualized. In a second step the structures of interest are segmented and a label is assigned to each of them. The averaging takes place in step three. Therefor all labeled data sets have to be registered into a common reference and a mean or median has to be derived. Finally three-dimensional surface models of the averaged brain structures are reconstructed. Figure 1 shows a schematic overview of the proposed pipeline.

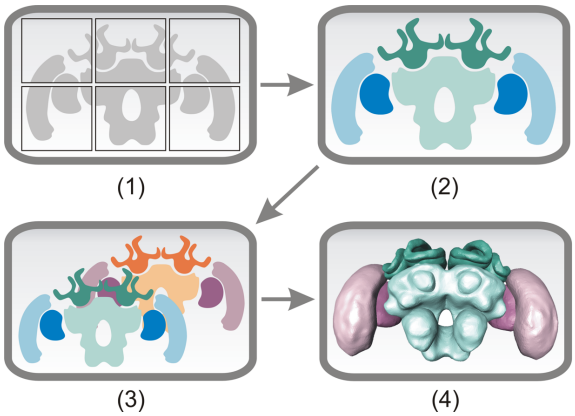


Figure 1. Schematic overview of the pipeline for the generation of surface-based insect brain atlases. After imaging and preprocessing (1), structures are segmented (2). The averaging of individuals (3) is followed by a surface reconstruction(4).

The underlying methods of each step have to fulfill several requirements. They must be applicable to the present imaging data and provide robust results. The techniques shall allow for modifications of final and intermediate data. For example the insertion and deletion of structures shall be supported. In addition the methods have to consider the user's skills and wishes. Finally the received results shall enable further processing.

We propose an iterative application of the pipeline. First coarse structures like neuropils are reconstructed and an initial atlas is generated. Using the same work steps, more detailed information such as substructures of neuropils or neurons is integrated. Thereby the initial standard serves as a reference frame for the registration procedure and the final visualization. This approach allows for a hierarchical organization of data, which in turn establishes a basis for the representation of the natural hierarchical organization of brain structures.

In the following we will describe the steps of the pipeline by means of brain data of the honeybee. Section 4 shows that the pipeline and its results are much more generally applicable.

Imaging and Preprocessing

Presently insect brains such as the fly brain or the honeybee brain are imaged with confocal laser scanning microscopy. After preparation including dissection and staining, the size of a honeybee brain (~$2.5 \times 1.6 \times 0.8\,mm$) extends the maximum field of view of the microscope with a common configuration. Therefore multiple stacks are used to acquire entire brains. Typically 2×3 partially overlapping tiles, each using

 ISBN 978-80-86943-98-5

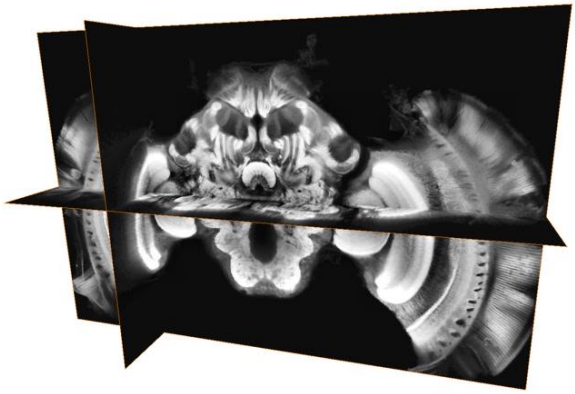


Figure 2. Slice view of a honeybee brain confocal microscopy image in xy-, yz-, and xz-plane. The data set is composed of six separately acquired image stacks.

between 80 and 120 slices of 512×512 pixels in plane and a thickness of $8\,\mu m$ are imaged.

For the combination of the image stacks landmarks that define corresponding points are placed manually. Afterward the stacks are merged using a conventional pyramidal blending. The landmarks are used to define the regions that will be blended. The scanning technique usually causes a shortening of distance in the z-direction. This is compensated by applying a linear scaling to the data [BRR*05].

By means of slicing or direct volume rendering or the combination of the two a first insight into the edited data is gained and the image and preprocessing quality can be checked. A slice view of a complete bee brain data set can be found in Figure 2.

Segmentation

In the next step anatomically distinct structures of the brain are segmented and assigned unique labels. The results are stored in so-called label fields in which each voxel represents a coding for a particular brain structure. These label fields will be used in the following averaging step and form the basis of the surface reconstruction. For several insects a coarse division and nomenclature of the brain into neuropils already has been defined and can be used as a guideline for the segmentation. In the honeybee brain we currently distinguish 22 major compartments.

The development of fully automatic segmentation methods is impeded by several difficulties. For instance, the borders of neuropils are often fuzzy, thus lacking strong gray value gradients. Gray values of structures highly depend on the chosen staining. Additionally they vary within the regions and between different individuals. Furthermore research in neurobiology still remains explorative aiming at discovering new structures and correlations. So frequently there is little a priori information and results are strongly influenced by the user's knowledge and experience.

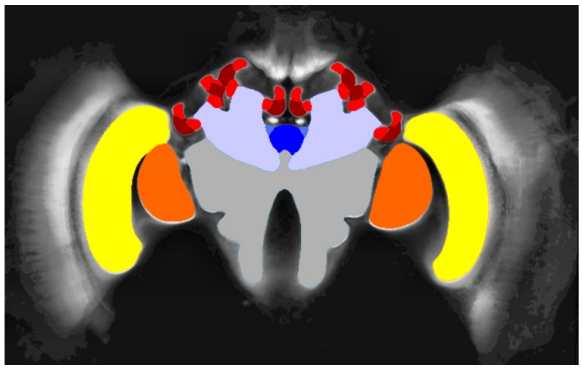


Figure 3. Segmentation of neuropils in the honeybee brain. Anatomical structures of interest were segmented and assigned individual labels. Here they are overlaid their corresponding gray value data set.

For the segmentation of neuropils we provide simple but robust, and user-friendly tools, which support manual segmentation. The user can choose from freehand boundary drawing, interval thresholding, and propagating contours based on a level set method [WnZ03]. For the latter a small circle is placed on mouse click and then the contour of the circle is extended as the user moves the mouse. All tools require the adjustment of only few parameters and allow for an interactive correction of the segmentation results. Their efficiency is improved by the integration of interpolation and extrapolation of label segmentation between slices. At any time structures can be added, removed or refined easily within the provided editor.

Figure 3 shows an example segmentation of honeybee brain neuropils.

Averaging

From the preprocessed and labeled data sets of individuals an atlas representing the brain characteristics of the analyzed species has to be derived. For this purpose an average of the sample is determined. This is usually done by computing the mean or median. In general the procedure comprises a registration of the data sets to a common reference and a subsequent averaging step.

We use an iterative averaging scheme which is based on the work presented in [RBMM01]. First global differences in position, orientation, and size of all data sets are compensated and an initial average image is generated. Using a nonrigid registration the data sets are then registered to the initial reference. This leads to a new less blurred average image to which the transformed images are registered again, and so forth.

Affine registration. The procedure starts with an affine registration of the individual images to an arbitrarily chosen template. Usually a subjective choice of a data set such as the original individual brain which appears most average to the user works as an adequate

 ISBN 978-80-86943-98-5

first reference. In order to allow for standardization we select the individual brain which has the smallest deviation from the mean volume over all components in the reconstruction as it has been used in [RZM*02]. Our affine registration implementation is based on the multiresolution search method presented in [SHH97].

Label and intensity averaging. Having registered all original individual images, a mean or median can be determined from them. Here the computation of the voxelwise arithmetic average of data values forms the most intuitive and general approach. For label fields, which are non-numerical images, a non-arithmetic average is needed. We do use a method that has been proposed in [BRR*05]. It selects the label that occurs most frequently and represents at least t percent of the valid voxels. A result voxel is termed 'undecided' if there is no unique most frequent label.

Elastic registration. Due to the remaining local shape varieties an average image solely based on a global affine registration might be inaccurate resulting in blurred average intensities and average labels with a large quantity of undecided voxels. Application of elastic registration methods helps to reduce this inaccuracy.

Our pipeline contains a non rigid registration method described in [RZM*02]. The algorithm computes an individual rigid transformation for each labeled brain structure. These transformations are relatively small due to the prior global alignment. For each voxel within a segmented substructure the transformation vector is calculated based on the per-structure alignment. Afterward the piece-wise defined vector field between the substructures is component-wise interpolated. This is done by using a heat transfer equation: at locations where the vectors are known the 'temperature', which means the component of the transformation vector, is kept fixed and in-between the resulting equilibrium state is computed.

We also offer a multiresolution and multigrid cubic B-spline registration based on the developments published in [RBMM01] and [RSH*99]. This approach pays more attention to fine details than the above outlined method.

In general, elastic registration is a delicate step, since the transformation might eliminate not only preparation artifacts like mechanical deformations, but also anatomical variability. If one is interested in this, elastic registration should be employed particularly conservatively.

Convergence. For some applications an average resulting from the affine registration step may be sufficient. It preserves possibly relevant shape differences and requires a comparatively lower computation time. However, to minimize the number of undecided voxels and to avoid disappearance of small structures the application of subsequent non-rigid registration often is necessary. Here the fraction of undecided voxels in the average label images and a visual observation coupled with an entropy analysis define the choice of the algorithm and the termination of the averaging procedure. In [RBMM01] a maximum iteration number of four has been proposed. No significant decrease in entropy and the number of undecided voxels could be observed in subsequent registration steps.

Surface generation

In this step surfaces are created that separate regions with different labels. If more than two differently labelled regions meet in space, complex topological situations may arise and the resulting separating surface in general is not a manifold. In order to generate such non-manifold surfaces from label fields, we utilize a generalized surface reconstruction algorithm [HSSZ97]. It produces interfaces between all neighboring voxels of different type, nicely stitched together. Like standard marching-cubes, it works by processing each hexahedral cell (dual to the voxel grid) individually and utilizing tables that contain triangulations for different equivalence classes. An equivalence class is a set of cells with labeled corners, where the cells can be transformed into each other by discrete rotations, reflections, and/or label permutations. While in the marching-cubes case $1 + 13$ such topologically distinct classes exist, for more than 2 labels on a cell more classes may exist. The number of equivalence classes can be computed by group theoretical means [BLS05, Heg98]: employing e.g. all before mentioned symmetry groups simultaneously, for 1 to 8 different labels on the eight corners of a cell there are 1, 13, 44, 66, 43, 15, 3, 1 different classes. Employing less symmetry groups, these numbers are significantly higher. Practically we use pre-computed tables only for configurations with up to three different labels. More complex cases occur rarely and the corresponding triangulations are computed on-the-fly. Figure 4(a) depicts the outer average surfaces of the major substructures of the honeybee brain. For alternative algorithms see [BDS*03] and [BRvL*05].

The number of triangles resulting from the surface generation algorithm typically has to be reduced. Since down-sampling the labeled input data may result in loss of tiny but relevant structures, we generate the surface in high resolution and simplify it afterwards, using the method presented in [GH97]. Figures 4(b) and 4(c) show detailed views after reducing the number of triangles to $20\,\%$ and $5\,\%$, respectively, of the original number of triangles. The simplified surface finally is remeshed in Figure 4(d), using

 ISBN 978-80-86943-98-5

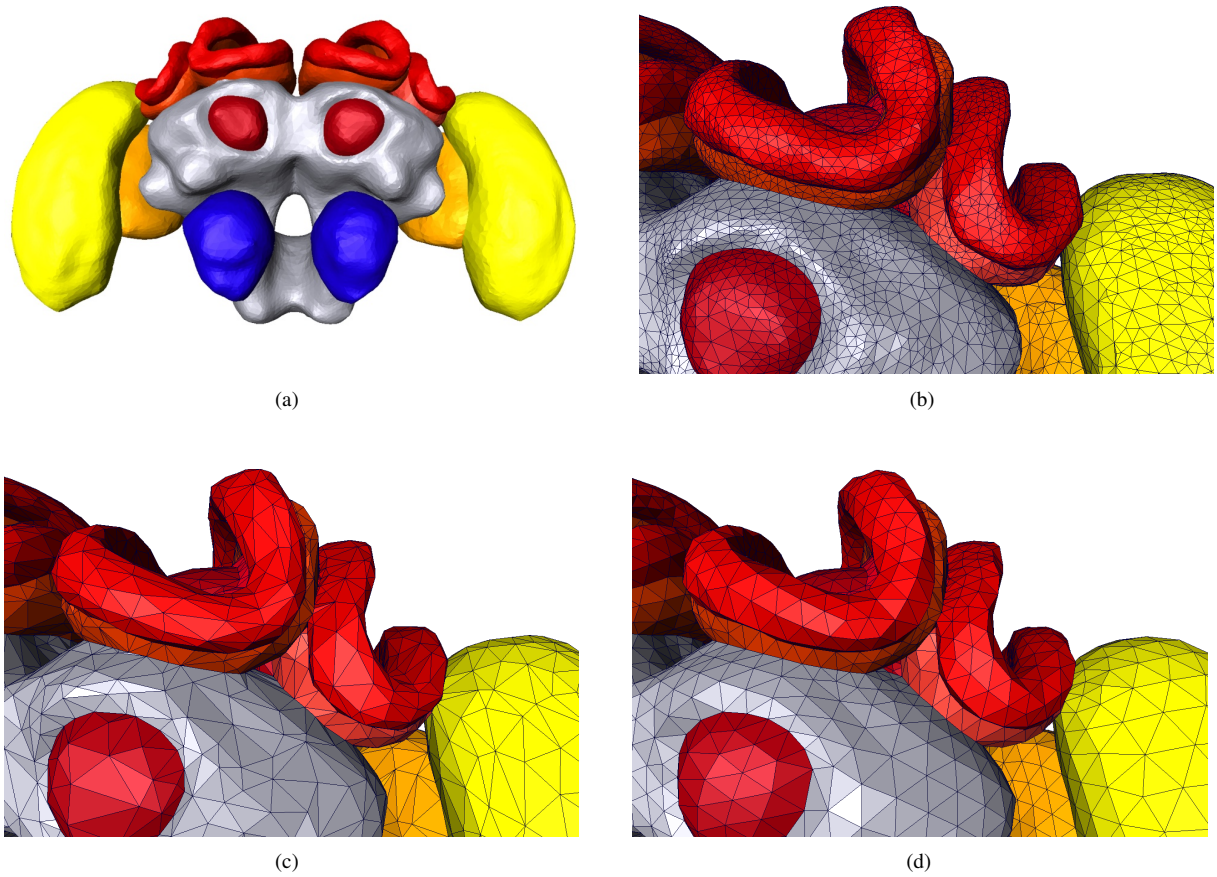


Figure 4. Generation of surfaces separating different biological structures. (a) outer average surfaces of the major substructures of the honeybee brain, (b) after reducing the number of triangles to $20\,\%$, (c) after reducing the number of triangles to $5\,\%$, and (d) after remeshing the simplified surface.

an implementation based on algorithms proposed in [SG03].

A surface editor enables the modification of the resulting surface. It provides the subdivision, collapse, removement or translation of selected triangles. Additional patches can be defined. Structures can be selected and their associated triangles can be saved as individual surfaces. These tools are mainly used for final visualization purpose. They also support further processing such as the insertion of new structures and the subdivision of existing ones.

Data integration

The resulting atlas represents a group of structures which can be regarded as a certain level of a hierarchy. On the next level substructures as for example single neurons or the glomeruli are added. The integration of additional data follows the above process. Again preprocessing, segmentation, averaging and surface reconstruction are needed in order to insert more detail into the existing atlas. We exemplary describe the approach for the integration of a projection neuron. See Figure 5 for the visualization of some work steps.

Image acquisition and preprocessing. Projection neurons (PN) are stained and imaged using confocal microscopy. Typically 2×2 tiled stacks of 350 sections each with 1024×1024 voxels are scanned. The integration of the PN also requires an acquisition of its surrounding neuropil regions which are needed for the registration. Therefore a generic counterstaining is applied and imaging is done according to the above protocol.

To avoid loss of detail usually neither resampling nor merging of the PN image stacks takes place. In contrast aiming at lower computation times in the registration step the size of the neuropil data is reduced and the stacks are composed into one data set. Neuron images usually contain only parts of the brain and subsequent steps of the pipeline have to be restricted to these parts. For this purpose several regions in the label image of the atlas have to be removed and the volume has to be cropped.

 ISBN 978-80-86943-98-5

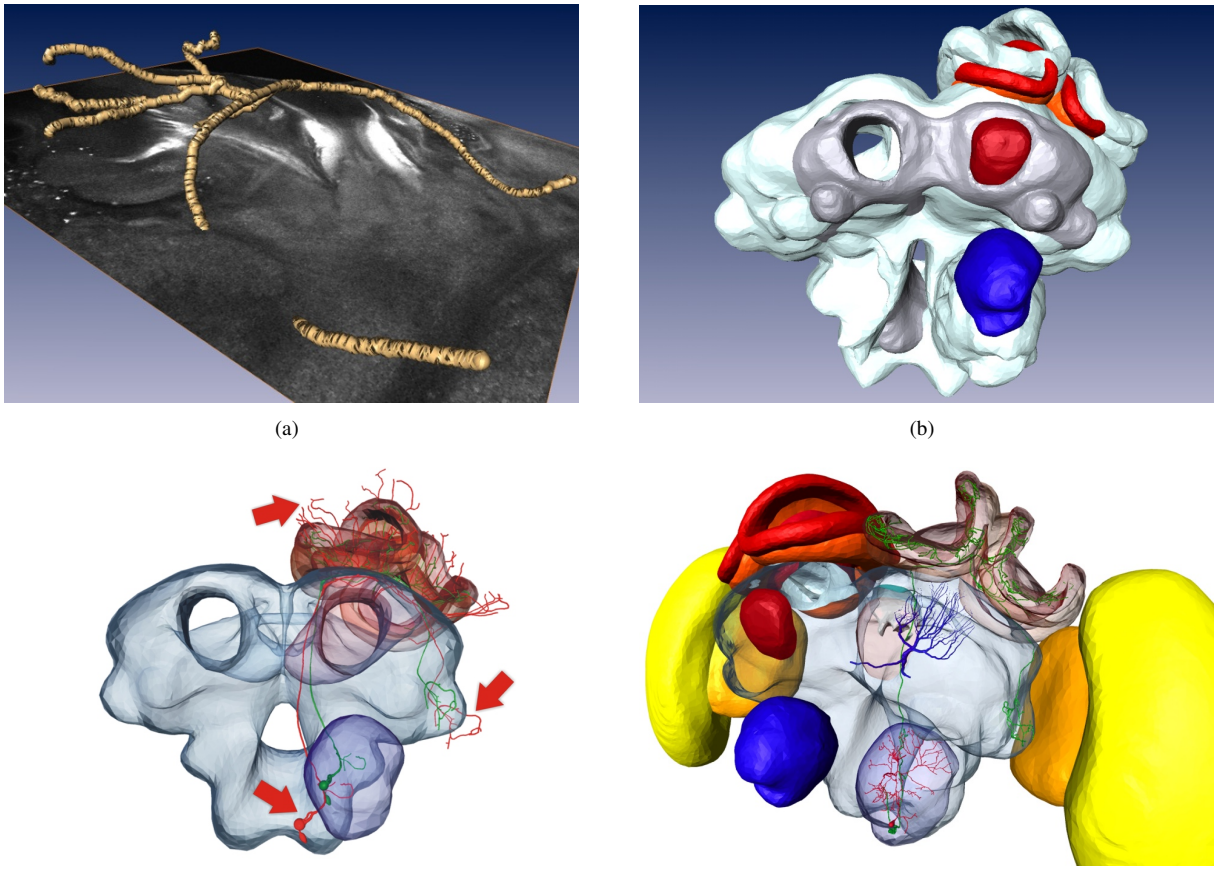


Figure 5. Neuron integration. (a) Segmentation of a neuronal part in one of the four commonly acquired image stacks. (b) Combined display of surface reconstructions of neuropils surrounding the considered neuron. The individual (light) and the standard neuropils (dark) differ in size and orientation. (c) After affine registration of the individual to the standard neuropils, the neuron (arrows) sticks out the atlas. Nonrigid registration can compensate for this inaccuracy. (d) View of a complete bee brain with a selection of integrated neurons

Neuron segmentation. The neuropils belonging to the considered neuron are segmented as described in Section Segmentation. For the segmentation of the neuron itself more specific methods are needed. Due to staining intensity variations, the existence of fine structures close to the resolution limit, and a partly insufficient resolution in z-direction fully automatic segmentation has not been achieved yet. In our pipeline promising results can be obtained by means of line extraction algorithms [SED*04, WnZ03]. Thereby the user manually sets branching or end points thus roughly tracing the graph like structures of the neuron. The centerline and thickness of the neuronal segment between these points are then detected automatically for example by minimizing an energy functional.

Registration of surrounding neuropils. For the integration of the neuron into the atlas the individual neuropils and the corresponding neuropils of the standard are registered using the same methods as for computing the standard. Afterward the resulting transformation is applied to the neuron data.

Surface reconstruction. Using a thickness-annotated graph, which represents the neuron, a surface reconstruction can be easily obtained. Our pipeline contains an implementation of the method described in [WnZ03]. Here the neuronal surface is initially described as a union of a set of spheres and cylinders. Applying an implicit surface algorithm, consistent and closed surfaces are generated that can be used for volume or surface rendering.

Computational performance

The computation time for the generation of an atlas highly depends on the number of individual data sets and the chosen registration strategy. An averaged atlas composed of ten individual brains can be created within one week. The following measurements apply to a single data set of size $650 \times 400 \times 120$. Alignment and merging of the stacks took about two min-

 ISBN 978-80-86943-98-5

utes. Manual segmentation of the neuropils required between two and six hours depending on the number of neuropils and the user's knowledge. Affine registration needed one hour. Non-rigid registration according to [RZM*02] required one hour whereas the method presented in [RBMM01] used up to 12 hours. Label voting and intensity averaging required about two minutes. The surface was generated within 38 seconds and produced $1.2 \cdot 10^6$. triangles. The simplification down to $2 \cdot 10^4$ triangles required 168 seconds and the final remeshing took about 3.5 minutes. The computation times for the steps of the integration of neuron data are similar. Speed-up can be observed during registrations. This results from the reduction of the data to only concerning neuropils. All techniques were tested on a current PC (3.0GHz Intel Pentium 4 processor).

4. APPLICATIONS

Using the proposed pipeline, a standard atlas of the honeybee brain has been generated [BRR*05]. This standard initially composed of 22 neuropils, is currently extended and refined. In [KMK*04] progress has been made in the investigation of the temporal dynamics of olfactory coding, and the neuronal wiring within different integration areas of the olfactory path. Here experimental data were acquired by stimulating the antennae with a range of odors and recording the response from single projection neurons. Individual neurons and accordant neuropils were reconstructed and collected into the framework of the standard atlas of the honeybee brain. Thus morphological as well as physiological properties of brain structures have been integrated. This work directly supports the analysis of neuronal circuits and explores the role of single neurons within the network.

The standardized atlas of the fly brain (see [RZM*02] for details) is upgraded by an average model of the *thoracic-abdominal nervous system*, which contains deeply analyzed insect motor pattern generators and most of the insect individually identified neurons. Multiple stained whole-mount preparations of the thoracic ganglia are averaged (see Figure 6 for first results). Afterward morphological details such as projections of legs, wings and halteres are added to the standard. On the next level neuropil structures are subdivided based on functional issues and finally groups of identified neurons are integrated into the atlas.

Parts of the pipeline have also been used for the generation of a standard atlas out of ten individual locust (*Schistocerca gregaria*) brains. Here neuropils belonging to the first hierarchy of the brain structure ontology have been averaged. The atlas is intended to form a spatial reference for the analysis of the central complex, its substructures, and its nervous system down to single neurons [KSH05].

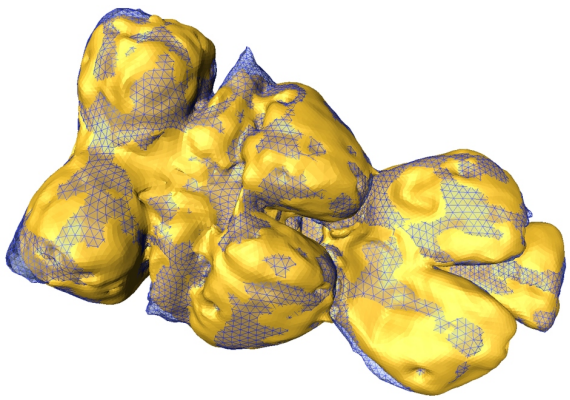


Figure 6. Atlas of fly thoracic ganglion. Shown are the average surface reconstruction of three individual fly thoracic ganglions and an overlaid individual surface reconstruction (grid)

The most time-consuming step in geometry reconstruction from image data is image segmentation. Using atlases, fully automatic segmentation of image data from an unknown subject can be performed. In [RBMMJ04] confocal microscopy images of 20 individual bee brains have been segmented by nonrigid registration to an atlas. The results were compared to a manually generated gold standard and showed a very high accuracy for most of the considered structures.

5. CONCLUSION AND FUTURE WORK

We presented a standardized pipeline for the creation of population-averaged atlases from 3D image data. The atlases are surface-based and extendable. Our pipeline consists of a sequence of steps, of which some require fine-tuned and sophisticated image and geometry processing algorithms. The proposed pipeline has been used to create a honeybee 'standard brain' and it is applied in a variety of other biological fields now.

Atlases resulting from the proposed procedure serve as spatial references, into which data from other experiments can be integrated – while maintaining their spatial context. Thus a natural organization of information is achieved that is well suited for visually supported analysis. Utilization of such reference models, augmented with empirical data of diverse provenance, greatly facilitates many scientific investigations. Examples in neurobiology are investigations of relations between brain morphology, brain function, and genetic factors.

In the future we will also associate ontological information with the spatially referenced substructures. This will provide not only a visual representation of the natural ontology of biological structures, but also will enable new navigation techniques.

 ISBN 978-80-86943-98-5

ACKNOWLEDGEMENTS

This work was supported by the German Research Foundation (DFG). The authors want to thank Torsten Rohlfing, Angela Kurylas and Jürgen Rybak for helpful information and fruitful discussions. Thanks to Jan Sahner and Tino Weinkauf for their company.

REFERENCES

[BDS*03] K.S. Bonnell, M.A. Duchaineau, D.R. Schikore, B. Hamann, and K.I. Joy. Material interface reconstruction. *IEEE Trans. Vis. Comput. Graph.*, 9(4):500–511, 2003.

[BLS05] D.C. Banks, S.A. Linton, and P.K. Stockmeyer. Counting Cases in Substitope Algorithms. *IEEE Trans. Vis. Comput. Graph.*, 10(4):371–384, 2005.

[BRR*05] R. Brandt, T. Rohlfing, J. Rybak, S. Krofczik, A. Maye, M. Westerhoff, H.-C. Hege, and R. Menzel. Three-Dimensional Average-Shape Atlas of the Honeybee Brain and Its Applications. *The Journal of Comparative Neurology*, 492(1):1–19, 2005.

[BRvL*05] Martin Bertram, Gerd Reis, Rolf H. van Lengen, Sascha Köhn, and Hans Hagen. Non-manifold Mesh Extraction from Time-varying Segmented Volumes used for Modeling a Human Heart. In K. W. Brodlie, D. J. Duke, and K. I. Joy, editors, *EUROGRAPHICS - IEEE VGTC Symposium on Visualization (2005)*, 2005.

[GH97] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[GMT00] A. Guimond, J. Meunier, and J.-P. Thirion. Average Brain Models: A Convergence Study. *Computer Vision and Image Understanding*, 77(2):192–210, 2000.

[Heg98] H.-C. Hege. Enumeration of Symmetry Classes in Mesh Generation. In M. Gross, H. Müller, P. Schröder, and Seidel H.-P., editors, *Report on Dagstuhl Seminar 9821: Hierarchical Methods in Computer Graphics*, pages 9–10, May 1998.

[HSSZ97] H.-C. Hege, M. Seebass, D. Stalling, and M. Zöckler. A Generalized Marching Cubes Algorithm Based On Non-Binary Classifications. Technical report, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1997.

[KMK*04] S. Krofczik, D. Mueller, A. Klawitter, A. Maye, and R. Menzel. Compiling functional and neuroanatomical features of individual neurons into a standard atlas of the bee brain Apis mellifera. In *FENS Abstr.*, volume 2, A193.5, 2004.

[KSH05] A.E. Kurylas, J. Schachtner, and U. Homberg. 3-D reconstruction of the central complex in the brain of the locust *Schistocerca gregaria*. In H. Zimmermann and K. Krieglstein, editors, *Proceedings of the 6th Meeting of the German Neuroscience Society 30th Göttingen Neurobiology Conference*, Neuroforum 2005, page 817. Thieme Verlag Stuttgart-New York 264B, 2005.

[PH04] Wayne Pereanu and Volker Hartenstein. Digital three-dimensional models of Drosophila development. *Current Opinion in Genetics & Development*, 14(4):382–392, 2004.

[RBMM01] T. Rohlfing, R. Brandt, C. R. Maurer, and R. Menzel. Bee Brains, B-Splines and Computational Democracy: Generating an Average Shape Atlas. In *IEEE Workshop on Mathematical Methods in Biomedical Image Analysis (MMBIA'01)*, page 187, 2001.

[RBMMJ04] Torsten Rohlfing, Robert Brandt, Randolf Menzel, and Calvin R. Maurer Jr. Evaluation of atlas selection strategies for atlas-based image segmentation with application to confocal microscopy images of bee brains. *NeuroImage*, 21(4):1428–1442, 2004.

[RMT03] D.E. Rex, J.Q. Ma, and A.W. Tofa. The LONI Pipeline Processing Environment. *Neuroimage*, 19(3):1033–1048, 2003.

[RSH*99] D. Rueckert, L. I. Sonoda, C. Hayes, D. L. G. Hill, M. O. Leach, and D. J. Hawkes. Nonrigid registration using free-form deformations: Application to breast MR images. *IEEE Transactions on Medical Imaging*, 18(8):712–721, 1999.

[RZM*02] K. Rein, M. Zoeckler, M. T. Mader, C. Gruebel, and M. Heisenberg. The Drosophila Standard Brain. *Current Biology*, 12(3):227–231, 2002.

[SED*04] Stephan Schmitt, Jan F. Evers, Carsten Duch, Michael Scholz, and Klaus Obermayer. New methods for the computer-assisted 3-D reconstruction of neurons from confocal image stacks. *NeuroImage*, 23(4):1283–98, 2004.

[SG03] Vitaly Surazhsky and Craig Gotsman. Explicit Surface Remeshing. In L. Kobbelt, P. Schröder, and H. Hoppe, editors, *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, volume 43, pages 20–30, 2003.

[SHH97] Colin Studholme, Derek L. G. Hill, and David J. Hawkes. Automated three-dimensional registration of magnetic resonance and positron emission tomography brain images by multiresolution optimization of voxel similarity measures. *Medical Physics*, 24(1):25–35, 1997.

[SWH04] Detlev Stalling, Malte Westerhoff, and Hans-Christian Hege. Amira: A Highly Interactive System for Visual Data Analysis. In Charles D. Hansen and Christopher R. Johnson, editors, *The Visualization Handbook*, chapter 38, pages 749–767. Elsevier, 2004.

[TT01] Arthur W. Toga and Paul M. Thompson. The Anatomical Record. *The New Anatomist*, 265(2):37–53, 2001.

[WnZ03] Malte Westerhoff (né Zöckler). *Efficient Visualization and Reconstruction of 3D Geometric Models from Neuro-Biological Confocal Microscope Scans*. PhD thesis, Fachbereich Mathematik und Informatik, Freie Universität Berlin, January 2003.

Towards plan-based automatic exploration of virtual worlds

Sylvain Desroche, Vincent Jolivet, Dimitri Plemenos
Xlim Laboratory, University of Limoges
83, rue d'Isle
87000 Limoges, FRANCE
sylvain.desroche@xlim.fr, {jolivet, plemenos}@unilim.fr

ABSTRACT

In order to efficiently explore unknown virtual worlds with a camera, the camera has to be as autonomous as possible, in order to take decisions whenever it is necessary. In this paper, we present a formalisation of the automatic virtual world exploration problem, together with a technique allowing on-line incremental external intelligent exploration of a virtual world by a camera. This technique is based on an evaluation function which takes into account scene's covering and quality of points of view, together with a small number of positions that should be reached by the camera and tries to make exploration plans in order to reach them using incremental determination of the next camera position.

Keywords

Virtual worlds, camera's movement, heuristic search, good view, evaluation function, incremental exploration, on-line exploration

1.INTRODUCTION

Automatic exploration of virtual worlds becomes more and more necessary because of development of virtual reality applications on computers. This technique is especially useful to give elegant solutions in two kinds of problems: scene understanding and guided visits of a virtual world. When an unknown scene is found on the Internet, the user would like, generally, to know what it represents. In order to see the scene, he (she) must display it on the screen but the chosen point of view can be poor. It is difficult to look for a good point of view for a scene displayed on the screen because the screen is bidimensional while the scene is tridimensional. The problem becomes more complex when the found scene is very complex. In such a case, a single point of view is clearly not sufficient. How to find other interesting points of view? How to use them in order to better understand the scene? Unlike the user, the computer has a complete knowledge of the scene's geometry and topology. So, it should be able to propose interesting points of view, according to some view quality criteria, and even to intelligently explore the scene, because a set of views of the scene, without a good transition between these views, does not assure a good knowledge of it.

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.

A guided visit of a virtual world is quite different. The virtual world can be a virtual representation of a museum, a particular building or a (part of a) city. Geometry and topology of the virtual world are not sufficient for the guided visit. The designer of the virtual world would like to show a set of interesting places in the world and the guided visit should take into account the designer's wishes. Again, in such a case, the computer has more information on the virtual world to visit than the visitor and it should be able to propose a guided exploration of the world, taking into account the desires of the world designer.

In section 2, the problem of virtual worlds exploration will be discussed. Authors conception of heuristic search-based scene exploration and the corresponding implemented techniques will be proposed in section 3. In section 4, plan-based improvements of heuristic search-based scene exploration will be presented, while in section 5 results, obtained by those of the proposed techniques which are currently implemented, will be presented and discussed. Section 6 will conclude the paper.

 ISBN 978-80-86943-98-5

2.THE PROBLEM OF VIRTUAL WORLDS EXPLORATION

There are two possibilities when one wishes to explore a virtual world with a camera. The first kind of exploration is *global exploration*, where the camera remains always outside the world to be explored. The second one is *local exploration*, where there is immersion of the camera inside the world.

Global exploration is based on a camera which moves around the scene and tries to give the user a good general knowledge of it by exploring first the most interesting parts of the scene. Having a global knowledge of the world to be explored is always necessary and, for a great part of scenes, global exploration gives the user sufficient knowledge of the scene.

The purpose of local exploration is a more precise knowledge of a scene, or of a part of it, by immersion of the camera inside the scene. The camera is now a part of the world and it must take into account possible obstacles which could obstruct its movement. Local exploration is rarely sufficient to give the user a good knowledge of the explored world. However, this kind of exploration could be seen as a complement of global exploration whenever global exploration does not allow to see some details.

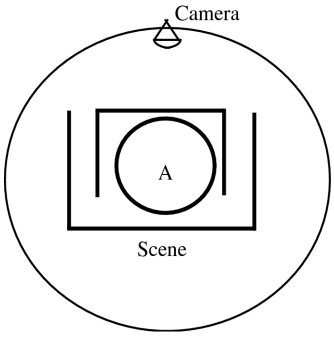


Figure 1. The camera cannot see object A

Let us consider the case of figure 1, where the scene is represented in 2 dimensions in order to make the problem easier to understand. The object A of the scene cannot be seen by a camera which moves around the scene. In such a case it is necessary to add local exploration, with immersion of the camera in the scene, in order to give the user a complete knowledge.

In some cases, the two scene exploration approaches can be combined. Let us consider the case of a virtual city. It is important to have sufficient knowledge of the general structure of the city before choosing to visit a particular place in it.

On the other hand, virtual world exploration may be *on-line* or *off-line* exploration.

On-line virtual world exploration may occur when the user discovers a new virtual world and would like to understand it as soon as possible. In such a case the camera path has to be determined in an incremental way, in real time, with only local knowledge of the world to explore.

In off-line virtual world exploration the user, having discovered a new virtual world, cannot explore it immediately. In such a case a camera path determination module may compute an optimised path for the camera, which may be used by the user for later exploration. In off-line virtual world exploration, camera path determination is more precise but it may be time consuming. This is not really a drawback, as the user decided to understand the virtual world later.

In this paper, only incremental global on-line exploration is concerned. Techniques for off-line exploration, as well as techniques to improve local exploration have been presented in other papers [Sok06a, Sok06b, Sok05, Sbe05, Jau06].

3.HEURISTIC SEARCH BASED VIRTUAL CAMERA'S MOVEMENT AROUND A SCENE

What are the main requirements for a good on-line global exploration of a virtual world with a camera? All that we need is :

- a criterion of “good view”,
- some general purpose rules on the camera’s movement,
- an evaluation function allowing the camera to choose its next position.

It is admitted that the camera’s direction of view relies the camera to the centre of the scene and that the camera’s cone of vision contains the whole scene. In the following lines we will explain how the three main requirements: “good view” criterion, evaluation function and camera’s movement rules are chosen for heuristic search-based scene exploration.

3.1.The “good view” criterion

What can be a “good” view for a virtual world? If the virtual world is completely unknown, the only possible criterion would necessarily be a geometry-based one because the geometry of the world is the only thing the computer knows. If the virtual world to be explored is sufficiently known and the purpose of the exploration is to show some important or interesting places of this world, the criterion should take into account the view of these selected places.

Some authors have worked to find a criterion of “good view” and methods to locate good points of view for a scene [Col90, Kk88, Ple91, Ple96]. Other

 ISBN 978-80-86943-98-5

authors are working to define a criterion of good point of view based on information theory [Sbe02]. A review of virtual world exploration techniques can be found in [Ple03]. More recently, some work was made to define more accurate viewpoint quality criteria [Sok06, Ple06, Lee05, Vaz03].

In this paper, we use a geometry-based criterion [Bar99], even if the proposed virtual world exploration method works with other criteria as well.

The used "good view" criterion is a combination of two notions: the notion of "number of visible details" and the notion of "importance of a detail". More precisely, considering that a scene is a set of surfaces, the good view criterion is a combination of two quantities:

- number of visible surfaces,
- area of the projected visible part of each surface.

More precisely, the importance of a point of view will be computed using the following formula:

$$I(V)=\frac{\sum_{i=1}^{n}\left[\frac{P_i(V)}{P_i(V)+1}\right]}{n}+\frac{\sum_{i=1}^{n}P_i(V)}{r}$$

where:

- I(V) is the importance of the view point V,
- Pi(V) is the projected visible area of the polygon number i obtained from the point of view V,
- r is the total projected area,
- n is the total number of polygons of the scene.

In this formula, [e] denotes the smallest integer, greater than or equal to e, for any expression e.

The problem with automatic computation of good view directions is that it is a time consuming process, hardly compatible with a real time smooth movement of a camera. So, in order to reduce the time cost of this task, we apply a computation technique using image analysis. Based on the use of the OpenGL graphical library and its integrated z-buffer, the used technique is described in this section.

If a distinct colour is given to each surface of the scene, displaying the scene using OpenGL allows to obtain a histogram which gives information on the number of displayed colours and the ratio of the image space occupied by each color.

As each surface has a distinct colour, the number of displayed colours is the number of visible surfaces of the scene from the current position of the camera. The ratio of the image space occupied by a colour is the area of the projection of the visual part of the corresponding surface. The sum of these ratios is the projected area of the visible part of the scene. In this manner, the two good view criteria are computed directly by means of an integrated fast display method.

With this technique, the same formula as above is used to compute the importance of a point of view but now:

Pi(V) is the number of pixels corresponding to the polygon number i in the image obtained from the view point V and,

r is the total number of pixels of the image (resolution of the image),

The main advantages of the used technique are the following:

- Approximated computing of the number of visible surfaces and of the projected area of each visible surface by image analysis is very easy. The total time cost of the technique is O(d) + O(m+n), where O(d) is the image computing cost, m is the number of pixels of the image and n the number of polygons (surfaces) of the scene.
- The display cost with a hardware acceleration based z-buffer is not important and a large number of polygons can be displayed very quickly.

To determine the starting point of the scene exploration process, that is, a point of view with a high probability to be the best one, the following technique is used:

The surface of the sphere where the camera is moving is divided in 8 spherical triangles. The best spherical triangle is determined by positioning the camera at each intersection point of the three main axes with the sphere and computing its importance as a point of view. The best intersection point on each axis is selected. These three points on the sphere determine a spherical triangle, selected as the best one.

The next problem to resolve is the selection the best point of view on the best spherical triangle. The following heuristic search technique is used to resolve this problem:

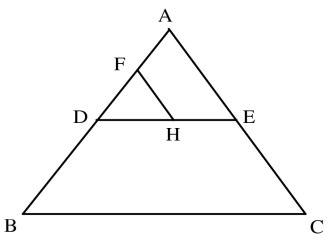


Figure 2. Heuristic search of the best point of view by subdivision of a spherical triangle

If the vertex A (figure 2) is the vertex with the best evaluation of the spherical triangle ABC, two new

 ISBN 978-80-86943-98-5

vertices E and F are chosen at the middles of the edges AB and AC respectively and the new spherical triangle ADE becomes the current spherical triangle. This process is recursively repeated until the quality of obtained points of view does not increase. The vertex of the final spherical triangle with the best evaluation is chosen as the best point of view.

3.2.The general rules guiding the camera's movement

The problem of virtual world exploration by a camera is a very interesting and useful problem but, surprisingly, very few authors have been interested in it [Fou96, Mou96, Jar98, Bar99, Bar00a, Bar00b]. In the following lines our conception of the rules guiding the camera's movement is presented.

During its movement, the camera has to explore the scene as well as possible. So, the camera's movement must apply some intuitive heuristic rules insuring a good exploration. The more important rules are the following :

- It is important that the camera moves along positions which are good points of view.
- The camera must avoid fast returns to the starting point or to already visited points.
- The camera's path must be as smooth as possible in order to allow the user to better understand the explored scene. A movement with brusque changes of direction is confusing for the user and must be avoided.

The first and second rules will be taken into account by the camera's position evaluation function which will be presented below. In order to always apply the third rule, the following technique is used.

Once the supposed best point of view has been computed on the surface of the sphere, the camera is set on this point and the movement of the camera starts by a decision for the camera's initial movement direction. Initially, all directions are plausible and eight movement directions are considered.

For each possible new position of the camera on the surface of the sphere, corresponding to a movement direction, the view direction from this point is evaluated and the chosen position is the one corresponding to the best view direction value.

After the first movement of the camera, a movement direction is defined by the previous and the current position of the camera. As blunt changes of movement direction have to be avoided, in order to obtain a smooth movement of the camera, the number of possible new directions of the camera is reduced and only 3 directions are considered for each new movement of the camera (figure 3).

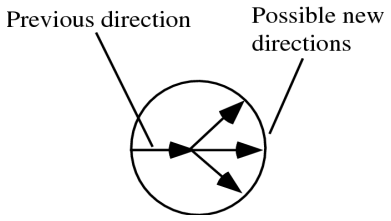


Figure 3. Only 3 directions are considered for a smooth movement of the camera

One of the three possible directions will be chosen by the position evaluation function which will integrate heuristic rules taking into account the first and second heuristic rules presented above, at the beginning of this section. This position evaluation function will be presented in the following sub-section.

3.3.The camera position evaluation function

The purpose of the virtual camera's movement around the scene is to give the user a good knowledge of the scene's properties. To do this, a maximum of interesting regions of the scene must be seen by the camera, with a minimum displacement from the starting point.

The three heuristic rules presented at the beginning of this section must be taken into account for the camera's movement. The third rule is already integrated as explained in sub-section 3.2. In the following lines we will explain how the first and second rules will be integrated in the evaluation function of the camera's position.

In order to avoid a fast return of the camera to the starting point, because of attraction due to the fact that this point determines a good view position, a weight is assigned to each position of the camera. The weight of a position is proportional to the distance of the virtual camera from the starting point. What is the distance of a point on the surface of a sphere from the starting point? We define this distance as the length of the arc of the circle obtained by the intersection of the sphere by the plane defined by the starting point, the current position of the camera and the centre of the scene.

In fact, we must consider two kinds of distance.

- The length of the path traced by the camera's movement (figure 6a). The length of this path can be computed as the sum of the chords of elementary arcs obtained by decomposition of the path.
- The minimal length arc between the starting point and the current position of the virtual camera, that is, the distance of the current position from the starting point (figure 4).

 ISBN 978-80-86943-98-5

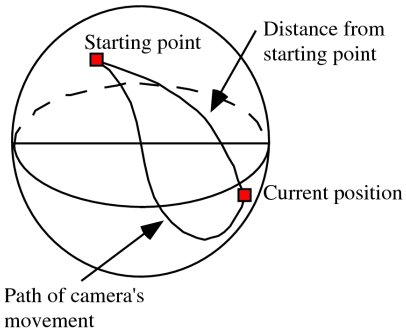


Figure 4. Distance of the current position of the camera from the starting point

In order to create a heuristic function guiding the movement of the camera, we can observe that the importance of the camera's distance from the starting point is inversely proportional to the length of the path traced by the camera.

Thus, our heuristic function computing the weight of a position for the camera on the surface of the sphere must take into account :

- The global evaluation of the camera's current position as a point of view (n_c).
- The path traced by the camera from the starting point to the current position (p_c).
- The distance of the current position from the starting point (d_c).

Finally, the main function we have chosen is the following :

$$w_c = \frac{n_c}{2}\left(1 + \frac{d_c}{p_c}\right)$$

where w denotes the weight and c the current position of the camera.

4.PRINCIPLES OF GOAL-BASED CAMERA MOVEMENT

The problem with the scene exploration techniques proposed in section 3 is that the choice of the camera's next position is based on local estimation of the position, trying only to push the camera far from the starting position. That is, there is not a real scene exploration strategy.

It would be interesting if the camera could elaborate exploration plans, with the purpose to reach interesting view positions.

Currently, we are only sure that the camera reaches a single interesting position, the starting position. With a strategy based on plan elaboration, reaching interesting positions should be considered as a permanent goal.

As a partial goal of scene exploration could be to reach an interesting position for the camera, it is possible to imagine three different plans:

1. Direct movement of the camera to the interesting point to reach, taking into account the general purpose rules on the camera's movement.
2. Progressive movement of the camera towards the interesting point to reach, taking into account the quality of the possible next positions of the camera.
3. Progressive movement of the camera towards the interesting point to reach, taking into account the quality of the possible next positions of the camera and the quality of the possible last positions before reaching the goal position.

In figure 5, P1 is the current position of the camera and P2 is an interesting position to reach. The first possibility is to choose the shortest path from P1 to P2, in order to reach P2 as soon as possible. The second possibility tries to reach P2 but the evaluation function determining the next position of the camera takes into account the quality as a point of view of every possible next position. So, the path of the camera is not always the shortest one from P1 to P2. Figure 8 illustrates above all the third possibility: here, the evaluation function is used to determine the next position of the camera, whose current position is P1, and the last position before the goal position P2, taking into account the quality as a point of view of each candidate position. If the determined next and last positions are respectively P3 and P4, the same process is applied to determine a path for the camera, where P3 is its current position and P4 the position to reach.

This determination of next and last positions can be made using a heuristic search at one level of depth or at many levels. The evaluation function computing the importance of a position must take into account, at each step, the current distance between the starting position and the position to reach.

So, the proposed goal-based strategy is made of two steps:

- Selection of a small number of interesting position for the camera. This number has to be small in order to allow on-line exploration.
- Virtual world exploration, using one of the three plans proposed above.

During exploration, only not yet seen polygons are taken into account to determine the quality of a camera position.

How to choose a small set of interesting positions to reach on the surface of the bounding sphere? An interesting position is a position being a good point of view. The chosen solution is to divide the bounding sphere in 8 spherical triangles and to compute a good

 ISBN 978-80-86943-98-5

point of view for each spherical triangle, using the technique described in section 3.1.

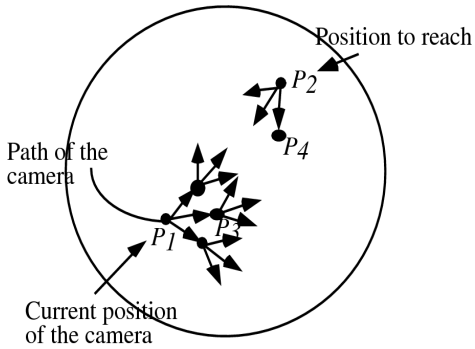


Figure 5. Plan elaboration using heuristic search from the starting position to the goal position

Now, given the current position of the camera and an interesting position to reach, the following strategy is used when applying the third, most general, exploration plan presented above. If I_P denotes the interest of the position P, n(P) the next position from position P, p(P) the previous position from the position P to reach, pv_P the quality as a point of view of the position P, d(P1, P2) the distance between positions P1 and P2 on the surface of the sphere and P_c and P_r respectivelly the current position and the position to reach, the next position of the camera can be evaluated using the following formula:

$$I_{n(P_c)} = \left(pv_{n(P_c)} + pv_{p(P_r)}\right) \frac{d(P_c, P_r)}{d\left(n(P_c), p(P_r)\right)}$$

In this formula, the next position of the camera is chosen in order to have the following conditions verified:

- The next camera position is a good point of view
- There exists a position, before the position to reach, which is a good point of view
- The distance between the next position of the camera and the position before the position to reach is minimal.

It is possible to transform the above formula in order to apply the second exploration plan presented at the beginning of this section. Here, only the next position of the camera has to be computed, the position to reach remaining the same (P_2). The new formula will be the following:

$$I_{n(P_c)} = \left(pv_{n(P_c)} + pv_{p(P_2)}\right) \frac{d(P_c, P_2)}{d\left(n(P_c), P_2\right)}$$

The main advantage of the second exploration plan is that exploration is faster. On the other hand, the control of the camera trajectory is less precise.

5.IMPLEMENTATION OF EXPLORATION TECHNIQUES AND RESULTS

A great part of the techniques presented in this paper and allowing intelligent exploration of virtual worlds has been implemented. In figure 6, one can see an exploration of a scene representing an office. The movement of the camera is represented by the dotted line. This exploration is based on the method described in section 3. The camera's motion is smooth and the scene is well understood after a relatively short exploration. In this kind of exploration, the camera's next position is computed in incremental manner but the exploration is not goal-oriented, as there are not goals to reach.

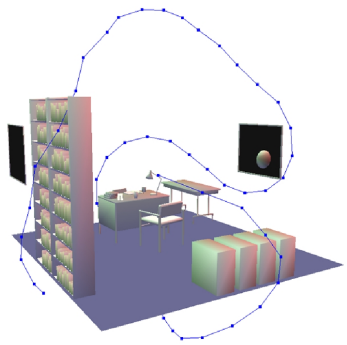


Figure 6. Exploration of an office by the techniques explained in the previous section

The first exploration plan described in section 4 was implemented but it is not very interesting, as smooth movement of the camera is not guarantied. The second exploration plan produces interesting results, which will be commented in this section. The third exploration strategy is under implementation.

We have chosen to comment on results of the proposed plan-based method, obtained with two scenes: The office scene, already seen in figure 6, and a sphere with 6 holes, containing various objects inside. This second test scene is very interesting because most of the scene details are visible only through the holes.

Four cases were studied, corresponding to variants of the second formula of section 4:

1. Only the distance between the current position of the camera and the position to reach is considered. The viewpoint qualities of positions to reach are not taken into account. See figure 7.
2. Only the viewpoint qualities of positions to reach are considered, that is, the next point to reach is the closest to the current camera position. See Figure 8.

 ISBN 978-80-86943-98-5

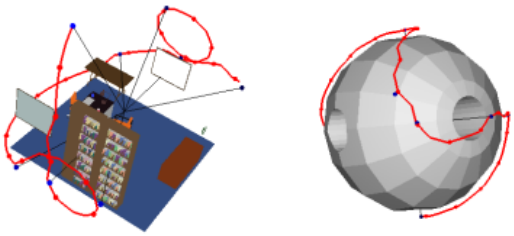


Figure 7. Only distance to the next position to reach is taken into account

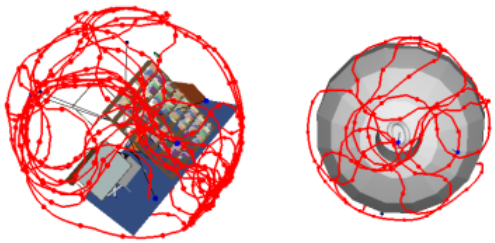


Figure 8. Only viewpoint qualities of the positions to reach is taken into account

3. Both distance and viewpoint quality of the position to reach are considered, with equal weights (ratio v*iewpoint quality/distance* = 1). See Figure 9.

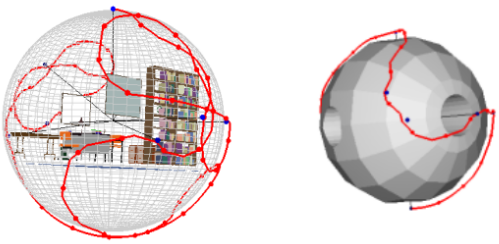


Figure 9. Both distance and viewpoint quality are equally considered

4. Both distance and viewpoint quality of the position to reach are considered, with the distance weight equal to twice the viewpoint quality weight (ratio *viewpoint quality/distance* = 1/2). See Figure 10.

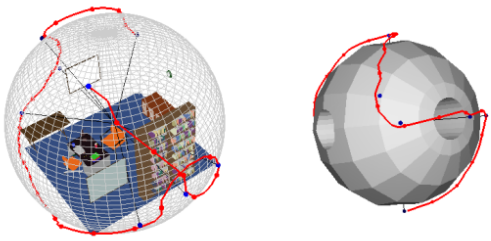


Figure 10. The distance is twice more important than viewpoint quality

One can see that the fourth case gives the best results with both test scenes. With the sphere test scene, the fourth case gives results very close to those of the first case (distance only), as it can be seen in figure 11.

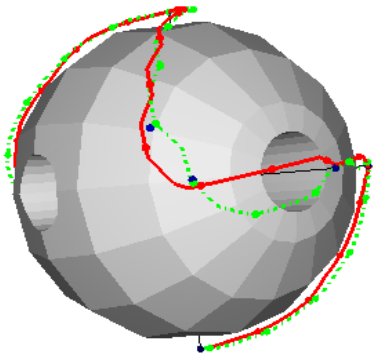


Figure 11. The distance-based path(in green) is very close to the best path(in red)

Graphic representations of results (Figures 12 and 13) show that, in all cases, more than 80% of the scene is seen when the last of the initially defined positions is reached by the camera. This proves that the chosen method for creating the small initial set of positions to reach is well adapted to the purpose of the plan-based automatic exploration. The number of positions to reach of the initial set is generally small (usually 8), so that the added extra time cost is negligible.

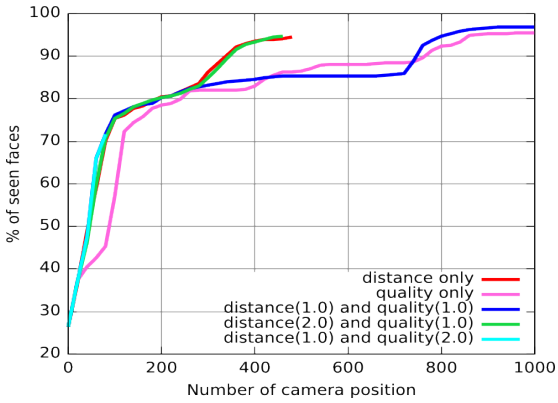


Figure 12. Graphic representation of results with the Office test scene

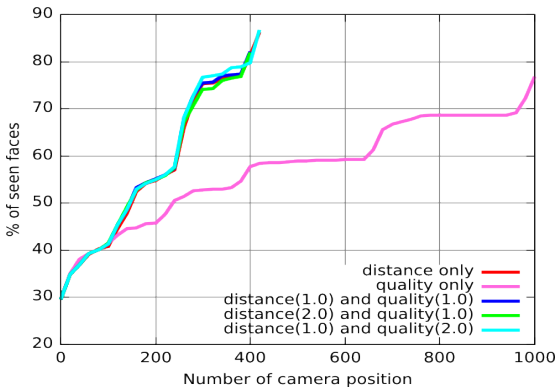


Figure 13. Graphic representation of results with the Sphere test scene

 ISBN 978-80-86943-98-5

6.CONCLUSION AND FUTURE WORK

In this paper we have presented techniques allowing external intelligent exploration of a scene by a virtual camera. In these techniques the camera is supposed to move on the surface of a sphere bounding the scene.

The presented techniques perform incremental determination of the next position of the camera and take into account both viewpoint quality and positions that should be reached by the camera and try to elaborate exploration plans in order to reach these positions when they try to determine the next camera position.

The study of obtained results shows that the distance of the current camera position from the position to reach is often more important than the viewpoint quality of the current position. In our opinion this fact can be explained by the mode of choice of the initial set of positions to reach. As they are chosen to be good viewpoints and well distributed on the surface of the surrounding sphere, it is natural to discover a big part of the virtual world by visiting the positions to reach only according to their distance from the current position. The optimal ratio viewpoint quality/distance is 1/2. Obtained trajectories with this ratio are always interesting and the used methods are not time consuming. In implemented techniques, the movement of the camera is smooth and the chosen camera paths interesting.

Taking into account that the implemented plan-based technique uses the second of the three possible camera's movement plans(see section 4), one can hope that implementation of the more accurate third plan will give much more interesting results. Our next work will be implementation of the remaining more elaborated plan of the plan-based technique. We are also trying to find an objective measure in order to evaluate the different scene exploration techniques.

7.REFERENCES

[bar99] BARRAL P., DORME G. PLEMENOS D., Visual understanding of a scene by automatic movement of a camera, GraphiCon'99, Moscow (Russia), August 26 - September 3, 1999.

[bar00a] BARRAL P., DORME G. PLEMENOS D., Scene understanding techniques using a virtual camera, Short paper, Eurographics'2000, Interlagen (Switzerland), August 20 - 25, 2000.

[bar00b] BARRAL P., DORME G. PLEMENOS D., Intelligent scene exploration with a camera, 3IA'2000, Limoges (France), May 3 - 4, 2000.

[Col90] COLIN C., Automatic computing of good views of a scene, MICAD'90, Paris (France), February 1990 (in French).

[Fou96] FOURNIER L., Automotive vehicle and environment modeling. Machine learning algorithms for automatic transmission control. PhD Thesis, Limoges (France), January 1996 (in French).

[Jar98] JARDILLIER F., LANGUENOU E., Screen-Space Constraints for Camera Movements: the Virtual Cameraman., Computer Graphics Forum, Volume 17, number 3, 1998.

[Jau06] JAUBERT B., TAMINE K., PLEMENOS D., Techniques for off-line exploration using a virtual camera. 9th International Conference 3IA'2006, May 23-24, 2006, pp. 31-44. Dimitri PLEMENOS (Editor).

[Kk88] KAMADA T., KAWAI S., A Simple Method for Computing General Position in Displaying Three-dimensional Objects, Computer Vision, Graphics and Image Processing, 41 (1988).

[Lee05] LEE C. H., VARSHNEY A., JACOBS D. W., Mesh saliency. ACM TOG, 24(3), 659-666 (2005).

[Mou98] MOUNIER J.-P., The use of genetic algorithms for path planning in the field of declarative modeling., International Conference 3IA'98, Limoges (FRANCE), April 28-29 1998.

[Ple91] PLEMENOS D., Contribution to the study and development of techniques of modelling, generation and display of scenes. The MultiFormes project., Professorial dissertation, Nantes (France), November 1991 (in French).

[Ple96] PLEMENOS D., BENAYADA M., Intelligent display in scene modelling. New techniques to automatically compute good views., GraphiCon'96, Saint Petersburg, July 1996.

[Ple03] PLEMENOS D., Exploring virtual worlds: current techniques and future issues. STAR report. GraphiCon'2003, Moscow, September 5-10, 2003.

[Ple06] PLEMENOS D., SOKOLOV, D., Intelligent scene display and exploration. International Conference GraphiCon'2006, Novosibirsk (Russia), June 26 - July 1, 2006.

[Sbe02] SBERT M., FEIXAS M., RIGAU J., CASTRO F., VAZQUEZ P.-P., Application of information theory to computer graphics, 3IA'2002, Limoges (France), May 14 - 15, 2002.

[Sbe05] SBERT M., PLEMENOS D., FEIXAS M., GONZALEZ F., Viewpoint Quality: Measures and Applications. Computational Aesthetics in Graphics, Visualization and Imaging, Girona, May 2005, pp.1-8.

[Sok05] SOKOLOV D., PLEMENOS D., Viewpoint quality and scene understanding. The 6th International Eurographics Symposium on Virtual Reality, Archaeulogy and Cultural Heritage (VAST'2005), Pisa (Italy), November 8-11, 2005, pp. 67-73.

[Sok06a] SOKOLOV D., PLEMENOS D., TAMINE K., Methods and data structures for virtual world exploration., "The Visual Computer",Volume 22,no 7,pp. 506-516, 2006.

[Sok06b] SOKOLOV D., PLEMENOS D., TAMINE K., Viewpoint quality and global scene exploration strategies. International Conference in Computer Graphics and Applications (GRAPP'2006), Setubal (Portugal), February 2006, pp.184-191.

[Vaz03] VAZQUEZ P. P. and SBERT M., Perception-based illumination information measurement and light source placement. Lecture Notes in Computer Science, 2003 (Proc. of ICCSA'2003).

 ISBN 978-80-86943-98-5

Memory-Efficient Sliding Window Progressive Meshes

Pavlo Turchyn
University of Jyväskylä, Finland
pturchy@cc.jyu.fi

Abstract

Progressive mesh is a data structure that encodes a continuous spectrum of mesh approximations. Sliding window progressive meshes (SWPM) minimize data transfers between CPU and GPU by storing mesh data in static on-GPU memory buffers [For01]. The main disadvantages of the original SWPM algorithm are poor vertex cache usage efficiency, and big resulting datasets. Connectivity-based algorithm [KT04] achieves a good vertex cache coherence but it does not address the problem of high memory utilization. In this paper, we describe estimates for the size of memory buffers, and describe methods to reduce the index datasets. We achieve 20% reduction due to the use hierarchical data structures (clustered patches); further reduction (50% or more) is possible if one can optimize connectivity of input meshes, or is willing to decrease the number of mesh approximations stored in the progressive mesh.

Keywords: level of detail, GPU, progressive meshes, independent sets, face clusters.

1 INTRODUCTION

Progressive mesh data structure encodes a continuous spectrum of mesh approximations [LRC*02, pp.70–72]. Efficient hardware rendering of the progressive meshes is discussed in a number of papers. Many proposed methods do not depend on mesh simplification procedure; they address only creation and maintenance of GPU-optimal sequences of triangles, such as triangle strips, while permitting the use of an arbitrary simplification algorithm (see e.g. [LRC*02, pp.168–169], [Ste01, BG01, SP03]). These methods process the data on CPU, and then transfer the results to GPU via generally slow procedure. Since the use of progressive meshes implies frequent LOD changes, the data transfers will be a bottleneck in most cases.

The problem of CPU-GPU data transfers is addressed in some view-dependent schemes, which partition a mesh into the set of triangle patches, e.g. [YSG05, CGG*05]. Each patch is updated once per several frames, thus the cost of updating the entire model is distributed over time. However, these schemes impose significant overheads when rendering smaller meshes (e.g. <100K triangles) with multiple instances, which are found in many applications, such as videogames.

Sliding window progressive meshes (SWPM) allow both efficient rendering of the instantiated meshes, and fast geomorphing that exploits frame-to-frame coherence. SWPM simplification algorithm partitions a mesh into the set of patches, which are sequentially replaced with simpler triangulations in a *view-independent* fashion. Unlike the view-dependent schemes, SWPM use very small patches (about six triangles each), so the typical difference between two subsequent mesh approximations is two triangles. The patches are stored in static on-GPU memory buffers that tend to be relatively large; this problem is augmented by the fact that simplification with higher quality requires even larger buffers.

This paper is organized as follows. Sec. 2 describes preliminaries about SWPM algorithm. Sec. 3 contains contribution of this paper; we derive estimates for the size of memory buffers, and propose methods to reduce memory consumption.

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

2 SLIDING WINDOW ALGORITHM

Simplification operators. In this paper, we consider *half-edge collapse* and *vertex removal* operators. Let $M = (V,T)$ be a 2-manifold triangle mesh, where $V = \{v_i | v_i \in \mathbb{R}^3\}$ is a set of vertices, and $T = \{\triangle_\ell | \triangle_\ell = (v_i, v_j, v_k), v_i, v_j, v_k \in V\}$ is a set of triangles. Vertex removal operator $\mathscr{R}(v_i)$ deletes from the mesh the vertex v_i and all incident triangles, and then re-triangulates the resulting hole with a triangulation algorithm. Half-edge collapse operator $\mathscr{H}(v_i, v_j)$ replaces the vertex v_i with the vertex v_j in all triangles of the mesh M and then removes the resulting degenerate edge and corresponding degenerate triangles from the mesh. One can interpret $\mathscr{H}$ as a vertex removal with the specific triangulation algorithm.

Mesh simplification algorithm builds a sequence of mesh approximations $\mathbf{M} = \{M_0, M_1, \ldots, M_n\}$ from the initial mesh M_0. An approximation $M_{k+1} \in \mathbf{M}$ is the result of applying the operator $\mathscr{H}(v_i, v_j)$ (or $\mathscr{R}(v_i)$) with some $v_i, v_j \in V$ to $M_k \in \mathbf{M}$ (the vertices v_i and v_j are chosen according to the value of an *error metric*).

Sliding window algorithm. Sliding window scheme was described in [For01] as an efficient method for rendering of instantiated geometry. Consider a mesh $M = (V,T)$. Let $S(v_i)$ be the set of the triangles incident

 ISBN 978-80-86943-98-5

to the vertex $v_i \in V$, i.e. $S(v_i) = \{\triangle_\ell | \triangle_\ell \ni v_i, \triangle_\ell \in T\}$ ($S(v_i)$ is also called *star* of the vertex v_i). We say that a subset $P(v_i)$ of the set T is a *patch*, if $S(v_i) \subseteq P(v_i) \subseteq T$; here v_i is the *inner vertex*.

Let $\mathbf{P} = \{P_1, P_2, \ldots, P_n\}$ be a list of patches, where $P_1 \cup P_2 \cup \cdots \cup P_n = T$, and any triangle belong to only one patch. The inner vertices of such patches belong to an independent set. We define a list of *simplified patches* $\mathbf{Q} = \{Q_1, Q_2, \ldots, Q_n\}$. Each *simplified patch* $Q_i \in \mathbf{Q}$ is obtained by applying a simplification operator to the corresponding patch $P_i \in \mathbf{P}$, i.e. $\mathscr{R}(v_j) : P_i(v_j) \longmapsto Q_i$, or $\mathscr{H}(v_j, v_k) : P_i(v_j) \longmapsto Q_i$, where $v_k \in P_i(v_j)$. We stress that a simplification operator removes the inner vertex only. Thus, we can replace any P_i with Q_i while maintaining conformity (i.e. no cracks or T-joints) of the mesh.

Let $||$ denote the lists concatenation operator. $A||B$ builds the list that contains the elements of A followed by the elements of B, i.e. $\{a_1, \ldots, a_n\}||\{b_1, \ldots, b_m\} = \{a_1, \ldots, a_n, b_1, \ldots, b_m\}$. We construct the *index buffer* (which is a memory buffer that holds triangles connectivity information) from the list of patches $\mathbf{P}$ and the corresponding list of simplified patches $\mathbf{Q}$ as follows

$$I(\mathbf{P}, \mathbf{Q}) = P_1||P_2||\cdots||P_n||Q_1||Q_2||\cdots||Q_n. \quad (1)$$

The important outcome of obtaining such I is that we can render a LOD simply by choosing the appropriate window in I, e.g. the original mesh M_0 is rendered using the window $P_1||P_2||\cdots||P_n$, M_1 is rendered using the window $P_2||P_3||\cdots||Q_1$, M_2 is rendered using the window $P_3||P_4||\cdots||Q_1||Q_2$, etc. No memory update is required to change mesh LOD; the index buffer may be shared among all mesh instances.

Geomorphing. The sliding window scheme enables efficient geomorphing to smooth LOD transitions. Consider $M_0 \to M_1$ transition; only the patch P_1 changes and actually needs morphing. In the case of an arbitrary $M_i \to M_j$ transition ($i < j$), the patches $P_{i+1}, P_{i+2}, \ldots, P_j$, which require morphing, form a continuous subsequence that may be rendered separately from the patches, which do not need morphing. One can expect that the majority of patches may be processed without morphing because in practice i is ofen close to j due to the frame-to-frame coherence. Since the morphing via linear interpolation effectively doubles the amount of data transferred per vertex, the sliding window scheme reduces by a factor of two the memory bandwidth consumed by geomorphing.

Vertex caching. One have to construct the list of patches $\mathbf{P}$ in such a way that any two subsequent patches $P_i \in \mathbf{P}$ and $P_{i+1} \in \mathbf{P}$ share via a vertex cache as many vertices as possible [KT04]. Thus, the sequence of simplified patches $\mathbf{Q}$ is also cache-optimized. Therefore, vertex cache hit rate remains high for any mesh LOD. Alternatively, one can optimize patches order for efficient Z-culling [NBS06].

3 SWPM INDEX DATASET

Consider a mesh that contains 60K triangles and 30K vertices. Below we show that SWPM index dataset takes about 2Mb of memory; on the other hand, only 0.7Mb buffer is required for storing two 3-vectors (e.g. position and normal) per vertex. Thus, reduction of the index dataset, which is described in this section, may be the main tool to reduce the size of geometric data for memory-limited applications.

Cascade simplification. Suppose, we have obtained the list of patches $\mathbf{P}_0$ (and the corresponding list of simplified patches $\mathbf{Q}_0$) on the mesh M_0. We construct SWPM index buffer $B_0 := I(\mathbf{P}_0, \mathbf{Q}_0)$. Let M_n be the simplest mesh approximation that can be rendered using the buffer B_0. It is possible to derive the limits (7) for the number of triangles M_n contains. However, many application may require simpler approximations. Thus, the simplest approximation M_n must be simplified further; we have to construct the buffer $B_1 := I(\mathbf{P}_1, \mathbf{Q}_1)$ where M_n serves as an initial mesh. Generally, we construct the buffers $B_0, B_1, \ldots, B_z$, where the buffer B_z contains a sufficiently coarse mesh.

We denote $|B_i|$ the number of triangles, which are stored in the SWPM buffer B_i. Let β be the total number of triangles in all SWPM buffers

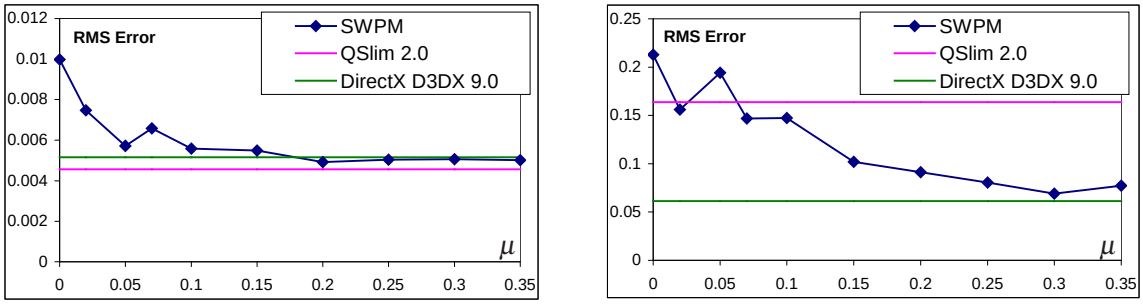
$$\beta := \sum_{i=0}^{z} |B_i|. \quad (2)$$

Under certain assumptions it is possible to derive a simple estimate (10) for the minimum value of β. The mesh-dependent parameter $\overline{d}$ is a mean degree of the vertices of the *maximum independent set*. Using this estimator, one can expect that SWPM index dataset is at least 3–5 times larger than the index buffer of the original mesh (since typically $\overline{d} \in [4;6]$).

The factor of error control. A drawback of the simplification algorithms that remove independent vertices is their limited ability to adapt mesh triangulation according to the original surface geometry. Consider the mesh shown in Fig. 5a. Conservatively, we start simplification by reducing complexity of the flat upper part of the mesh while leaving the curved bottom part untouched as shown in Fig. 5c. However, we delete vertices uniformly from the top and the bottom when removing the maximum independent set.

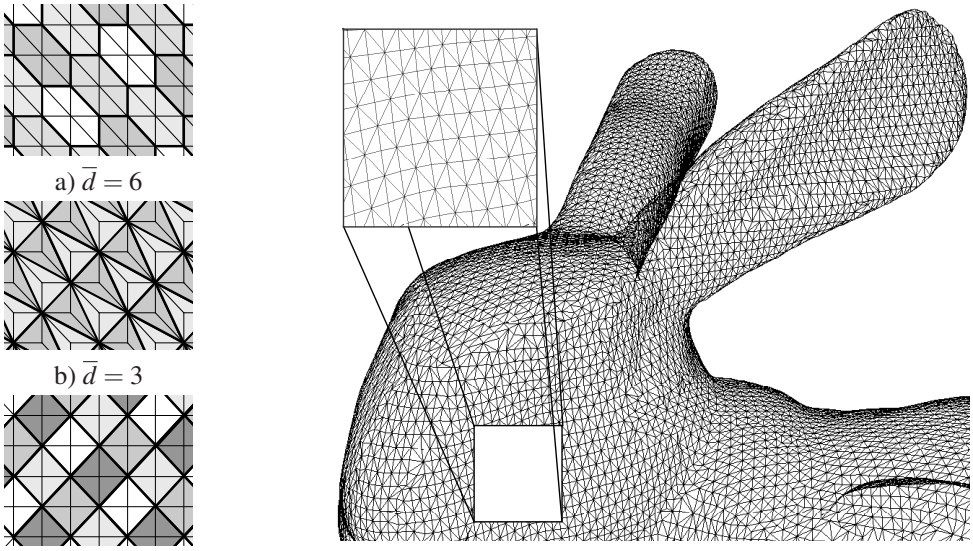
One possible solution is as follows. First, we sort all vertices of the mesh $M = (V, T)$ according to the error induced by the removal of each vertex. Then we prohibit removal of k vertices with the largest error; any such vertex may not become an inner vertex. Let $\mu = k/|V|$ be a user-defined fraction of the vertices that may not be removed. Typical relations between μ and the simplification error are shown in Fig. 1. In practice, setting $\mu \in [0.05; 0.2]$ is often sufficient to achieve plausible simplification results. Similar argumentation concerning large meshes can be found in [FS01].

 ISBN 978-80-86943-98-5



a) *t72* mesh is simplified from 2450 to 640 triangles b) *cessna* mesh is simplified from 7446 to 580 triangles

Figure 1: Relationship between the simplification error and the fraction of non-removable vertices μ



a) $\overline{d} = 6$

b) $\overline{d} = 3$

c) $\overline{d} = 4$

d) optimized *Bunny* ($\overline{d} \approx 4$)

Figure 2: a) the patches on regular triangulation b) the result of applying Optimize procedure to the mesh shown in Fig. 2a c) the result of Optimize when $\overline{d}$ is limited to $\overline{d} \geq 4$ d) the latter procedure is applied to Bunny mesh (the model is courtesy of Stanford Computer Graphics Lab)

When $\mu > 0$, the following estimator tends to be relatively accurate

$$\beta = |T_0|(\overline{d} - 1)(\mu + 1). \tag{3}$$

Thus, one can expect that SWPM index dataset is 4–7 times larger than the index buffer of the original mesh (since typically $\overline{d} \in [4;6]$ and $\mu \in [0.05;0.2]$).

3.1 Mesh connectivity optimization

It follows from (10) and (3) that the size of index dataset is proportional to the mean degree of the inner vertices. At the same time, we notice that a greedy search for the maximum independent set, which is based on the minimum-degree heuristics, tends to include the vertices of lower degree to the independent set. Therefore, one way of minimizing $\overline{d}$ is creating low-degree vertices in the mesh artificially, e.g. at the preprocessing step that precedes the simplification process. One may perform such preprocessing by applying a sequence of *edge swap* operators. *Edge swap* operator $\mathscr{E}(v_k, v_\ell, v_m, v_n)$ replaces two triangles $\triangle_i = (v_k, v_\ell, v_m)$ and $\triangle_j = (v_\ell, v_k, v_n)$, which share the edge (v_k, v_ℓ), with the triangles $\triangle'_i = (v_k, v_n, v_m)$ and $\triangle'_j = (v_\ell, v_m, v_n)$. As a result, degrees of the vertices v_k and v_ℓ decrease.

One simple strategy is applying the operator $\mathscr{E}$ iteratively until the degree of any vertex cannot be reduced any further.

function *Optimize*(M)
  $X \leftarrow \{v_1\}$, $Y \leftarrow \{v_n\}$, where v_n is connected to v_1
  for each $v_i \in X$
    find the edge (v_i, v_m) such that $v_m \in Y$
    for each edge (v_i, v_j) starting with (v_i, v_m) in *ccw* order
      if $v_j \notin Y$ and $v_k, v_\ell \notin X$ **then**
        $M \leftarrow \mathscr{E}(v_i, v_j, v_k, v_\ell)M$
        $X \leftarrow X \cup \{v_j\}$
        $Y \leftarrow Y \cup \{v_k, v_\ell\}$
  return M

For example, *Optimize* produces the triangulation shown in Fig. 2b from the triangulation shown in Fig. 2a. It decreases the mean degree $\overline{d}$ down to three, which is the lowest value. However, it is easy to see that such optimization also dramatically worsens mesh

 ISBN 978-80-86943-98-5

quality in planar mesh regions since the majority of resulting triangles would have obtuse angles. Therefore, one have to implement a check for planarity; if a curvature in a vertex is below some threshold, then degree of this vertex should not be decreased below four. Such modification of the algorithm is straightforward; we omit its details. The modified optimization produces the mesh with *diamond* pattern (see Fig. 2c) from the regular triangulation shown in Fig. 2a. Optimized *Bunny* model is shown in Fig. 2d.

3.2 Clustered patches

In this subsection we describe an improvement to the algorithm of SWPM dataset creation. In order to reduce the size of index dataset, we reuse the simplified patches to construct a new set of the patches via clustering.

We demonstrate the idea of clustering on a simple example. We define the list of seven patches $\mathbf{P} = \{P_1, P_2, \ldots, P_7\}$ and the corresponding list of simplified patches $\mathbf{Q} = \{Q_1, Q_2, \ldots, Q_7\}$. The simplified patches are shown in Fig. 3b. The simplified patches contain 28 triangles; the numbering of triangles is shown in Fig. 3a. By X denote the fragment of the index buffer $X := Q_1||Q_2||\cdots||Q_7 = \{\triangle_1, \triangle_2, \ldots, \triangle_{28}\}$. The fragment X is shown in Fig. 4a; it is located at the end of index buffer $I(\mathbf{P}, \mathbf{Q})$.

Using a clustering algorithm, we construct a new list of patches $\mathbf{P}^h = \{P_1^h, P_2^h, \ldots, P_n^h\}$, where each patch $P_i^h \in \mathbf{P}^h$ is a concatenation of one or more consecutive simplified patches of the list $\mathbf{Q}$. In our example, we define two patches $P_1^h = Q_1||Q_2||Q_3$ and $P_2^h = Q_4||Q_5||Q_6||Q_7$ (see Fig. 3c), thus $\mathbf{P}^h = \{P_1^h, P_2^h\}$. Then, we build the corresponding list of simplified patches $\mathbf{Q}^h = \{Q_1^h, Q_2^h\}$, and the index buffer $I(\mathbf{P}^h, \mathbf{Q}^h)$ in the same way as we do it for the basic sliding window algorithm. The index buffer starts with the fragment $Y := P_1^h||P_2^h = Q_1||Q_2||\cdots||Q_7 = \{\triangle_1, \triangle_2, \ldots, \triangle_{28}\}$ (see Fig. 4b), which coincides with the fragment X located at the end of index buffer $I(\mathbf{P}, \mathbf{Q})$ (see Fig 4a). The memory buffer $D := I(\mathbf{P}, \mathbf{Q})||I(\mathbf{P}^h, \mathbf{Q}^h) = P_1||P_2||\cdots||P_7||X||Y||Q_1^h||Q_2^h$ contains a large amount of duplicated data because $X \equiv Y$. However, instead of D we can use the buffer $D^h := P_1||P_2||\cdots||P_7||X||Q_1^h||Q_2^h$ that contains both $I(\mathbf{P}, \mathbf{Q})$ and $I(\mathbf{P}^h, \mathbf{Q}^h)$. The buffer D^h is smaller than D because the repeated subsequence is omitted.

The goal of clustering is building the lists of patches in such a way, that two or more index buffers are partially coincident, thus we can construct their compact memory representation, which is smaller than simple union of the buffers. As shown in Appendix, the size of SWPM index dataset decreases as the number of patches increases. In order to increase the number of patches, we introduce a notion of *linked patches*.

In our example, define patches $P_1^\ell = Q_1||Q_2||\{\triangle_9 \in Q_3\}$, $P_2^\ell = (Q_3 \setminus \{\triangle_9\})||Q_4||\{\triangle_{17} \in Q_5\}$, and $P_3^\ell = (Q_5 \setminus \{\triangle_{17}\})||Q_6||Q_7$ (see Fig. 3d). We call P_i^ℓ and P_{i+1}^ℓ *linked patches* when they both include triangles of the same simplified patch. Linked patches P_1^ℓ and P_2^ℓ share the simplified patch Q_3; P_2^ℓ and P_3^ℓ share Q_5. We build the list of patches $\mathbf{P}^\ell = \{P_1^\ell, P_2^\ell, P_3^\ell\}$ and its corresponding list of simplified patches $\mathbf{Q}^\ell = \{Q_1^\ell, Q_2^\ell, Q_3^\ell\}$. Again, a fragment $P_1^\ell||P_2^\ell||P_3^\ell$ of the index buffer $I(\mathbf{P}^\ell, \mathbf{Q}^\ell)$ (see Fig. 4c) coincide with the fragment $X \subset I(\mathbf{P}, \mathbf{Q})$, thus we can construct more compact memory representation of these index buffers.

Linked patches have to be subsequent in the list $\mathbf{P}^\ell$. As a counterexample, consider such list $\mathbf{P}^\ell = \{P_2^\ell, P_1^\ell, P_3^\ell\}$, where linked patches P_2^ℓ and P_3^ℓ are not subsequent. Any continuous triangles subsequence of $I(\mathbf{P}^\ell, \mathbf{Q}^\ell)$ cannot coincide with Q_5; thus, $I(\mathbf{P}^\ell, \mathbf{Q}^\ell)$ and $I(\mathbf{P}, \mathbf{Q})$ cannot partially coincide, and the advantage of the use of clustering is lost.

Agglomerative clustering algorithm. The input of clustering algorithm is the set of simplified patches $\mathbf{Q}_0$ and the simplest mesh $M_n = (V_n, T_n)$ produced as a result of simplification of the mesh M_0. The only clustering criterion is maximization of the number of clusters. We can formulate this problem as a maximum independent set problem on a graph G^h. Each vertex v_i of the independent set is the inner vertex of a patch $P^h(v_i)$, which is formed by the simplified patches that include the vertex v_i. By $G(T)$ denote the connectivity graph of the triangulation T. The graph G^h is initialized with $G(T_x)$; then for each simplified patch $Q_j \in \mathbf{Q}_0$ we find the additional edges, which connect all previously unconnected vertices of Q_j (i.e. the edges required to make $G(Q_j)$ complete), and insert them to G^h.

In order to find the maximum independent set on G^h, one may use the greedy search described in [SvK97]). First, we mark high-error vertices (the number of such vertices is controlled by the user-defined parameter μ), then we delete all marked vertices by decrementing the degrees of their neighbors. While unmarked vertices remain, we choose a vertex of lowest degree for the independent set and delete it and its neighbors by marking them and decrementing the degrees of their neighbors.

Additionally, we implement a single-step lookahead to search for linked patches. For example, suppose the patch $P_1^\ell = Q_1||Q_2||Q_3$ is found in the first iteration of the greedy search. Then we attempt to find a patch P_2^ℓ, which shares Q_1, Q_2, or Q_3 with P_1^ℓ. This is a local search since the linked patches are in the neighborhood of P_1^ℓ. If several candidates are found, we choose the one of the lowest inner vertex degree. If the linked patch P_2^ℓ is found, we attempt to find a patch P_3^ℓ, which is linked to P_2^ℓ, etc. When no more linked patches can be constructed, we proceed with the basic greedy search for a next independent vertex.

 ISBN 978-80-86943-98-5

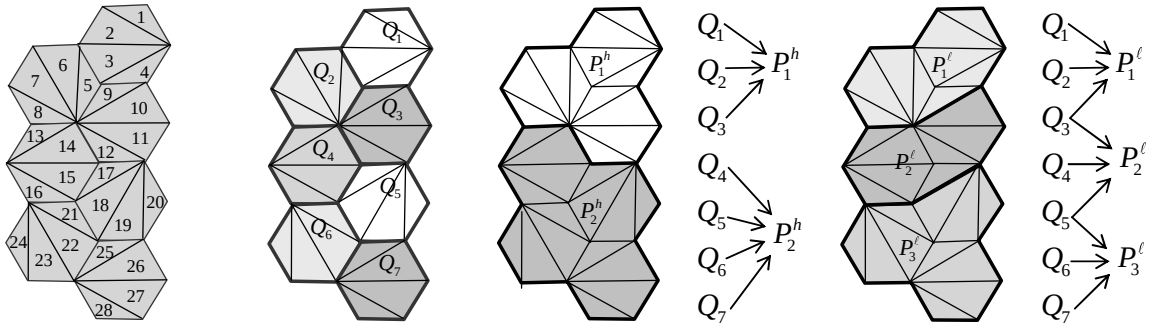


Figure 3: The use of clustering to construct new patches from the set of simplified patches

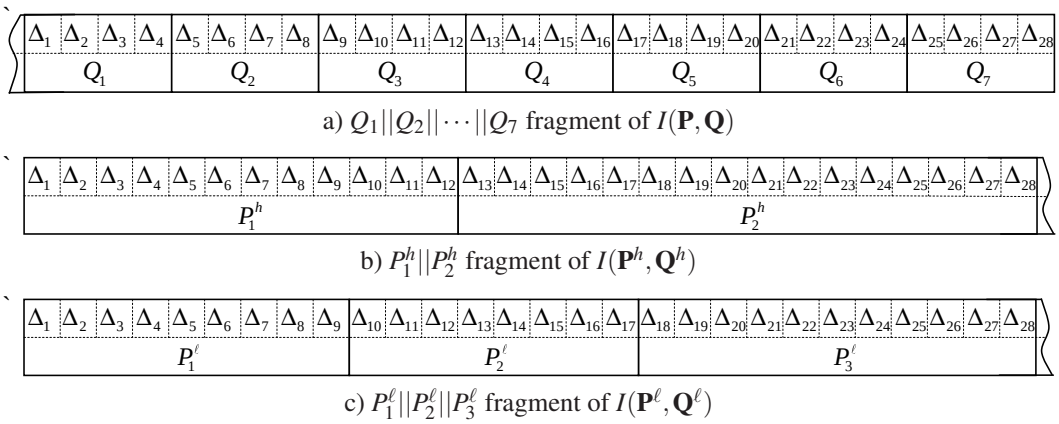


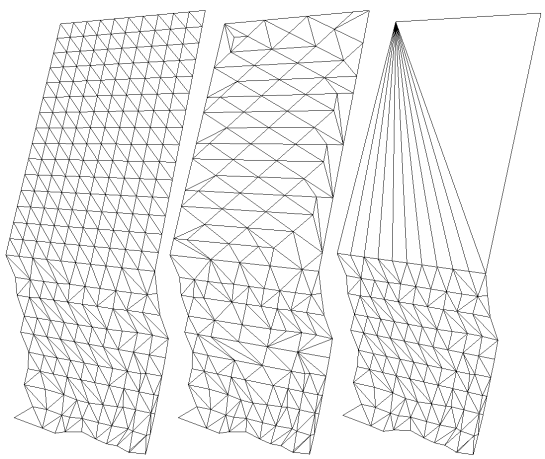
Figure 4: Coinciding fragments of the index buffers from the example shown in Fig. 3

3.3 Patches with multiple inner vertices

Consider the patch $P_1^h = Q_1||Q_2||Q_3$ and its corresponding simplified patch Q_1^h. As we simplify the mesh, the following sequence of replacements occurs: $P_1 \to Q_1$, $P_2 \to Q_2$, $P_3 \to Q_3$, $P_1^h \to Q_1^h$. To reduce the size of index dataset, we may choose not to store Q_1, Q_2, and Q_3; we directly replace $P_1||P_2||P_3 \to Q_1^h$ during rendering. Thus, we can reduce the size of index data at the expense of storing fewer approximations. In order to describe such process we have to extend our previous definition of a patch.

We define a *patch with multiple inner vertices* $P^*(v_i, v_j, \ldots, v_k)$ as a list of triangles, such that $S(v_i) \cup S(v_j) \cup \cdots \cup S(v_k) \subseteq P^*$. We construct the corresponding simplified patch Q^* by sequentially applying simplification operator $\mathscr{R}$ (or $\mathscr{H}$), which removes the inner vertices, e.g. $Q^* = \mathscr{R}(v_i)\mathscr{R}(v_j)\cdots\mathscr{R}(v_k)P^*(v_i, v_j, \ldots, v_k)$. Given the list of patches $\mathbf{P}^* = \{P_1^*, P_2^*, \ldots, P_n^*\}$ and its corresponding list of simplified patches $\mathbf{Q}^* = \{Q_1^*, Q_2^*, \ldots, Q_n^*\}$, we build the index buffer $I(\mathbf{P}^*, \mathbf{Q}^*)$ and render the mesh approximations as described in Sec. 2.

A patch with several inner vertices is simplified more aggressively than a patch with single inner vertex. Thus, the patches with multiple inner vertices are particularly useful for the overtesselated models that contain a lot of redundant coplanar triangles. Suppose the entire flat upper part of the mesh shown in Fig. 5a is included into a single patch P^*. Fig. 5c shows that we avoid simplifying the curved bottom part by simplifying P^* only, and reduce the size of index data. A planarity measure (e.g. *dual quadratic metrics* [GWH01]) may be used to select the patches that are suitable for having more than one inner vertex.



a) b) $\lambda = 5.7$ c) $\lambda = 2.4$

Figure 5: a) original mesh, b) simplification using single inner vertex per patch ($\mu = 0.25$), c) the entire flat upper part is simplified as a patch with several inner vertices

One method to build the patches with multiple inner vertices is the clustering algorithm; we simply discard

 ISBN 978-80-86943-98-5

the patches P^h as described above. This can be done in run-time when there is not enough videomemory available in the system. Alternatively, a mesh partitioning algorithm may be used to directly obtain the patches with several inner vertices.

Generally, the patches with multiple inner vertices provide to the user a way to trade the number of mesh approximations in SWPM for the smaller index dataset.

4 NUMERICAL RESULTS

Rendering performance. One popular method for rendering of instantiated geometry is the *discrete LOD* method where several discrete mesh approximations are created using a simplification software, such as QSlim. Then each approximation is optimized for the efficient hardware processing (we use *NVTriStrip* package for this purpose). In run-time, the system selects the most appropriate mesh approximation to display for an object. We compare discrete LOD and SWPM in an application that renders 7000 diffusely lit instances of *t72* mesh on the system with Athlon64 2.2Ghz and GeForce 7800GTX. The application is not shader-bounded since we use relatively simple vertex shader (diffuse lighting only). Two 3-vectors (position and normal) are defined per vertex. In this test, the camera moves with varying velocity over the scene that contains regularly spaced mesh instances. The results are shown in Fig. 6. LOD selection function is the same for all methods. When geomorphing is not in use, performance of the discrete LOD (a) and SWPM (b) is the same because their rendering paths coincide. However, geomorphing causes a notable slowdown of the discrete LOD rendering (c) since it doubles the bandwidth required for the vertex data (note that its impact may be less serious for shader-bounded applications since geomorphing is arithmetically inexpensive). Geomorphing in the sliding window scheme (d) is not expensive due to the use of frame-to-frame coherence (as discussed in Sec. 2). This advantage disappears when the frames are highly incoherent. For example, a fast camera movement lasts for 8 seconds starting from 9th second of the timedemo; both methods (c) and (d) demonstrate similar performance during this period. The curve (e) shows performance of CPU-based progressive meshes implementation (as found e.g. in DirectX D3DX 9.0) that streams the list of indices from CPU to GPU each frame; the application generates approx. 500Mb of indices per second, which makes such scheme impractical.

Index dataset reduction. Table 1 demonstrates the results of dataset reduction techniques. The parameter λ in (9) relates the size of resulting SWPM index dataset to the number of triangles in the original mesh. In Table 1, the coefficient λ corresponds to the progressive mesh obtained using the basic sliding window algorithm; dataset reduction techniques decrease this value according to the percentages (a)–(d). The values

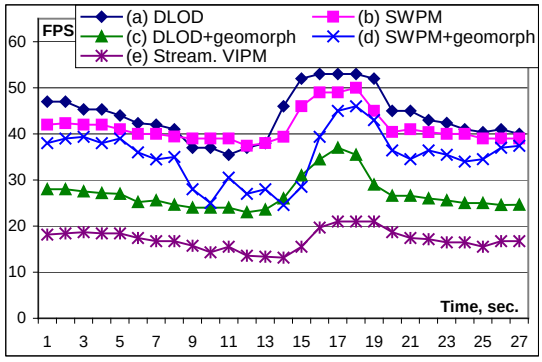


Figure 6: Rendering speed of several LOD algorithms in frames per second over time

Basic SWPM			Index dataset reduction, %			
Mesh	$\lvert T_0 \rvert$	λ	(a)	(b)	(c)	(d)
Regular	49920	5.4	39%	18%	41%	58%
TIN	77632	5.2	28%	16%	31%	51%
Hugo	16374	5.1	23%	18%	29%	51%
Bunny	69451	5.4	19%	15%	27%	49%
Cessna	7446	5.3	9%	19%	27%	53%
Cow	5804	5.5	11%	18%	26%	49%
t72	2450	3.8	12%	19%	29%	50%
Elf	1274	4.6	6%	20%	31%	57%

Table 1: Index dataset reduction via connectivity optimization (a), clustering (b), the combination of optimization and clustering (c), or the combination of optimization and the patches with three inner vertices per patch (d)

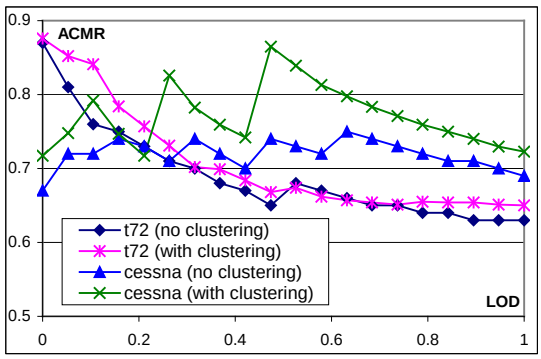


Figure 7: ACMR (vertex cache size is 16 entries) for all resolutions of t72 and cessna SWPM created with or without clustering

(d) are obtained by discarding some simplified patches (as discussed in Sec. 3.3) from the progressive meshes obtained for the case (c); this normally produces the patches with approximately three inner vertices.

Following [BG01], we use the *average number of cache misses per triangle* (ACMR) as a measure of vertex cache usage efficiency. The clustering is an additional factor (in addition to vertex cache coherence), which is taken into account when the order of patches is determined. Thus, the clustering may cause an increase in ACMR in some cases (see Fig. 7).

 ISBN 978-80-86943-98-5

Simplification quality. We use *quadratic error metrics* [LRC*02, pp.133–134] and half-edge collapse operator to create SWPM. Fig. 8 shows *t72* mesh simplified with help of various algorithms. Generally, although SWPM is an *ad-hoc* algorithm for efficient rendering, it typically produces mesh approximations comparable with the results of conservative mesh simplification algorithms.

CONCLUSIONS

Proposed methods for datasets reduction extend applicability of SWPM algorithm for memory-limited applications. The use of SWPM may be an interesting extension to batch-processing view-dependent LOD schemes, such as [YSG05, CGG*05].

References

[BG01] BOGOMJAKOV A., GOTSMAN C.: Universal rendering sequences for transparent vertex caching of progressive meshes. In *Proceedings of Graphics Interface 2001* (2001), Watson B., Buchanan J. W., (Eds.), pp. 81–90.

[CGG*05] CIGNONI P., GANOVELLI F., GOBBETTI E., MARTON F., PONCHIO F., SCOPIGNO R.: Batched multi triangulation. In *Proceedings IEEE Visualization* (2005), IEEE Computer Society Press, pp. 157–164.

[For01] FORSYTH T.: Comparison of vipm methods. In *Game Programming Gems 2* (2001), DeLoura M., (Ed.), Charles River Media, pp. 363–376.

[FS01] FRANC M., SKALA V.: Parallel triangular mesh decimation without sorting. In *SCCG'01: Proceedings of the 17th Spring conference on Computer graphics* (Washington, DC, USA, 2001), IEEE Computer Society, p. 22.

[GWH01] GARLAND M., WILLMOTT A., HECKBERT P. S.: Hierarchical face clustering on polygonal surfaces. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics* (New York, NY, USA, 2001), ACM Press, pp. 49–58.

[KT04] KOROTOV S., TURCHYN P.: Topology-driven progressive mesh construction for hardware-accelerated rendering. *Computer Graphics and Geometry (Internet) 6*, 3 (2004), 100–119.

[LRC*02] LUEBKE D., REDDY M., COHEN J., VARSHNEY A., WATSON B., HUEBNER R.: *Level of Detail for 3D Graphics*. Computer Graphics and Geometric Modeling. Morgan Kaufmann, 2002.

[NBS06] NEHAB D., BARCZAK J., SANDER P. V.: Triangle order optimization for graphics hardware computation culling. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2006), ACM Press, pp. 207–211.

[SP03] SHAFAE M., PAJAROLA R.: Dstrips: Dynamic triangle strips for real-time mesh simplification and rendering. In *Proceedings Pacific Graphics Conference* (2003).

[Ste01] STEWART A. J.: Tunneling for triangle strips in continuous level-of-detail meshes. In *Proceedings of Graphics Interface* (2001), Watson B., Buchanan J. W., (Eds.), pp. 91–100.

[SvK97] SNOEYINK J., VAN KREVELD M. J.: Linear-time reconstruction of delaunay triangulations with applications. In *ESA '97: Proceedings of the 5th Annual European Symposium on Algorithms* (London, UK, 1997), Springer-Verlag, pp. 459–471.

[YSG05] YOON S.-E., SALOMON B., GAYLE R.: Quick-vdr: Out-of-core view-dependent rendering of gigantic models. *IEEE Transactions on Visualization and Computer Graphics 11*, 4 (2005), 369–382. Member-Dinesh Manocha.

A THE BOUNDS FOR SWPM INDEX DATASETS

Consider the buffer $B_0 := I(\mathbf{P}_0, \mathbf{Q}_0)$, where the approximations $M_0, M_1, \ldots, M_n$ are stored. First part of the buffer, which contains the non-simplified patches from the list $\mathbf{P}$, forms the set of triangles T_0 of the initial mesh $M_0 = (V_0, T_0)$, i.e. $T_0 = P_1 \cup P_2 \cup \cdots \cup P_n$. Remaining part of the buffer, which contains the simplified patches from the list $\mathbf{Q}$, forms the triangles of the approximation $M_n = (V_n, T_n)$, where $T_n = Q_1 \cup Q_2 \cup \cdots \cup Q_n$. Then, the number of triangles in the buffer B_0 is a sum of the number of triangles in T_0 and the number of triangles in T_n, i.e. $|B_0| = |T_0| + |T_n|$. In what follows we assume that the meshes are 2-manifolds without holes. Thus, the operator $\mathscr{H}$ (or $\mathscr{R}$) removes exactly two triangles from a patch, so

$$|T_n| = \sum_{i=1}^{n} |Q_i| = \sum_{i=1}^{n} (|P_i| - 2) = |T_0| - 2n, \tag{4}$$

where n is the number of patches. Hence, one have to maximize n in order to minimize $|B_0|$. It follows from the definition of a patch that maximization of n requires solving the *maximum independent set* problem on a mesh connectivity graph (the set of inner vertices is a *maximum independent set*). This is a classical *NP-hard* problem. There exist polynomial-time algorithms for finding its approximate solution with a good accuracy. One greedy algorithm for planar graphs is described in [SvK97]. The algorithm's lower bound for the size of independent set is $n > |V_0|/6$. It follows from the Euler's relation that

$$n > \frac{|V_0|}{6} > \frac{|T_0|}{12} \tag{5}$$

On the other hand, we note that each patch contains at least three triangles, so

$$n \leqslant \frac{|T_0|}{3}. \tag{6}$$

Combining (4)–(6), we obtain the bounds

$$\frac{1}{3}|T_0| \leqslant |T_n| < \frac{5}{6}|T_0|. \tag{7}$$

Using M_n as an initial mesh, we construct another SWPM buffer. Denote this buffer B_1. The simplest triangulation within the buffer B_1 may not be sufficient for the application. Thus, one has to continue constructing SWPM buffers $B_2, B_3, \ldots, B_z$ until the required mesh complexity is achieved. Our main interest is estimating the parameter β in (2), which determines the amount of memory required for storing these buffers.

The approximation M_n serves as an initial mesh when we construct the buffer B_1. Let $M_m = (V_m, T_m)$ be the simplest

 ISBN 978-80-86943-98-5

approximation of the buffer B_1. Following the same reasoning, which is used to obtain (7), we have

$$\frac{1}{3}|T_n| \leqslant |T_m| < \frac{5}{6}|T_n|.$$

In order to obtain the upper bound for $|T_m|$, for $|T_n|$ we take the upper bound provided by (7). Then

$$|T_m| < \frac{5}{6}|T_n| < \left(\frac{5}{6}\right)^2 |T_0|,$$

so

$$|B_1| = |T_n| + |T_m| < \frac{5}{6}|T_0| + \left(\frac{5}{6}\right)^2 |T_0|.$$

Generally,

$$|B_i| < \left(\frac{5}{6}\right)^i |T_0| + \left(\frac{5}{6}\right)^{i+1} |T_0|.$$

By summing up the inequalities for every $|B_i|$, we obtain upper bound for β

$$\beta < |T_0| + 2\left(\frac{5}{6}\right)|T_0| + \cdots + 2\left(\frac{5}{6}\right)^{z-1}|T_0| + \left(\frac{5}{6}\right)^z |T_0|.$$

We can interpret the right hand side as a converging geometric series. This is justified since in practice z is relatively big. Thus, we conclude that

$$\beta < 11|T_0|.$$

It is possible derive the lower bound for β using a similar reasoning. Finally, we arrive to the following inequalities

$$2|T_0| \leqslant \beta < 11|T_0|. \tag{8}$$

It is natural to express β as a size of initial mesh triangulation multiplied by a coefficient

$$\beta = |T_0|\lambda, \tag{9}$$

where $\lambda \in [2;11)$ according to (8). We derive the bounds for the coefficient λ under relatively general assumptions about the input mesh. However, memory-limited applications require more precise estimation of λ for specific meshes. Thus, we have to derive an estimate for λ that exploits properties of a particular mesh.

A *degree* of a mesh vertex is the number of triangles incident to this vertex. Let $\overline{d}$ be a mean degree of the mesh vertices of *maximum independent set*. The value of $\overline{d}$ depends mainly on the structure of triangulation. We assume that

1. the choice of simplification operators and their parameters preserves the structure of initial triangulation, so $\overline{d}$ is approximately the same for any approximation M_i;
2. the size of a patch is approximately equal to the degree of its inner vertex, thus $\overline{d}$ is a mean patch size.

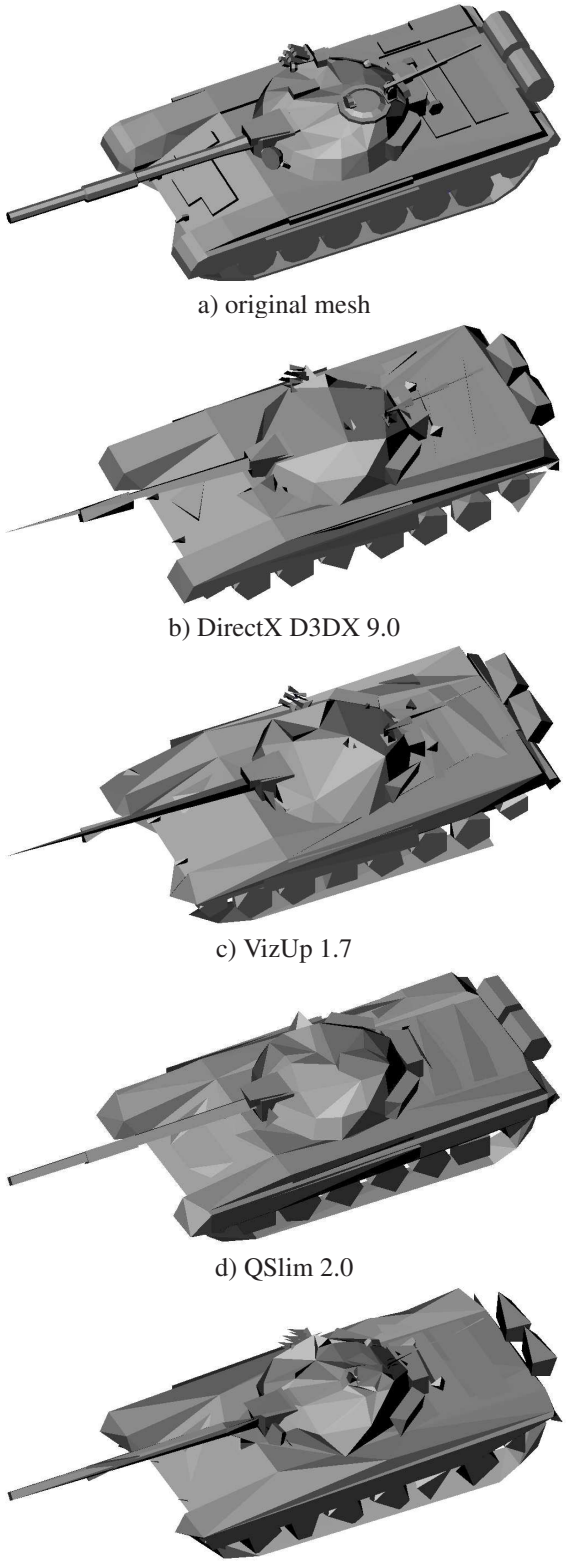
Under these assumptions, the number of patches is a ratio of total number of triangles to $\overline{d}$. Applying this relation to (4), it is easy to show that

$$|T_n| = |T_0|(1 - 2/\overline{d}),$$
$$|B_0| = |T_0| + |T_n| = |T_0|(2 - 2/\overline{d}),$$
$$|B_i| = |T_0|(2 - 2/\overline{d})(1 - 2/\overline{d})^i.$$

Again, β is expressed as a converging geometric series

$$\beta = |T_0|(\overline{d} - 1). \tag{10}$$

Thus, the size of resulting dataset is proportional to $\overline{d}$, which is a mesh-dependent parameter.



a) original mesh

b) DirectX D3DX 9.0

c) VizUp 1.7

d) QSlim 2.0

e) SWPM ($\mu = 0.2$)

Figure 8: t72 mesh is simplified from 2450 to 640 triangles using different simplification algorithms. As discussed in Sec. 3, the parameter μ determines tradeoff between the size of SWPM index dataset and the simplification quality; relations between μ and RMS simplification error are shown in Fig. 1.

 ISBN 978-80-86943-98-5

Hardware-accelerated point-based rendering of surfaces and volumes

Eduardo Tejada Tobias Schafhitzel Thomas Ertl

Universität Stuttgart
{eduardo.tejada|tobias.schafhitzel|thomas.ertl}@vis.uni-stuttgart.de

ABSTRACT

In this paper, we present a fast GPU-based algorithm for ray-tracing point-based models, which includes an efficient computation of secondary and shadow rays, contrary to previous work which supported ray-surface intersections for primary rays only. Volumetric effects are added to the models by means of scattered data interpolation in order to combine point-based surface and volume rendering in the same scene. This allows us to obtain effects such as refraction within volumetric objects. The flexibility of our method is demonstrated by combining shadows, textured objects, refraction and volumetric effects in the same scene comprised uniquely by point sets.

Keywords: Point-based; Point Set Surfaces; Graphics hardware; Ray-tracing.

1 INTRODUCTION

Mesh-based methods have become the standard *de facto* for a wide range of applications. The advances in mesh-based techniques stimulated the development of hardware specific to meshes, which in turn encouraged the development of new and better mesh-based methods. However, during the last five years, the increase in the computation power of todays processors and the flexibility provided by graphics hardware, allowed the arising of a new set of point-based techniques, which represent an alternative to mesh-based methods. By working directly with point clouds, the processing is done on the raw data without the need for any intermediate representation and generally artificial connectivity relations[18]. Since the only information needed is the geometry given by the points, larger models can be stored in memory. Also, re-sampling during extreme deformations and topology changes is supported efficiently.

Among the point-based surface approximation methods developed in the last years are Point Set Surfaces (PSS)[6;30] As stated by Adamson and Alexa[2], PSS have clear advantages over other point-based surface approximation techniques when ray-tracing is to be used, namely the locality of the computations, the possibility of defining a minimum feature size and the fact that the surface is smooth and manifold. Beside the inherent implications of these three characteristics, the second advantage can be exploited when computing the intersection of the ray with the surface, whilst the last one makes CSG operations feasible.

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.

However, ray-tracing a PSS on current standard PC processors is not interactive. To tackle this problem, implementations on special hardware have been presented. Wald and Seidel[28] implemented a highly-tuned ray-tracer that used Adamson and Alexa's implicit surface definition[3] on an Opteron PC, making use of SIMD operations and a number of acceleration techniques originally developed by the authors in the last years for mesh models. Tejada et al.[27] presented a GPU implementation to compute ray-surface intersections of primary with the PSS proposed by Alexa et al.[6].

Thus, based on the work by Tejada et al., we chose to render the PSS on the GPU exploiting the flexibility and new functionality offered by currently available commodity graphics hardware. As mentioned above, this approach includes a fast computation of the intersection of primary rays with the PSS. However, intersections with secondary and shadow rays are computed by means of a brute force scheme. Therefore, computing efficiently these intersections with the PSS on the GPU is our main concern in this work. The resulting algorithm, which we present in this paper, allows us to work with nested surfaces and to include self-shadowing in the scene. In order to include volumetric effects in the scene, we exploited the texture-based data structure we use for ray-tracing the PSS, to interpolate the scalar at the sampling points. This way, we are able to create scenes, where both surface rendering of the PSS and volumetric effects are combined. Furthermore, if the points belonging to a material boundary of the volumetric object are properly identified, we are able to com-

bine refraction with volume rendering to obtain interesting visual effects.

2 RELATED WORK

As mentioned before, during the last years point-based methods have gained a new popularity. Modeling tools and techniques for creating and editing point-based models have been developed [12;20;30]. Also deformable bodies simulation [16], animation [19;21], and rendering algorithms [1;3;9;25;29] have been proposed. Alexa et al. [6] proposed a resampling technique for generating dense samplings in order to cover the image space consistently by simply projecting the points onto the screen. The 'surfaces' obtained this way are therefore known as Point Set Surfaces (PSS). PSS is a surface approximation method that, as stated in the previous section, presents important advantages for ray-tracing. PSS have been further studied by Alexa et al. [5], by Adamson and Alexa [3;4], and by Amenta and Kil [7;8], who noted that PSS are an special case of *extremal surfaces* [15]. Amenta and Kil [7] also studied important characteristics about the domain of point set surfaces.

In general, PSS methods make use of classical differential geometry results to ensure consistent local representation through polynomial approximations. In this respect, Zwicker et al. [30] use linear minimization functions based on Weighted Least-Squares (WLS) to define the local polynomial functions, whilst Alexa et al. [6] employ a non-linear strategy based on Moving Least-Squares (MLS) to define a local coordinate system on which a local polynomial approximation of the surface is calculated using WLS. This approach was based on the work by Levin [14] on the approximation power of the MLS method. Point-based techniques that make use of implicit functions whose zero set is guaranteed to generate surface representations were also proposed [3;8;13]. Also, guarantees for homeomorphical approximations to the original object based on this implicit function have been presented [11].

Ray-tracing PSS has been previously addressed both on the CPU [2;28] and the GPU [27]. However, as already mentioned, ray-tracing a PSS on the CPU is prohibitively slow. Therefore, Wald and Seidel [28] presented a ray-caster, implemented on the Opteron PC, for the implicit surface definition by Adamson and Alexa [3]. Also, a GPU-based technique was proposed [27], that efficiently computes the intersection of primary rays with the surface. Although, the advantage of this implementation is the use of commodity hardware, a description of how to compute the intersection between secondary and shadow rays with the PSS efficiently is not presented. Ray-tracing on the GPU has been addressed in a more general sense by Purcell [23].

3 POINT SET SURFACES

Given a set of points $p_i \in \mathbb{R}^3, i \in \{1,...,N\}$, Alexa et al. [6] define the corresponding Point Set Surface S_p as the set of stationary points for the projection procedure described in the following. First a local orthonormal coordinate system is built upon the plane $H(n, r+tn)$, where the normal n to the plane and the scalar t are obtained by minimizing

$$\sum_{p_i \in P(r)} < p_i - r - tn, n >^2 \theta(\|p_i - r - tn\|) \quad (1)$$

with respect to both n and t. Here $\theta(x)$ is a non-negative, monotonically decreasing function and $P(r)$ is the set of points in the neighborhood of r. For the weighting function $\theta(x)$, authors frequently make use of a Gaussian $\theta(x) = e^{-\frac{x^2}{h^2}}$, where h represents the local level of detail of the object (feature size). Note that the constraint $\|n\| = 1$ must be observed during the minimization. Also note that, as a result of this minimization, H will be a plane close to r and quasi-tangent to S_p.

Once the local coordinate system is built over H with origin in $q = r + tn$, a local bivariate polynomial approximation g to the surface is computed using the points in $P(r)$. If q_i is the projection of p_i onto H, (x_i, y_i) is the local representation of q_i in the local system and $f_i = n \cdot (p_i - q)$ (the height of p_i over H), then the coefficients of this polynomial are found by minimizing

$$\sum_{p_i \in P} (g(x_i, y_i) - f_i)^2 \theta(\|p_i - q\|). \quad (2)$$

Then, the projection $\mathscr{P}(r)$ onto S_p is defined by $\mathscr{P}(r) = q + g(0,0)$.

The minimization defined in Equation 1 is a non-linear optimization problem. To solve it, we use an iterative process that descends to the next local minima. This iterative process consists of two steps: (1) minimize with respect to t and (2) minimize with respect to n. In order to start the process a first approximation of n must be computed using covariance analysis [18]. Then, the minimization with respect to t is performed using the *Brent with derivative* algorithm [22], and the resulting t is then fixed for minimizing with respect to n. Since minimizing with respect to n in $\mathbb{S}^2$ (the space of directions) is computational expensive, an approximation could be obtained as in the work by Alexa et al. [6] by minimizing q on the plane defined by $q = \mathscr{P}(r) + tn$ and the previous n using the *conjugate gradient* algorithm [26]. These two steps are repeated until the change in both parameters is smaller than a pre-defined threshold. With these results we build the local coordinate system and compute the local polynomial approximation as described

 ISBN 978-80-86943-98-5

above. The minimization of Equation 2 can also be performed using *conjugate gradient*.

4 RAY-TRACING POINT SET SURFACES ON THE GPU

Based on the observation that the influence of a sample point p_i on the shape of the PSS can be limited, Adamson and Alexa[2] defined a trust region T_i for each sample point p_i as the ball with center on p_i and radius b, where $0.5h < b < h$. The idea for the ray-tracing process is to compute a first approximation of the intersection between the ray and the PSS and then converge iteratively to the actual intersection. For this, in a pre-processing step, the local polynomial approximation at each point p_i is calculated as described above, i.e. $r = p_i$. The first approximation for the intersection is then the nearest intersection x_t of the ray with these pre-computed local polynomials, that lies within the trust region of the corresponding p_i. To converge to the actual intersection between the ray and S_p, $\mathscr{P}(x_t)$ is computed. Note that during this projection process a local polynomial approximation g_t is found. The distance between $\mathscr{P}(x_t)$ and x_t gives us a measure for the error e_t of the current intersection x_t. If this error is greater than an user-defined threshold, the intersection between g_t and the ray is calculated and taken as the new x_t. If x_t lies outside the current trust region, the next nearest intersection of the ray with the pre-stored local polynomials is found and the process repeated.

Some simplifications were introduced into the GPU implementation of the projection operator[27]. Firstly, since the pre-computation of the local polynomials stored in each p_i requires projecting the sample points, which are almost on S_p, the iterative process for minimizing Equation 1 is not performed. Instead, the initial n is calculated using covariance analysis and the local coordinate system is built over $H(n, p_i)$. The same applies to the computation of $\mathscr{P}(x_t)$ for the current intersection x_t. Secondly, in order to make feasible the computation of the coefficients of the local polynomial approximations on the GPU, an incomplete polynomial of degree 2 is used, namely $g(x,y) = Ax^2 + By^2 + Cxy + D$.

We introduce here a further simplification based on the observation that the intersection of a ray with the stored polynomials often suffices for obtaining good approximations to the actual intersection point between the ray and the point set surface. This means, that the iterative process used to converge to the actual intersection with S_p is completely avoided. As can be appreciated in Section 7, the resulting renderings are of good quality even with this first rough approximation, whilst gaining performance.

To store the information needed on the GPU, we use a 2D texture `tex_positions` with the positions in space of the sampling points p_i and a 3D texture `tex_neighbors` with the pointers to the nearest neighbors of each p_i. These textures are used to calculate the coefficients of the polynomial $g_i(x,y)$ for each p_i, which is written to the target texture `tex_polynomial`, and the orthonormal base of the local coordinate system, written to textures `tex_basea`, `tex_baseb` and `tex_basec`, as done in previous work[27].

With this information, the intersection of the primary rays with S_p can be efficiently computed[27]. However, contrary to previous work, we approximate this intersection as the intersection of the ray with the nearest polynomial that lies inside the corresponding trust region. The process is started by rendering for each p_i a viewport-aligned disc with center in p_i and radius b. Each fragment generated this way represents a ray that intersects the trust region T_i of p_i. In the fragment shader, the ray corresponding to the fragment is transformed to the local system, its intersection with the local polynomial g_i is computed and, in case the intersection lies within T_i, the intersection point is written to texture `tex_intersection`, and the normal at the intersection point to texture `tex_normal`. With this information the viewable part of the surface can be lit and displayed, as shown in Figure 1 where the surface rendering of the Armadillo model (172974 points) obtained with our GPU renderer is shown. The performance for this model was 6 fps on an Nvidia 7900 GT grahics card.

5 SHADOWS, REFLECTIONS AND REFRACTION ON THE GPU

To limit the number of points tested to find the intersection of secondary rays with the local polynomials, we build a Cartesian grid of reduced resolution (about 20^3 for a point set of size 35000) covering the domain of the point set as depicted in Figure 3a. Then, to find the intersection of a given ray with the point set surface we traverse the grid starting in the cell where the origin of the ray lies and test the points stored in each traversed cell. If the origin of the ray lies outside the grid, the intersection of the ray with the bounding box is found and used as the new origin of the ray.

One important issue to consider is shown in Figure 3b, where the case of a polynomial of a point in a neighboring cell intersecting the current cell is exemplified. To deal with this case during the test we must include in each cell the points in neighboring cells that lie within a distance b from the boundaries of the cell as shown in Figure 3a.

 ISBN 978-80-86943-98-5

Figure 1: The Armadillo model rendered with our GPU-based point set surfaces renderer.

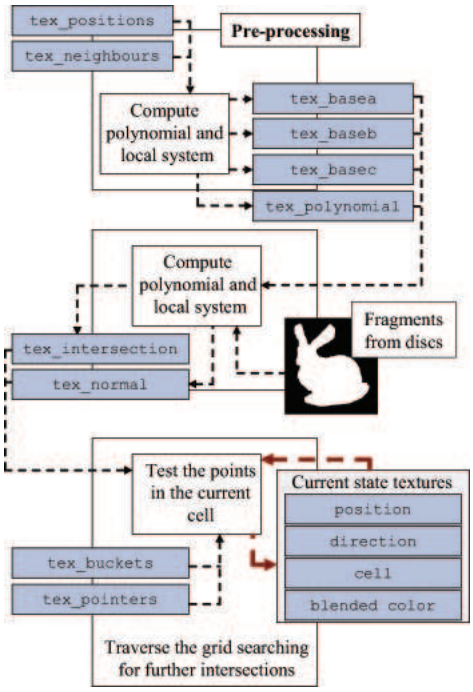


Figure 2: Ray-tracing a PSS following the secondary refracted rays only. The red pointed arrows represent the loop on the CPU used for reducing the number of instructions performed in a single pass.

This process is also performed on the GPU. Thus, a 3D texture, `tex_buckets`, for the grid is created where a pointer to a position in an intermediary 2D texture, `tex_pointers`, is stored. In texture `tex_pointers` pointers to the positions in `tex_position` and `tex_neighbors` corresponding to the points in the cell are stored sequentially. Then, given a fragment corresponding to an arbitrary ray (generated by rendering a single quad covering the viewport and following the refracting ray) the points corresponding to the cells intersected by the ray are tested in the fragment shader using this texture-based data structure.

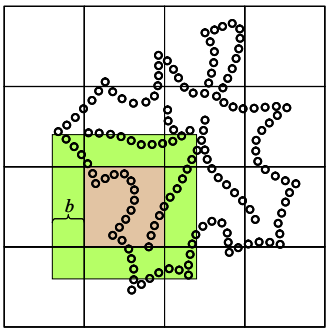
In Figure 2 we show how this can be used for ray-tracing a PSS following only the secondary refracted rays. A complete ray-tracing algorithm on the GPU could be easily implemented from this results (See [10;24] for details on the implementation of a GPU-based ray-tracer). One important issue shown in Figure 2 is the need for a CPU loop for traversing the grid. This loop could also be implemented in a fragment shader and only one render pass would be necessary for traversing the complete object finding all intersections of the ray (refracted each time an intersection is found) with the object. However, the number of instructions executed by the fragment processor is limited and when the number of points in the object over-passes a certain limit, regions with green pixels are obtained. Therefore, we opted for performing a render pass for each traversed cell. We use ping-pong rendering in order to fetch the results of the previous pass in the current pass, which included the position, direction, cell and accumulated color (therefore, we render to four render targets).

Also, support for different indices of refraction for each channel was implemented. Since it is only possible to have four render targets, we have to execute the process above described for each channel and combine their results in a further render pass. Shadows, including self-shadowing, are easily implemented by traversing the grid following the light vector. Figure 4 shows visual results of self-shadowing and shadows between the Stanford Bunny and the point set of an artifical terrain.

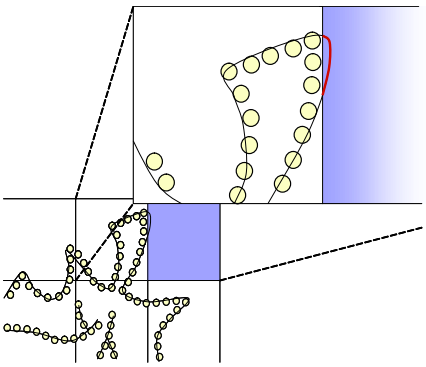
6 INTRODUCING VOLUME RENDERING EFFECTS

As mentioned before, we also included volumetric effects in the scene. For this, we make use of the texture-based data structure to perform scattered data interpolation [17] of scalar values stored at the sampling points. These points are classified as belonging to a material boundary or not. The integration for a ray is started by using the boundary points to find the first intersection x_p of the ray with the object, with the process described in the previous section. The ray is then sampled using a constant step. The neighbors of the current sampling point x_p are found using the grid described in the previous section, by accessing the points in the cell where x_p is stored, as well as the points belonging to

 ISBN 978-80-86943-98-5



(a)



(b)

Figure 3: The Cartesian grid to traverse the object. In (a) the points within the bright green and red areas are included in the list of points of the cell colored light red, due to the case shown in (b) for the shaded cell. Although there is no point inside the cell, a segment (shown in red) of the Point Set Surface could intersect it.

Figure 4: The Stanford Bunny model generating a shadow on the point-based texturized terrain. Self-shadowing is also computed.

the neighboring cells. From these points we only use those within a pre-defined distance k to the integration point. The interpolated value $s(x_p)$ at x_p is then given by

$$s(x_p) = \frac{\sum_{p_i \in N(x_p)} s(p_i) \exp(-d(x_p, p_i)^2/2\rho)}{\sum_{p_i \in N(x_p)} \exp(-d(x_p, p_i)^2/2\rho)} \quad (3)$$

where $N(x_p)$ is the set of points in the k-neighborhood of x_p, $d(x_p, p_i)$ is the distance between x_p and p_i and ρ is a smoothing parameter.

This process is easily combined with the one depicted in Figure 2 to render mixed scenes with both point-based surfaces and scattered volumetric data. Furthermore, refraction at the boundaries of a volume is possible. During grid traversal, for each visited cell, we test the intersection between the ray and the polynomials of the points stored in the cell that belong to a material boundary. If an intersection is found, the direction of the refracted ray is computed and the integration process continues with this new ray direction. This rendering process results in interesting visual effects, as can be seen in the next section.

7 RESULTS

Here we present performance and visual results obtained with known data sets from tests carried out on a PC equipped with an NVidia 7900 GT graphics card and rendering to a viewport of size 640×480. We run tests with the models using refraction, shadows computation, volume rendering and volume rendering with refraction. Also, results for surface rendering with different indices of refraction (IOR) for each channel are presented. All these rendering modes can be seen in Figure 7. For surface rendering without refraction we achieved frames rates of 28 and 20 fps for the Stanford Bunny dataset (35947 points) and the Horse dataset (48485 points) respectively. Including shadows reduced the performance to 10 fps for the Bunny and to 11 fps for the Horse.

Complex scenes can be described by means of point clouds as seen in the figures, where examples with texturized point-based terrains, scenes with refracted surfaces and volumes, self-shadowing and refracted volumes are shown. Furthermore, scenes with combined modalities, e.g. refracted volumes and texturized terrains, are supported. Following the refracted ray through the object we obtained frame rates of 1.33 fps for the Stanford Bunny and of 2.16 fps for the Horse model. As stated above, multiple indices of refraction were also implemented using multiple passes, which turned into a slow down of the rendering process (0.25 and 0.22 fps for the Bunny and the Horse respectively).

On the other hand, including volumetric effects in the rendering does not have a significant impact on the performance, since the scalar values are interpolated at the

 ISBN 978-80-86943-98-5

Figure 5: The Stanford Bunny and Horse point sets rendered with the techniques implemented. Starting with the left-top image row-wise: the raw point set, the point set surface, refraction with single IOR, refraction with multiple IOR, volume rendering with refraction and volume rendering using transfer functions.

same positions on the ray we use to traverse the grid structure while computing the intersections of the ray with the surface. Thus, 1.08 fps were achieved for the Bunny using combined refraction and volumetric effects. In this case, the performance for the Horse was 0.96 fps.

8 CONCLUSION

In this paper, we presented GPU-based rendering techniques to render scenes comprised by volumes and surfaces described only by points with no other information than the position of the points in space and the scalar value associated to them. By implementing our techniques on the GPU, we managed to reduce the computation time compared to CPU implementations. Although higher frame rates have been achieved with implementations on specialized hardware, the advantage of working with commodity graphics hardware is the accessibility to desktop users and the rapid increase in computational power and flexibility compared to CPUs.

Currently the most important problem we find for continuing with the development of point-based methods is the complexity of the proximity queries. In order to reduce the time spent in this queries, we had to precompute the nearest neighbors for the sample points and use a grid-based structure during ray traversal as

 ISBN 978-80-86943-98-5

in previous work on ray-tracing on the GPU. However, this has the disadvantage of using additional storage space that, although is not as large as the required by surface meshes or structured and non-structured volumes, could become a serious problem for larger models. On the other hand, we believe that the advances on point-based methods will capture the attention of researchers and hardware manufacturers in the near future, resulting in the inclusion of such expensive operations in hardware allowing the further developing of point-based computer graphics.

ACKNOWLEDGMENTS

Research supported in part by the DAAD (German Academic Exchange Service) grant Nr. A/04/08711.

REFERENCES

[1] Bart Adams, Richard Keiser, Mark Pauly, Leonidas J. Guibas, Markus Gross, and Philip Dutré. Efficient raytracing of deforming point-sampled surfaces. *Computer Graphics Forum*, 24(3):677–684, 2005.

[2] A. Adamson and M. Alexa. Ray tracing point set surfaces. In *Proc. of Shape Modeling International*, page 298. IEEE Computer Society, 2003.

[3] Anders Adamson and Marc Alexa. Approximating and intersecting surfaces from points. In *Proc. of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 230–239. Eurographics Assoc., 2003.

[4] M. Alexa and A. Adamson. On normals and projection operators for surfaces defined by point sets. In *Eurographics Symposium on Point-Based Graphics*, pages 149–155. Eurographics Assoc., 2004.

[5] 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.

[6] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. In *Proc. of IEEE Visualization*, pages 21–28. IEEE Computer Society, October 2001.

[7] Nina Amenta and Yong J Kil. The domain of a point set surfaces. *Eurographics Symposium on Point-based Graphics*, 1(1):139–147, 2004.

[8] Nina Amenta and Yong Joo Kil. Defining point-set surfaces. *ACM Transactions on Graphics*, 23(3):264–270, 2004.

[9] Mario Botsch and Leif Kobbelt. High-quality point-based rendering on modern GPUs. In *Proc. of Pacific Conference on Computer Graphics and Applications*, page 335. IEEE Computer Society, 2003.

[10] Nathan A. Carr, Jesse D. Hall, and John C. Hart. 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.

[11] Tamal K Dey, S Goswami, and J Sun. Extremal surface based projections converge and reconstruct with isotopy. Technical Report OSU-CISRC-05-TR25, Ohio State University, 2005.

[12] Leif Kobblet and Mario Botsch. A survey of point-based techniques in computer graphics. *Computer & Graphics*, 28(6):801–814, 2004.

[13] Ravikrishna Kolluri. Provably good moving least squares. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1008–1017, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics. Vencedor do prêmio: *Best Student Paper Award*.

[14] David Levin. The approximation power of moving least-squares. *Mathematics of Computation*, 67(224):1517–1531, 1998.

[15] Gerard Medioni, Mi-Suen Lee, and Chi-Keung Tang. *Computational Framework for Segmentation and Grouping*. Elsevier Science, 2000.

[16] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *Proc. of ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 141–151. ACM Press, 2004.

[17] Gregory M. Nielson. Scattered data modeling. *IEEE Computer Graphics and Applications*, 13(1):60–70, 1993.

[18] Mark Pauly. *Point Primitives for Interactive Modeling and Processing of 3D Geometry*. PhD thesis, Federal Institute of Technology (ETH) of Zurick, 2003.

[19] Mark Pauly, Richard Keiser, Bart Adams, Philip Dutré, Markus Gross, and Leonidas J. Guibas. Meshless animation of fracturing solids. *ACM Transactions on Graphics*, 24(3):957–964, 2005.

[20] Mark Pauly, Richard Keiser, Leif P. Kobbelt, and Markus Gross. Shape modeling with point-

sampled geometry. *ACM Transactions on Graphics*, 22(3):641–650, 2003.

[21] Mark Pauly, Dinesh K. Pai, and Leonidas J. Guibas. Quasi-rigid objects in contact. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 109–119. ACM Press, 2004.

[22] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 1st edition, 1986.

[23] T. Purcell. *Ray Tracing on a Stream Processor*. PhD thesis, Stanford University, 2004.

[24] 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, July 2002. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002).

[25] Gernot Schaufler and Henrik Wann Jensen. Ray tracing point sampled geometry. In *Proc. of Eurographics Workshop on Rendering Techniques*, pages 319–328. Springer-Verlag, 2000.

[26] Jonathan R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Carnegie Mellon University, 1994.

[27] E. Tejada, J.P. Gois, L. G. Nonato, A. Castelo, and T. Ertl. Hardware-accelerated Extraction and Rendering of Point Set Surfaces. In *Proceedings of EUROGRAPHICS - IEEE VGTC Symposium on Visualization*, pages 21–28, 2006.

[28] Ingo Wald and Hans-Peter Seidel. Interactive Ray Tracing of Point Based Models. In *Proceedings of 2005 Symposium on Point Based Graphics*, 2005.

[29] Michael Wand and Wolfgang Straßer. Multi-resolution point-sample raytracing. In *Graphics Interface*, pages 139–148, 2003.

[30] Matthias Zwicker, Mark Pauly, Oliver Knoll, and Markus Gross. Pointshop 3D: an interactive system for point-based surface editing. In *SIGGRAPH : Proc. of Computer Graphics and Interactive Techniques*, pages 322–329. ACM Press, 2002.

 ISBN 978-80-86943-98-5

Image Registration for Multi-exposure High Dynamic Range Image Acquisition

Anna Tomaszewska
Szczecin University of Technology
atomaszewska@wi.ps.pl

Radoslaw Mantiuk
Szczecin University of Technology
rmantiuk@wi.ps.pl

ABSTRACT

We present a fully automatic method for eliminating misalignments between a sequence of hand-held photographs taken at different exposures. The key component of the technique is the SIFT method that is employed to search for key-points (or feature-points) in consecutive images. The key-points are used to find matrices, that transform a set of images to a single coordinate system and eliminate any global misalignments (including general planar homography). We employ this technique to capture high dynamic range images from a set of photographs taken at different exposures, where misalignments can cause blurring and artifacts, and prevent achieving high quality HDR images. The proposed alignment technique works well for over- and under-exposed images and is not sensitive to an image content. We present our implementation of the technique and the results of tests made for variety of photographs.

Keywords: image alignment, image registration, contrast domain, HDR capture, High Dynamic Range Imaging.

1 INTRODUCTION

There is tremendous progress in the development and accessibility of high dynamic range (HDR) imaging technology [RWPD05]. Modern image processing and graphics software becomes HDR enabled. Also HDR digital photography replaces low dynamic range (LDR) technologies. HDR photographs are of much better quality and easier to be processed in a digital darkroom. Unfortunately, HDR cameras are still very expensive and not available for average users. On the other hand, taking HDR photographs seems to be legitimate and crucial. In the near future LDR images may become almost obsolete due to the progress in LCD technology [SHS+04] and it will not be easy to display LDR image correctly. LDR photographs will look pale and not interesting on HDR LCD displays.

The multi-exposure HDR capture technique [MP95] seems to be a good alternative to HDR cameras, which can be used to create an HDR image from photographs taken with a conventional LDR camera. The technique uses differently exposed photographs to recover the response function of a camera. From the response function, the algorithm creates an HDR image whose pixel values are proportional to the true radiance (or luminance) value of a scene. Because this technique requires multiple input photographs, there is a high likelihood of misalignments between pixels in the sequence of exposures due to moving of a hand-held camera (global motion) or dynamic object in a scene (local motion causing ghosting). It is crucial that misalignments between input photographs should be removed before fusing an HDR image.

Our work addresses the problem of global alignment (registration) of hand-held photographs taken with different exposures. The method compensates misalignments caused by any movement of a camera. In particular, general planar homography (or linear planar projective transform) is considered so any linear transformation of a point position between planes on images is recovered (see [HZ06] for details on the properties and limitations of planar homography). The key component of the technique is the scale-invariant feature transform (SIFT) [Low04] that is employed to search for key-points (often called feature-points) in consecutive images. We modified the algorithm by automatizing contrast threshold value calculation. The key-point correspondences are used to find matrices, that map a set of images to a single coordinate system. The bilinear interpolation of the output images based on the calculated matrices completes the algorithm. The registered photographs are fused using our implementation of multi-exposure HDR capture technique proposed by Mitsunaga and Nayar [MN99]. Our alignment technique is fully automatic and is not sensitive to image content. It works correctly even for highly under- and over-exposed images. We modified registration algorithm so it can be applied not only for standard LDR

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

 ISBN 978-80-86943-98-5

images but also for HDR images. We can use extended dynamic range photographs stored in RAW format as an input to alignment algorithm. This way, less input photographs can be used to create high dynamic range image. Moreover, we noticed that the accuracy of image registration is much better for input RAWs than for standard LDR photographs.

In Section 2 we present a review of previous work related to image alignment for multi-exposure HDR capture techniques. A detailed description of our SIFT based HDR alignment technique is presented in Section 3. In Section 4 we show implementation of the method followed by the results of the tests run on hand-captured photographs (Section 5). Section 6 presents conclusions and discusses future work directions.

2 RELATED WORK

In recent years significant progress has been made in the development of algorithms that allow to capture HDR images using low dynamic range sensors (standard LDR cameras) [MP95, DM97, MN99, RBS99]. These algorithms retrieve high dynamic range information from a sequence of photographs. The authors suggest using tripod to avoid camera movement and they do not address the problem of eliminating misalignments.

The problem of image alignment and matching was intensively studied during last years [ZF03, Bro92] but not for registration of images of different exposures. The only solution that addressed exactly the problem of capture of HDR photographs, was proposed in [War03, RWPD05]. The technique employs conversion of input photographs into percentile threshold bitmaps. The bitmaps are analyzed and then aligned horizontally and vertically using shift and difference operations over each image. In chapter 5 we compare the results achieved with this algorithm to our method.

Kang et al. [KUWS03] described a technique for creating high dynamic range video from a sequence of altering light and dark exposures. A part of the technique is a HDR stitching process, which includes global and local alignment to compensate for pixel motion. The stitching process can be also used to compensate for camera movement when creating an HDR still from a series of bracketed still photographs. However, the presented technique seems to be suitable for video where there are no large differences between consecutive frames.

In [ST04] Sand and Teller present a global and local matching algorithm, which is robust to changes in exposure of photographs. The key idea behind this technique is to identify which parts of the image can be matched without being confused by parts that are difficult to match. Such assumption seems to be not valid for images with large differences in exposures, where there is usually not enough information for correct matching. The technique was designed for matching two video sequences and was not tested on still photographs.

Recently, Cerman and Hlavac [CH06] presented an alignment method based on unconstrained nonlinear optimization. In this method, each image is linearized using the estimated camera response function and multiplied by the exposure ratio. Then, a normalized difference summed across all corresponding pixels is used to estimate misalignments. The method can compensate global rotation and horizontal and vertical shifts.

There are a few techniques which compute camera response function based on misaligned photographs [GN03, KP04]. However, these methods are not meant to create HDR images. The problem of removing ghosting artifacts in a multi-exposure sequence of photographs was also investigated [EAKR06, RWPD05] but proposed algorithms do not take into consideration a compensation of camera movements.

In the next section, we present our technique of removing global misalignments between photographs in a multi-exposure sequence. This technique can be applied to still images and it compensates any camera movement.

3 MISALIGNMENT COMPENSATION

Our modification of multi-exposure HDR capture method is presented in Figure 1. We align all input photographs to a selected reference (the image with the best exposure) before fusing an HDR image rather than use a tripod as suggested in the previous methods.

The algorithm of our alignment technique is shown in Figure 2. The technique establishes correspondences between points, lines or other geometrical entities in a set of images. We use a modified SIFT (Scale Invariant Feature Transform) [Low04] algorithm to extract local features called key-points. They are located at scale-space maxima/minima of a difference of Gaussian function. The key-points descriptors accumulated in orientation histograms form the invariant descriptors.

In the next step, these descriptors are used to find correspondences between key-points in the reference image and remaining images. The number of the correspondences is reduced by RANSAC [FB81] algorithm because only four pairs of key-points (8 coefficients) are needed to calculate transformation matrix. The RANSAC selects a set of key-points correspondences that are compatible with a homography between the images. It uses the singular value composition (SVD) method to solve over-determined system of linear equations required by the optimization algorithm.

 ISBN 978-80-86943-98-5

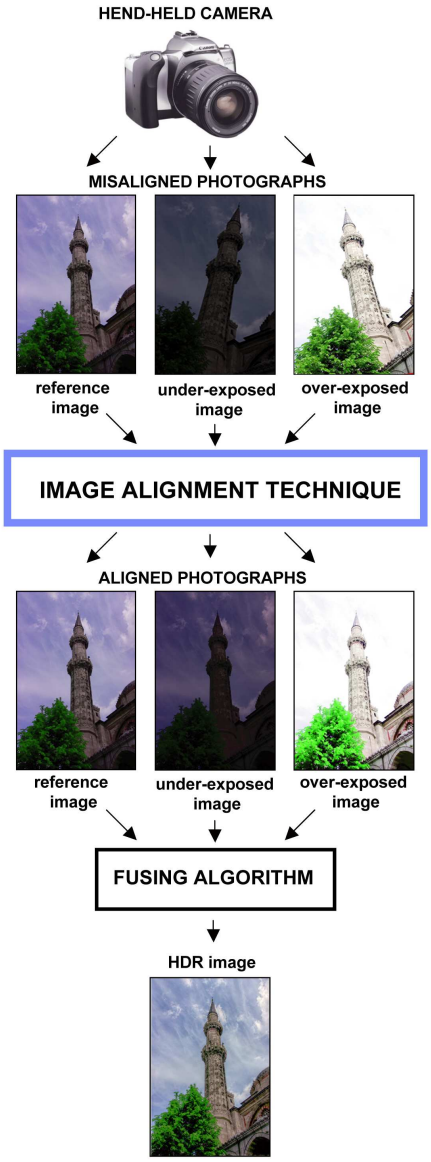


Figure 1: A technique of capturing HDR photographs with a hand-held camera based on the image alignment and image fusing techniques.

The transformation matrix is computed by the direct linear transform (DLT) algorithm. This matrix is used to transform a given image to the coordinates of the reference image:

$$x' \sim \mathbf{H} \cdot x, \tag{1}$$

where x' and x are points from destination and source images respectively, and H is a transformation matrix:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix},$$

h_{ij} are unknowns.

The algorithm computes the color values of consecutive output pixels using bilinear interpolation of appropriate input pixels. This solution ensures that there are no holes in an output image. Moreover a sub-pixel precision of calculation improves the quality of images (decreases aliasing).

We noticed that the accuracy of the transformation matrix can be decreased due to doubtful correspondences. The RANSAC algorithm chooses the best pairs of key-points based on a random technique, which can fail. We improve the accuracy of the correspondences selection by choosing only these key-points which occur in all photographs simultaneously.

The key-points searching algorithm works in contrast domain so it is intensity invariant. Therefore, we can use it to find the similarities between images of different exposures, where intensity values can change significantly. The algorithm uses a multi-scale pyramid of Difference-of-Gaussian images to improve the key-points detection (a larger neighborhood of a pixel is considered).

As has been mentioned above the key-searching algorithm operates also with RAW format as an input data. However, in order to improve the accuracy of image registration for such pictures we modified it. Instead of using constant contrast threshold value we calculate if for each image separate in relation its dynamics. This modification makes it possible to detect far more key-points in RAW images in comparison to LDR ones. It enhances accuracy for pictures taken in bad light conditions. The results are presented in section 5.

4 IMPLEMENTATION

The algorithm for multi-exposure image alignment has been implemented as a part of *pfstools* [1] package [MKM07]. The main components of *pfstools* are programs for reading and writing images in all major HDR and LDR formats (e.g. OpenEXR, Radiance's RGBE, etc.) and programs for basic image manipulation (rotation, scaling, cropping, etc.). The typical usage of *pfstools* involves executing several programs joined by UNIX pipes. The first program transmits the current frame or image to the next one in the chain. The final program should either display an image or write it to a disk. An example of command line is given below:

[1] *pfstools* is distributed as Open Source under the GPL license and the project web page can be found at `http://pfstools.sourceforge.net/`

 ISBN 978-80-86943-98-5

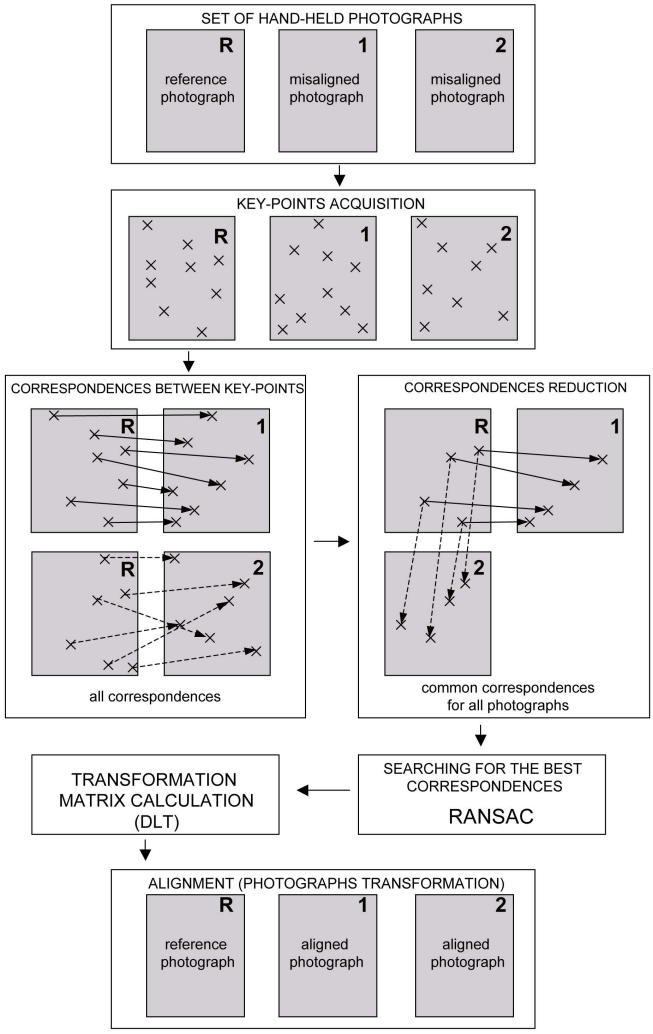


Figure 2: The algorithm of the image alignment technique.

```
pfsin␣input1.jpg␣input2.jpg␣|␣pfsalign␣/
␣|␣pfshdrcalibrate␣|␣pfsout␣output.exr
```

Read the images `input1.jpg` and `input2.jpg`, align the images (`pfsalign`), create HDR image (`pfshdrcalibrate`) and write the output to `output.exr`.

The *pfsalign* program (see the example above) is responsible for recovering misalignments and transforming input images to a single coordinate space. This program was implemented in C++ and, like the *pfstools* package, can be run under Mac OS, Linux and Windows (with cygwin support).

The *pfsalign* consists of two main modules. The first one looks for key-points and correspondences between them, and then calculates transformation matrix. We used SVD solver implementation from numerical recipes book [PTVF92] to aid the calculation of the transformation matrix. The second module transforms images to single coordinate space with sub-pixel precision so that there are no holes in resulting images.

The *pfshdrcalibrate* program (see the example above) fuses images of different exposures and calculates a HDR radiance map. We implemented the fusing algorithm proposed by Mitsunaga and Nayar [MN99] which is based on polynomial approximation of the camera response function. We find this algorithm very suitable, especially in case of the input sequence of a limited number of photographs (e.g. 2-3). The polynomial function fills gaps in a response function so that it is possible to calculate correct radiance value even for pixels without good representation in the input photographs (e.g. pixels with maximum value (255) in all photographs).

5 RESULTS

To test the usability of our multi-exposure image alignment method, we have executed an extensive set of tests on photograph sequences. The main goal of the

 ISBN 978-80-86943-98-5

tests was to check the accuracy of alignment for images of short and long exposures (very light and very dark) and of different image content. All photographs were taken with hand-held digital camera (Canon 10D with Sigma 18-125 mm F3.5-5.6 DC lens) and encompassed any possible combination of camera movements (shifts, rotations and zooming). To evaluate the accuracy of the image alignment, we fuse a set of aligned images into a HDR image and subjectively estimate the quality of that image.

In the top row of Figure 3 a set of misaligned photographs of the same scene is depicted (the set consists of three photographs of resolution 1536x1024 pixels stored in 8-bits JPEG format). The photographs were aligned using our algorithm (the second row) and then a HDR image was created (bottom row). The photographs were aligned with sufficient accuracy so that there are no visible artifacts in the final HDR image. There are also no visible holes in the aligned images (from the middle row), as they were filled with bilinearly interpolated pixel values. The algorithm requires about 450 seconds to align three presented photographs on the CPU. The time of registration depends on an image content. For comparison, we also show a false HDR (the third row - left) which was fussed based on misaligned images from the top row. In the third row (right), the HDR image created by *hdrgen* (available at `http://www.anyhere.com/`) program is presented. It can be seen in this figure that simple transformations implemented in *hdrgen* are not enough for accurate image registration.

In Figure 4 we show a few additional results of image alignment. In these examples the capabilities of removing shifts, rotations, zooms and a combination of these transformations are presented. The algorithm correctly recognizes even large movements of a camera.

Figure 5 presents another example of input image set with very light (over-exposed) and very dark (under-exposed) photographs. One can see that despite a large difference in exposure, sufficient number of key-points was found to establish a correspondence between images and calculate transformation matrices. Moreover, there are no key-points in unsteady too dark or too light areas.

Figure 6 depicts the arrangement and number of key-points detected for LDR and RAW images respectively. The pictures were taken in the same light conditions. In RAW images, in contrast to LDR pictures, features were detected also in poorly lit fragments of images. Due to that we have a more regular arrangement of key-points in a picture compared with LDR images, and what follows we can calculate a more accurate transformation matrix.

We successfully tested our alignment technique for many photographs of different content. The technique requires only four pairs of corresponding key-points to calculate transformation matrix so it will give correct results also for images with large non-textured areas and parallax effects.

The algorithm fails for images that contain many recurrent textures occupying a large portion of an image. In such cases, many key-points have the same descriptors and RANSAC (more precisely SVD) cannot compute good (it means falling below an error threshold) key-point correspondences. We find this problem difficult to solve based on the feature based alignment method (our algorithm belongs to this group). Fortunately, such kinds of photographs are rather rare in practice.

6 CONCLUSION AND FUTURE WORK

The major outcome of this work is an image registration method that can be used to remove misalignments between images in a sequence of differently exposed photographs (in RAW and/or JPEG format). The method is fully automatic and compensates misalignments caused by any movement of a camera. It works in contrast domain so is less sensitive to image content and changes in exposure. The method employs SWIFT algorithm to find key-points, RANSAC algorithm to choose the best key-point correspondences and DLT technique to calculate transformation matrix. The input photographs are transformed to a single coordinate system with sub-pixel precision to avoid holes and aliasing artifacts. The results of this work facilitate capturing HDR images based on the multi-exposure techniques. Our method simplifies acquisition of HDR images by removing the main disadvantage of the multi exposure techniques, which is a necessity of using a tripod.

We implemented and tested our method for a variety of photographs. We find it effective in most cases but several difficult images cause the method to fail. The solution of this problem is further optimization of SIFT and RANSAC algorithms for adjusting them to high dynamic range input images. We also plan to estimate the transformation matrix not only for planar homography but also for more general 3D case with depth reconstruction. Since our current implementation does not deal with local changes, we intend to implement a de-ghosting technique in the future. We plan to speed-up the implementation based on the GPU programming to facilitate the practical usage of our method.

ACKNOWLEDGMENTS

We would like to thank Piotr Ostiak for basic implementation of the image matching algorithm. Special thanks go to Rafal Mantiuk and Grzegorz Krawczyk

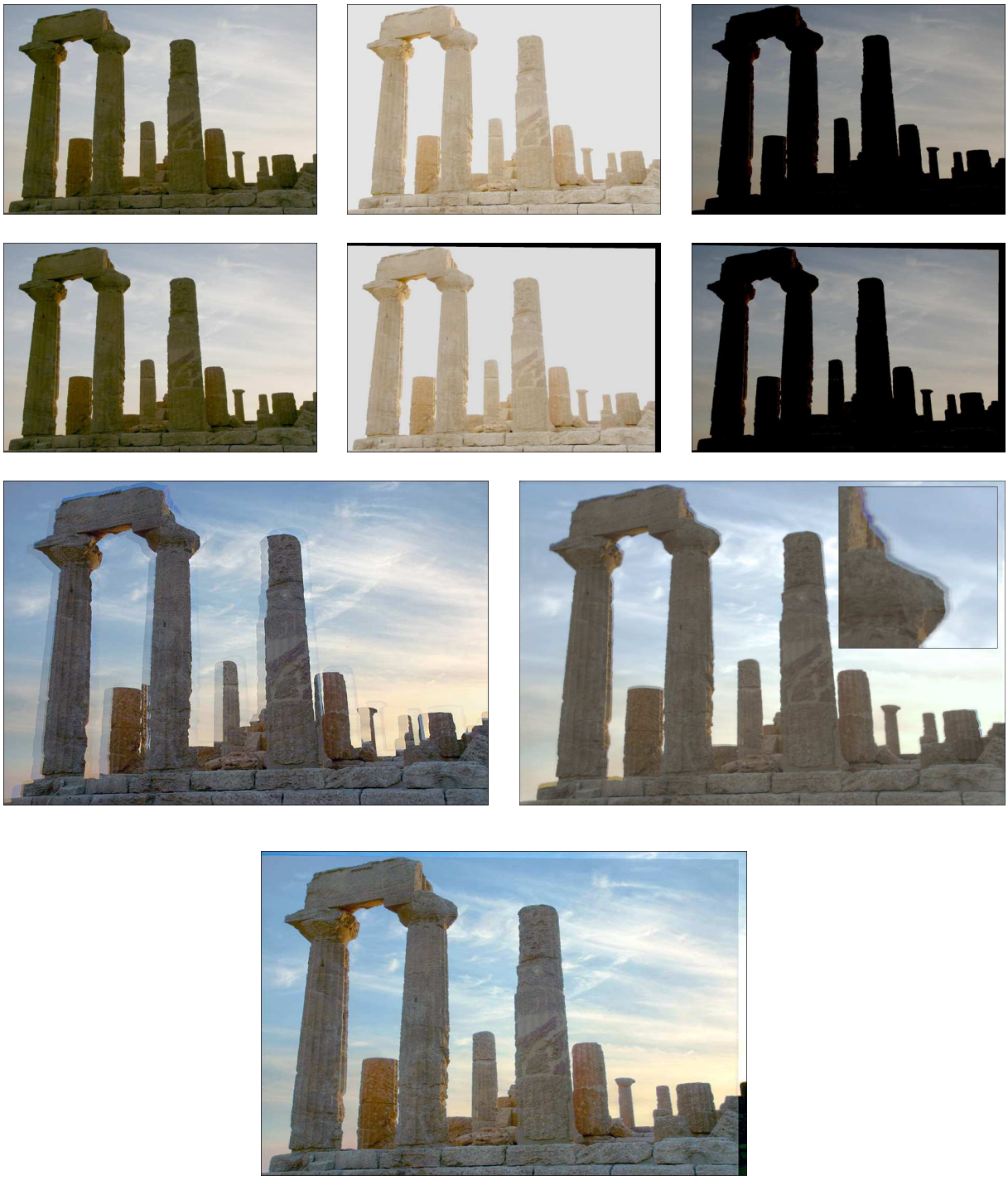


Figure 3: The HDR image captured with a hand-held camera. Top row: misaligned photographs. The second row from top: aligned photographs, the first photograph is a reference image. The third row: HDR image created by fusing the misaligned photographs (left) and by *hdrgen* software (right) (the inset depicts misalignment artifacts). Bottom row: HDR image created from aligned photographs. All HDR images were tone mapped using contrast domain operator. [MMS05])

 ISBN 978-80-86943-98-5

Figure 4: The results of image alignment. Left column: reference photographs. Middle column: photographs before alignment. Right column: photographs after alignment (transformed to the coordinate system of the reference photograph). From the top to the bottom row, the examples show shift, rotation, zoom, and complex camera movement (general homography).

for their helpful comments concerning this work and for delivering the *pfstools* software.

REFERENCES

[Bro92] Lisa Gottesfeld Brown. A survey of image registration techniques. *ACM Comput. Surv.*, 24(4):325–376, 1992.

[CH06] Lukas Cerman and Vaclav Hlavac. Exposure time estimation for high dynamic range imaging with hand held camera. In *Proc. of Computer Vision Winter Workshop, Czech Republic*, February 2006.

[DM97] P.E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of ACM SIGGRAPH 1997*, pages 369–378, 1997.

[EAKR06] Ahmet Oguz Akyuz Erum Arif Khan and Erik Reinhard. Ghost removal in high dynamic range images. In *Proc. of IEEE International Conference on Image Processing*, October 2006.

[FB81] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381Ð395, 1981.

[GN03] M.D. Grossberg and S.K. Nayar. Determining the camera response from images: What is knowable? *PAMI*, 25(11):1455–1467, November 2003.

[HZ06] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2006.

[KP04] S.J. Kim and M. Pollefeys. Radiometric self-alignment of image sequences. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 2004.

[KUWS03] Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High dynamic range video. *ACM Trans. Graph.*, 22(3):319–325, 2003.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[MKM07] Rafal Mantiuk, Grzegorz Krawczyk, and Radoslaw Mantiuk. High dynamic range imaging pipeline: Perception-motivated representation of visual content. In *To be published in Proc. of Human Vision and Electronic Imaging XII, IS&T/SPIE's Annual Symposium on Electronic Imaging (2007)*, SPIE Proceedings Series, San Jose, California USA, January 2007.

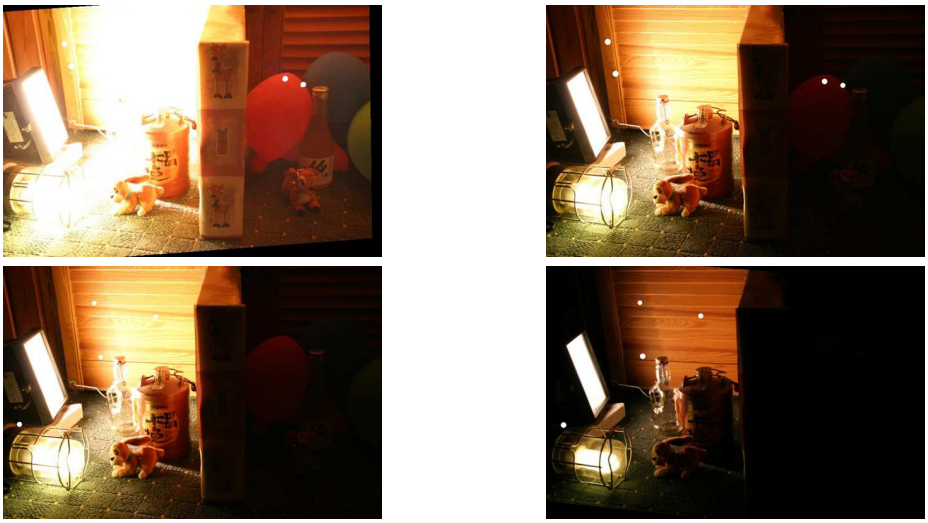


Figure 5: Example photographs taken with a very short and very long exposures. Valid key-points are marked as white dots. There are no key-points in too dark and too light areas.

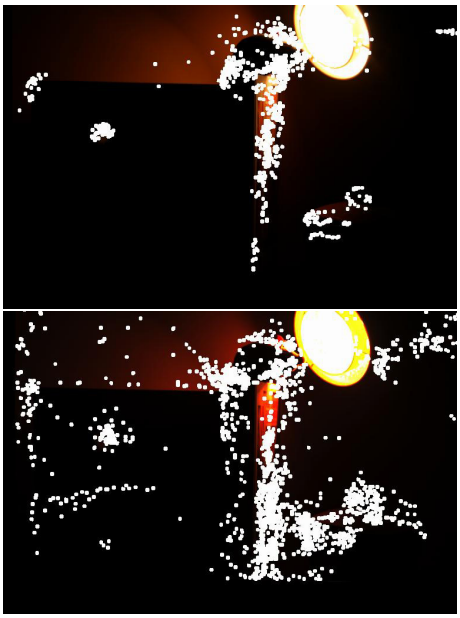


Figure 6: An arrangement and number of key-points detected for LDR (top) and RAW (bottom) images taken in the same light conditions.

SPIE.

[MMS05] Rafał Mantiuk, Karol Myszkowski, and Hans-Peter Seidel. A perceptual framework for contrast processing of high dynamic range images. In *APGV '05: 2nd Symposium on Appied Perception in Graphics and Visualization*, pages 87–94, 2005.

[MN99] T. Mitsunaga and S. K. Nayar. Radiometric self calibration. pages 374–380, 1999.

[MP95] S. Mann and R. Picard. Being 'Undigital' with Digital Cameras: Extending Dynamic Range by Combining Differently Exposed Pictures. *Proc. of IST's 48th Annual Conference*, pages 442–448, May 1995.

[PTVF92] William Press, Saul Teukolsky, William Vetterling, and Brian Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, 2nd edition, 1992.

[RBS99] M. Robertson, S. Borman, and R. Stevenson. Dynamic range improvement through multiple exposures. In *Proceedings of the 1999 International Conference on Image Processing (ICIP-99)*, pages 159–163, Los Alamitos, CA, October 24–28 1999.

[RWPD05] Erik Reinhard, Greg Ward, Sumanta Pattanaik, and Paul Debevec. *High Dynamic Range Imaging. Acquisition, Manipulation, and Display*. Morgan Kaufmann, 2005.

[SHS+04] H. Seetzen, W. Heidrich, W. Stuerzlinger, G. Ward, L. Whitehead, M. Trentacoste, A. Ghosh, and A. Vorozcovs. High dynamic range display systems. *ACM Trans. on Graph.*, 23(3):757–765, 2004.

[ST04] Peter Sand and Seth Teller. Video matching. *ACM Trans. Graph.*, 23(3):592–599, 2004.

[War03] Greg Ward. Fast, robust image registration for compositing high dynamic range photographs from hand-held exposures. *journal of graphics tools*, 8(2):17–30, 2003.

[ZF03] Barbara Zitova and Jan Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21(11):977–1000, October 2003.

Face Recognition under Varying Illumination

ABSTRACT

This paper proposes a novel pipeline to develop a Face Recognition System robust to illumination variation. We consider the case when only one single image per person is available during the training phase. In order to utilize the superiority of Linear Discriminant Analysis (LDA) over Principal Component Analysis (PCA) in regard to variable illumination, a number of new images illuminated from different directions are synthesized from a single image by means of the Quotient Image. Furthermore, during the testing phase, an iterative algorithm is used for the restoration of frontal illumination of a face illuminated from any arbitrary angle. Experimental results on the YaleB database show that our approach can achieve a top recognition rate compared to existing methods and can be integrated into real time face recognition system.

Keywords

Face Recognition, Image Synthesis, Illumination Restoration, Dimensionality Reduction.

1. INTRODUCTION

Face Recognition has been recently receiving particular attention especially in security related fields. As for the early researches, both geometric feature based methods and template-matching methods were regarded as typical technologies. Since the 1990s appearance-based methods have been playing a dominant role in the area.

However, face recognition remains a difficult, unsolved problem in general. The changes induced by illumination are often larger than the differences between individuals, causing systems based directly on comparing images to misclassify input images. Approaches for handling variable illumination can be divided into four main categories: (1) extraction of illumination invariant features [Adi97, Lai01] (2) transformation of images with variable illuminations to a canonical representation [Xie05, Liu05, Sha01] (3) modeling the illumination variations [Geo01] (4) utilization of some 3D face models whose facial shapes and albedos are obtained in advance [Wey04, Zhan05].

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.

Another effort related to varying illumination is the creation or synthesis of the image space of a novel image illuminated from an arbitrary angle. The works done on this topic consider face images as Lambertian surfaces, and at the same time assume faces as objects having the same shape but different surface texture [Sha01, Geo01, Geo03, Zhan05, Zhao00, Zhao03]. While Georghiades et al. [Geo01] do not deal with cast and attached shadows, Zhang et al. [Zhan05] minimize shadow effects by applying surface reconstruction.

In this work we propose a novel approach to minimize the effects of illumination variations on face recognition performance. Our method requires only one training image for any subject that will undergo testing. We try to solve the Small Sample Size (SSS) problem regarding this case. Our pipeline consists of three main steps: (1) synthesis of the image space for any input image, (2) training using a class-based linear discriminant approach, and (3) illumination restoration [Liu05] of any incoming face image during the recognition process. In Section 2 we will give a short introduction to PCA and LDA as dimensionality reduction techniques. An image synthesis method will be introduced in section 3. Section 4 will present an iterative process of illumination restoration. Finally experimental results will be given in section 5 and conclusions are drawn in Section 6.

2. DIMENSIONALITY REDUCTION

When using appearance based methods, we usually represent an image of width w and height h as a vector in a $w \cdot h$ dimensional space. In practice, this space, i.e. the full image space, is too large to allow robust and fast object recognition. A common way to attempt to resolve this problem is to use dimensionality reduction techniques.

2.1 Principal Component Analysis

Principal Component Analysis [Tur91] is a technique that is useful for the compression and classification of data. More formally let us consider a set of N sample images $\{p_1,p_2,\ldots,p_N\}$ taking values in an n-dimensional image space, and assume that each image belongs to one of c classes $\{P_1,P_2,\ldots,P_c\}$. We consider a linear transformation mapping the original n-dimensional image space into an m-dimensional feature space, where (m < n). If the total scatter matrix S_T (covariance matrix) is defined as:

$$S_T = \sum_{k=1}^{N} (p_k - \mu)(p_k - \mu)^T \qquad (1)$$

the dimension reduction is realized by solving the eigenvalues problem:

$$S_T \Phi = \Phi \Lambda \qquad (2)$$

where μ is the mean image, Λ is a diagonal matrix whose diagonal elements are the eigenvalues of S_T with their magnitudes in descending order, and Φ is a matrix whose i^{th} column is the ith eigenvector of S_T. In order to obtain the eigenspace we generally choose m eigenvectors corresponding to the m largest eigenvalues, which capture over 95% of the variations in the images. After calculating the eigenfaces the projection is the only process left to be done. Let W_S be the matrix consisting of the m eigenvectors, and I_n be a new face image. The projection of I_n onto the eigenspace is represented as follows:

$$a = W_S^T (I_n - \mu) \qquad (3)$$

where a is mx1 vector containing the m projection coefficients. The reconstructed image is then given as:

$$I_n' = W_S a + \mu \,. \qquad (4)$$

The reconstructed image is the best approximation of the input image in the mean square sense. An advantage of using such representations is their reduced sensitivity to noise. A drawback of this approach is that the scatter being maximized is due not only to the between-class scatter that is useful for classification, but also to the within-class scatter that, for classification purposes, is unwanted information. Thus if PCA is presented with images of faces under varying illumination, the projection matrix $\boldsymbol{W_S}$ will contain principal components which retain, in the projected feature space, the variation due to lighting. Consequently, the points in the projected space will not be well clustered, and worse, the classes may be smeared together. It has been suggested that by discarding the three most significant principal components, the variation due to lighting is reduced [Bel97].

2.2 Fisher Linear Discriminant Analysis

Fisher's Linear Discriminant Analysis (FLDA) [Bel97] uses an important fact of photometric stereo: In the absence of shadowing, given three images of a Lambertian surface taken from the same viewpoint under three known, linearly independent light source directions, the albedo and surface normal can be recovered. For classification, this fact has great importance. It shows that, for a fixed viewpoint, the images of a Lambertian surface lie in a 3D linear subspace of the high-dimensional image space. One can perform dimensionality reduction using linear projection and still preserve linear separability. More formally, let us continue our previous study of PCA from this new perspective. The Linear Discriminant Analysis (LDA) selects W in such a way that the ratio of the between-class scatter and the within-class scatter is maximized. Let the between-class scatter matrix be defined as:

$$S_B = \sum_{i=1}^{C} N_i (\mu_i - \mu)(\mu_i - \mu)^T \qquad (5)$$

and the within-class scatter be defined as:

$$S_W = \sum_{i=1}^{C} \sum_{k=1}^{N} (p_k - \mu_i)(p_k - \mu_i)^T \qquad (6)$$

where μ_i is the mean image of class P_i , and N_i is the number of samples in class P_i. If S_W is nonsingular, the optimal projection $\boldsymbol{W_{opt}}$ is chosen as the matrix with orthonormal columns which maximizes the ratio of the determinant of the between-class scatter matrix of the projected samples to the determinant of the within-class scatter matrix of the projected samples, i.e.:

$$W_{opt} = \arg\max_{W} \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|} = [w_1\, w_2 \ldots w_m] \qquad (7)$$

In the face recognition problem, one is confronted with the difficulty that the within-class scatter matrix S_W is always singular. This stems from the fact that the rank of S_W is at most N - c, and, in general, the number of images in the learning set N is much smaller than the number of pixels in each image n. This means that it is possible to choose the matrix W such that the within-class scatter of the projected samples can be made exactly zero. To overcome the complication of a singular S_W, an alternative method has been proposed called Fisherfaces. It avoids this

 ISBN 978-80-86943-98-5

problem by projecting the image set to a lower dimensional space so that the resulting within-class scatter matrix S_W is nonsingular. This is achieved by using PCA to reduce the dimension of the feature space to N - c, and then applying the standard LDA defined by Eq. (7) to reduce the dimension to c - 1.

Recently it has been shown that the null space of S_W may contain significant discriminatory information [Lu03, Gao06]. As a consequence, some of the significant discriminatory information may be lost due to the preprocessing PCA step. Many methods classified as Direct LDA [Chen00, Yu00] have been developed to deal with this problem. However, the Fisherface method appears to be the best simultaneously handling variation in lighting. It has lower error rate than the PCA method.

3. IMAGE SYNTHESIS

Nearly all approaches to view synthesis take a set of images gathered from multiple viewpoints and apply techniques related to structure from motion, stereopsis, image transfer, image warping, or image morphing. Each of these methods requires the establishment of correspondences between image data (e.g., pixels) across the set. Since dense correspondence is difficult to obtain, most methods extract sparse image features (e.g., corners, lines), and may use multi-view geometric constraints or scene-dependent geometric constraints to reduce the search process and constrain the estimates. For these approaches to be effective, there must be sufficient texture or viewpoint-independent scene features, such as albedo discontinuities or surface normal discontinuities. Underlying nearly all such stereo algorithms is a constant brightness assumption, that is, the intensity (irradiance) of corresponding pixels should be the same.

This section is based on a work showing that the set of all images generated by varying lighting conditions on a collection of Lambertian objects can be characterized analytically using images of a prototype object and a (illumination invariant) "signature" image per object of the class called Quotient Image [Sha01]. In this approach the consideration will be restricted to objects with a Lambertian reflectance:

$$I(k,l) = \rho(k,l)n(k,l)^T s \qquad (8)$$

where $0 \le \rho(k, l) \le 1$ is the surface texture, n(k, l) is the surface normal direction, and s is the light source direction whose magnitude is the light source intensity. Furthermore it is assumed that the faces belonging to a class have the same shape but differ in surface texture. Although this is a very strong assumption it can be shown that it holds if faces are roughly aligned.

Given two objects *y* and *a*, let the quotient image Q be the ratio of their albedos:

$$Q_y(u,v) = \frac{\rho_y(u,v)}{\rho_a(u,v)} \qquad (9)$$

where u, v change over the image. Clearly, Q is illumination invariant. The importance of this ratio becomes clear by the following statement:

> Given three images $\mathbf{a_1}$, $\mathbf{a_2}$, $\mathbf{a_3}$ of object **a**, illuminated by any three linearly independent lighting conditions and an image y_s of y illuminated by some light sources, then there exists coefficient x_1, x_2, x_3 that satisfy:
>
> $$y_S = (\sum_j x_j a_j) \otimes Q_y \qquad (10)$$

where $\otimes$ denotes the Cartesian product (pixel by pixel multiplication).

We see that once Q_y is given, we can generate y_s (the novel image) and all other images of the image space of *y*. The key is to obtain the correct coefficients x_j which can be done by using a bootstrap. Let the bootstrap set of 3N pictures be taken from three fixed (linearly independent) not necessarily known light sources s_1, s_2 and s_3. Let $\boldsymbol{A_i}$, i = 1,..., N be a matrix whose columns are the three pictures of object $\boldsymbol{a_i}$ with albedo function $\boldsymbol{\rho_i}$. Thus, $A_1,...,A_N$ represent the bootstrap set of N matrices, each is a (nx3) matrix, where n is the number of pixels of the image. Let y_s be an image of some novel object y (not part of the bootstrap set) illuminated by some light source $s=\sum_j x_j s_j$. We wish to recover $x=\{x_1,x_2,x_3\}$ given the *N* matrices $A_1,..., A_N$ and the vector y_s. This can be done by solving a bilinear problem in the N+3 unknowns x and α_i, which can be obtained by minimizing the function:

$$f(x) = \frac{1}{2}\sum_{i=1}^{N} \left| A_i x - \alpha_i y_S \right|^2 \qquad (11)$$

for the unknown *x*. To find the desired global minimum we apply the Euler-Lagrange equation related with the variables *x* and *α*. This can be done by derivation of *f(x)* through these variables. We get the following relations:

$$x = \sum_{i=1}^{N} \alpha_i v_i \qquad (12)$$

 ISBN 978-80-86943-98-5

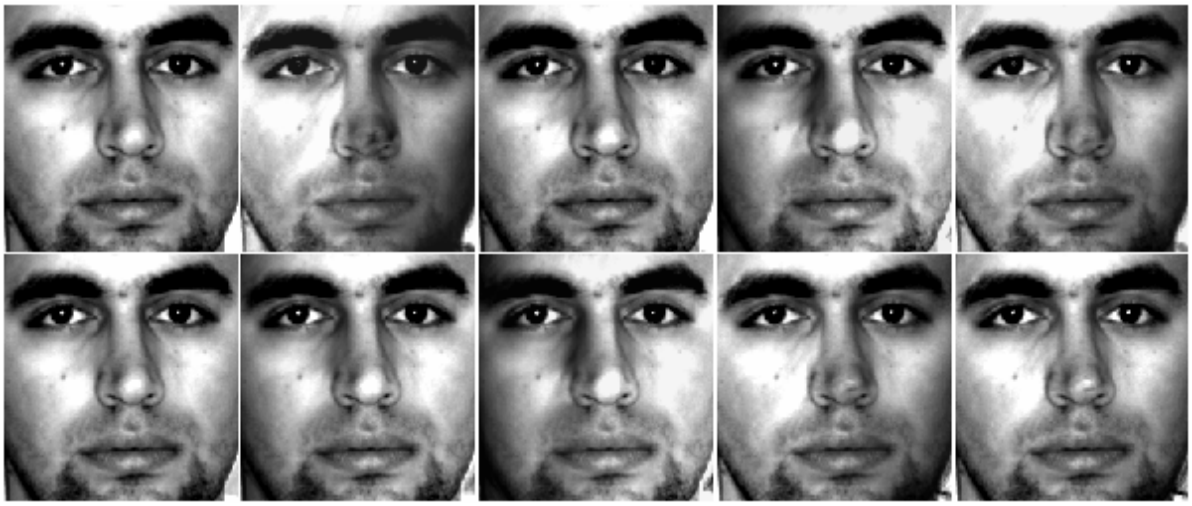


Fig. 1. Example of the image synthesis. The input image is in the upper leftmost position

$$v_i = \left(\sum_{r=1}^{N} A_r^T A_r \right)^{-1} A_i^T y_s \qquad (13)$$

$$\alpha_i y_S^T y_S - \left(\sum_{r=1}^{N} \alpha_r v_r \right)^T A_i^T y_S = 0 \qquad (14)$$

So we first find the v vectors (3x1) in Eq. (13), and we solve the homogeneous linear equation in Eq. (14) for the α_i. Then by using Eq. (12) the desired minimum can be found. Finally we compute the quotient image $Q_y = y_s / Ax$, where $\boldsymbol{A}$ is the average of $A_1, \ldots, A_N$. The image space is spanned by the product of images $\boldsymbol{Q_y}$ and $\boldsymbol{Az}$ for all choices of $\boldsymbol{z}$. An example of an output based on a training bootstrap (10 persons) from YaleB is given in Fig 1.

As we see, the results of this algorithm are quite satisfactory in spanning the image space of a given input image. This helps to overcome the SSS problem and creates the possibility to use LDA-based methods even when only one image per person is provided during the learning phase. An inherent assumption throughout the algorithm is that for a given pixel (x, y), n(x, y) is the same for all the images, i.e, the bootstrap set as well as the test images. The performance degrades when dominant features between the bootstrap set and the test are misaligned. As this step will occur after the face detection it is supposed that dominant features will have been depicted and aligned previously.

4. ITERATIVE METHOD IN ILLUMINATION RESTORATION

The major advantages of the algorithm explained in this section are that no facial feature extraction is needed and the generated face images will be visually natural looking. The method is based on the general idea that the ratio of two images of the same person is simpler to deal with than directly comparing images of different persons.

It uses a ratio-image, which is the quotient between a face image whose lighting condition is to be normalized and a reference face image. The two images are blurred using a Gaussian filter, and the reference image is then updated by an iterative strategy in order to further improve the quality of the restored face image. In this approach, a face image with arbitrary illumination can be restored so as to have frontal illumination.

Let I_{ik} denote a face image of the ith person captured under the s_k light source direction, where a light source is classified according to its direction. I_{r0} represents a face image of another person captured under the frontal light source s_0 and is used as a reference image. Then, we give the two blurred images of I_{ik} and I_{r0} denoted as B_{ik} and B_{r0}, respectively, as:

$$B_{ik} = F * I_{ik} = F * (\rho_i n_i^T s_k) = (F * \rho_i n_i^T) s_k \qquad (15)$$

$$B_{r0} = F * I_{r0} = F * (\rho_r n_r^T s_0) = (F * \rho_r n_r^T) s_0 \qquad (16)$$

where (*) is the convolution operation and F is a 2D Gaussian low-pass filter, with $\sigma_x = \sigma_y = \sigma$, given by the following formula:

$$F(x, y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \qquad (17)$$

As the shape and albedos of all faces are similar if the size of F is big enough, we can assume that $B_{i0} \approx B_{ro}$. By using the formulas (15)-(17) and this assumption, we can obtain the face image under frontal illumination for the ith person from I_{ik} captured under an arbitrary lighting direction s_k by:

 ISBN 978-80-86943-98-5

$$H_{io} = \rho_i n_i^T s_o \approx \rho_i n_i^T s_k \frac{\left(F * \rho_r n_r^T\right) s_o}{\left(F * \rho_i n_i^T\right) s_k} = I_{ik} \frac{B_{ro}}{B_{ik}} \quad (18)$$

H_{i0} is an estimation of I_{i0} or a restored image of I_{ik}. This approach can be summarized by the following algorithm:

1. A mean face image and an eigenspace Φ_S introduced in section 2 are computed based on a set of training images, which are all captured under frontal light source
2. An initial restored image can be calculated using Eq. (15)-(18), where the mean face image is used as the initial reference image and the size of the Gaussian filter F is 5. A reconstructed image is obtained from the initial restored image based on the computed eigenspace, and should have a smaller number of noise points.
3. An iterative procedure is used to obtain a final restored image with frontal illumination. During the iterative procedure, the reference image is updated with the new reconstructed image so as to obtain a visually better restored image.
4. The iterative procedure continues until a stopping criterion is met. In this approach the stopping criterion is the difference between two consecutive outputs of Eq. (18), or a specified maximum number of iterations.

5. EXPERIMENTAL RESULTS

All the experiments have been done using the YaleB database. Despite its relatively small size, this database contains samples from the whole illumination space and has become a testing standard for variable-illumination recognition-methods. This database consists of 10 distinct persons and 45 images for each person, divided in four subsets. The first subset includes 70 images captured under light source directions within a span of 12˚. The second and third subsets have respectively 120 images and 140 images each, captured under light source directions within 25˚ and within 50˚. The fourth subset contains 120 images captured under light source directions within 75˚. In all the images the position of the two eyes of each face were located and translated to the same position. The images were cropped to a size of 180x190. In order to improve the performance of dimensionality reduction and recognition, all the images were normalized to zero mean and unit variance. After that the pipeline was tested when histogram equalization (HE) and adaptive histogram equalization (AHE) was applied as a further preprocessing step. Table 1 shows the effect of the preprocessing on the recognition rate, where it is obvious that AHE gives the best result among these preprocessing techniques.

	No	HE	AHE
Recognition	43.4	74	81.5

Table 1. Results with the YaleB database (PCA used)

A wide range of experiments have been conducted to test the Quotient Image algorithm. First, synthesizing new images from any arbitrarily illuminated input image outside the YaleB database is considered by using a bootstrap consisting of 30 pictures of 10 persons of YaleB (Fig 1). Furthermore we examined the performance when only 15 images (5 persons), 9 images (3 persons) and 3 images (1 person) respectively were available in the bootstrap (Fig 2). In all these cases, after calculating the Quotient Image by means of $Q_y = y_s / Ax$, we create the image space of the novel image by the product of Q_y and Az for all choices of z. Some examples of the calculated **x** variable are given in Table 2.

Coeff/#Person	5	3	1
x_1	0.11302	0.23729	0.15915
x_2	0.38648	0.31587	0.45989
x_3	0.41526	0.35312	0.29723

Table 2. Coefficient results for different bootstrap combinations

Using these coefficients, we create the image space of the input image by randomly assigning different values for z or using a normal distribution around the original values of ***X***. From Fig. 2 we can see that a bootstrap consisting of 10 persons is quite consistent for creating the image space of an input image. Even when we reduce it by half, the results are quite satisfactory. This is because the albedos of possible faces occupy only a small part of the dimension in which they are spanned. Of course the larger the bootstrap size the more accurate will be the recovery of **x** and the quotient image.

In order to prepare the training set for the LDA process we create the image space of all 10 persons of the YaleB database. For these we used 15 images for bootstrap where the object being reconstructed has been left out. The results of the LDA step are given in Table 3 and Table 4. The final step of our approach is to reconstruct any incoming image in order to have a frontal illuminated image. In the experiments with the YaleB database subsets the results were almost identical to the frontal illuminated image for the first and second subset.

 ISBN 978-80-86943-98-5

Fig. 2. Image re-rendering results for different bootstrap combinations: (a) 10 persons (b) 5 persons; (c) 3 persons; (d) 1 person

The other two subsets where the illumination conditions are worse also produce high performance but it has to be stated that some noise and feature corruption became visible (Fig 3).

After the illumination restoration process the performance of the whole approach was tested with 450 input images. Different distance measurements were experimented with:

- Manhattan distance (L1- norm)
- Cosine angle between two vector representations
- Euclidian distance (L2 - norm)

The Euclidian distance gives the best results for this classification purpose when used in a one-nearest neighbor classifier (Table 3, 4).

In order to further increase the recognition rate several combinations during the training phase have been applied. For the LDA step the best performance was achieved when 10 synthesized images per object were available during the training phase and all the discriminatory feature vectors of the LDA-projection matrix were used (feature vectors of length 9). The use of a higher number of synthesized images slightly increases the performance. In all the experiments the results were compared with PCA because it is the most important discriminatory technique used when only one image per person is available during training.

	Subset1	Subset2	Subset3	Subset4	Total
HE+PCA (%)	100	97.5	66.42	44.16	74
HE+New (%)	100	100	92.8	91.66	95.56

Table 3. Recognition rates with histogram equalization preprocessing

	Subset1	Subset2	Subset3	Subset4	Total
AHE+PCA	100	100	83.57	50	81.55
AHE+New	100	100	100	95	98.6

Table 4. Recognition rates with adaptive histogram equalization preprocessing

 ISBN 978-80-86943-98-5

(a)

(b)

Fig. 3. Illumination restoration for images of Subset4 (up to 70◦) (a) Before preprocessing and illumination restoration (b) After illumination restoration

Obviously our proposed approach can significantly improve the recognition rates. Other methods have achieved high recognition rates on the YaleB database, but they require a large number of images for each person. A recent work [Zhan05] proposes an illumination invariant face recognition system from a single input image. It tries to solve the problem by using spherical-harmonics basis-images and they achieve very good results. However they specify that the computational burden of such an algorithm is high for the integration in a real time system.

6. CONCLUSION

A novel pipeline for dealing with illumination variation was introduced. In this work was aimed at a solution of the SSS problem of class-based discrimination problems. For this an image-space synthesis-method was explained and the image space of each image of the training set was created. After creating the image space of each training image, FLDA was applied in order to best use the discriminatory features of the system.

Every incoming image was processed with the illumination restoration algorithm and then a projection was done in order to extract the discriminatory features. The recognition rate with the YaleB database consisting of 450 images was 98.66% which can be considered very successful when compared to existing methods. Another approach [Geo01a] claimed 100% recognition rates in all data subsets, but seven images of each person in Subset1 have to be used to obtain the shape and albedo of the face.

As a conclusion, this work proposes an innovative approach for creating a robust face recognition system under varying illumination. This study offers the possibility of creating a real time system because it is not computationally complex. In the future this study will be extended to deal not only with upright frontal views but also with different poses. One possible approach based on this study is to apply multiple reference subspaces for different poses.

 ISBN 978-80-86943-98-5

ACKNOWLEDGEMENTS

The work presented in this publication is partially carried out as part of the PVG - Point-based Volume Graphics project supported by the Austrian Science Fund (FWF) under grant no. P18547-N04, and partly funded by TUBITAK (The Scientific & Technological Research Council of Turkey), project EEEAG-104E121.

7. REFERENCES

[Adi97] Adini, Y., Moses, Y., and Ullman, S.: Face recognition: The problem of compensating for changes in illumination direction. PAMI, Vol. 19, No. 7, pp. 721–732, 1997.

[Bel97] Belhumeur, P.N, Hespanha, J.P., and Kriegman, D.J.: Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. IEEE Trans. PAMI, Vol. 19, No. 7, pp 711-720, 1997.

[Chen00] Chen, L., Liao, H.M., Ko, M., Lin, J., Yu, G.: A new LDA-based face recognition system which can solve the small sample size problem. Pattern Recognition Letters, Vol. 33, pp 1713-1726, 2000.

[Gao06] Gao, H., Davis, J.W.: Why direct LDA is not equivalent to LDA. Pattern Recognition Letters, Vol. 39, pp 1002-1006, 2006.

[Geo01] Georghiades, A.S., Belhumeur, P.N., Kriegman, D.J.: From few to many: Illumination cone models for face recognition under variable lighting and pose. IEEE Trans. Pattern Anal. Mach. Intell. 23 (2) pp. 643–660, 2001.

[Geo03] Georghiades, A.S., Belhumeur, P.N., Kriegman, D.J.: Illumination-based image synthesis: Creating novel images of human faces under differing pose and lighting. IEEE workshop on Multi-View Modeling and Analysis of Visual Scenes, pp 47 - 54, 1999.

[Lai01] Lai, J.H., Yuen, P.C., Feng, G.C.: Face recognition using holistic Fourier invariant features. Pattern Recognition Letters, Vol. 34, pp 95-109, 2001.

[Liu05] Liu, D.H., Lam, K. M., Shen, L. S.: Illumination invariant face recognition. Pattern Recognition, Vol. 38, No 10, pp. 1705-1716, 2005.

[Lu03] Lu, J., Plataniotis, K.N., Venetsanopoulos, A.N.: Regularization studies of linear discriminant analysis in small sample size scenarios with application to face recognition. Pattern Recognition Letters, Vol. 26, No. 2, pp 181-191, 2005.

[Sha01] Shashua, A., Tammy, R.R.: The quotient image: Class-based re-rendering and recognition with varying illuminations. IEEE Trans. Pattern Anal. Mach. Intell. 23 (2), pp. 129–139, 2001.

[Tur91] Turk, M., Pentland, A.: Eigenfaces for recognition. Journal of Cognitive Neuroscience, Vol. 3, Num. 1, pp. 71-86, 1991.

[Wey04] Weyrauch, B., Heisele, B., Huang, J., Blanz, V.: Component-Based face recognition with 3D morphable models. IEEE Trans. CVPRW'04, Vol. 5, pp 85-90, 2004.

[Xie05] Xie, X., Lam, K.M.: Face recognition under varying illumination based on a 2D face shape model. Pattern Recognition, Vol.38, No 2, pp. 221-230, 2005.

[Yu00] Yu, H., Yang, J.: A direct LDA algorithm for high-dimensional data-with application to face recognition. Pattern Recognition Letters, Vol. 34, pp 2067-2070, 2000.

[Zhan05] Zhang, L., Wang, S., Samaras, D.: Face synthesis and recognition from a single image under arbitrary unknown lighting using a spherical harmonic basis morphable model. CVPR 2005, Volume 2, pp. 209 – 216, 2005.

[Zhao00] Zhao, W., Chellapa, R: SFS based view synthesis for robust face recognition. Proceedings of the 4th Conference on Automatic Face and Gesture Recognition, pp 285-292, 2000.

[Zhao03] Zhao, W., Chellapa, R., Philips, P.J., Rosenfeld, A.: Face Recognition: A literature Survey. ACM Computing Surveys, Vol. 35, No. 4, pp. 399-458, 2003.

Converting RGB Volume Data to Scalar Fields for Segmentation Purposes

Tetyana Ivanovska[†] Lars Linsen[‡]
School of Engineering and Science
International University Bremen[*]
Bremen, Germany

ABSTRACT

Most medical scanning techniques generate scalar fields, for which a large range of segmentation algorithms exists. Some scanning techniques like cryosections, however, generate color data typically stored in *RGB* format. Since standard segmentation algorithms such as isosurface extraction, level-set and region growing methods all have their advantages and drawbacks and many extensions and specializations of the algorithms have been developed to solve specific problems, one would need to generalize all these approaches to color data to have the full range of algorithmic solutions at hand. A more viable way to proceed is to convert the color data field to a scalar field in a preprocessing step, which allows for the direct application of all above-mentioned segmentation approaches. We propose a procedure that converts color to scalar data while preserving the properties that are important for segmentation purposes. We first convert the colors from *RGB* to $L^*a^*b^*$ color space, which separates the luminance channel from the chrominance channels and distributes the chrominance with respect to human perception. Then, we cluster the colors present in the data using a number of approaches and discuss the advantages and drawbacks. In order to assign to each cluster an appropriate scalar value, we use the ideas of the recently presented *Color2Gray* algorithm and generalize it for application to volume data. The Color2Gray algorithm in its originally proposed form is too inefficient to be applied to volume data, but a restructuring of the algorithm coupled with a prior clusterization step allows us to apply the algorithm even to large volume data. We segment the resulting scalar field using standard segmentation algorithms and discuss our results in comparison to standard conversion results.

Keywords: Color-to-scalar conversion, clusterization, segmentation.

1 INTRODUCTION

Visualizing medical imaging data is one of the traditional tasks in scientific visualization. The in vivo medical scanning techniques typically generate stacks of 2D grayscale images, where the grayscale values represent, for example, the tissues' densities. From the stack of images a 3D scalar field can be reconstructed. Examples of such medical imaging techniques are CT, MRI, PET, etc. More thorough examinations can be made when using ex vivo scanning techniques. The most prominent example would be cryosections. When generating cryosections the individual slices are scanned using digital photography. Thus, the resulting images are not grayscale but colored. Typically, they are stored in *RGB* color space. Reconstruction of the volumetric data set leads to a 3D *RGB* color field.

[*]Jacobs University Bremen as of spring 2007
[†]t.ivanovska@iu-bremen.de
[‡]l.linsen@iu-bremen.de

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.

Two main directions can be distinguished in the context of 3D medical imaging visualization, namely direct volume rendering and segmentation. While direct volume rendering displays the 3D data allowing for interactive change of the viewing parameters, segmentation extracts geometry that can be used for rendering purposes as well as for quantitative measurements and further processing. The main objective of the segmentation process is to extract boundary surfaces of certain objects, separating them from the surrounding tissues. In medical terms, one would like to extract the 3D geometry of the scanned organs. Standard segmentation algorithms include isosurface extraction, level-set methods, or region-growing approaches. These segmentation algorithms (as well as most direct volume rendering approaches) operate on 3D scalar fields. Generalization to color data is often not straihgt forward and only few attempts have been taken. Since different approaches are more or less suitable for different segmentation purposes, one would like to have the whole range of all segmentation algorithms generalized to color data. A more viable way to achieve this goal is to convert the color data into an appropriate scalar field without losing the property of being able to segment distinguishable regions.

While in the imaging community many sophisticated *RGB*-to-grayscale conversion algorithms exist, in visualization one mostly converts the *RGB* values to a color

 ISBN 978-80-86943-98-5

model with a separate luminance channel and uses the luminance channel for segmentation purposes. Obviously, the creation of a 3D luminance field leads to a loss in data information. In particular, two colors with same luminance but different chrominance are mapped to the same scalar value. Thus, any automatic segmentation algorithm would fail in separating two adjacent regions that are colored with the respective two colors. Any linear mapping that projects the 3D color space onto one axis (even chrominance-based approaches or Principal Component Analysis) would suffer from this drawback. We present a procedure for converting *RGB* color fields to scalar fields under the consideration that the scalar field will be used for segmentation purposes. Thus, we would like to distinguish all significant colors (especially when they occur in adjacent regions) while maintaining luminance order for colors with similar chrominance values. Our general approach is described in Section 3.

Our conversion method is based on two steps. In a first step, we analyze the color data in terms of occurring colors, the number of occurrences, and their distribution in the color space. We use the $L^*a^*b^*$ color space, since it separates the luminance channel from the chrominance channels and distributes the chrominance values such that the Euclidean norm in the $L^*a^*b^*$ space approximately captures perceptual dissimilarity. Exploiting the color distribution in the $L^*a^*b^*$ space for a given data set, we cluster the colors into n regions, where n is the predefined cardinal number of the range of the resulting scalar field.

We use several approaches for clusterization. One of them is based on a genetic algorithm. It produces excellent results but has high computational costs and, therefore, is not applicable to larger 3D data sets. We use it to validate with it other clusterization algorithms, e. g. axes-aligned binary space splitting, k-means, c-means, and median cut. The details are described in Section 4.

In the second step, a representative of each cluster has to be mapped to an appropriate scalar value. For this mapping we use the ideas of the recently presented *Color2Gray* approach [GOTG05]. The Color2Gray algorithm uses three parameters that determine the mapping in terms of luminance and chrominance. We can set these parameters such that the mapping has the desired before-mentioned properties.

The main drawback of the Color2Gray algorithm is its asymptotic computation time that is $O(N^6)$ for a $N \times N \times N$ volume data set. This runtime does not make it practical for 3D applications. However, when restructuring the algorithm and applying it to the clustered colors instead of the original image colors, we can reduce the computation time to $O(k^2)$, where k is the number of clusters, e. g. $k = 256$ for byte-sized output. Details are given in Section 5.

In Section 6, we give results that document the efficacy of our method. In particular, we compare them to luminance-based conversion and to other clusterization techniques common in the imaging community. We apply standard segmentation techniques to illustrate the advantages of our method when coupling the conversion with segmentation procedures.

2 RELATED WORK

The standard mapping for a color-to-grayscale transformation is the projection of the colors of a color image to the gray axis in the respective color space. Thus, color is mapped to its luminance. Such a method is inadequate for the pixels with same luminance but different chrominance. Several methods have been proposed for solving the general problem of reducing an n-dimensional set of data to m dimensions, where $m < n$. Since all standard color spaces have three dimensions, $n = 3$ and $m = 1$ in our case. Principal Component Analysis (PCA) is one of such methods [Jol02]. PCA can be used for computation of an ellipsoid in color space (principal components). Color values in the image can then be projected on a (luminance) axis defined by the primary axis of this ellipsoid. The efficacy of this method depends on the color space. However, PCA can also suffer from the problem of projecting colors of different chrominance to the same position an the axis.

Another way of generating grayscale images out of color images is to match local contrast in color images [SB02]. The contrast is regarded as a gradient and then the grayscale is recovered by solving a Poisson equation. This method has difficulties with certain classes of images as the global contrast influence is avoided. Moreover, local approaches do not work for our purposes, as one color may be mapped to different scalar values in different regions.

Another recently presented method maintained the proportionality between perceived color difference and perceived luminance difference, but ignores spatial arrangement of pixels [RGW05].

The goals of the listed approaches are indeed diverse, but none of them is targeted towards a subsequent segmentation step. Our approach instead aims for the necessary properties, which are to distinguish all significant colors and to maintain luminance order for colors with comparable chrominance values. We use a generalization of the recently developed two-dimensional Color2Gray algorithm [GOTG05] coupled with a clusterization / quantization procedure.

We have analysed a number of quantization algorithms which could be used for the prior quantization of the initial image before using the Color2Gray. The objective of color quantization is displaying a full color image with a restricted set of representative colors without a significant, i. e. perceptually almost not noticeable, loss of color impression. Colors are to be approx-

 ISBN 978-80-86943-98-5

imated as closely as possible when quantized. Quantization techniques consider quality criteria such as human perception, computation time, and memory requirements [Sch97]. We have investigated the following existing quantization algorithms: static color look-up algorithm using look-up tables [Hec82], popularity algorithm [Hec82], median cut algorithm [Hec82], k-means algorithm [Mac67], fuzzy c-means algorithm [Bez81].

The idea of static color look-up table algorithms is to divide the color cube into equally thick slices in each dimension. The crossproduct of these color levels can be used as the entities of the color look-up table. A significant drawback of this method are artifacts in form of edges in the resulting image. The main idea of popularity algorithms is to build a colormap by finding the K most frequently occurring colors in the original image. The colors are stored in a histogram. These K most frequently occurring colors are extracted and used as entries in the color table. The image is quantized with respect to that table. The question that remains is how to map the colors that appear in the original image but are not stored in the color table. A method that detects the most frequently used color out of the colors stored in the color table within a small neighborhood of the regarded pixel. Thus, in general, each pixel has to be tested to find the shortest distance to one of the K most frequently used color values. The main drawback of this method is that some important but "unpopular" image colors could be lost. The median-cut method was originally described by P. Heckbert [Hec82]. The idea behind it is to use each of the color in a synthesized look-up table to represent an equal number of pixels of the original image. The algorithm partitions the color space iteratively into subspaces of decreasing size. The algorithm starts with an axes-aligned bounding box that encloses all the different color values present in the original image. The "size" of the box is given by the minimum and maximum of each of the color coordinates that encloses the current box. For splitting the box one determines the dimension, in which the box will be (further) subdivided. The splitting is executed by sorting the points by increasing values in the dimension, where the current box has its largest edge and partitioning the box into two halves at the position of the median. Approximately equal numbers of points are generated on each side of the cutting plane. Splitting is applied iteratively and continued until K boxes are generated. The number K may be chosen to be the maximum number of color entries in the available colormap. The color assigned to each of the K boxes is calculated by averaging the colors of each box. The median-cut method performs well for pixels, whose colors lie in a high-density region of the color space, where repeated divisions resulted in cells of small size and, hence, small color errors. However, colors that fall in low-density regions of the color space are within large cells, where large color errors are to be expected. The main idea of the k-means algorithm is to define K centroids, one for each cluster. These centroids should be placed as far from each other as possible. Then each point from the initial data set is associated with the nearest centroid. When all the points have "their" centroids, the K centroids are recalculated as the average centers of each cluster.Then a new binding has to be done between the same data set points and the nearest new centroid. This loop is continued until no more changes are done. The idea of the fuzzy c-means method is similar to the k-means approach, but it allows one data point to belong to two or more clusters. Fuzzy partitioning is carried out through an iterative optimization of the data points membership in the clusters and the corresponding update of the cluster centers. The iteration terminates when there is no difference between the membership results with respect to some given precision. The results of the k-means and c-means approaches depend on the choice of the initial centroids. We take as initial centroids the data points that are distributed as far as possible from each other in the color space.

3 GENERAL APPROACH

We present a procedure for converting color data to a scalar field in a way that is amenable for subsequent segmentation of the volume. Any segmentation technique may be applied to the resulting scalar field.

Our conversion approach consists of three main steps. Assuming that the 3D color data is given in form of *RGB* data, we first convert the *RGB* values to a color representation in the $L^*a^*b^*$ color space. Secondly, we apply a quantization step that reduces the number of used colors in the given color data from their original number to a typically much smaller number of colors in the same color space. The reduced amount of colors is chosen with respect to the number of distinguishable output values in the to be generated scalar field. When considering a scalar field that allows us to store 1 byte information per sample point, we would set the reduced number of colors for quantization to 256. The quantization step assures that all important colors can still be distinguished after quantization. The quantization step also generates a unique mapping of each color of the original color set to one color of the reduced color set. Finally, the reduced set of colors is mapped to scalar values in a way that luminance order is preserved for colors with similar chrominance values.

Operating in *RGB* space is inadequate for our purposes, since it does not distinguish between luminance and chrominance. Several color models such as *HLS* or *HSV* have this property and are widely used for this particular reason. However, we decided to use the $L^*a^*b^*$ color space, because its Euclidean norm closely correspond to perceptual dissimilarity [Pas03]. *CIE* $L^*a^*b^*$

 ISBN 978-80-86943-98-5

is the most complete color model used conventionally to describe all the colors visible to the human eye. It was developed for this specific purpose by the International Commission on Illumination or Commission Internationale d'Eclairage (CIE). The three parameters in the model represent the luminance L of the color, where $L = 0$ yields black and $L = 100$ indicates white, its position a between red and green, where negative values indicate green while positive values indicate red, and its position b between yellow and blue, where negative values indicate blue and positive values indicate yellow.

Assuming that the original data is given in *RGB* color space, we need to convert the colors to a representation in the $L^*a^*b^*$ color space. To do so, we first transform the *RGB* data to the *CIE XYZ* color space and, afterwards, convert the *XYZ* values to $L^*a^*b^*$ colors. Note that the matrix of transformation from *RGB* data to *XYZ* depends on the chosen *RGB* standard. We consider the *R*709 *RGB* standard. Hence, the three channels of the $L^*a^*b^*$ colors are computed by

$$
\begin{aligned}
L^* &= \begin{cases} 116 \cdot \left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - 16, & \text{if } \frac{Y}{Y_n} > 0.008856 \\ 903.3 \cdot \left(\frac{Y}{Y_n}\right), & \text{otherwise} \end{cases}, \\
a^* &= 500 \cdot \left(\left(\frac{X}{X_n}\right)^{\frac{1}{3}} - \left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} \right), \text{ and} \\
b^* &= 200 \cdot \left(\left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - \left(\frac{Z}{Z_n}\right)^{\frac{1}{3}} \right),
\end{aligned}
$$

where X_n,Y_n,and Z_n are the values of X,Y,and Z, respectively, for a specified reference of the white, i. e. illuminant, color, and X, Y, and Z are computed by

$$
\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{pmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{pmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}
$$

For the quantization step, we first choose the number K of clusters to be generated, which is equal to the number of distinguishable values in the outputted scalar field. If the number of unique colors in the image is less than the available output values, the quantization step can be skipped. Each color defines its own cluster. However, for real data we are dealing with, this case does never occur in practice when using byte-sized output values. Since we are computing a scalar field rather than a greyscale image, we are not limited to typical image format sizes. Any number of output clusters can be chosen. An increasing number of output clusters only affects the performance of the subsequent conversion algorithm.

Given the number K of output colors, we execute some clusterization algorithm in order to generate K clusters of colors and determine the representative colors for each cluster. The clusterization algorithms we have developed are described in Section 4.

The actual quantization of the picture is performed by substitution the initial color value by the value of the representative of the cluster, to which the initial color value has been assigned. The output of this step is a color data set in $L^*a^*b^*$ color space that only makes use of a restricted number of K different colors.

Finally, K representative colors of the clusters are to be mapped to K scalar values. For this purpose we use the Color2Gray algorithm. However, we do not apply the Color2Gray algorithm directly, as the computational costs are too high for running this algorithm on standard-sized 3D data sets. Instead, we apply the mapping idea of the Color2Gray algorithm to the K representative colors of the cluster. Since the Color2Gray algorithm can be fine-tuned by three intuitive parameters, we examine, which of these parameters would be most suitable for our purposes, such that all significant parts in the data are distinguishable. Details on this last step are given in Section 5.

The final output of our processing pipeline is a scalar field of same dimensions as the initial color data field. We use the scalar values for segmentation purposes. We apply standard segmentation algorithms for a proof of concept.

4 CLUSTERIZATION

We introduce two novel clusterization techniques and compare them to the clusterization techniques described in Section 2. We came up with novel methods ourselves, as we observed that the known clusterization methods we applied did not lead to fully satisfiable results. We discovered that some important color differencies can get lost during clusterization. This observation is due to the fact that algorithms such as median cut yield only average quality results. We achieved very good results while using k-means and c-means clustering algorithms on some of our examples.

We propose two novel clusterization methods for a more adequate clustering. We first propose a genetic algorithm that generates high-quality results and can be used as a standard for other clusterization algorithms. For the genetic algorithm, we have to define the genetic material, its initialization, the update rules that are iteratively applied, and the fitness function.

The genetic material of the individual is stored in a chromosome made up of basic genes which define the physical features of the individual. A previously presented genetic clusterization algorithms [TAE98] takes the mapping of all sample colors to the K palette colors as a genetic chromosome. This choice leads to an extremely high memory consumption, as each chromosome consists of a number of genes equal to the number of unique colors in the picture. In our algorithm, we take the centers of all cluster, i. e. points in the $L^*a^*b^*$ color space, as a gene. Thus, a chromosome consists of K genes only, where K is the number of clusters.

 ISBN 978-80-86943-98-5

For the initial population generation we build the median cut tree, take the cubes lying on the depth equal to $log_2(K)$, where K is the number of clusters, and determine which unique colors of the image belong to which cube. Then, we find the average centers of each cluster as it is done in the median-cut algorithm. The initial population is formed by the centers of each cube.

We define an overall fitness function F_l that characterizes each individual l by

$$F_l = \sum_{m=1}^{K} \left\{ D(\bar{c}_m, \tilde{c}_m) + \max_{i,j \in P_m} (D(i,j)) \right\},$$

where $\bar{c}_m$ is the average center of the image colors, which belong to cluster m, $\tilde{c}_m$ is the cluster center chosen by the genetic algorithm, D is the Euclidean distance function in the $L^*a^*b^*$ color space, and P_m is the set of the colors that belong to cluster m.

We minimize this function for the individuals by generating new populations using reproductions (mutations, crossovers) and comparing the fitness functions.

On the test examples the genetic algorithm achieved a clusterization that distinguishes all important colors and generates an equal distribution among the remaining colors.

Unfortunately, the computation times for the genetic algorithm tend to be rather high, such that it is not practical to apply it to larger 3D data sets. Therefore, we will use this algorithms only as a standard, to which we compare the results generated by faster clusterization algorithms. The goal was to develop a clusterization algorithm which is much faster than the genetic clusterization algorithm and, at the same time, more adequate than median cut, k-means and c-means algorithm in certain cases.

We developed the axes-aligned binary-tree partitioning approach for clusterization. Like in median cut algorithm our algorithm starts with an axes-aligned box that encloses all the different color values from the original image in the $L^*a^*b^*$ color space. The "size" of the box is given by the minimum and maximum of each of the color coordinates that enclose the box . For splitting the box we have to decide, in which dimension we want to perform the splitting step. We choose the dimension, in which the box has its longest edges. The points are sorted in this dimension. In this order we are looking for the largest gap in the splitting dimension. The partitioning of the box into two halves is done at the lower end of that largest gap. We iterate these splitting steps until all unique colors that have a representative in the given image are split. This procedure forms a binary tree with the unique colors stored in the leaves. As a result, pairs of closest colors are the pairs stored in leaves with the same parent. Note that the binary tree is typically unbalanced.

In order to determine K clusters out of the binary-tree structure, we iteratively merge leaves of the tree. In each step, we merge the two leaves, whose colors have the minimal Euclidean distance in $L^*a^*b^*$ color space. The color assigned to the new leaf is the average color of the colors stored in the leaves that have been merged. The average color c_{av} is computed by

$$c_{av} = \frac{1}{M} \cdot \sum_{i=1}^{M} c_i \,, \tag{1}$$

where M is the number of all leaves which belong to this subtree, and c_i are the colors that have been stored in the leaves. We apply this step until the number of leaves is equal to the number of cluster centers.

In general, the decision which method to take depends on the initial data.

5 COLOR-TO-SCALAR CONVERSION

The most common way of converting *RGB* images to grayscale in terms of image processing or converting an *RGB* color data set to a scalar field in terms of visualization, is to operate on the luminance. First, the image colors are converted from *RGB* to a color space with a luminance channel, for example, to the $L^*a^*b^*$ color space. Then, luminance values are taken as resulting scalar values. More sophisticated techniques for mapping the 3D color space to one particular axis have been developed, but all these methods are ineffective at preserving different colors orthogonal to the chosen axis. Often the axis is oriented close to the luminance axis such that isochromatic colors are projected to similar regions on the axis.

The Color2Gray algorithm allows to take into account both luminance and chrominance differencies in a source image and construct an appropriate grayscale image. It was introduced for the conversion of 2D images, but could be generalized to voume data.

The user can influence the output of the Color2Gray algorithm using three simple and intuitive parameters. The first parameter θ controls whether chromatic differences are mapped to increases or decreases in luminance value. The second parameter α determines how much chromatic variation is allowed to change the source luminance value. The third parameter μ sets the neighbourhood size used for chrominance estimation and luminance gradients.

The color differences between pixels in the color image are expressed as a set of signed scalar values. The differences are measured in the various channels of the $L^*a^*b^*$ color space. Thus, both luminance and chrominance differences are computed. The generation of the output in form of a grayscale version of the image is based on these differences. For each pixel i and each neighbor pixel j, the signed distance scalar δ_{ij} based on luminance and chrominance differences is computed by

$$\delta_{ij}(\alpha,\theta) = \begin{cases} \triangle L_{ij}, & \text{if } \|\triangle L_{ij}\| > crunch(\| \triangle \vec{C}_{ij} \|) \\ crunch(\| \triangle \vec{C}_{ij} \|), & \text{if } crunch(\| \triangle \vec{C}_{ij} \|) \cdot \vec{V}_\theta \geq 0 \\ crunch(- \| \triangle \vec{C}_{ij} \|), & \text{otherwise} \end{cases}$$

 ISBN 978-80-86943-98-5

where L_i is the luminance of ith pixel, $\triangle L_{ij} = L_i - L_j$, $\| \triangle \vec{C}_{ij} \|$ is the Euclidean norm of the vector $\triangle \vec{C}_{ij} = (\triangle A_{ij}, \triangle B_{ij})$ with $\triangle A_{ij}$ and $\triangle B_{ij}$ being the differencies between pixels i and j in the chrominance channels a^* and b^*, respectively, $\vec{V}_\theta = (cos\theta, sin\theta)$ is a normalized vector defined by θ, and $crunch(x) = \alpha * tanh(\frac{x}{\alpha})$. For a detailed derivation of the formula, we refer to the original work on the Color2Gray algorithm [GOTG05].

Given a set of signed differencies δ_{ij} for pixel pairs (i, j) of an ordered set S, a scalar field g is computed such that g minimizes a target function $f(g)$. The target function $f(g)$ is given by

$$f(g) = \sum_{(i,j)\in S} ((g_i - g_j) - \delta_{ij})^2 \, .$$

Hence, the minimization problem can be written in the form of

$$min(f(g)) = min\left(\frac{1}{2}\sum_{(i,j)\in K} (x_i - x_j - b_{ij})^2\right) \tag{2}$$

where the x_i, x_j, and b_{ij} can easily be derived. This is a least-squares problem of the form

$$min\left(\frac{1}{2}(Ax-b)^T(Ax-b)\right) \, ,$$

which can be rearranged to

$$min\left(\frac{1}{2}x^T A^T Ax - (A^T)^T x + \frac{1}{2}b^T b\right) \, .$$

This equation is quadratic with a symmetric, positive semi-definite Hessian. Therefore, minimizing it is equivalent to satisfying the linear equation:

$$A^T Ax = A^T b \, .$$

Deriving the terms d_k on the right-hand side of the equation, we obtain

$$d_k = \left[A^T b\right]_k = \sum_{j\geq k} \delta_{kj} - \sum_{i<k} \delta_{ik} \, .$$

Because of the regular form of the Hessian, $A^T Ax = A^T b$ expands to

$$(N-1)\cdot x_k - \sum_{l\neq k} x_l = d_k$$

with A being a $N \times N$ matrix. For any two indices i and j, we obtain

$$d_i - d_j = ((N-1)\cdot x_i - x_j) - ((N-1)\cdot x_j - x_i)$$

leading to

$$x_i = \frac{d_i - d_j + N\cdot x_j}{N} \, . \tag{3}$$

For any $x_i = c$ there is exactly one solution to Equation 2, which may be obtained by taking any known minimal vector x', and shifting all of its elements by $x_i - x'_i$. So, the problem can be solved by setting $x_0 = 0$ and getting all other x_i from Equation 3. Then, the found grayscale values are shifted to be as close to the source luminances as possible.

The bottleneck of the algorithm is the calculation of the coefficients d_k. The cost of the calculations is $O(N^6)$ for a $N \times N \times N$ volume image. Using local variants of the algorithm by adjusting parameter μ is inappropriate for our purposes, as one color value should always be assigned to the same greyscale value, which is only assured by using a global version of the algorithm.

However, if we do the prior clusterization of the image, the calculations can be reduced dramatically. After the quantization of the image we are left with arrays of length K, where K is the number of generated clusters. Let the output generated by the cluster be given in form of the color values of each cluster's center stored in array *Centers*, the number of occurrences of colors from each cluster stored in array *Occurs*, and the indices that assign to each pixel of the color the appropriate cluster stored in array *Indices*.

For each cluster k we can calculate d'_k by

$$d'_k = \sum_{j\geq k} \delta_{kj} * Occurs[j] - \sum_{i<k} \delta_{ik} * Occurs[i] \tag{4}$$

The δ_{kj} and δ_{ik} are computed on the cluster colors only using the information stored in array *Centers*. The cost of this calculation is $O(K^2)$. Thus, it only depends on the typically small number of clusters and is independent of the number of pixels/voxels in the image. In particular, it does not matter, of which dimension the original image is. Our approach scales to arbitrary dimensions, as it only operates on the clusters and their centers.

Finally, for each pixel or voxel i of the original 2D or 3D image, we get the desired value d_i by determining the cluster it has been assigned to and using the respective value d'_k. Thus, we retrieve $d_i = D[Indices[i]]$, where D is the array that stores the d'_k we computed for all clusters k. These values d_i are used, as before, to compute the x_i from Equation 3.

By bringing down the computational costs for the bottleneck computation from $O(N^6)$ to $O(K^2)$, we significantly speed up the procedure, which makes it applicable to larger 3D data sets.

6 RESULTS AND DISCUSSION

Since the improvement of our approach over luminance-based conversion models can be documented best by looking at 2D images, we first want to give some examples, where we convert individual slices through a 3D color data set.

For one of the examples we use a part of a horizontal slice through the Visible Female data set[1]. The data set is obtained by taking cryosections of an entire female human body. The first image of the Figure 2 shows a red organ surrounded by a yellowish tissue. The luminance of the surrounding tissue varies, but the chrominance values of the surrounding yellow region are clearly distinguishable from the chrominance value of the red organ.

[1]Data set courtesy of the National Institute of Health.

 ISBN 978-80-86943-98-5

Figure 2 shows a conversion of the slice to a greyscale image using a luminance-based approach on the left-hand side and our approach on the right-hand side. Using the luminance-based approach, the boundary of the initially red region to the left gets lost. Using our approach, the initially red region is still clearly distinguishable from the surrounding tissue.

For generating the results of our approach throughout the paper, we used the axes-aligned binary-space partitioning or median cut clusterization and the following Color2Gray parameters: parameter $\alpha = 40$ and parameter $\theta = \frac{\pi}{4}$ or $\theta = \frac{3\pi}{2}$. Parameter μ always has to chosen such that the entire image is considered as a neighborhood to obtain a global approach.

For the generation of Figure 3 we applied a segmentation algorithm to the images of Figure 2. For segmentation purposes, we used a standard approach for isosurface segmentation based on marching squares for 2D images and marching cubes [LC87] for 3D images. However, any other segmentation approaches such as other isosurface extraction methods, level-set methods, or region-growing approaches could be used instead. In Figure 3, the contours are shown in red. The figure illustrates that the segmentation algorithm was not able to segment the initially red region when applied to the image converted by the luminance-based approach on the left-hand side. When applied to the image generated with our conversion algorithm, the region can be segmented well.

In Figure 1 on the left-hand-side, we show a part of a slice of a cryosection of a Macaque monkey brain.[2] The brain slices have been digitized using high-resolution digital photography that allows to scan even very small structures up to single neurons. We pick a region of interest with different tissues. The encircled tissue of interest has a purple color and is surrounded by tissue with brownish color shades.

The next two images of Figure 1 show greyscale images after conversion using a luminance-based approach (second) and our approach (third). Again, our method manages to let the tissues distinguishable, while the boundary of the purple tissue gets lost when using the luminance-based approach, as the luminance values for the purple and the brownish tissue were about the same. The last two images of Figure 1 show the results when applying a segmentation algorithm to separate the two tissues. Segmentation after luminance-based conversion only leaves us with a segmentation of the black spots. The purple regions cannot be segmented. Segmentation after our conversion allows for segmentation of the purple tissue.

For the generation of Figure 4 we used a data set of a cancer cell that has been scanned using fluorescence microscopy[3]. The 3D cancer cell data set is used to extract the yellow regions. The results are shown in Figure 4. Our conversion approach allows us to exactly extract the yellow regions by using isosurface extraction (middle), while the segmentation run on the luminance-based converted scalar field (right) extracts much larger regions that also include many originally green and red parts.

The results document that our color-to-scalar conversion method allows us to convert data sets with neighbored isoluminant areas that can still be separated by a standard segmentation method after conversion. Obviously, when the original color data only contains colors that vary in their luminance, a luminance-based conversion produces an optimal conversion that cannot be improved by our approach.

7 CONCLUSIONS

We have presented an approach for conversion of *RGB* color data to scalar data that is amenable for subsequent segmentation of the scalar field. In particular, our conversion method does not map isoluminant colors to the same scalar value, but allows us to preserve all important colors such that they are still distinguishable by a segmentation algorithm after conversion. Moreover, the order in luminance for isochrominant colors is maintained during conversion.

Our method operates in $L^*a^*b^*$ color space and uses clusterization algorithms for quantization. Afterwards, the quantized colors are mapped to the appropriate scalar values. We have presented two novel clusterization methods based on a genetic algorithm and an axes-aligned binary space partitioning. For the final assignment of the cluster colors to the scalar values we used ideas from the Color2Gray approach. By only applying the algorithm to the clustered data, we were able to achieve computation times that also allow for the application of our methods to larger 3D data sets.

We presented results of 2D and 3D color data set that document the improvements over other approaches when converting data sets with different isoluminant colors. We have applied standard segmentation techniques to show how the subsequent segmentation improved when using our approach. Our approach allows us to apply the entire range of all segmentation techniques to color data via our conversion step. This is obviously a more viable way than generalizing all existing and useful scalar field segmentation approaches to color volume data, which may not be straightforward for many of them.

REFERENCES

[Bez81] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press - New York, 1981.

[2]Data set courtesy of Edward G. Jones, Center for Neuroscience, University of Davis, California.

[3]Data set courtesy of the Department of Medicine, University of California, San Diego.

 ISBN 978-80-86943-98-5

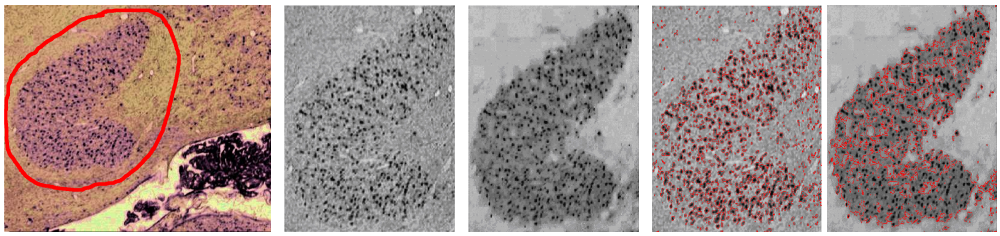


Figure 1: *First*: Cryosection of Macaque monkey brain with encircled region of interest. *Second*: Converted images using luminance-based approach. The two tissues cannot be distinguished anymore. *Third*: Converted images using our approach. The two tissues can clearly be separated by a segmentation algorithm. *Fourth*: Segmentation after luminance-based conversion cannot detect the formally purple region. *Fifth*: Segmentation after conversion with our approach allows to separate the tissues.

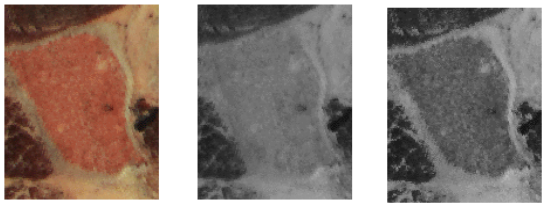


Figure 2: Comparison of converted images using luminance-based approach (middle) and our approach (right) when applied to the region of interest (left). Using the luminance-based approach, the boundary of the initially red region to the left gets lost. Using our approach, the initially red region is clearly distinguishable from the surrounding tissue.

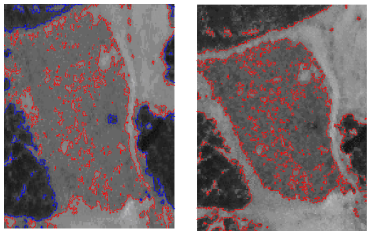


Figure 3: Comparison of segmentation of the converted images from Figure 2. For the image generated by a luminance-based conversion, the segmentation of the initially red region fails (left). For the image generated with our conversion algorithm, the region can be segmented very well (right).

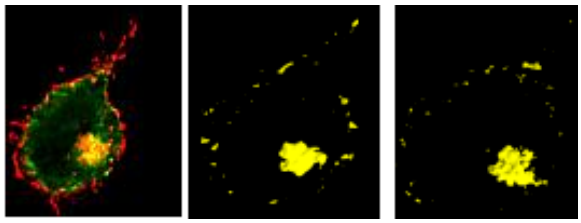


Figure 4: Fluorescence microscopy data set of a cancer cell (left). 3D segmentation of yellow regions via our color-to-scalar conversion approach (right) works well, while the segmentation after luminance-based conversion (middle) does not separate the yellow from th green and red regions completely.

[GOTG05] Amy A. Gooch, Sven C. Olsen, Jack Tumblin, and Bruce Gooch. Color2gray: salience-preserving color removal. *ACM Trans. Graph.*, 24(3):634–639, 2005.

[Hec82] P. Heckbert. Color image quantization for frame buffer display. In *Computer Graphics (Proceedings of ACM SIGGRAPH 82)*, pages 297–307, 1982.

[Jol02] I. T. Jolliffe. *Principal Component analysis*. Springer - Verlag, 2002.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, volume 21, pages 163–169, July 1987.

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[Pas03] D. Pascale. *A Review of RGB Color Spaces...from xyY to R'G'B'*. Babel Color, 2003.

[RGW05] Karl Rasche, Robert Geist, and James Westall. Re-coloring images for gamuts of lower dimension. *Eurographics/Computer Graphics Forum*, 24(3), 2005.

[SB02] D. A. Socolinsky and L. B.Wolff. Multispectral image visualization through first-order fusion. *IEEE Transactions on Image Processing*, 11(8), 2002.

[Sch97] P. Scheunders. A comparison of clustering algorithms applied to color image quantization. *Pattern Recognition Letters*, 18, 1997.

[TAE98] Tolga Tagdizen, Lale Akarun, and Cem Ersoy. Color quantization with genetic algorithms1. *Signal Processing: Image Communication*, 12, 1998.

 ISBN 978-80-86943-98-5

Estimation of Planar Surfaces in Noisy Range Images for the RoboCup Rescue Competition

Sarah Steinmetz
Active Vision Group
University of Koblenz-Landau
Universitätsstr. 1
56070 Koblenz, Germany
xena@uni-koblenz.de

Johannes Pellenz
Active Vision Group
University of Koblenz-Landau
Universitätsstr. 1
56070 Koblenz, Germany
pellenz@uni-koblenz.de

Dietrich Paulus
Active Vision Group
University of Koblenz-Landau
Universitätsstr. 1
56070 Koblenz, Germany
paulus@uni-koblenz.de

ABSTRACT

In this contribution we present a comprehensive method to extract connected planar polygons from a range image acquired by a laser range camera. The result is a boundary representation of the objects in the scene. The approximation of the detected planes by three–dimensional polygons can be used as a source for feature–based registration of sequential range images. We focus on the processing of range images that are segmented into planar regions. The polygon of each plane is extracted by incremental line fitting on the 2D contour of the segmented region projected onto the xy–plane followed by the propagation to the corresponding 3D–plane. We present a novel idea for joining these three–dimensional polygons. Due to sparsely distributed depth values of inclined planes and noise in areas of object edges, some planes cannot be segmented completely. Therefore object edges that actually represent one edge drift apart in 3D. Such edges are detected and joined. The direction in which each edge is moved, is determined by a confidence measure, depending on the slope of the plane.
We describe our experiments on range data of scenes containing planar as well as curved surfaces and give quantitative results. The estimates are compared to measurements that were manually taken. For cubic objects we compare the angles of the estimated polygons to the expected orthogonality. For approximately 70 planar surface patches we get an average error of 5 degrees. 83 percent of successfully segmented lines are joint correctly by the algorithm.

Keywords: 3D model reconstruction, boundary representation, range image processing, computer vision.

1 INTRODUCTION

Many publications propose and compare algorithms for the segmentation of dense range images into planar patches. An extensive survey of different techniques can be found in [HJBJ+96] and [CC05]. Here, we go further and process the segmented range images in order to extract a boundary representation of the scene from a single view. For our future goal, the autonomous indoor 3D map generation done by a robot at the RoboCup Rescue League, we are interested in the representation of the boundaries of planes as planar polygons. Such features may also be used for the registration of sequential range images. Concerning the problem of finding a high–level description of the segmented surfaces, there exist comparatively few techniques. In [BS92], after the segmentation of the range image, roof edges are built by the segments of the intersection lines of neighboured planes, whose 2D projection is verified by the roof edge contour pixels of these planes. Vertices result from the intersection of three planes. Step edges are found by 2D Hough clustering and a line tracing on the step edge contour pixels. The technique of intersection of surfaces can also be found in [FEF95], [Liu93] and [Koc96]. In section 3 we will see that this method is not applicable to our segmented range images due to to the noisy range data and unreliable orientation of some segmented planes. The approach in [HGB95] determines the relationship between planes (connected by edge or vertex) as well, by classifying the boundary pixels between regions as roof or jump edge pixels in an early processing step in order to determine which planes to intersect. However, it is demonstrated that one cannot fully trust in the correctness of these intersections, thus the authors use so–called "glue patches" in order to handle the problem. With this method they even connect vertices that result from the intersection of more than three surfaces by a glue patch. We chose another approach to avoid faulty intersection of planes by determining common edges of planes after the extraction of the planar polygonal boundary of each plane. This higher–level description is more reliable than working on pixel level as we show in section 2.
Our approach expects a segmented range image as input. For the segmentation of the range images we use a modified reimplementation [Zin01] of the approach developed by Liu [Liu93]. The segmentation returns 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.
Copyright UNION Agency - Science Press, Plzen, Czech Republic.

 ISBN 978-80-86943-98-5

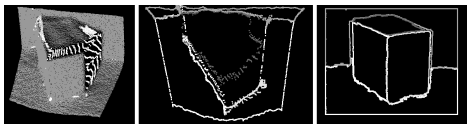


Figure 1: Left: outliers (*white points*) that are not assigned to any region, middle: the 3D contour coordinates for every region, right: orthographic projection of the contour coordinates onto the *xy*–plane

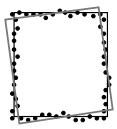


Figure 2: Incremental line fitting

region image that labels each pixel with the identifier of the region it belongs to and the parameters of the planes that have been fitted through the points of the regions. Some segmentation results are shown in Figure 7 in the lower left of each scene. After the segmentation there are still some erroneous outliers that actually belong to a region. The white points in the top plane of the box in the left image of Figure 1 represent such outliers. During the outlier elimination of the segmentation process the points remain marked as outliers due to their sparse distribution. The figure shows that range values in some visible areas, like the top of the displayed box, are as sparsely distributed as in areas of range values that represent artefacts. The white points between the front of the box and the wall represent artefacts. Particularly the areas of object edges are very noisy so that one can hardly distinguish a step edge from a roof edge by analyzing the neighbourhood of the border of a surface. Thus, in contrast to the approaches in [FEF95] and [HGB95], we do no initial classification of boundary pixels in order to determine the relationship between planes. Instead, we first determine the polygon for each plane (section 2), after that we analyze the relationship between edges in order to join common edges and vertices (section 3).

As imaging device we use a laser range camera, that provides dense range images with depth values in cartesian coordinates. Section 4 gives further information on the range images and the current results.

2 3D POLYGON EXTRACTION

2.1 2D boundary extraction

The first step is to determine the contour points of each region. For the boundary extraction we implemented the crack following algorithm [Kin97]. To identify the starting coordinate for this contour following method, the bounding box of the corresponding region is used. Starting from the bottom right corner the first pixel of the region is found by moving to the left. This procedure allows to assume that the contour starts with a corner pixel. The result is a sequential, counterclockwise list of the contour pixel coordinates for every region. The range image contains the *z*–coordinate for each pixel, thus the corresponding 3D contour coordinates are directly given, as shown in Figure 1.

2.2 Determination of 3D polygons

To determine the 3D polygon representing a plane, the best fitting lines through the contour points are needed. As depicted in Figure 1, the orthographic projection of the contour points onto the *xy*–plane represents the actual outline of the surface much better than the 3D coordinates. Thus the fitting is done in the 2D space, and the result is transferred to the 3D space.

In [NMTS05] several line extraction algorithms are compared. Incremental Line Fitting and Split and Merge are evaluated as best in speed and correctness. We implemented the incremental line fitting. With noisy data the standard incremental line fitting has a disadvantage: If a high residual threshold is chosen in order to avoid that a noisy point sequence is divided into many short segments, the resulting polygon is skewed, as depicted in Figure 2: the desired result is shown (*black rectangle*) overlayed to the result with high residual threshold (*skew, grey rectangle*).

Thus we extended the algorithm as follows: Every time when the next point is added to the fitting line, the current fitting error is compared to the fitting error of the previous step. If the new error is larger than the previous plus an offset, the current line is considered as getting worse. With the offset one can control the strictness of the judgement. When the current line finally exceeds the root–mean–square deviation threshold, not only the last bad point is put back, but all points since the fitting error became from bad to worse. Consequently one can choose a high fitting error threshold and fit lines through noisy data points, without fitting beyond corners. As line fitting we implemented an Orthogonal Distance Regression Fitting as described in [AV05]. The incremental line fitting results in a list of straight lines, the intersections of these subsequent lines represent the vertices of the desired polygon. The resulting 2D–polygon is transferred to 3D by computing the *z*–coordinate of the vertices using the parameters of the associated plane. If there are subsequent 3D–lines that are nearly parallel, the intersections are located outside of the polygon. Thus two lines that enclose an angle that is smaller than a threshold are merged by refitting the 2D–points of both lines. Figure 3 shows the result of the polygon extraction. The advantage of our approach is that we get closed linear contours (polygons) for each plane, thus we do not need any vertex refinement in this step.

3 JOINING OF POLYGONS

As shown in Figure 3 there are large gaps between polygons that should be actually connected. Especially the top of the box did ”move backwards”. Since some of such gaps are as large as distances between edges that do not belong together, a distance criterion in 3D space would be inappropriate, in order to detect which polygons (more precisely: which edges and vertices) should

 ISBN 978-80-86943-98-5

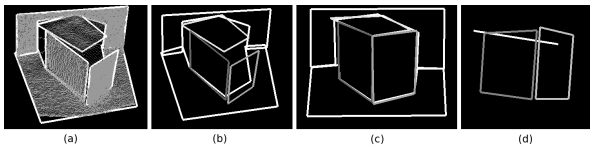


Figure 3: Result of the incremental line fitting on contour points of each plane. (a) with original range data, (b) extracted polygons, (c) orthographic projection onto the *xy*–plane, (d) the incoherent polygons of the box, view along the top plane

be joined. Again the analysis of the 2D orthogonal projection onto the *xy*–plane, compare Figure 3 (c), is advantageous. Criteria for joining two edges:

1. The 2D edges are nearly parallel.
2. The midpoints of the 2D edges are near.
3. The 2D edges have nearly equal length.
4. The 3D edges are nearly parallel.

But how to join two edges? Similar to the approaches mentioned in section 1, one could replace the concerned edges by an adequate part of the intersection line of the two planes that share the edge. Figure 3 (d) shows that this would cause an incorrect result concerning the shape of the box due to the erroneous orientation of the top plane.
Planes whose normal encloses a large angle with the optical axis of the camera, cannot be segmented completely due to sparse range data near edges, resulting in an erroneous orientation of the fitted plane. The idea is to adapt the orientation of such planes during the joining of edges. In order to be able to decide which plane should be adapted, a confidence value is assigned to every plane first. The chosen confidence measure is the angle between the normal of the plane and the *z*–axis, the smaller the angle the higher the confidence. A plane with a higher confidence (the "winner" plane) is allowed to influence the orientation of a less confidential plane (the "target" plane). Depending on the current state of the target plane, the influence of the winner plane decreases. States are:

1. FREE No edge of the target polygon has been joined yet. The orientation and position of the plane might be adjusted.

2. JOINED One edge of the target polygon has been joined. There is still one degree of freedom: the rotation around the joined edge.

3. FIXED The orientation of the target polygon is fixed, the vertices of edges that are not joined are allowed to move in its plane.

The algorithm is described in Figure 4, the operations in bold type are explained in the following.

3.1 Attracting the target plane

If the target polygon is FREE the influence of the winner polygon is unrestricted, as long as the adapted polygon keeps its planarity. The process of attracting the target plane is depicted in Figure 5. At first the target polygon is translated towards the winner edge. The translation vector $\boldsymbol{t}$ is the vector between that endpoint A_v of the target edge and that endpoint B_w of the winner edge that have the shortest distance. Then the target polygon is rotated in a way that the winner edge is part of the plane. According to Eulers's theorem, any rotation in 3D can be expressed as a rotation with respect to a single unit norm axis by an angle [TV98]. The rotation axis is the normalized vector $\boldsymbol{u}$ that is perpendicular to the direction vector $\boldsymbol{e}_\mathrm{w}$ of the winner edge and the direction vector $\boldsymbol{l}$ of the intersection line of the two planes. The rotation angle α is the acute angle enclosed by $\boldsymbol{l}$ and $\boldsymbol{e}_\mathrm{w}$. The rotation matrix $\boldsymbol{R}$ is computed as follows [TV98]:

$$\boldsymbol{R} = Id_3 \cos\alpha + (1-\cos\alpha)\cdot \boldsymbol{u}\boldsymbol{u}^\mathrm{T} + \sin(\alpha)\cdot[\boldsymbol{u}]_\times, \quad (1)$$

with identity matrix Id_3 and cross product matrix $[\boldsymbol{u}]_\times$. The rotation must not influence the already joined vertex, the intersection point of $\boldsymbol{u}$ and $\boldsymbol{e}_\mathrm{w}$. Therefore, before the polygon is rotated, the target polygon is moved to the origin using the negative position vector $\boldsymbol{t}_\mathrm{o} = -\boldsymbol{b}_\mathrm{w}$ of the joined vertex[1]. After the rotation the polygon is translated back. Using homogeneous coordinates the whole transformation can be expressed by a single transformation matrix $\boldsymbol{M} \in \mathbb{R}^{4\times 4}$:

$$\boldsymbol{M} = (-\boldsymbol{T}_\mathrm{o})\boldsymbol{R}'\boldsymbol{T}_\mathrm{o}\boldsymbol{T} \quad (2)$$

$$\boldsymbol{T} = \begin{pmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & \boldsymbol{t} \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{pmatrix} \boldsymbol{T}_o = \begin{pmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & \boldsymbol{t}_\mathrm{o} \\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3)$$

$$\boldsymbol{R}' = \begin{pmatrix} & & & 0 \\ & \boldsymbol{R} & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4)$$

Each vertex $\boldsymbol{p}^i, i = 1,\ldots,n$, of the target polygon is expressed in homogeneous coordinates and multiplied by the matrix.

$$\left(p_1'^i, p_2'^i, p_3'^i, 1\right)^\mathrm{T} = \boldsymbol{M}\left(p_1^i, p_2^i, p_3^i, 1\right)^\mathrm{T} \quad (5)$$

The homogeneous coordinate is not changed, thus the new vertex $\boldsymbol{p}'^i$ is given by the first three components of the result.

$$\boldsymbol{p}'^i = \left(p_1'^i, p_2'^i, p_3'^i\right)^\mathrm{T} \quad (6)$$

[1] Position vectors of points are written in lower case.

 ISBN 978-80-86943-98-5

```
Order planes descendant by confidence
Associate every plane with a descending list of its possible target planes
WHILE (there are still winner planes that have not been tested for joining yet)
    Take next winner plane p_w
    FOR every edge e_{w_i} of p_w
        WHILE (e_{w_i} not joined yet & there are still possible target planes)
            Take next target plane p_v
            WHILE (e_{w_i} not joined yet & there are still edges of p_v)
                Take next not joined edge e_{v_j} of p_v
                IF (e_{w_i} and e_{v_j} meet joining criteria)
                    IF (p_v is FREE)
                        attract p_v to e_{w_i}
                        mark p_v as JOINED
                    ELSE IF (p_v is JOINED)
                        IF ( e_{w_i} and e_{v_j} share a common joined vertex)
                            turn p_v round joined edge onto e_{w_i}
                            mark p_v as FIXED
                        ELSE
                            intersect p_w and p_v
                            mark p_v as FIXED
                    ELSE
                        intersect p_w and p_v
                        mark p_v as FIXED
```

Figure 4: Structogram: algorithm JoinPolygons

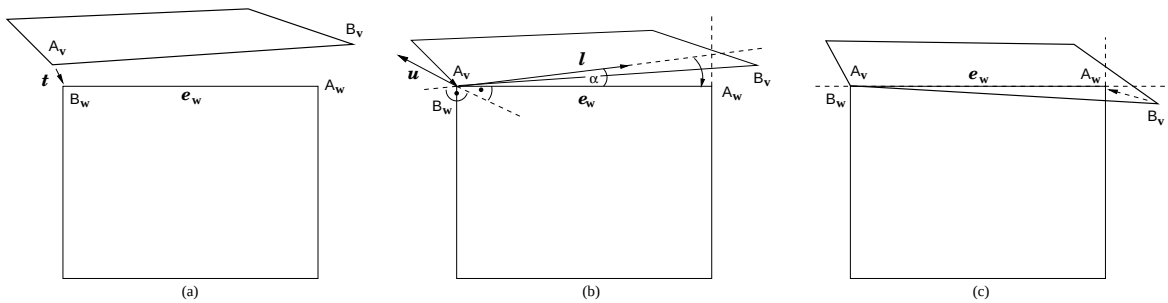


Figure 5: Attracting the target plane (a) Translation towards winner plane (b) Rotation onto winner edge (c) Substitution of the endpoint of the target edge by the endpoint of the winner edge.

After the transformation the target polygon and the winner edge lie in the same plane. The last step is to replace the second vertex B_v of the target edge with the corresponding endpoint A_w of the winner edge.
As this procedure changes the orientation of the target polygon, the execution is only allowed if the joining target edge is not too short with respect to the perimeter of the polygon. Otherwise the *Intersection* (see below) is performed. The condition prevents, that a small, error-prone edge propagates its possible error.

3.2 Turning of target plane

This part is executed if the winner polygon and the target polygon are already joined in a common vertex V_j, and if this vertex is the endpoint of a joined edge of the target polygon. Thus the target polygon is in state JOINED. The situation is depicted in Figure 6 (a). The target polygon is fixed at the joined edge with direction vector $\boldsymbol{e}_j$ but may rotate around it towards the winner polygon. As mentioned above, after the rotation the winner edge shall be part of the rotated target plane. For the computation of the rotation angle α we need a vector $\boldsymbol{l}_v$ in the target and a vector $\boldsymbol{l}_w$ in the winner

 ISBN 978-80-86943-98-5

plane that are perpendicular to the rotation axis. These are computed by dropping a perpendicular from the free endpoints A_v and B_w of the edges that have to be joined to the line through V_j with direction vector $\boldsymbol{e}_j$.

$$t = \frac{(\boldsymbol{a}_v - \boldsymbol{v}_j) \cdot \boldsymbol{e}_j}{\left\|\boldsymbol{e}_j\right\|^2} \tag{7}$$

$$\boldsymbol{f} = \boldsymbol{v}_j + t \cdot \boldsymbol{e}_j \tag{8}$$

$$\boldsymbol{l}_v = \boldsymbol{a}_v - \boldsymbol{f} \tag{9}$$

$\boldsymbol{f}$ is the base of the perpendicular of A_v to the line. The computation of $\boldsymbol{l}_w$ is analogous. As the direction of the rotation depends on the direction of the rotation axis $\boldsymbol{u}$, it is computed by the normalized cross product $\frac{\boldsymbol{l}_v \times \boldsymbol{l}_w}{\|\boldsymbol{l}_v \times \boldsymbol{l}_w\|}$ The rotation angle α is the angle between $\boldsymbol{l}_v$ and $\boldsymbol{l}_w$. The rotation matrix $\boldsymbol{R}$ is computed as shown in formula 1. The final transformation is given by matrix $\boldsymbol{M} \in \mathbb{R}^{4\times4}$:

$$\boldsymbol{M} = (-\boldsymbol{T}_o)\boldsymbol{R}'\boldsymbol{T}_o \tag{10}$$

The transformation matrixes correspond to 4, with translation vector $\boldsymbol{t}_o = (-1) \cdot \boldsymbol{v}_j$ for the translation of the target plane to the origin. After the transformation of each vertex of the target polygon, the free endpoint A_v of the target edge is replaced by the corresponding endpoint B_w of the winner edge.

Intersection The intersection causes the adaption of both, the target edge and the winner edge. The edges are replaced by a segment of the intersection line of the concerned planes. The segment is the line composed by the intersection points of the preceding and the following edge of the winner edge with the target plane.

The result of the process is a boundary representation of the scene. The geometry is given by the 3D polygons and the according plane parameters, the topology is given by the connection of edges and trihedral vertices.

4 EXPERIMENTS AND RESULTS

For the experiments we used a laser range camera of Daimler-Benz Aerospace, that provides dense range images. We used a resolution of 320×240 pixels and a measuring range of 90 to 300 cm, the distance resolution is about two percent of the measuring range, as evaluated in [APZN01]. We tested the method with artificial indoor scenes, 8 images that show objects with planar surfaces (image number p01 to p08) and 5 images that show also objects with curved surfaces (image number m01 to m05). Figure 7 shows several results. The top left chart shows the original range image in 3D space, the bottom left displays the segmentation result and the right chart shows the final result. All images result from the same parameter settings.

Image	Edges to join		Joined	
	before	after	correct	wrong
	polygon extraction			
p01 (a)	7	7	7	0
p02	4	4	4	0
p03	7	5	4	0
p04	7	7	5	0
p05 (b)	7	7	7	0
p06	2	2	2	0
p07	7	3	3	0
p08 (c)	8	4	4	0
m01 (d)	6	5	3	1
m02	6	6	3	0
m03	2	2	0	0
m04	9	6	6	0
m05	6	4	3	0
Total	78	62	52	1

Table 1: Evaluation of joining of edges of polygons. The letters in brackets in the first column refer to the images in figure 7

Image	Trihedral vertices to join		Joined
	before	after	
	polygon extraction		
p01 (a)	2	2	2
p02	1	1	1
p03	1	0	0
p04	2	1	1
p05 (b)	2	2	1
p06	0	0	0
p07	2	0	0
p08 (c)	1	0	0
m01 (d)	1	1	1
m02	2	2	1
m03	0	0	0
m04	2	1	1
m05	0	0	0
Total	16	10	8

Table 2: Evaluation of joining of vertices of polygons. The letters in brackets in the first column refer to the images in figure 7

Table 1 and table 2 list the results of the joining of edges. The evaluation shows that some edges and accordingly trihedral vertices are lost due to under-segmentation, but 83.87% of the edges that remain to join after the extraction of polygons for each plane are merged correctly. Image 7(a) shows an example where all edges could be joined and the shape of the box is well approximated. Especially the top of the box, that had shown an erroneous orientation after the segmentation, has been adjusted properly. Table 3 shows the reconstructed angles between planes that are known to be perpendicular. The mean difference to the optimum of 90° averages 5.34°.

 ISBN 978-80-86943-98-5

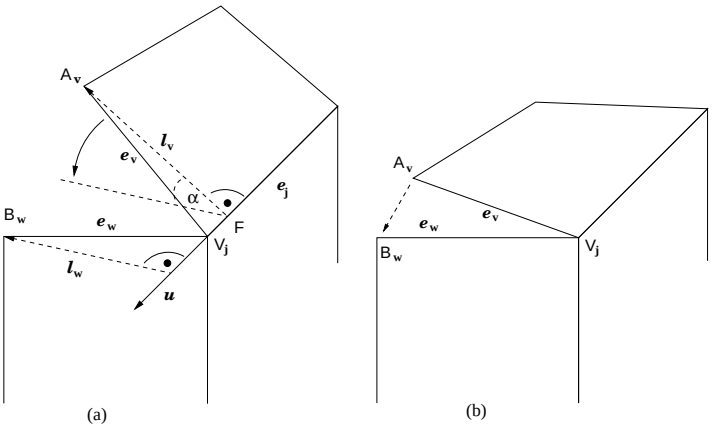


Figure 6: Turning of target polygon around the joined edge $\boldsymbol{e}_j$ (a) Turning of the target polygon onto the winner edge (b) Substitution of the free endpoint of the target edge.

Image	Number of orthogonal planes	Average angle between planes	Diff. to 90°
p01 (a)	7	86.94	3.06
p02	7	84.14	5.86
p03	7	83.99	6.01
p04	10	85.54	4.46
p05 (b)	7	85.47	4.53
p07	1	83.90	6.10
p08 (c)	5	83.32	6.68
m01 (d)	10	84.00	6.00
m02	7	83.49	6.51
m03	4	82.65	7.35
m04	3	87.27	2.73
Total	68	84.66	5.34

Table 3: Evaluation of angles between orthogonal planes. Angles are given in degree. The letters in brackets in the first column refer to the images in figure 7

Image 7(b) shows an artefact caused by the proceeding during intersection: In this case the front plane of the box has been the winner polygon and the ground the target polygon. The preceding edge of the bottom edge of the front had been a short sloped edge and this skewness is optically enforced because this edge becomes longer due to the intersection process. Image 7(c) shows the already mentioned problem of undersegmentation concerning planes, whose normal has a large angle to the optical axis. Nevertheless all edges have been joined correctly in these images. Our experiments confirm that the joining criterion (section 2) should be adapted. The current criteria are not strict enough to ensure that two edges are only joined if they really represent a common edge, as shown in image 7(d). The slope of the top plane of the box is wrong as its rear edge had been joined erroneously with its shadow edge that belongs to the wall plane. The image 7(d) shows an example with objects with curved surfaces. These are approximated by several planar patches. Overall the results are promising, especially the joining and the adjustment of the orientation of polygons works very well.

We also tested the method on data that was acquired with a 3D laser range finder which is mounted on our mobile system. The 3D–scanner consists of a Hokuyo 2D range finder that can be turned by a servo [PDMP06]. It covers a sphere of $240° \times 360°$ degrees with a resolution of $0.352° \times 0.9°$. The scanner can detect objects in distances up to 4000 mm. It is lightweight and small and therefore very suitable for a mobile robot. For the experiments, range data of an almost empty room was captured (see figure 8). The only objects in that room were benches and some lamps that are attached to the wall. From the complete sphere, a window of about $139° \times 127°$ was extracted and projected on a plane of 640×480 pixels (see figure 9). For each pixel in the plane, the ϕ and θ of the angle from the camera center was calculated and the corresponding 3D–coordinate was determined by a linear interpolation of the 4 closest laser range measurements. The result of this transformation is a range image comparable to the data that comes from the laser camera, but with a much larger field of view (the field of view of the laser range camera is only 42° in horizontal and 32° in vertical direction). This results in a high resolution of 3D–points along the z–axis even if the object is located in the area above and laterally of the laser scanner.

Therefore the processing of the images created by the laser scanner requires a few adaptions of the algorithm. The reason is that the projection of the computed 3D–data onto the xy–plane leads to a representation that is not adequate for the incremental line fitting process. In contrast to the images of the laser camera the new 3D–data may contain planes whose normal is nearly orthogonal to the principal axis and whose projection onto

 ISBN 978-80-86943-98-5

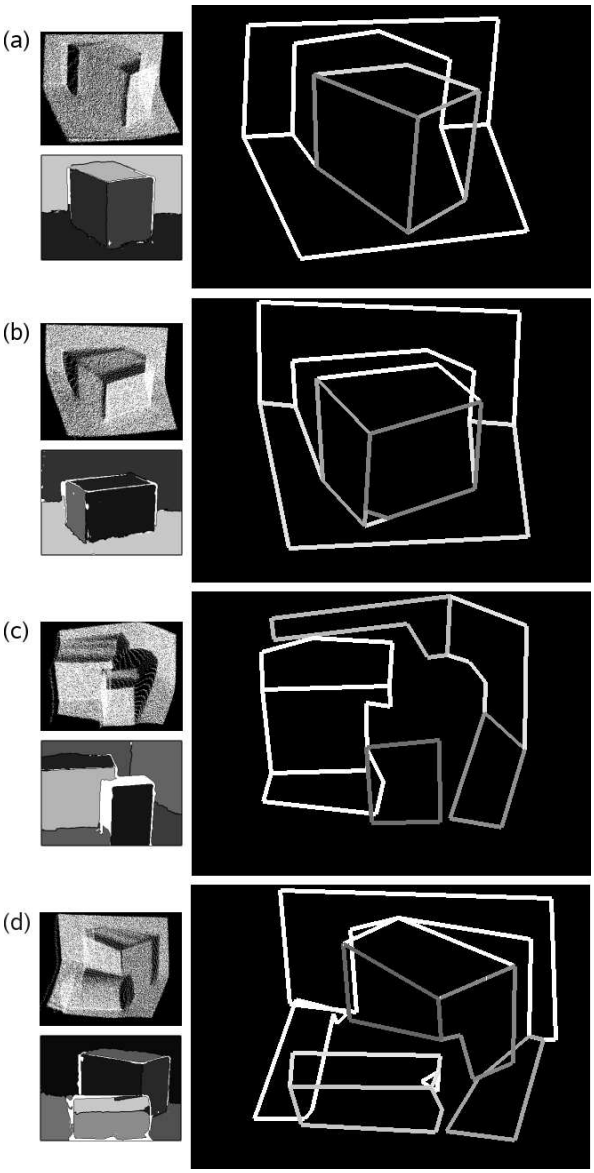


Figure 7: Some results of the whole process. Top left chart: original range image in 3D space, bottom left chart: segmentation result, outliers are depicted as white points, right chart: final result

the xy–plane results in collapsed shapes with nearly no area. Thus the incremental line fitting and the transfer of the 2D–polygons onto the according 3D–plane may fail. The propagation of a 2D–vertex to the corresponding plane could result in a point in the infinite. The new approach is that the incremental line fitting is not always processed with the projection onto the xy–plane but with the projection onto that coordinate plane whose direction of projection is closest to the direction of the normal of the plane. Further the second joining criterion, that checks if the midpoints of the 2D–edges are close enough, is now performed in 3D–space.

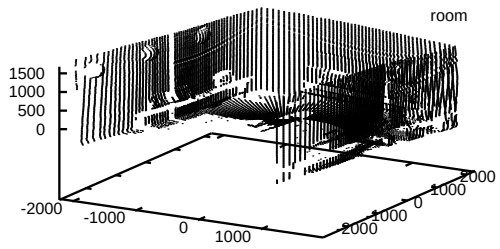


Figure 8: Point cloud of the scanned room (acquired by the 3D laser scanner; without the ceiling)

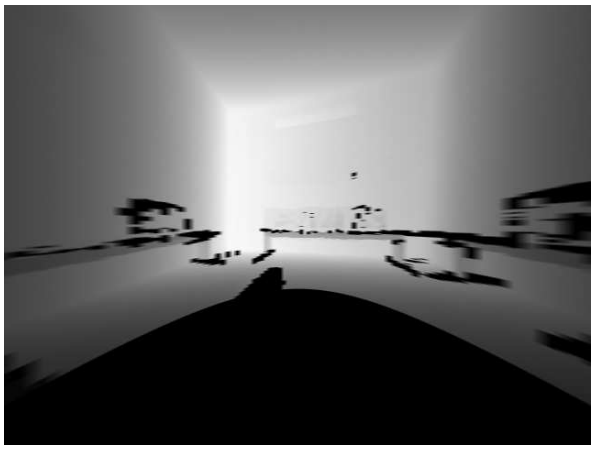


Figure 9: Plane with depth data as grey values. The black artefacts represent areas where the laser scanner did not receive an echo

Figure 10 displays the result for the described scene. The example demonstrates that smaller artefacts (also inside of regions) are suppressed. The qualitative analysis shows that the slightly modified algorithm works also well with the range data from the laser scanner.

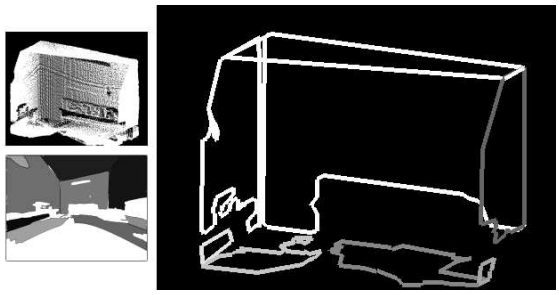


Figure 10: Result using an image created with the laser range scanner. Top left chart: original range image in 3D space, bottom left chart: segmentation result, outliers are depicted as white points, right chart: final result

 ISBN 978-80-86943-98-5

5 CONCLUSION

We presented a method to extraxt a boundary representation from dense range images based on their segmentation into planar surfaces. Using a confidence measure the polygons are joined by edges and trihedral vertices. In addition the orientation of less reliable polygons is adjusted during this process. The result is a high level description of the scene that contains its geometry and topology. In the future we want to enforce the joining criterion, in order to assure that different edges are not joined. The technique will be used for data from a 3D laser range finder which is mounted on our mobile robot "Robbie" at the RoboCup Rescue championship next year.

REFERENCES

[APZN01] U. Ahlrichs, D. Paulus, T. Zinsser, and H. Niemann. Segmentation of range images. In S. Donati, editor, *Optoelectronic Distance / Displacement Measurements and Applications*, pages 416–421, Pavia, Italy, 2001.

[AV05] H. Aceves and H. Velazquez. Scalings between physical and their observationally related quantities of merger remnants. *Revista Mexicana de Astronomia y Astrofisica*, (41):523–532, 2005.

[BS92] Suchendra M. Bhandarkar and Andreas Siebert. Integrating edge and surface information for range image segmentation. *Pattern Recognition*, 25(9):947–962, 1992. `http://dx.doi.org/10.1016/0031-3203(92)90060-V`.

[CC05] C. Chao Chen. A survey of techniques on range image segmentation and registration. Ph.D. Second-level Exam Part I, `http://web.cs.gc.cuny.edu/~cchen/`, 2005.

[FEF95] A. W. Fitzgibbon, D. W. Eggert, and R. B. Fisher. High-level cad model acquisition from range images. In *University of Edinburgh*, Department of Artificial Intelligence, 1995. University of Edinburgh.

[HGB95] A. Hoover, D. B. Goldgof, and K. W. Bowyer. Extracting a valid boundary representation from a segmented range image. 17(9):920–924, 1995.

[HJBJ+96] A. Hoover, G. Jean-Baptiste, X. Jiang, P. Flynn, H. Bunke, D. Goldgof, K. Bowyer, D. Eggert, A. Fitzgibbon, and R. Fisher. An experimental comparison of range image segmentation algorithms. *IEEE Trans. on PAMI*, 18(7):673–689, 1996.

[Kin97] V. Kindratenko. *Development and Application of Image Analysis Techniques for Identification and Classification of Microscopic Particles*. PhD thesis, University of Antwerp, Department of Chemistry, 1997.

[Koc96] R. Koch. Surface segmentation and modelling of 3-d polygonal objects from stereoscopic image pairs. In *International Conference on Pattern Recognition*, volume 1, pages 233–237, 1996.

[Liu93] H. Liu. *Oberflächenbasierte Segmentierung von Tiefenbildern*, volume 36 of *Dissertationen zur Künstlichen Intelligenz*. Infix Verlag, St. Augustin, Germany, 1993.

[NMTS05] V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart. A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics. *IEEE International Conference on Intelligent Robots and Systems*, 2005.

[PDMP06] Johannes Pellenz, Christian Delis, Ioannis Mihailidis, and Dietrich Paulus. Low-cost 3D-Laserscanner für mobile Systeme im RoboCup Rescue Wettbewerb. *9. Anwendungsbezogener Workshop zur Erfassung, Modellierung, Verarbeitung und Auswertung von 3D-Daten*, 2006.

[TV98] E. Trucco and A. Verri. *Introductory Techniques for 3–D Computer Vision*. Prentice Hall, New York, 1998.

[Zin01] T. Zinsser. Oberflächensegmentierung in Tiefenbildern. Master's thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, http://www.informatik.uni-erlangen.de, 2001.

Color Textured Image Segmentation Based on Spatial Dependence Using 3D Co-occurrence Matrices and Markov Random Fields

William Robson Schwartz
Department of Computer Science,
University of Maryland
College Park, MD, USA, 20742-327
schwartz@cs.umd.edu

Hélio Pedrini
Department of Computer Science,
Federal University of Paraná
Curitiba-PR, Brazil, 81531-990
helio@inf.ufpr.br

ABSTRACT

Image segmentation is a primary step in many computer vision tasks. Although many segmentation methods based on either color or texture have been proposed in the last decades, there have been only few approaches combining both these features. This work presents a new image segmentation method using color texture features extracted from 3D co-occurrence matrices combined with spatial dependence, this modeled by a Markov random field. The 3D co-occurrence matrices provide features which summarize statistical interaction both between pixels and different color bands, which is not usually accomplished by other segmentation methods. After a preliminary segmentation of the image into homogeneous regions, the ICM method is applied only to pixels located in the boundaries between regions, providing a fine segmentation with a reduced computational cost, since a small portion of the image is considered in the last stage. A set of synthetic and natural color images is used to show the results by applying the proposed method.

Keywords: Image segmentation; Spatial Dependence; Markov Random Fields; Color; Texture Features.

1 INTRODUCTION

The primary purpose of an image segmentation system is to extract information from the images to allow the discrimination among different objects of interest. Image segmentation is of great interest in a variety of scientific and industrial fields, with applications in medicine, microscopy, remote sensing, control of quality, retrieval of information in graphic databases, among others. The segmentation process is usually based on gray level intensity, color, shape, or texture.

Texture can be characterized by local variations of pixel values that repeat in a regular or random pattern on the object or image. It can also be defined as a repetitive arrangement of patterns over a region. Although several methods for unsupervised and supervised texture segmentation and classification have been proposed in the literature, there are neither formal approaches nor generic methods that are useful for a great variety of images.

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.

The main texture feature extraction methods can be categorized into structural, statistical and spectral [Palm04]. Structural methods are based on an arrangement of textural elements. Statistical methods define textures as stochastic processes and characterize them by a number of statistical measures. Most important statistical approaches are co-occurrence matrices, autocorrelation methods, and gray level run length matrices. Spectral approaches focus on periodic pattern resulting in peaks in the frequency domain, for instance, Gabor filtering and wavelet decomposition.

The use of colors also plays an important role in the description of regions contained in an image [Lucch01]. Many techniques for feature extraction, capable of summarizing the region properties, are based on color histogram. Although such techniques are widely used in certain applications, a serious disadvantage is their incapability of incorporating spatial information into the histogram. Furthermore, histograms are susceptible to global variations in pixel intensity.

While significant advance has been achieved in texture segmentation [Tucer98] and in color image segmentation [Cheng01] separately, the combination of texture and color properties is considered as a much more challenging problem [Deng01]. However, such combination can provide more accurate information to guide the description of the image regions [Panjw95, Shafa97, Belon98, Shi95, Palm04, Chen05].

Due to the satisfactory performance in several areas, image segmentation methods based on spatial dependence have received increasing attention from sci-

entific community [Deng04, Krish97]. Besides considering the features extracted from image regions, as usually used in region-based segmentation techniques, such methods incorporate information about a region neighborhood through the Bayesian formulation.

Among the main advantages in using segmentation based on random fields are the integration of spatial relationship between adjacent regions of the image [Dubes89], the use of several features for image description by means of the Bayesian formulation, the region labeling for generating the final segmentation obtained directly from the random field [Deng04], and the incorporation of constraints into the energy function to be minimized [Geman90].

This work describes a new image segmentation method using color texture features extracted directly from 3D co-occurrence matrices combined with spatial dependence, this modeled by a Markov random field.

The method is divided into two stages. In the first one, the centers of homogeneous regions are located by using the Dog Rabbit clustering algorithm, resulting in a coarse segmentation. The second stage is responsible for determinating the boundaries between regions, resulting in a fine segmentation. In contrast with the approach described by Fwu and Djuric [Fwu96], which employs the ICM (iterated conditional modes) in the whole image, the proposed method applies the ICM only to those pixels located in the boundaries of adjacent regions, therefore, reducing the computational cost.

The paper is organized as follows. Section 2 describes the techniques used to develop the method. In Section 3, the proposed method is presented and discussed. Experimental results obtained by applying the segmentation method are shown in Section 4. Finally, Section 5 concludes the paper with some final remarks.

2 RELATED TECHNIQUES

This section describes the techniques used to develop the proposed method, which include the Dog Rabbit clustering algorithm for locating the cluster centers in the first stage of the method, and color texture features and Markov random fields that are combined by the Bayesian formulation to allow spatial dependence during the second stage of the segmentation method.

2.1 Dog Rabbit Clustering Algorithm

The Dog Rabbit clustering method is an iterative procedure proposed by McKenzie and Alder [McKen94] and recently improved by Hill et al. [Hill05]. It uses a dynamic process to move G points to positions near the centers of clusters in the feature space.

The main idea consists of sequentially taking the sample points and moving the cluster centers toward each sample, as it is considered. The closest center is moved to the sample point under consideration, while the other ones move a lesser amount, such that only one center is more strongly attracted to each data cluster. After all iterations, each one of the G points is considered as the center of a cluster.

The displacement of the center C_j can be modeled by Equation 1, where D_j is the distance between the feature vector of the data point y_i and jth center. f_j is the *fatigue* parameter, responsible for keeping each C_j in the centroid of its corresponding cluster, and Λ is the *inhibition* parameter which allows only one center for each data cluster.

$$C_j^{'} = C_j + \alpha_j \frac{2D_{i,j}}{(1+D_{i,j})^{f_j}}(y_i - C_j) \qquad (1)$$

where

$$\alpha_j = \begin{cases} 1 & \text{if } C_j \text{ is the closest center to } y_i \\ \dfrac{D_{i,j}}{\Lambda + D_{i,j}} & \text{otherwise} \end{cases}$$

According to McKenzie and Alder [McKen94], the Dog Rabbit algorithm is more reliable than the well-known K-means clustering algorithm that is sensitive to the order in which the data are presented.

2.2 Segmentation Based on Spatial Dependence

Methods based on information regarding spatial dependence use the Bayesian formulation to relate the features of a region to its certain neighborhood [Winkl03]. In order to divide an image into homogeneous regions by grouping pixels having similar characteristics, such methods consider the existence of an observation (input image with n pixels) and a correctly, but unknown, segmented image.

Each pixel of the image is considered as a random variable that assumes values in $L = \{0, 1, \ldots, G-1\}$, where G denotes the number of regions with distinct characteristics in the image. $Y = \{y_1, y_2, \ldots, y_n\}$ denotes the set composed of the feature vectors representing the observed variables. Information about spatial dependence is modeled by a MRF (Markov random field), represented by the set of random variables $X = \{X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n\}$, where x_i belongs to the set L.

The Bayes' theorem, given in Equation 2, is used to establish the relationship between features Y and the spatial dependence of variables X, where $P(X)$ is often called the *a priori probability*. The correct segmentation for an image is that one which the labelling in set X maximizes the *a posteriori probability* $P(X|Y)$.

$$P(X|Y) = \frac{P(X)P(Y|X)}{P(Y)} \qquad (2)$$

However, the computational cost needed to determine the optimal segmentation is extremely high since it is

 ISBN 978-80-86943-98-5

not trivial to define the prior probability $P(X)$ and there are an exponential number of possible segmentations, even for images with small sizes.

Therefore, for practical purpose, segmentation techniques for approximating the optimal solution must be used. Such approximations are obtained by means of relaxation methods such as SA (simulated annealing) [Geman84], ICM (iterated conditional modes) [Besag86], MMP (maximum marginal probability) [Marro87], belief propagation [Pearl88], and graph cuts [Boyko01], which iteratively maximize probability $P(X|Y)$, after making a set of assumptions.

2.3 Color Texture Feature Extraction

In gray level images, texture can be described as an attribute representing the spatial arrangement of the pixel intensities in a region of the image. A classical approach is based on extracting several statistical measures from a gray level co-occurrence matrix, such as contrast, correlation, energy, and homogeneity [Baral95, Haral73].

The co-occurrence matrix of an image is an estimation of the second-order joint probability density of the intensity changes between pair of pixels, separated by a given distance at a certain orientation. Typically, the distance is one pixel and the orientation is quantized into four different orientations ($\theta = 0^o$, 45°, 90°, 135°). Therefore, four co-occurrence matrices are generated.

Color is also a very important attribute in image analysis. Many techniques for color feature extraction are based on the color histogram. Although such techniques have been extensively used in certain applications, they have some disadvantages since spatial information is not incorporated into the histogram.

Since the segmentation method proposed in this work uses color textured images, an extension of the traditional gray level co-occurrence matrices is required to extract proper features. The approach proposed by Dacheng et al. [Dache02] is used to describe texture features in color images. Initially, the RGB color model is converted into HSI model, since it separates the intensity and color components. Then, H, S and I components are quantized into 8, 4 and 4 bins, respectively. Nine orientations are used to define the neighborhood of a pixel along the H, S and I planes, as shown in Figure 1.

From these 3D co-occurrence matrices, four textural features are calculated (angular second moment, contrast, correlation, and entropy), producing a total of 36 measures, since nine matrices are calculated.

2.4 Markov Random Field

In order to define the image pixel interaction, an image is considered as a stochastic process, that is, $X = (X_1 = x_1, X_2 = x_2, ..., X_n = x_n)$, where x_i belongs to set

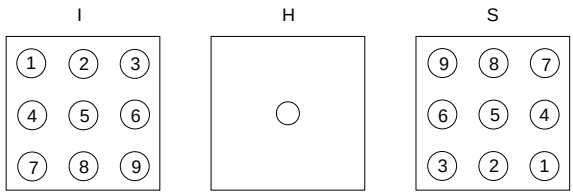


Figure 1. Orientations used to define the neighborhood of a pixel using the HSI model. Equal labels indicate the considered variations of intensity, crossing the central pixel in H plane.

L, defined in the beginning of Section 2.2. A realization of the image is vector $x = (x_1, x_2, \ldots, x_n)$. Thus, the sample space has $\{0, 1, \ldots, G-1\}^n$ different realizations.

Through the stochastic modeling, an image can be considered as a set of random variables. Thus, it is possible to determine the joint distribution of such variables, denoted $P(X)$. However, due to the lack of information regarding the dependence of the random variables and the exponential number of parameters that have to be estimated, Abend et al. [Abend65] concluded that only the local interactions between neighbors can be considered. As a consequence, only approximated solutions for the optimum segmentation may be obtained in acceptable computational time.

Some stochastic models based on local dependence have been proposed, such as Markov Mesh, Pickard model, and Markov random field. All these models estimate the joint probability $P(X)$ by means of Equation 3, where Z is called partition function and $H(X)$ represents an energy function which depends only on local interactions among the random variables [Winkl03].

$$P(X) = \frac{1}{Z}\exp(-H(X)) \tag{3}$$

3 PROPOSED METHOD

The aim of the proposed method is to segment a color textured image in G regions having similar features. The method is composed of two stages. The first stage segments the homogeneous regions of the image by using the Dog Rabbit clustering method and a bidimensional histogram, resulting in a coarse segmentation. In the second stage, considering that the features follow the Gaussian probability distribution, the ICM is used to determine the location of boundaries between adjacent regions.

The diagram in Figure 2 illustrates the steps of each stage. Since the ICM is applied only to pixels located in boundary regions, this approach significantly reduces the computational costs.

During the first stage, the input image is converted from RGB into HSI color model. The image is divided into a number of square windows, being allowed the overlapping of distinct regions. For each window, 3D

 ISBN 978-80-86943-98-5

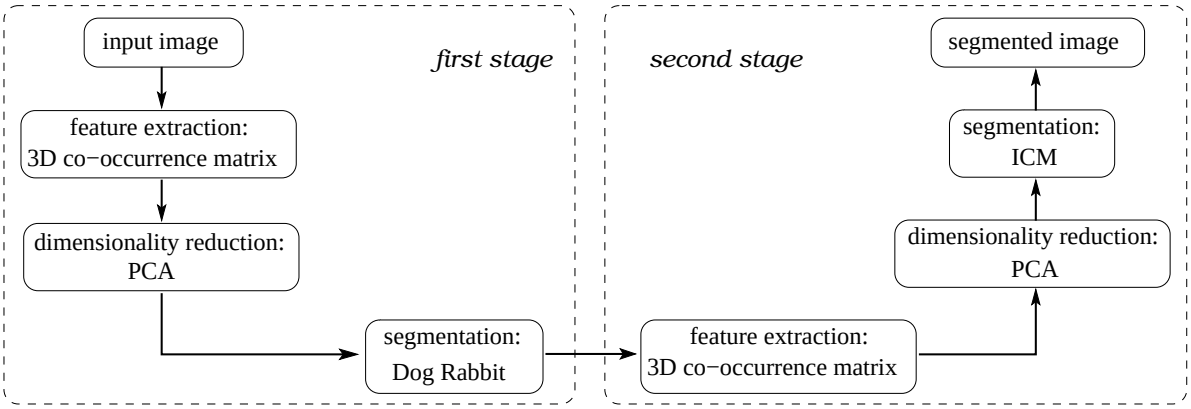
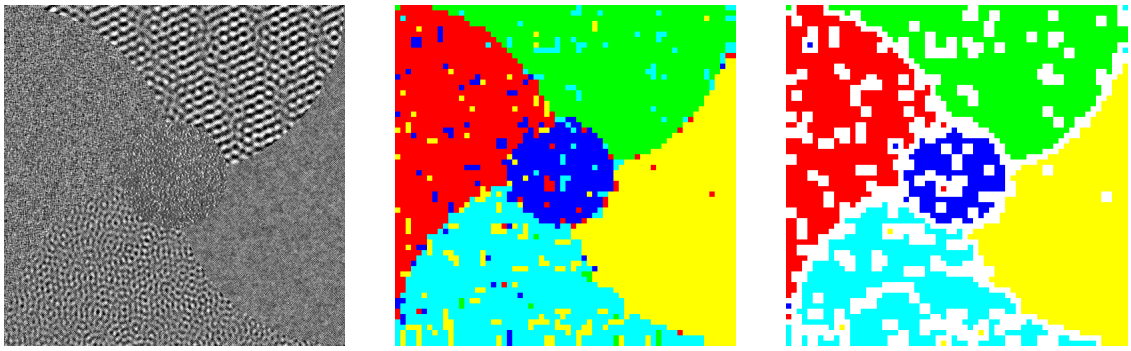


Figure 2. Diagram illustrating the two stages of the proposed segmentation method.



(a) (b) (c)

Figure 3. Example of coarse segmentations. (a) texture mosaic; (b) resulting segmentation by applying only a clustering algorithm; (c) segmentation obtained with the bidimensional histogram.

co-occurrence matrices are calculated to generate 36 statistical measures. Such measures compose a feature vector, which is used to describe the region of the image contained in the sampled window.

Additionally, principal component analysis (PCA) is applied to each feature vector to reduce the data dimensionality. Afterwards, all measures obtained from the co-occurrence matrices are rescaled to have unit variance and zero mean. The Dog Rabbit clustering method is applied in the feature vectors aiming at locating the G cluster centers which divide the image in homogeneous classes. Finally, each sampled window is labeled with a value in set L.

Once the initial clustering is concluded, a bidimensional histogram is calculated. In this histogram, the entry (x,y) contains the occurrence frequency of each G possible labels. A pixel located at (x,y) is assigned to the class i if the entry (x,y) has only non zero values in i; otherwise, that pixel will be considered in the second stage of the algorithm. Therefore, the more overlapping windows, the more precise the result of the first stage will be.

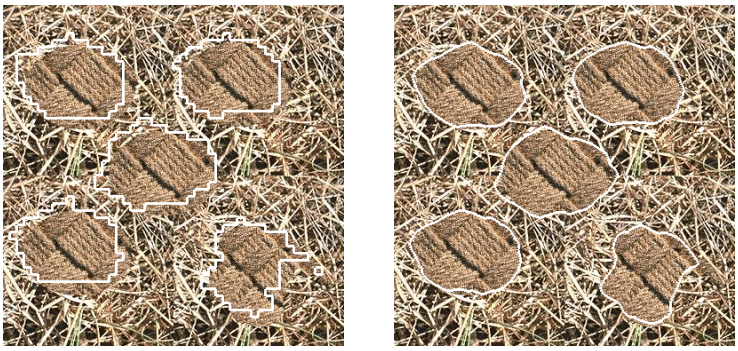
Figure 3 shows an instance of a coarse segmentation that may be obtained by computing the bidimensional histogram and setting as segmented only those regions belonging to a single class. As a result, the parameter estimation required by the second stage is performed without considering regions located in the boundaries between two classes (white regions in Figure 3(c)). Therefore, the parameters are estimated within samples in a single class, as desired.

To segment those pixels not labeled in the first stage, the ICM method is applied to estimate an labeling which maximizes the probability *a posteriori*, shown in Equation 2. Since the ICM maximizes only the local probability of a random variable, given by Equation 4, its computational cost is lower than the costs presented by other methods. The Equation 4 shows the terms which need to be maximized, where η_i represents a local neighborhood of random variable.

$$x_i \leftarrow \arg\max_{v \in L} \{P(X_i = v|\eta_i) \\ P(Y_i = y_i|X_i = v)\} \tag{4}$$

Assuming that the features extracted to each pixel follow a Gaussian distribution, Jackson and Landgrebe [Jacks02] approximated Equation 4 by 5. In such approximation, μ_v and Σ_v denote, respectively, the mean vector and the covariance matrix of the vth class in the image; m represents the number of neighbors for x_i that belong to regions different from v; finally, β is a constant weight coefficient which defines how strong is

 ISBN 978-80-86943-98-5



(a) (b)

Figure 4. Segmentation of a texture mosaic image. (a) after the Dog Rabbit clustering (first stage); (b) after 25 cycles of ICM using β=0.8 (second stage).

the interaction among adjacent pixel within a neighborhood [Winkl03].

$$x_i \leftarrow \arg\min_{v \in L} \left\{ \ln|\Sigma_v| + 2m\beta + (y_i - \mu_v)^{\mathrm{T}} \Sigma_v^{-1} (y_i - \mu_v) \right\} \quad (5)$$

Although the feature extraction and dimensionality reduction are performed in the same manner as that in the first stage, the windows are sampled differently, centered in each unsegmented pixel. Therefore, each vector y_i contains the result of PCA executed over features extracted from 3D co-occurrence matrices obtained from a window centered in ith pixel.

Once the feature vector computation is concluded, it is enough to execute a few cycles of ICM to label those pixels that remain unlabeled after the first stage. Finally, after the union of results obtained in both stages, the image is completely segmented.

4 EXPERIMENTAL RESULTS

A set of synthetic and natural color images is used to illustrate the results by applying the proposed method. Figure 4 shows the segmentation of a color mosaic image, using only the Dog Rabbit clustering algorithm (first stage of the proposed method) and after the second stage, respectively.

It can be observed that the last stage provides an improved segmentation due to the local adaptation of the ICM algorithm. The boundaries between the regions are smoother and better identified.

To evaluate our methodology, experiments were performed by using three texture mosaics and real images, shown in Figure 6. Each image was partitioned into windows with size of 24×24 pixels for feature extraction in both stages.

Since the ICM is used during the second stage of the method, the final segmentation presents a fine adaptation to the boundaries between adjacent regions, such as the result shown in Figure 5.

Although the ICM algorithm is used in the second stage of our method, when compared to methods that apply ICM in the entire image, the computational cost is lower since only the pixels located in the region boundaries are considered in the second stage.

Figure 5. Segmentation with a fine adaptation to the boundaries between regions.

The number of segmented pixels and corresponding percentage during the two stages are shown in Table 1. The percentage depends on the amount of boundary regions present in the image.

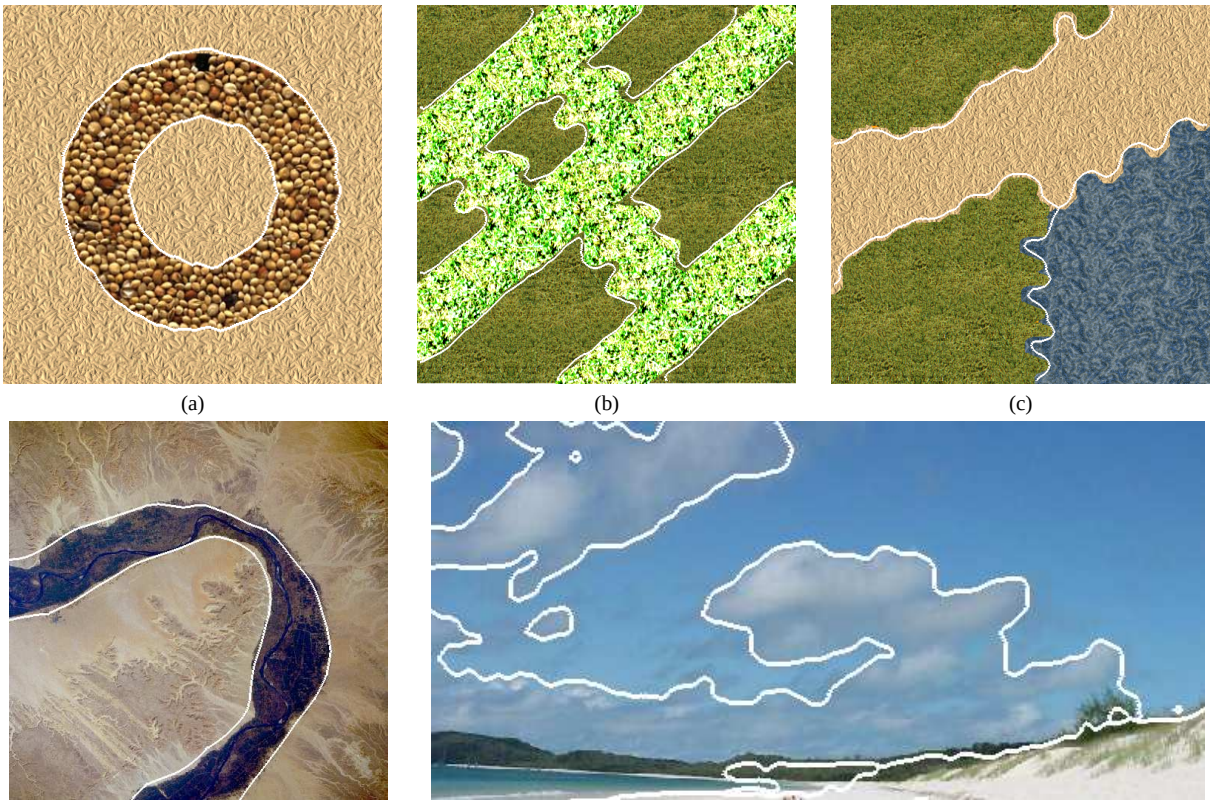
Image	number of segmented pixels first stage	second stage
Figure 4(b)	129,920 (88.11%)	17,536 (11.89%)
Figure 6(a)	118,848 (80.60%)	28,608 (19.40%)
Figure 6(b)	189,744 (72.38%)	72,400 (27.62%)
Figure 6(c)	243,520 (92.90%)	18,624 (7.10%)
Figure 6(d)	228,352 (87.11%)	33,792 (12.89%)

Table 1. Number of segmented pixels during the first (Dog Rabbit) and second stage (ICM) of the proposed method.

5 CONCLUSIONS

This work presented a new image segmentation method using color texture features extracted from 3D co-

 ISBN 978-80-86943-98-5



(a) (b) (c)

(d) (e)

Figure 6. Results obtained by our segmentation method after 50 cycles of ICM, with β=0.8 and windows composed of 24×24 pixels for feature extraction. (a)-(b) texture mosaics with 384×384 pixels; (c) texture mosaic with 512×512 pixels; (d) satellite image composed of 512×512 pixels; (e) natural image with 512×256 pixels.

occurrence matrices combined with spatial dependence modeled by a Markov random field.

The application of ICM only to pixels located in the boundaries of regions reduced the computational cost, also providing an improved segmentation due to its local adaptation.

The effectiveness of the proposed method was demonstrated by several experiments using synthetic and real color images. An extension of the method for segmenting color-texture regions in video data is planned as future work.

6 ACKNOWLEDGEMENTS

Partial support of this work was provided by a grant Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

REFERENCES

[Abend65] K. Abend, T. J. Harley, and L. N. Kanal. Classification of Binary Random Patterns. *IEEE Transactions on Information Theory*, IT-11(4):538–544, October 1965.

[Baral95] A. Baraldi and F. Parmiggiani. An Investigation of the Textural Characteristics Associated with Gray Level Cooccurrence Matrix Statistical Parameters. *IEEE Transactions on Geoscience and Remote Sensing*, 33(2):293–304, March 1995.

[Belon98] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Color- and Texture-Based Image Segmentation Using EM and Its Application to Content-Based Image Retrieval. In *Proceedings of the Sixth International Conference on Computer Vision*, pages 675–682, Washington, DC, USA, 1998.

[Besag86] J. E. Besag. On the Statistical Analysis of Dirty Pictures. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 48(3):259–302, September 1986.

[Boyko01] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization Via Graph Cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, November 2001.

[Chen05] J. Chen, T. N. Pappas, A. Mojsilovic, and

B. Rogowitz. Adaptive Perceptual Color-Texture Image Segmentation. *IEEE Transactions on Image Processing*, 14(10):1524–1536, October 2005.

[Cheng01] H. D. Cheng, X. H. Jiang, Y. Sun, and Jingli Wang. Color Image Segmentation: Advances and Prospects. *Pattern Recognition*, 34(12):2259–2281, December 2001.

[Dache02] T. Dacheng, L. Xuelong, Y. Yuan, Y. Nenghai, L. Zhengkai, and T. Xiao-ou. A Set of Novel Textural Features Based on 3D Co-occurrence Matrix for Content-based Image Retrieval. In *Proceedings of the Fifth International Conference on Information Fusion*, volume 2, pages 1403–1407, 2002.

[Deng01] Y. Deng and B. S. Manjunath. Unsupervised Segmentation of Color-Texture Regions in Images and Video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):800–810, August 2001.

[Deng04] H. Deng and D. A. Clausi. Unsupervised Image Segmentation Using a Simple MRF Model with a New Implementation Scheme. *Pattern Recognition*, 37(12):2323–2335, December 2004.

[Dubes89] R. C. Dubes and A. K. Jain. Random Field Models in Image Analysis. *Journal of Applied Statistics*, 16(2):131–164, June 1989.

[Fwu96] J. K. Fwu and P. M. Djuric. Unsupervised Vector Image Segmentation by a Tree Structure-ICM Algorithm. *IEEE Transaction on Medical Imaging*, 15(6):871–880, December 1996.

[Geman84] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, November 1984.

[Geman90] D. Geman, S. Geman, C. Graffigne, and P. Dong. Boundary Detection by Constrained Optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):609–628, July 1990.

[Haral73] R. M. Haralick, K. Shanmugam, and I. H. Dinstein. Textural Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, November 1973.

[Hill05] E. J. Hill, M. Alder, and C. J. S. deSilva. An Improvement to the DR Clustering Algorithm. *Pattern Recognition Letters*, 26(1):101–107, January 2005.

[Jacks02] Q. Jackson and D. A. Landgrebe. Adaptative Bayesian Contextual Classification Based on Markov Random Fields. *IEEE Transactions on Geoscience and Remote Sensing*, 40(11):2454–2463, November 2002.

[Krish97] S. Krishnamachari and R. Chellappa. Multiresolution Gauss-Markov Random Field Models for Texture Segmentation. *IEEE Transactions on Image Processing*, 6(2):251–267, February 1997.

[Lucch01] L. Lucchese and S.K. Mitra. Color Image Segmentation: A State-of-the-Art Survey. *Proceedings of the Indian National Science Academy*, 67(2):207–221, March 2001.

[Marro87] J. Marroquin, S. Mitter, and T. Poggio. Probabilistic Solution of Ill-posed Problems in Computacional Vision. *Journal of the American Statistical Association*, 82(397):76–89, March 1987.

[McKen94] P. McKenzie and M. Alder. Unsupervised Learning: The Dog Rabbit Strategy. In *IEEE International Conference on Neural Networks*, volume 2, pages 616–621, June 1994.

[Palm04] C. Palm. Color Texture Classification by Integrative Co-occurrence Matrices. *Pattern Recognition*, 37(5):965–976, May 2004.

[Panjw95] D. K. Panjwani and G. Healey. Markov Random Field Models for Unsupervised Segmentation of Textured Color Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(10):939–954, October 1995.

[Pearl88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, California, 1988.

[Shafa97] L. Shafarenko, M. Petrou, and J. Kittler. Automatic Watershed Segmentation of Randomly Textured Color Images. *IEEE Transactions on Image Processing*, 6(11):1530–1544, November 1997.

[Shi95] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 1995.

[Tucer98] M. Tuceryan and A. K. Jain. Texture Analysis. In C. H. Chen, L. F. Pau, and P.S.P. Wang, editors, *The Handbook of Pattern Recognition and Computer Vision*, pages 207–248. World Scientific Publishing Co., August 1998.

[Winkl03] G. Winkler. *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods, A Mathematical Introduction*. Application of Mathematics, Stochastic Modeling and Applied Probability. Springer, 2003.

Rendering diffuse objects using particle systems inside voxelized surface geometry

Thorsten Juckel Steffi Beckhaus
University of Hamburg
Vogt-Koelln-Strasse 30
22527 Hamburg, Germany
juckel@blurredvision.de
steffi.beckhaus@uni-hamburg.de

ABSTRACT

This paper presents an unique method for rendering complex shapes as fuzzy or diffuse objects inside virtual environments. It uses surface geometry that is converted into a voxel-like grid to specify the appearance of the shape. A particle system displays the outline of the object at runtime. Particles are allowed to move freely inside the voxel-grid but obtain certain attributes from the voxels they currently reside in. Those attributes assign color, size, textures, transparency to the particles, as well as forces that influence the movement of the particles. Force effects include linear and spiral movements, gravitational points, and helix-shaped motion of particles. Through this, complex movement of the particles inside the voxel-space can be created at interactive rates, while still maintaining the approximate form of the original surface geometry.

Keywords: Particle systems, voxels, interaction, real-time rendering

1 INTRODUCTION

Rendering diffuse changing shapes like fire, dust, or smoke is a common task in computer graphics. Since such shapes have a limited complexity, rendering them can be done efficiently in real-time using simplified models and techniques. When extending those shapes to more complex models, such as the rendering of ghost-like figures, this is no longer easily performed. Our aim is the rendering of diffuse, fuzzy, potentially ghost-like objects and characters. Like classic polygonal 3D objects, they have a specific shape and texture, but react to collisions, for example, by floating around the colliding solid object and re-assembling their shape afterwards. Waving inside the ghost-like character would result in stirring up the colors and rough shape of the character, as happens when color drops in water are stirred. Once left alone, however, the character should gradually re-adjust itself to its original form. If the character moves or changes shape, this may have a temporary blurring effect or may happen immediately. Furthermore, forces inside and around the character could make the character itself include dynamic components, such that, for example, the eyes glow by emitting fire or light particles moving in form of tornados.

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 Pilzen, Czech Republic.

The application of such characters is best exemplified by 3D ghost-like or cartoon characters in 3D games that behave like normal figures, but additionally show the described fuzzy and dynamic visual and shape behavior. Also, virtual narrators and guides in 3D environments and storytelling applications could have these features. They are often superimposed onto the running story and, as they are superimposed external "add-on"s, may potentially be more believable inside a story if they show a more fantasy-like appearance.

The idea is not bound to characters but can also comprise the visualization of normally invisible phenomena like energy fields in open space and aura-like phenomena around characters.

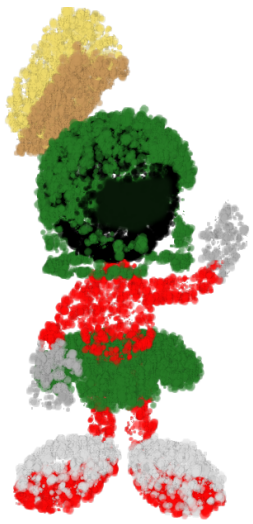


Figure 1: Marvin the Martian rendered as a diffuse character with point-sprite particles.

 ISBN 978-80-86943-98-5

To visualize characters with the described features and behavior using traditional polygonal representations of objects, complex models are needed. Those tend not to act well in interactive applications such as real-time interactive virtual environments or games. However, a simplified approach can be found using voxelized geometry in combination with common particle systems.

This paper introduces a unique method for rendering diffuse figures that extends the normal use of particle systems to display fuzzy characters and complex shapes (see Figure 1). It uses a representation of surface geometry transformed into the voxel-space to approximate the outline and appearance of the shape. This voxelized grid is generated in advance as a computational step. Particles are later generated inside this space to visualize the diffuse shape in real-time. Additional attributes can be stored inside the voxels to obtain a more complex behavioral model for the fuzzy shape. Currently, size, color, animation speed, and some sophisticated movement models can be used inside the voxels.

In Sections 2 and 3, we will give an overview of the work previously done in this area. A short introduction on particle systems and also on methods used to create a voxel-space model of surface geometry is given. This information is applied later in the paper to give a detailed description of the particle system used. Furthermore, the voxelization algorithm is explained in detail. In Section 4, the implementation of the method is presented. We describe the changes made to the original voxelization algorithm to obtain further information about the original surface geometry. We also describe, how the particle system is attached to the voxel-space and how the attributes of the voxels influence the particles. That information is later extended in Section 5 to include common interaction techniques. Specifically, manipulation practices, complex forces inside the voxels, and collision with solid models will be discussed. Section 6 presents the results of the particle shape and Section 7 concludes the work with an outlook on further work.

2 RELATED WORK

To render atmospheric effects, such as dust, smoke, or fire, the number of approaches is numerous. Realistic rendering of dust has been developed by Blinn [2] to display dusty surfaces and Dobashi et al. [3] introduced a hardware-accelerated method for rendering atmospheric scattering. While those approximations produce extremely good results, they are hardly capable of real-time rendering inside large interactive environments that are often present in 3D virtual environments. They furthermore do not support direct interaction with the effect itself.

In 1983, Reeves introduced *particle systems* to render fuzzy objects like fire, explosions and forests [7]. The complex phenomena were separated into tiny parts that could individually move around given some common constraints. This concept was later converted to use parallel computations [9] and also used by Reynolds inside behavioral systems to model the complex movement of flocks of birds [8]. Particles were not only seen as individual points anymore, but could interact with each other and form a much more complex behavior. Using constraints for the particle system, like attaching the particles to vertex elements, fairly complex shapes can be achieved.

Oriented particles [11] can be attached to surface geometry or be used for point-based rendering or sometimes to display clothing. Here, particles are interconnected by springs to make up a piece of cloth. Simple dynamic computations are performed to simulate the movement of the cloth. Since the tessellation can be varied in the cloth simulation, this method is also capable of running in real-time. Today, particle systems have advanced to be the common way for displaying fuzzy phenomena like smoke, dust, or fire. Unfortunately, those phenomena are often limited in their behavior and complexity, when rendering them in real-time. Current systems, like the one Kipfer and Segal introduced, are capable of rendering up to a million particles in real-time [4]. Even though a mass of particles are computed and rendered, interaction with the particles and control over them is very limited. Other uses of particles include point-based rendering that very closely defines the shell of a geometric object but act as another approach for surface modeling and accelerated rendering.

Since we are looking for dynamic movement of the surface and close interaction with the particles, normal particle systems are sufficient. Particle systems are also used as the integral part of fluid simulations using the Navier-Stokes equation as proposed by Stam [10]. Here, particles are moved through a computational grid to represent the density of a simulated fluid. Even though particles are used there, they only contribute to the calculation of the fluid dynamics and not directly to the visual appearance.

An image-based method to represent three dimensional objects is the use of voxels. Individual pixels are extended into 3D as a grid of small equidistant boxes to build the shape of an object. This method is often used in interactive visualization of medical data, but can also be used in other fields of real-time computer graphics. Mostly, it is used to represent three-dimensional images of scanned data.

There are several methods to convert image data to surface geometry, with the most prominent being the marching cubes algorithm [5]. Oomes, Snoeren, and Dijkstra looked at the other direction [6], trying to create voxel objects from surface geometry. While their method yields good results, the cost of the computation

 ISBN 978-80-86943-98-5

is high. Since that would have to be done every time the triangle model is updated, this becomes impracticable for real-time animations. Another method, using todays standard graphics hardware, can be used to approximate surface geometry using voxels. Beckhaus et al. used a method to render whole scenes as slices to create a voxel-based representation of the scene used for spacial analysis and collision detection [1].

3 BACKGROUND AND DESIGN

As particle systems and voxelization are the basic techniques from which our method draws, both are introduced here in more detail.

3.1 Particle systems

Most objects in a virtual environment can easily be created as surface geometry and animated using keyframes, functions, or other methods of direct manipulation. Some elements, though, that are common in the real world, are too complex to animate and to display manually. Those include the rendering of rain, where it is not possible to create every single raindrop in the scene. However, since most raindrops exhibit the same behavioral model, it is efficient to just define one object that is generating all the raindrops inside the scene automatically.

Particle system are often used in real-time graphics for this kind of common special effects, because they are very easy to implement and have the visual consistency needed by the programmers or animators. They consist mainly of an emitter, that manages the creation and update cycle of the individual particles. For every time-step in the simulation, the emitter has to perform a couple of tasks to keep the particle system alive. First, old particles that have exceeded their lifetime are extinguished. Then, new particles are generated by the emitter and initial attributes are assigned. The current set of living particles is moved and transformed according to the rules defined inside the particle system.

In the particles themselves, individual attributes are stored. In Reeves' original work, those included color, size, lifetime, and transparency. Additionally, a position and velocity were stored to determine the movement of the particles. As individual particles reach their predetermined lifetime, they fade away and can be deleted by the emitter, since they do not contribute to the simulation any further. During their lifetime, they can be influenced by outside forces, such as wind or gravity, but also other, more direct, forces, like the collision with solid objects.

When rendering the particles, they are often represented as tiny primitives, such as points, lines or small polygons. Later, their representation was extended to use textured billboarded quads or point sprites. Even animated sprites or video-textures can be used.

3.2 Voxelization

Using several slices of image scans through a human body, a three dimensional image can be created. Commonly used in the medical field, voxels are volume elements of a certain size, that are defined by their color or opacity. The information stored in the image scans can be used to obtain a volume representation of a model. Voxels are not limited to store just color information inside them. They can, for example, be easily extended to include velocity information about movement to form a vector field. They can also be used to include all kinds of attributes. We decided to store all the visual characteristics that were discussed earlier as well as information about the forces influencing the particles.

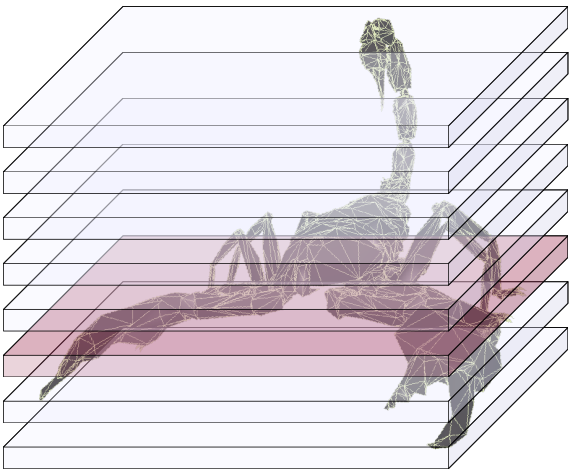


Figure 2: Several render passes through the model create the voxel representation of the shape

With todays standard graphics hardware, a special method can be used to approximate surface geometry using voxels. [1] used a method to render whole scenes as slices to create a voxel representation used for spacial analysis and collision detection. While they used this method to create a voxel representation of a complete scene in a virtual environment, it can also be applied to create texture layers from a single model.

The Marching Cubes algorithm took into account that a triangle, passing through a set of four voxels is always coloring the voxel the same way. Accordingly, a small set of triangles can be defined for different coloring configurations of the voxel-space. Using this information, creation of voxel-based images is similar. At positions where an object, composed of triangles, is inside the viewing plane, the intersection points are colored by the rasterizer unit of the graphics card. Using several viewing planes that are closely fitted behind each other, a whole set of layers can be created (see Figure 2). These layers are then used to generate a voxel representation of the surface geometry that can be used to determine the particles appearance.

 ISBN 978-80-86943-98-5

4 IMPLEMENTATION

The presented method combines the use of voxelized objects with the benefits of displaying diffuse sets of particle clouds to create complex looking objects or characters in virtual environments. The focus of the design was more on the simulation and interaction process and not on the efficiency of the rendering step, even though different methods were considered or implemented.

4.1 System design

The system is made up of two layers (see Figure 3). In the representation layer, the voxelized version of the surface geometry that was previously created is used. Voxels are either active, or deactivated. If they are activated, the voxels are part of the original surface geometry outline. Those voxels contribute directly to the particle system. Deactivated voxels do not contribute to the particle system in any way. Inside each activated voxel, a set of attributes is stored that defines the behavior and appearance of the particles currently residing inside it. The voxel-space is generated automatically as an early computational step and can later be edited as needed by the animator or designer of the model.

The second layer contains the particle-emitter that is generating new particles inside the voxel-space during runtime. Even though the voxel-space is defined by the whole bounding box of the original surface geometry, particles are only created in activated voxels, i.e. voxels that have color and attributes assigned to them. They define the outline of the surface geometry.

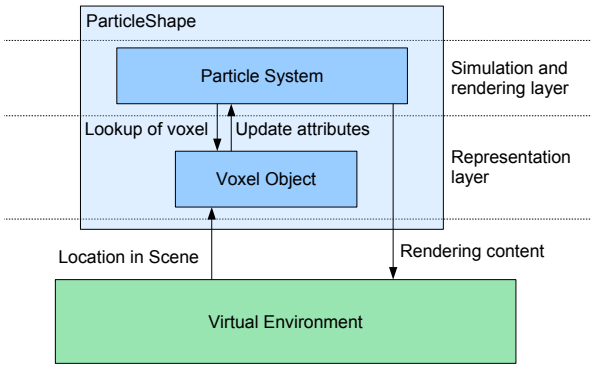


Figure 3: The layout of the system.

4.2 Voxelization

To use the voxelization algorithm of Beckhaus et al. [1] for our purposes, their method had to be slightly modified. Since they needed only spatial information about the geometry, the color information was ignored. We use the color information to store our initial attributes inside the voxel. Since this color can also be the same as the color of the back buffer, we take into consideration values stored inside the depth buffer after the rendering pass. This saves a lot of time in terms of coloring the model afterwards, since this can be done as part of the modeling step. Other attribute values stored inside the voxel are current density, maximum density, force effect, particle color, particle size, particle fading color, particle fading size, particle lifetime, particle interpolation time, animation speed for textures, and a flag for indicating, if the voxel is active, i.e. contributing to the appearance of the shape. This flag can be used during collision detection and response to make the shape move fluently through solid objects. By temporarily deactivating the voxels, particles are not created inside a solid object the particle-based shape is moving through.

Every voxel can have its own set of attributes and manipulates only the particles currently residing inside the voxel. As particles move freely in world space, they check at every time-step, Δt, which voxel they currently reside in and receive updated attribute values from the voxel accordingly. Since the voxels are evenly spaced inside the object, this can be done very fast in one simple computational step for every particle.

4.3 Update cycle

New particles are generated by an emitter inside voxels that have not reached their maximum density yet. Unlike normal particle systems, particles receive their initial values not from that emitter, but from the voxel directly. Particle movement is determined by the current force effect inside the voxel. This can be either a linear force or a rotational force. Rotational forces are projected onto linear forces, because small time-steps are assumed during the calculation steps. This allows a simple calculation of movement inside the particles independent of the complexity of the original forces. For simplification of the calculation, forces are stored at the center of the voxels. Linear forces act uniformly inside the whole voxel. The axis for rotational forces or the gravitational point are also placed at the center of the voxel but act differently on particles at different locations.

Since the voxels are all evenly spaced, the particle–voxel assignment can easily be established as an array access in linear time. The size and resolution of the voxel-grid does not contribute notably to the update time of the animation.

Unlike the common implementations discussed earlier, particles are kept alive for an indefinite period of time while they are inside any activated voxels. This helps to keep the animation stable. When a particle enters a new voxel, it gets new target attributes assigned. Interpolation is done linearly for color and size and texture animation speed. The interpolation time is also retrieved from the new voxel. Particles can, therefore, change their appearance at different speeds inside dif-

 ISBN 978-80-86943-98-5

ferent voxels. This ensures more freedom for the animator as well as smooth transitions between the voxels, but still makes it possible to have the particles react totally independent inside the separate voxels. Having a set of target attributes also makes it possible to change the attribute instantly, for instance when the voxel attributes change or the particle leaves the defined voxel space, or changes into a new medium, such as bubbles moving from water into the air. This interpolation can be seen in Figure 4. The interpolation between the two voxels is surrounded by the green circle.

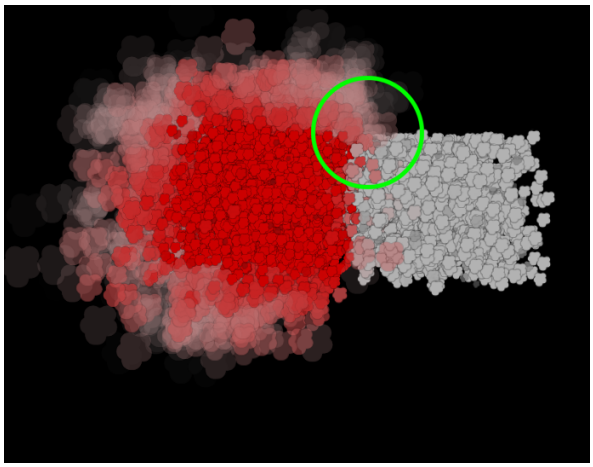


Figure 4: Particle shape with only two voxels. Particles on the left fade away slowly when leaving their voxel, on the right they fade away instantly.

As soon as a particle leaves an activated voxel and enters the free space surrounding the model, the normal fading behavior is triggered. The fading corresponds to the fading behavior attributes of the last visited active voxel and is consistent with the behavior described by Reeves. However, if a particle re-enters an activated voxel, it's lifetime will be reset to zero again and the particle will continue to live. The normal animation model can be followed then as described earlier.

4.4 Displaying the particle system

There are different ways to render particles. Reeves explained in his original paper that he draws the particles as simple points or lines for every particle. This is sufficient for small fast moving objects. However, when displaying larger, slow moving objects, it is common to use billboards to display the particles. These are commonly mapped with textures to provide additional level of detail. These textures can be animated or video textured. In our implementation, texture-sets can be defined and played at different intervals. Those intervals are stored inside the voxel definition.

Today's graphic frameworks like OpenGL or DirectX provide different kinds of functions that define how transparent objects are combined on screen. Blending is the mechanism for combining color already in the frame buffer with the color of the incoming primitive. The result is then stored back in the frame buffer.

Rendering every particle independently may be suitable if additive blending of colors is used. Many overlaid particles become very bright. They get a white, glowing look to them. This may be reasonable for effects like fire and other light emissive special effects. When rendering more complex phenomena, however, such as cloud layers or diffuse objects, it becomes important to use additional alpha values that are stored inside the color information or the alpha channel of the textures to perform the blending operation.

Rendering transparent objects is always performed with the depth-buffer being disabled. When displaying the particles in arbitrary order, overlapping particles from further away may occlude others that are closer to the viewing plane. Because of that, they have to be rendered in the right order from back to front. The hardware's method to achieve this, is writing values into the depth buffer. As new primitives are rendered, their z-values are checked against the ones stored inside the depth buffer. If they are smaller than the previous ones, the primitives can be rendered. To display particles that have alpha values stored for transparency and need to be rendered accordingly, a sorting algorithm should be used. We chose a simple quick sort algorithm, since it is directly supported by C++ through the standard template library and does not need any specialized hardware. Since sorting thousands of particles in real-time can limit the frame-rate drastically, sorting should be limited to a minimum. Sometimes, it is sufficient, to just sort parts of the particles for adequate visual effect.

5 INTERACTION, MANIPULATION, AND DYNAMICS

There are a few common interaction techniques for manipulating scenes and objects in 3D virtual environments. The most basic include translation, rotation, and scaling of objects. Normally those actions are performed by a transformation matrix that can be applied to the original vertices of the geometric models. This works fine there, since the geometry is connected and all the vertices still have the same relation to each other.

As the particles used in the presented method form a highly chaotic system and are not attached to anything solid, transformation matrices can not be applied that easily. Every manipulation of the whole system, i.e. a transformation of the voxelized model, acts as a kind of force onto the particles. The particles, in turn, have to move accordingly. Movement and rotation of the system influences different particles in differing ways. While old particles typically keep their current momentum and move according to their force values, newly generated particles receive additional momentum from the movement of the underlying structure. Therefore, interaction and manipulation with the objects mainly

 ISBN 978-80-86943-98-5

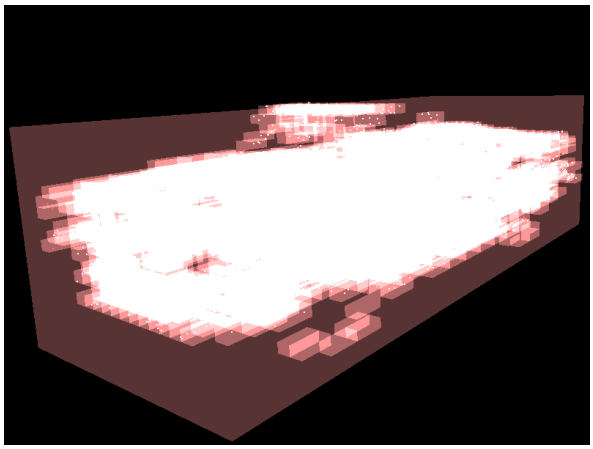


Figure 5: The bounding boxes of the voxelized shape of a car.

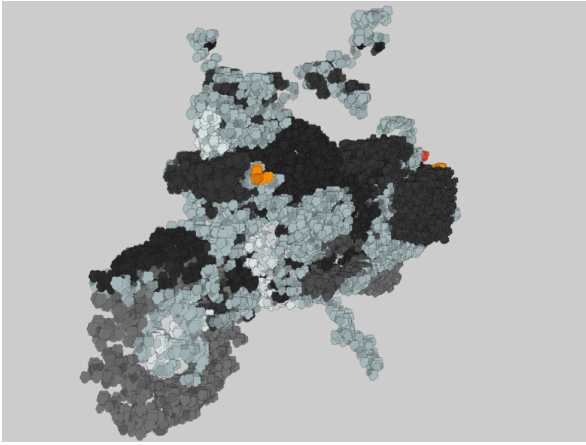


Figure 6: A chopper made of particles. Though it has a quite solid appearance, particles move constantly.

has to deal with the application of forces to the particle system, both overall and on voxel level. Those forces can in addition be used to create additional visual features in the resulting shapes and characters. The reminder of this section describes interaction with and animation of the fuzzy objects by discussing forces applied to the system or to single voxels and forces that happen through collisions.

5.1 External forces and forces inside a voxel

In addition to the color, size, or animation speed, particles can be influenced by different forces inside the voxel-space. We distinguish forces that act on the whole system from the outside, which include movement, wind, or gravity, and forces that act on every particle differently. Transformation of the whole system is modeled as just another external force that affects newly generated particles as an initial momentum acting on the particles.

In our implementation, the designer can decide, which force he wants to use in certain voxels and can set them individually. We implemented four different kinds of forces to be used inside every voxel. They are *linear forces*, like wind or gravity; *spiral forces*, that move the particle around an arbitrary axis at the center of the voxel; a *gravitational point* at the center of a voxel, that attracts particles and forces them to move around in an orbit; and a helix-like movement, that rotates the particles around an axis in a specified direction. Fluid dynamics were also considered, especially since the voxel-grid already implies the computational grid used there, but were dismissed in the initial approach, because their calculations were too complex for large voxel-grids in the real-time virtual environments that were targeted.

5.2 Collisions with solid objects

When looking at smoke in reality, any collision with the system, like moving a hand through a cloud of fog impacts the movement of the particles. However, since collision is also a very time consuming progress, especially when dealing with thousands of different particles, it has to be limited. To decrease the time for the calculation drastically, a hierarchical system has been added to the particle based shape.

Bounding volume collision of the whole system can nicely be performed by the application's framework. The voxel-object's bounding box is tested against every other object in the virtual environment. If an object is penetrating the surrounding bounding box of the whole system, a more detailed test can be performed. This is done using the volumes of the individual voxel (see Figure 5). If a voxel penetrates a solid object or vice versa, voxels are temporarily deactivated. This is important, since particles would otherwise still be generated inside the voxels, leading to particles appearing inside the solid object.

Additionally, every particle that is currently inside colliding voxels has to be checked for direct collision with the object. This has to be done on a per triangle basis, since particles hit an object directly and the collision response needs to be as accurate as possible. As the particles position before and after the update-step can be considered as a line, collision detection is done using a line-triangle algorithm.

6 RESULTS

This section shows some images we have generated using our particle based system. Figure 6 shows how closely surface geometry can be approximated with our particle system. Even though this example uses only a grid of 40^3 voxels and about 30,000 particles, many details of the chopper are visible. It can be noted, how the initial coloring of the model was used to include additional detail at the position of the indicator.

 ISBN 978-80-86943-98-5

Figure 7: Using just one voxel, even conventional effects, like this fire can be created.

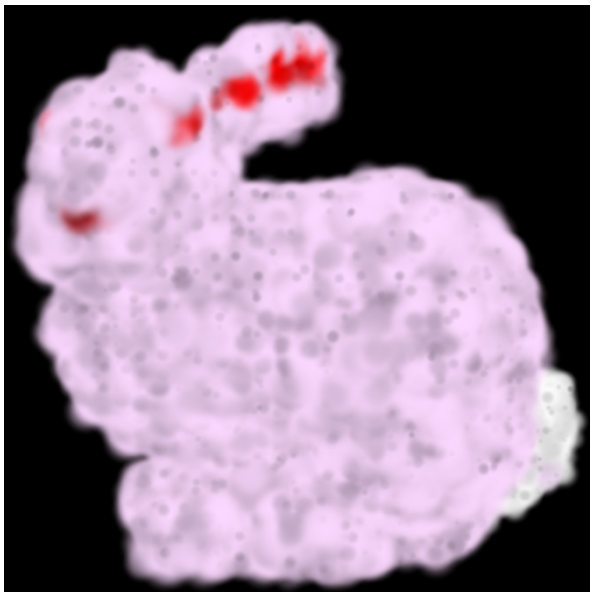


Figure 8: The Stanford bunny as a "fluffy" version using our particle based system.

Simple effects, like the fire shown in Figure 7 can easily be created by our system. This example uses just one voxel and an animated texture for the particles. More complex models or video textures may also be used. Rendering the model using additive blending is useful for rendering fire and other light emitting objects. All results were generated using an AthlonXP 2000+-based laptop with a ATI Radeon 9000 mobility graphic-card. Because of this, most shader-based optimization could not be employed. Further optimizations using more current graphic hardware or multi-processor systems will increase the frame output immensely.

For our method, most objects could be sufficiently represented as a voxel-grid with a dimension of 50^3. As the complexity of the models rises, the voxel-grid needs to be more precise. It should be mentioned though, that memory consumption increases rapidly and large voxel-grids are not encouraged, even though the time for the computation does not rise. Since packing the voxel-grid to save memory will result in eliminating the fast access of voxels through positions of the particles, compression cannot be performed easily.

The image on the first page (Figure 1) shows the popular TV character Marvin the Martian from the show Looney Tunes. The diffuse model was voxelized with a 50^3 grid and rendered with 40,000 particles visualized as point sprites. The rendering times are at about 10 to 15 frames per second.

The model of the Stanford bunny in Figure 8 was rendered to simulate the "fluffy" fur of the bunny. At the nose, eyes and ears voxel where colored red, at the rear, the tail was colored white. Particles were instantly colored in the correct way.

7 CONCLUSION AND FUTURE WORK

This paper presented a new method to approximate fully controllable, complex, fuzzy shapes to be used inside interactive 3D virtual environments. The shape approximates the provided polygonal geometry closely, while still having a fuzzy and dynamic exterior, with interaction and reaction to other objects being possible. Forces are integrated into the shape to change the appearance during runtime. For this method, a fast structure for creating, rendering, changing, and updating structures was required.

To achieve this, regular surface models are converted into voxelized versions. Particles are spawned afterward inside the voxels to create a highly dynamic version of the previous surface geometry. Forces that are stored inside the voxels act on the particles and move them along paths. Complex rotational forces make the particles move in spirals, orbits, or as vortices inside the voxel. Additional attributes inside the voxels, like size and textures define the appearance of the particles during the render process. Particles are created during runtime inside the voxels to approximate the original shape. Generation is done inside the world's coordinate system, to let simplify calculations when the particles interact directly with other objects.

Voxels are created efficiently through hardware, making it possible to create voxels as needed in limited time to let the user focus on the design and appearance of the model using traditional surface modeling tools. Vertex colors are used during the conversion step to create the initial coloring attributes of the voxels. Through simple force calculations inside the particles, the performance impact for updating the particle attributes could be reduced to a minimum. This leaves most of the compu-

 ISBN 978-80-86943-98-5

tational power to the application. Interaction between a particle system and rigid bodies inside the virtual world have been implemented through a structured collision detection and response algorithm.

The applications for this system range from traditional rendering of fire, clouds, and dust to diffuse ghost-like characters and objects. The presented particle system can also be used for visualization of stress fields inside materials. Particles would move through the force field created by the voxels to form clusters, where the forces are too high to maintain further structural integrity of the material. Other, more artistically oriented methods include drawing of voxels in an virtual environment directly to use as an interactive diffuse paint metaphor.

The new particle system produces satisfying results inside a controlled environment. Some design aspects had to be considered during the creation of the system. Using highly detailed geometry is only useful if the voxelization is performed with a fine resolution. This, necessarily, requires smaller particles and, therefore, a higher number of particles inside the system. As a result, the particle system's performance rapidly decreases as the amount of particles rises. Coarser models with a resolution of 60 voxels per side prove to be sufficient for the trade-off between visual complexity and real-time performance. Additional tuning of the particle system can also enhance the visual appearance.

When using models with very detailed parts or small, thin spikes, holes occur, where only a single row of voxels is representing the originating geometry. This can be avoided, by adjusting the forces inside the voxels along the path of the geometry. An automated approach could be employed to aid the design process.

The system works well with simple independent models. To further extend our system, interaction between more than one particle system should be implemented. Pressure fields and fluid dynamics were not added yet, because of calculation time issues, but would enhance the visual complexity of the system. As hardware implementations progress, fluid calculations should be possible even for large grids. Complex lighting calculations inside the particle system would also improve the system and will probably be integrated later.

REFERENCES

[1] Steffi Beckhaus, Jürgen Wind, and Thomas Strothotte. Hardware-based voxelization for 3d spatial analysis. In *Proceedings of CGIM '02*, pages 15–20. ACTA Press, 2002.

[2] James F. Blinn. Light reflection functions for simulation of clouds and dusty surfaces. In *Proceedings of the 9th annual conference on Computer graphics and interactive techniques*, pages 21–29. ACM Press, 1982.

[3] Yoshinori Dobashi, Tsuyoshi Yamamoto, and Tomoyuki Nishita. Interactive rendering of atmospheric scattering effects using graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 99–107. Eurographics Association, 2002.

[4] Peter Kipfer, Mark Segal, and Rüdiger Westermann. Uberflow: A gpu-based particle engine. *Graphics Hardware*, 2004.

[5] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169. ACM Press, 1987.

[6] Stijn Oomes, Peter Snoeren, and Tjeerd Dijkstra. 3d shape representation: Transforming polygons into voxels. In *Proceedings of the First International Conference on Scale-Space Theory in Computer Vision*, pages 349–352. Springer-Verlag, 1997.

[7] W. T. Reeves. Particle systems – a technique for modeling a class of fuzzy objects. *ACM Trans. Graph.*, 2(2):91–108, 1983.

[8] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.

[9] Karl Sims. Particle animation and rendering using data parallel computation. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 405–413, New York, NY, USA, 1990. ACM Press.

[10] Jos Stam. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[11] Richard Szeliski and David Tonnesen. Surface modeling with oriented particle systems. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 185–194. ACM Press, 1992.

 ISBN 978-80-86943-98-5

Motion Pattern Preserving IK Operating in the Motion Principal Coefficients Space

Schubert R. Carvalho
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Virtual Reality Lab
CH-1015 Lausanne, Switzerland
schubert.carvalho@epfl.ch

Ronan Boulic
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Virtual Reality Lab
CH-1015 Lausanne, Switzerland
ronan.boulic@epfl.ch

Daniel Thalmann
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Virtual Reality Lab
CH-1015 Lausanne, Switzerland
daniel.thalmann@epfl.ch

ABSTRACT

We introduce a constraint-based motion editing technique enforcing the intrinsic motion flow of a given motion pattern (e.g., golf swing). Its major characteristic is to operate in the *motion* Principal Coefficients (PCs) space instead of the *pose* PCs space. By construction, it is sufficient to constrain a single frame with Inverse Kinematics (e.g., the hitting position of the golf club head) to obtain a motion solution preserving the motion pattern style. Such an approach also reduces the size of the Jacobian matrix as the motion PCs space dimension is small for a coordinated movement (e.g. up to 9 for the golf swing illustrating this paper). We compare this approach against a per-frame prioritized IK method regarding performance, expressivity and simplicity.

Keywords

Motion Editing, Principal Coefficients, Principal Component Analysis, Inverse Kinematics.

1 INTRODUCTION

Technological advances in motion capture techniques make it possible to provide high quality motions for computer animation. However, the captured animations almost always attend to specific needs. Editing and reusing these motions in new situations (e.g., retargeting to new characters or to new environments) became an increasing area of research. Unfortunately, making little adjustments into a good motion may introduce some undesirable artifacts or discontinuities, that can modify the intrinsic style of the motion. Hence, providing ways to preserve the characteristics of the original animation and to impose continuity on the edited animation are the major challenges of motion editing techniques.

In this paper, we present a new method for motion editing. As in traditional per-frame constrained-based approaches the user can make adjustments to a character with direct manipulation, for example repositioning the character's end-effectors. However, two important differences arise between our approach and traditional per-frame constrained-based techniques. First, our method considers just the frame that should be edited, instead of a range of frames around the edited one. Second, to compute the new end-effector's position the system takes into account the style or characteristics of the original animation during optimization.

Although, traditional constrained-based techniques uses just the information of the input motion, we use a database of motion captured animations (not upper body cleaned) of the same behavior of the edited one. These motions are used to estimate a set of Principal Components (PC) and Principal Coefficients (PCs) [Jol86]. The PCs space is used to model the characteristics of the edited animation. The PC space is used for two purposes: 1) to bound the inverse kinematics (IK) solutions closer to the pose space of the edited animation; and 2) to reduce the size of the Jacobian matrix to invert. Our experiments indicate that our approach preserves the characteristics of the edited motion and impose continuity between frames just constraining the pose that should be edited. We also assess the quality of the edited animations against a traditional per-frame prioritized IK method [CB06].

The next section reviews related work. Section 3 provides the motivation and the overview of our approach. Section 4 describes our approach for motion editing in more details. Section 5 describes our motion normalization schema. Then, we illustrate our method with results in section 6. Finally, we conclude this paper by discussing current limitations of our method and possible extensions in section 7.

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.

 ISBN 978-80-86943-98-5

2 RELATED WORK

2.1 Motion Editing

Constrained-based techniques [Gle01] enables the user to specify constraints over the entire motion or at specific times, while editing an animation. An inverse kinematics (IK) solver computes the "best"motion, pose by pose, satisfying all these constraints. Lee and Shin [LS99] introduced the per-frame plus filtering class of motion editing techniques. Continuity between frames was maintained with a B-Spline filter. Kulpa et. al. [KMA05] proposed a motion adaptation methodology. Their method do not use any kind of filtering, so it provides very good results for real-time motion adaptation of virtual characters. To ease the adaptation, the underlying skeleton is divided into groups, each of which being an individual kinematic chain. This has the advantage to ease the computation of a solution. However, as no synergy exists between groups, it may lead to unrealistic results. Le Callennec and Boulic [CB06] proposed an off-line interactive motion editing technique with prioritized constraints. The characteristics of the original motion are preserved adding the difference between the motion before and after editing as the lowest priority constraint. Splines curves were used to impose smoothly varying constraints. Recently, Liu [LZqDMSH06] proposed a motion editing technique with collision avoidance to prevent the character's limbs to interpenetrate the body during editing. Displacement maps techniques and a Kalman filter were used to preserve the similarities between the edited and original motions, and to impose continuity between frames.

All these techniques do not explicitly model the characteristics of the original motion, mainly because style is difficulty to formulate. Our approach uses Principal Component Analysis(PCA) [Jol86] to represent style in the low-dimensional space of PCs. We also provide an IK solver, to treat geometric constraints, able to preserve motion patterns (e.g., the golf swing characteristics) during optimization, and a different approach to impose continuity between frames.

2.2 Principal Component Analysis

PCA is used in Computer Animation to treat different kinds of problems. Alexa and Müller [AM00] used PCA to represent animations as a set of principal animation components. Glardon [GBT04] used the PCA to synthesize walking, running and jumping motions into a low-dimensional space. Following this idea, Safanova et. al. [SHP04] proposed a motion synthesis framework able to synthesize new motions by optimizing a set of constraints in the low-dimensional space of the PCA. They used IK, just in the characters limbs, to prevent undesirable artifacts such as foot sliding. Urtasun et. al. [UGB$^+$04] proposed a style based motion synthesis framework able to produce motions with different styles by extrapolating the PCs parameters. Urtasun and Fua [UF04] also used the PCA to improve 3D body tracking. Their formulation is similar in spirit to ours because they need to evaluate the PCs that describes a tracked motion style. They relaxed the overconstrained problem of tracking multiple effectors for each frame by allowing the Principal Coefficients to evolve overtime too. On the contrary, we relax the problem by constraining only one frame for which the constraint is particularly easy to specify, e.g. hitting frame in golf. This reduces computation time as the solution is searched in the Principal Coefficients space, which dimension is small compared to the posture space.

3 MOTIVATION AND OVERVIEW

The motion editing framework proposed in this paper is motivated by the following question: how to edit a motion by constraining just one pose without adding discontinuities. We investigated this problem by considering three issues: first, we need to mathematically represent the characteristics of the edited animation; second, we need a constraint formulation that takes into account these characteristics during optimization; third, we need to impose continuity between frames considering the characteristics of the edited animation.

Our aim for motion editing is to take a motion that is good, but is in need of some adjustments, e.g., the motion has the basic format that we want. This is almost always the case of captured motions, where simple adjustments should be made [Gle97]. Now, consider one adjustment: at a specific time of the motion, there is a new position that should be achieved. Consider Figure 8 as an example. We have a good golf swing motion, but we want that the golf club head hits the ball onto another position. We can associate a constraint to the golf club head, to change its position, and compute the new motion.

A common technique used to address such problems is inverse kinematics. IK gives the possibility to edit a motion by dragging the character's end-effector to a new position. When highly articulated figures are considered, IK algorithms are iterative. They try to find the best solution, for some set of constraints, iteratively minimizing some residual error (section 4.2). IK is also the core of constraint-based motion editing techniques [Gle01]. However, when IK is used to deform a motion frame by frame, motion continuity can be broken [Gle97]. Instead of using traditional techniques (i.e., splines curves [LS99]), we proposed a new approach (section 4.3) by exploiting the low-dimensional parametrization of a set of motions with the same behavior as the edited one (sectio 4.1).

The present framework handles geometric constrains, such as position or rotation of end-effectors. Geometric constraints are more intuitive because they directly specify a goal for a specific end-effector [CB06]. Constraints such as position of the center of mass are not specified,

 ISBN 978-80-86943-98-5

because the solution is searched within the PC space of physically balanced motions.

4 A MODEL FOR EDITING

4.1 Motion Parametrization

In this section, we recall how a motion can be represented by PCA. The parameters of the PCA can be easily estimated from captured animations of people performing a specific activity (see, section 5 for more details). Let us define a pose, Θ, as a state vector describing the global position and orientation of the root node, and a set of joint angles:

$$\Theta = [\theta_1, P_1, \theta_j, ..., \theta_n] \tag{1}$$

where, P_1 and θ_1 represent the 3D global position and 3D orientation of the root node in the sacroiliac level and θ_j is the local transformation of the jth joint expressed using exponential map formulation [Gra98].

A motion can be represented as a normalized motion vector Ψ of dimension: $D = n * N$. Where $N = 132$ is the total number of poses and $n = 93$ is the number of degrees of freedom of the skeleton including the 3D global position and 3D orientation of the root node. Ψ is a column vector of the form:

$$\Psi = [\Theta_{t_0}, ..., \Theta_{t_k}, ..., \Theta_{t_1}]^T, \qquad 0 \leq k \leq 1 \tag{2}$$

where, Θ_{t_k} is a posture corresponding to the normalized time t_k (see, section 5 for more details).

Given the motion database, we compute the mean motion $\Psi_\circ$ and the covariance matrix $C_{M\times M}$, where $M = 10$ is the size of the motion database. The eigenvectors of the covariance matrix are computed and a linear map is constructed to obtain the principal components $\mathbf{E}_{i,1\leq i\leq M}$ [Jol86]. Assuming this set of examples to be representative for a golf swing motion, any motion vector Ψ can be approximated as a linear combination of the mean motion and the PC:

$$\Psi \approx \Psi_\circ + \sum_{i=1}^{m} \alpha_i \mathbf{E}_i \tag{3}$$

where, $\alpha = (\alpha_1, ..., \alpha_m)$ are the Principal Coefficients that characterize the motion and $m \leq M$. m represents the number of PC required to reconstruct a desired percentage σ of the database. This percentage is defined as:

$$\sigma = \frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{\mathbf{M}} \lambda_i} \tag{4}$$

where λ_i is the eigenvalue corresponding to the $\mathbf{E}_i$ eigenvector, ordered in decreasing order. The parameters of the linear model are estimated only once off-line, before the editing process starts.

4.2 Constraining a Single Posture in the Motion PC Space

In this section, we demonstrate how standard IK methods can be adapted to our needs. If we define the pose, including the root node, of a virtual character as a n-dimensional vector (joint space, Eq. 1) and the position of the end-effector, $\mathbf{x}$, as a p-dimensional vector (task space), the inverse kinematics function can be defined as:

$$\Theta = f^{-1}(\mathbf{x}) \tag{5}$$

for a redundant manipulator, i.e., $n > p$, the problem is always ill-posed (there is not unique solution) and even if $n = p$ multiple solutions can exist [Cra86]. Hence, IK algorithms should determine just one solution to Eq. 5 given many possibilities. In the presence of redundant manipulators IK can be formulated as the solution of an optimization problem:

$$\mathbf{g} = \mathbf{x}(\Theta) \tag{6}$$

where $\mathbf{x}(\Theta)$ and $\mathbf{g}$ are p-dimensional vectors expressed in the task space. The task function $\mathbf{x}(\Theta)$ represents the position or orientation of an end-effector frame while $\mathbf{g}$ is the goal to be reached.

One approach to solve IK problems, with optimization criteria, is the well known Resolved Motion Rate Control (RMRC) proposed by [Whi69]. The RMRC computes an optimal change, $\Delta\Theta$, to Θ, for a small change, $\Delta\mathbf{x}$, in $\mathbf{x}$. The final pose can be found adding $\Delta\Theta$ with respect to Θ ($\Theta = \Theta + \Delta\Theta$). To use the RMRC technique we need to compute the Jacobian matrix of the forward kinematics function, $\mathbf{x} = f(\Theta)$, and invert it:

$$\Delta\Theta = J(\Theta)^{-1}\Delta\mathbf{x} \tag{7}$$

where, $J(\Theta) = \partial\mathbf{x}/\partial\Theta$ is the Jacobian matrix of size $(p\times n)$. To solve Eq. 7 in this way $J(\Theta)$ should be square $(n = p)$ and non-singular. But, for a redundant manipulator this is not the case. The damped pseudo-inverse technique can be used deal with this problem [Mac90]:

$$\Delta\Theta = J(\Theta)^{\dagger^{\xi}}\Delta\mathbf{x} \tag{8}$$

where, $J(\Theta)^{\dagger^{\xi}}$ is the damped pseudo-inverse. The damped factor, ξ, is added to prevent Jacobian's singularities and to stabilize IK solutions [Mac90]. Eq. 8 can be solved by Singular Value Decomposition (SVD) exploring the null space of the Jacobian matrix. However, this Jacobian is not able to restrict the solutions to the motion pattern space, e.g., the space of golf swing motions, because it searches for solutions into the complete joint space.

We alleviate this problem by exploiting the PC space restricted to the single frame we want to constraint. This approach reduces the Jacobian's size and consequently restricts the solution to the motion pattern space. For

 ISBN 978-80-86943-98-5

example, as we want to preserve the golf swing characteristics as much as possible, we restrict the solution in the space of similar poses.

We underconstrain the problem by considering just the parameters corresponding to a specific normalized time, t_k. The pose to constrain can be represented as a function of the Principal Coefficients and the normalized time:

$$\Theta_{t_k} = Z\left(t_k, \alpha_1, .., \alpha_m\right) \tag{9}$$

So, to compute the new pose configuration for the displacement $\Delta\mathbf{x}$, we have:

$$\Delta\alpha = J(\alpha)^{\dagger^{\xi}}\Delta\mathbf{x} \tag{10}$$

where, $J(\alpha) = \partial\mathbf{x}/\partial\alpha$ is the Jacobian matrix, of size $(p \times m)$, relating the task increment with the α_i coefficients. Once, we have the new increment $\Delta\alpha$, the updated pose can be found computing Eq. 3 just for the normalized time of the constrained pose. The Jacobian, $J(\alpha)$, can be easily computed with the chain rule:

$$\left[\frac{\partial\mathbf{x}_l}{\partial\alpha_i}\right] = \left[\frac{\partial\mathbf{x}_l}{\partial\theta_j}\right] \cdot \left[\frac{\partial\theta_j}{\partial\alpha_i}\right]_{t_k} \tag{11}$$

The derivatives of $\left[\frac{\partial\mathbf{x}_l}{\partial\theta_j}\right]$ can be fast and easily computed [Bae01]. Besides, θ_j are linear combinations of the $\mathbf{E}_i$ principal components, so $\left[\frac{\partial\theta_j}{\partial\alpha_i}\right]_{t_k}$ is simply extracted from the jth coordinate of $\mathbf{E}_i$ [UF04] for the time t_k. We denote this Jacobian as $J(\mathbf{E})_{t_k}$ its size is only $n \times m$, to be compared to the much larger size $D \times m$ of the PC basis. The size of the Jacobian $J(\alpha)$ is much smaller than the Jacobian used in Eq. 8 (Figure 1).

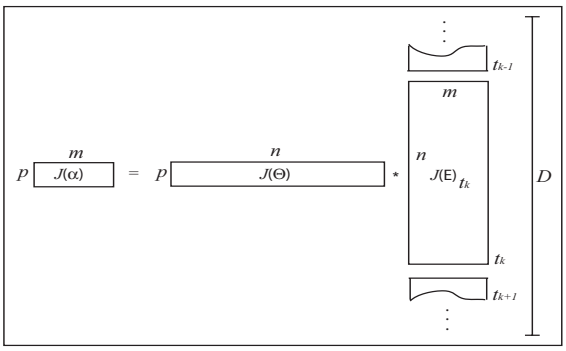


Figure 1. Obtaining $J(\alpha)$ from $J(\Theta)$ and the t_k slice of the Principal Components basis.

In the experiments reported in this paper (section 6), we have $p = 3$; $m = 9$ and $n = 93$. $J(\alpha)_{3\times 9}$; $J(\Theta)_{3\times 93}$; $J(\mathbf{E})_{t_k 93\times 9}$ and the PC basis is as 12276×9 already said.

4.3 Continuity

When a motion is deformed at specific times continuity between frames should be maintained [Gle01]. In our case, continuity is enforced through the PCA parameters. As the α_i coefficients (Eq. 3) are considered invariants for all frames of the motion (Eq. 2). Then, if we modify the α_i coefficients for a pose Θ_{t_k}, the updated $\widetilde{\alpha}_i$, generate a full motion belonging the PC space.

More precisely, to deform a given motion, Ψ_D. We estimate its α_D coefficients projecting this motion in the linear model described by Eq. 3. The next step is to choose the pose $\Theta_{D_{t_k}}$, which should be edited, and extract its corresponding Jacobian, $J(\mathbf{E})_{t_k}$. Then, Eq. 10 is iteratively solved to compute the desired increment $\Delta\alpha_D$. The final step is to use the new $\widetilde{\alpha}_D$ coefficients ($\widetilde{\alpha}_D = \alpha_D + \Delta\alpha_D$) into Eq. 3 to impose continuity. Algorithm 1 describes the whole solution:

Algorithm 1 Motion deformation and continuity.

1: $\widetilde{\alpha}_D \leftarrow \alpha_D$; $\Delta\alpha_D \leftarrow \mathbf{0}$; $J(\mathbf{E})_{t_k} \leftarrow \mathbf{E}_{t_k}$
2: **while** not converged **do**
3: Compute $\{J(\Theta_{D_{t_k}}), \Delta\mathbf{x}\}$
4: $J(\widetilde{\alpha}_D) \leftarrow J(\Theta_{D_{t_k}})J(\mathbf{E})_{t_k}$
5: $\Delta\alpha_D \leftarrow J(\widetilde{\alpha}_D)^{\dagger^{\xi}}\Delta\mathbf{x}$
6: $\widetilde{\alpha}_D \leftarrow \widetilde{\alpha}_D + \Delta\alpha_D$
7: $\Theta_{D_{t_k}} \leftarrow \Psi_{\circ t_k} + \widetilde{\alpha}_D\mathbf{E}_{t_k}$
8: **end while**
9: $\widetilde{\Psi}_D \leftarrow \Psi_\circ + \sum_{i=1}^{m}\widetilde{\alpha}_{D_i}\mathbf{E}_i$

The convergence of algorithm 1 is achieved by tracking the error of the norm of the difference between the current state and the desired state: $\mathbf{e} = \|\mathbf{x}(\Theta) - \mathbf{g}\|$. We verified the algorithm's convergence for a simple example depicted by Figure 2. In this example, the golf club head should hits the ball onto another position defined by the user.

Figure 2. Left side: original pose $\Theta_{D_{t_i}}$. Right side: edited pose. This final pose was computed using 9 PC.

Figure 3 shows the convergence of Algorithm 1 as a function of the number of α_D coefficients. The convergence is faster when more than five dimensions are used (Table 1) as the solution posture is more likely to belong to the generated pose space for t_k. When less than six dimensions were used, we noticed the presence of some undesirable artifacts (e.g., feet sliding) during reconstruction (step 9 of the Algorithm 1). This problem is a consequence of the low percentage, corresponding to such a small number of PC, used during reconstruc-

 ISBN 978-80-86943-98-5

tion [SHP04] and not of the quality of the updated $\widetilde{\alpha}_D$ coefficients.

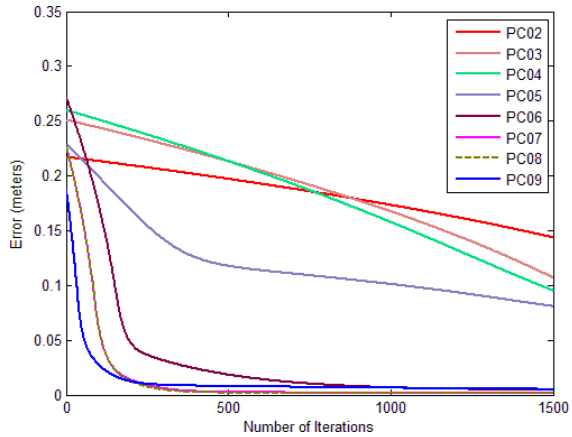


Figure 3. Convergence of algorithm 1. The convergence runs faster when more than five PC are used. We set 1500 iterations and chose a fixed value for the damping factor, $\xi = 10$.

PC	Time (seconds)	Nb. of Iterations
2	40.000	19106
3	110.000	50214
4	42.000	19929
5	9.000	4194
6	1.625	757
7	0.500	235
8	0.484	221
9	0.531	247

Table 1: Computation time, from lines 2 to 8 of the Algorithm 1, as a function of the number of PC. We used the values $\xi = 10$ and $\mathbf{e} < 0.01m$.

5 MOTION NORMALIZATION

A motion duration normalization is necessary in our approach because the PCA's parameters are estimated from complete motions. This is required as the single constrained posture should be as much as possible independent from duration variations of the original captured motions. For example, the golf club head should hit the ball at the same normalized time for all motions.

Let us recall the relationship between the motion index frame f_r, the frame-rate F_r (e.g., 25 fps), the motion duration T, and the normalized time:

$$f_r = TF_r t_k \tag{12}$$

Then, the total number of frames, F_T, can be expressed:

$$F_T = TF_r \tag{13}$$

The normalization is done in two stages. First, each motion Ψ_i has its corresponding total time T_i (Figure 4(a)). So, we preprocess the motion database to compute the mean time, T_μ (left side of Eq. 14), of all motions. We use this time directly into Eq. 13 to establish the final normalized size, which is $F = 132$. The normalization is done using quaternion spherical interpolation [Sho85] [1].But, for the root position we proceed with a simple linear interpolation.

The first step produces the normalized motions database, $\widetilde{\Psi}$ (Figure 4(b)). Second, we search in this normalized database for the key frame, K_{f_i}, for which the golf club should hits the ball. Then, we compute the mean key frame, K_{f_μ}, from all these motions:

$$T_\mu = \frac{1}{M}\sum_{i=1}^{M} T_i, \quad K_{f_\mu} = \frac{1}{M}\sum_{i=1}^{M} K_{f_i} \tag{14}$$

The mean key frame is used to compute the durations T_{μ_1} and T_{μ_2}, for the first half, $\left[K_{f_0}, ..., K_{f_\mu}\right]$, and second half, $\left[K_{f_{\mu+1}}, ..., K_{f_1}\right]$, of the final normalized motion, $\widetilde{\Psi}_{\mu_i}$ (Figure. 4(c)). Each interval is warped according to these two durations. The final normalized animations, $\widetilde{\Psi}_\mu$, are used to estimate the PCA's parameters (section 4.1).

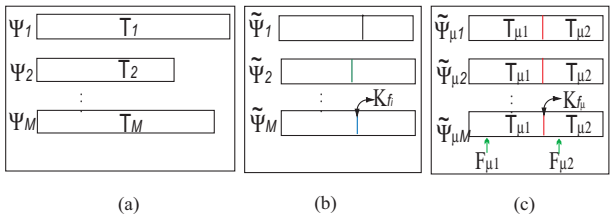


Figure 4. Motion normalization process.

6 EXPERIMENTS

We analyzed the performance of our approach in a number of experiments. We verified its performance, the style preservation of the edited motions, and its extrapolation capabilities. As a final experiment, we compare our model against a per-frame motion deformation technique [CB06] regarding performance, expressivity and simplicity.

6.1 Editing and Style

Figures 7(b) and 6 show the performance and style preservation results of the proposed model for motion editing. Figure 7(a) depicts the five goal positions (white balls) where the golf club head should hits the ball. The start position is the same for all five editing operations and is described by the blue ball. These positions were intentionally chosen to coincide with some of the positions of the motion database - to see if the edited motion would converge towards the original motion group or some of the others groups. For example, position 1

[1] The initial representation of the joint angles is exponential map. Before the interpolation, we convert them to quaternion. After the interpolation, we come back to exponential map to estimate the parameters of the linear model.

 ISBN 978-80-86943-98-5

(Figure 7(a)) corresponds to the 3D position of the golf club head for the motion of group 2 (Figure 6), the input motion, which belongs to group 1, should hits the ball in that position. Figure 6 depicts the first two α_i coefficients of Eq. 3 for the 10 golf swing motion vectors. Note that, vectors corresponding to similar styles of golf swing tend to cluster. Figure 5 shows the percentage of the database (Eq. 4) for the first 9 PC. The 10th PC is negligible.

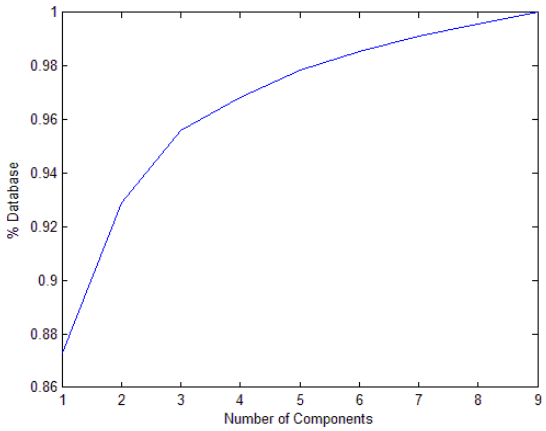


Figure 5. Percentage of the database.

Figure 7(b) shows the performance results. Each optimization took from 1 to 5 seconds approximately to converge. The algorithm's convergence depends of some parameters: the number of principal components (which we set to 8) and the intrinsic parameters of the optimizer (i.e., the damping factor, $\xi = 10$, the number of iterations, which we set to 1500, and the value of the tracked error, which we set to 0,01m). A high value of the regularization term ξ guarantees a stable but slow convergence. When ξ is zero the algorithm converges quickly (i.e, less than 12 iterations) but some undesirable poses and consequently discontinuities problems were produced for goals far from the initial end-effector position. This undesirable behavior happens because the IK solver tries to achieve unreachable poses, i.e, poses that are not in the database. The choice of a suitable value for ξ is thus one of the subtle difficulties to yield a good compromise between robustness and efficiency [Mac90, BB04]. The tracked error and the number of iterations were used to restrict the region of feasible poses during parameter's extrapolation. It means that, if the IK solver tries to achieve unreachable poses, the ones that are not in the motion database, the algorithm does not converge, but produces a result minimizing the norm of the error. Using these criterions a compromise between parameters' extrapolation and efficiency were established.

Finally, we verified that the edited motions preserved the same characteristics as the input motion (Figure 6). The optimization criteria used to solve the IK problem (Eq. 10) provides a local solution with minimum norm close to the starting state. Thereby, the final motion solution is found in the neighborhood of the starting state. This was checked by re-projecting the first two updated $\widetilde{\alpha}_{D_i}$ coefficients into the original golf motion database, as shown by Figure 6.

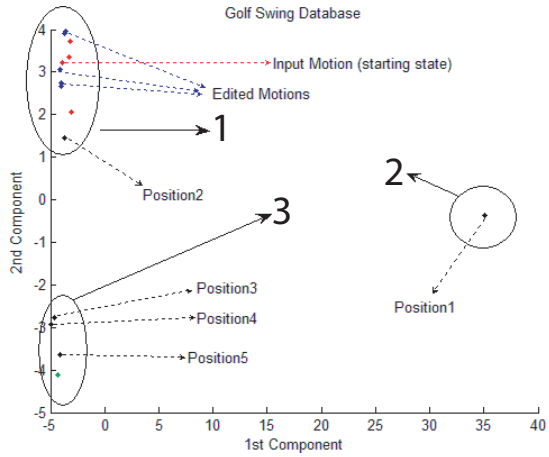


Figure 6. Clustering behavior of the motion database. Three groups of styles were found. The motions corresponding to the positions of Figure 7(a) are showed by black dots. The edited motions are showed by the blue dots. Three groups of style are showed.

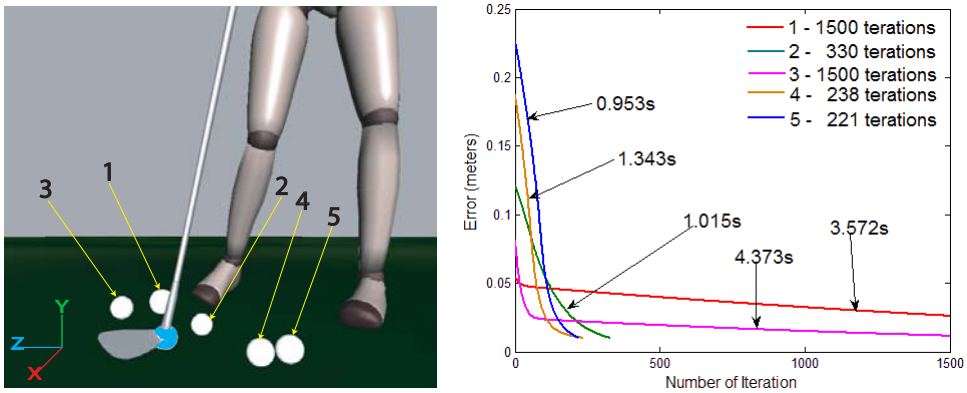
6.2 Visual Comparison

In this section, we compare the visual quality of the edited motion. To edit a motion by using the per-frame method proposed by [CB06] the user has to specify the trajectory of the end-effector (modeled as spline curves) and the time intervals for which the motion will be deformed. On the contrary, by using our approach the user needs to specify the final goal position of the end-effector just for one frame - the one that should be deformed. Now, for visual comparison of both techniques, we asked the user to achieve the editing task of shifting the hit of the golf club head (in front of the left foot, at frame 94). The first line of Figure 8 shows the starting motion with the ball in the middle of the feet, the second line shows the animation result with our approach and the third line the result with the per-frame motion deformation method.

To edit this simple motion by using the per-frame approach, the user needed to add four more constraints in addition the golf club head constraint: three on the feet to prevent foot sliding and one to control the center of mass, to prevent unbalance poses. The user has to establish the ease-in and ease-out time intervals to prevent discontinuities. These time intervals increase computation cost. For this simple example, the computing cost was approximately 100 sec, without visualization. We also noticed inter penetration of the arms at frame 93, and discontinuities between frames (e.g., body balance) around the hit position. Of course, these problems can be fixed, but more tuning is needed.

On the contrary, with the proposed motion editing model only one constraint was necessary - the golf club head constraint - to achieve more realistic results. Our ap-

 ISBN 978-80-86943-98-5



(a) (b)

Figure 7. Motion editing performance. The convergence is stopped when the end-effector is less than 1 cm of the goal. This threshold is used to delimit a region of feasible poses.

proach provided more natural poses (i.e., there is no inter penetration of the arms). The proposed model for continuity demonstrated more stable results, we did not noticed unbalanced poses or foot sliding. The computing cost was approximately 1 sec, without visualization. To summarize, if the motion needs just an adjustment of one key event, our approach can provide faster and more realistic results.

7 CONCLUSION AND FUTURE WORK

We have presented a new approach for interactive constraint-based motion editing, by constraining only a single pose in the motion space of the Principal Components and by imposing continuity exploiting this space. We demonstrated the performance, flexibility and simplicity of this approach through the editing of a golf swing motion and by comparing the results against a per-frame prioritized IK technique. Our approach provided faster and better quality results with less tuning from the user side.

One current limitation of our framework, compared with existing motion editing techniques [CB06], is that the constraint is applied to a single frame. Extending it to constrain multiple frames is possible but one has to keep in mind the small dimension of the PC space, that can quickly lead to an overconstrained problem. One possibility to alleviate this problem is to assign them a different priority [BB04]. Another possibility could be to solve them separately and interpolate the PC solutions between their associated frames. In any case the small computing cost for finding a solution in the presented context (less than 4s in general) makes our approach well suited for exploiting more versatile objectives.

8 ACKNOWLEDGEMENTS

Many thanks to Benoît Le Callennec, to providing access to his source code and for discussions about his method (with the support of the Swiss National Fund grant n° 200020-109989). The authors would like to thank to Autodesk/Maya for their donation of Maya software. The authors would like to thank Raquel Urtasun and Pascal Fua for proving the data used in this paper and for part of the PCA source code. Thanks to Mireille Clavien for video production. This project was sponsored by the EPFL - Sport and Rehabilitation Engineering program.

9 REFERENCES

[AM00] M. Alexa and W. Muller. Representing animations by principal components. *Computer Graphics Forum*, 19(3):411–418, 2000.

[Bae01] Paolo Baerlocher. *Inverse Kinematics Techniques Of The Interactive Posture Control Of Articulated Figures*. Phd thesis, École Polytechnique Fédéral de Lausanne, 2001.

[BB04] P. Baerlocher and R. Boulic. An inverse kinematic architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer*, 20(6), 2004.

[CB06] B. Le Callennec and R. Boulic. Interactive motion deformation with prioritized constraints. *Graphical Models*, 68 (2):175–193, March 2006. Special Issue on SCA 2004.

[Cra86] J. J. Craig. *Introduction to Robotics*. Addison-Wesley, Reading MA, 1986.

[GBT04] P. Glardon, R. Boulic, and D. Thalmann. A coherent locomotion engine extrapolating beyond experimental data. *Proceedings of CASA*, 2004.

[Gle97] M. Gleicher. Motion editing with spacetime constraints. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 139–ff., New York, NY, USA, 1997. ACM Press. ISBN 0-89791-884-3.

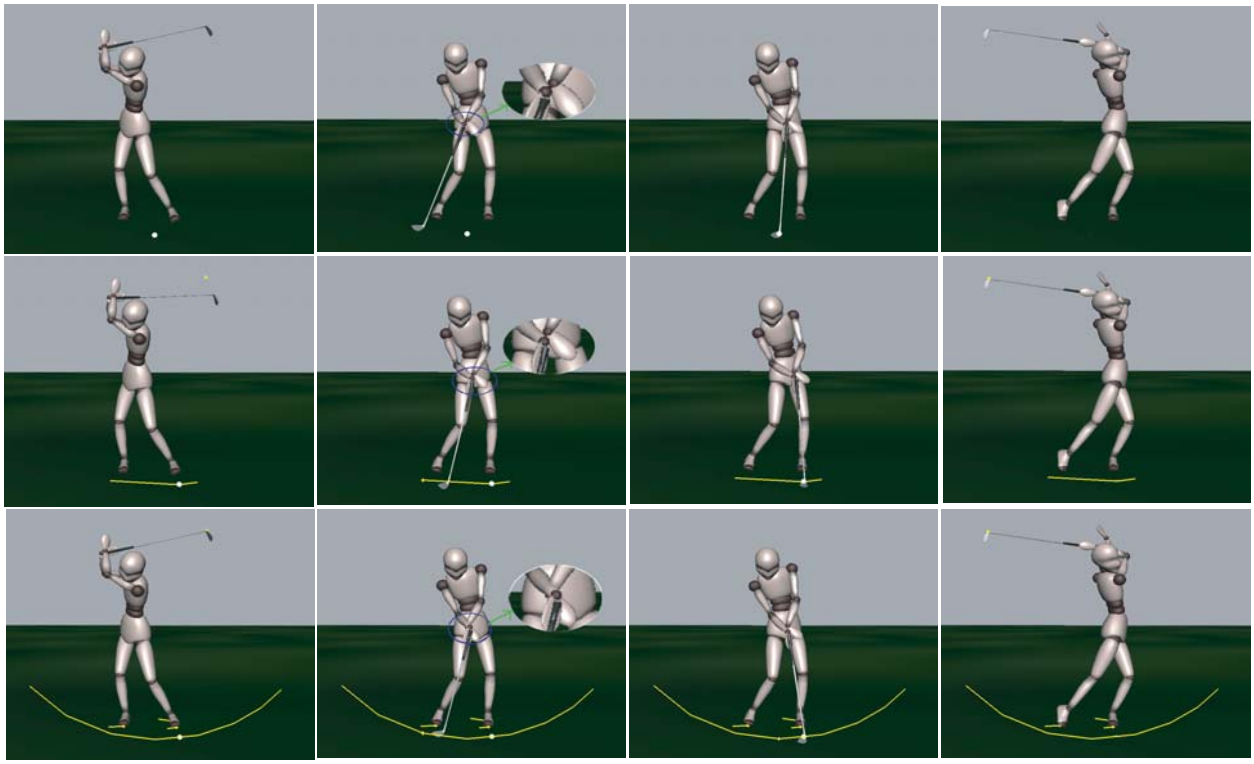


Figure 8. Animations results. Top row: original motion. Middle row: edited motion using the proposed approach. Bottom row: edited motion using the prioritized approach. Yellow lines show ease-in and ease-out time intervals. The prioritized approach produced a motion with inter penetration of the arms (second picture from left to right).

[Gle01] M. Gleicher. Comparing constraint-based motion editing methods. *Graphical models*, 63(2):107–134, 2001.

[Gra98] F. S. Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3(3): 29–48, 1998.

[Jol86] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

[KMA05] R. Kulpa, F. Multon, and B. Arnaldi1. Morphology-independent representation of motions for interactive human-like animation. In *EUROGRAPHICS*, volume 24, pages 343–352, Number 2005.

[LS99] J. Lee and S.Y. Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of ACM SIGGRAPH*, 1999.

[LZqDMSH06] L. Liu, W. Zhao-qi, Z. Deng-Ming, and X. Shi-Hong. Motion edit with collision avoidance. In *Proceedings of the WSCG*, pages 303–310, January 2006.

[Mac90] A.A. Maciejewski. Dealing with the ill-conditioned equations of motion forarticulated figures. In *Computer Graphics and Applications, IEEE*, volume 10, pages 63–71, May 1990.

[Sho85] K. Shoemake. Animating rotation with quaternion curves. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254, New York, NY, USA, 1985. ACM Press. ISBN 0-89791-166-0.

[SHP04] A. Safonova, J. K. Hodgins, and N. S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. Graph.*, 23(3):514–521, 2004.

[UF04] R. Urtasun and P. Fua. 3d human body tracking using deterministic temporal motion models. In *European Conference on Computer Vision, Prague, Czech Republic*, May 2004.

[UGB+04] R. Urtasun, P. Glardon, R. Boulic, D. Thalmann, and P. Fua. Style-based motion synthesis. *Computer Graphics Forum (CGF)*, 23(4):799–812, November 2004.

[Whi69] D. E. Whitney. Resolved motion rate control of manipulators and human prostheses. In *IEEE Trans. Man-Mach. Syst*, volume 10 (2), pages 47–53, June 1969.

 ISBN 978-80-86943-98-5

Benchmarking Speech Synchronized Facial Animation Based on Context-Dependent Visemes

José Mario De Martino
Dept. of Computer Eng. and Industrial Automation
School of Electrical and Computer Engineering
State University of Campinas, CP 6101
13083-970, Campinas, SP, Brazil
martino@dca.fee.unicamp.br

Fábio Violaro
Department of Communications
School of Electrical and Computer Engineering
State University of Campinas, CP 6101
13083-970, Campinas, SP, Brazil
fabio@decom.fee.unicamp.br

ABSTRACT

In this paper we evaluate the effectiveness in conveying speech information of a speech synchronized facial animation system based on context-dependent visemes. The evaluation procedure is based on an oral speech intelligibility test conducted with, and without, supplementary visual information provided by a real and a virtual speaker. Three situations (audio-only, audio+video and audio+animation) are compared and analysed under five different conditions of noise contamination of the audio signal. The results show that the virtual face driven by context-dependent visemes effectively contributes to speech intelligibility at high noise degradation levels (Signal to Noise Ratio (SNR) ≤ -18dB).

Keywords

Facial animation, Facial animation evaluation, Context-dependent visemes, Speech intelligibility test.

1. INTRODUCTION

Facial animation has been a research topic for more than three decades. Since Park's pioneer work [Par72], different approaches have been proposed to improve the various aspects involved in facial animation. Among them, the proper reproduction of the visible speech articulatory movements in synchronization with the audio is a crucial issue for many applications. Examples of such applications include virtual characters for films, embodied conversational agents and virtual helpers for language learning.

Viseme, the shorthand of visual phoneme, is the term used to denote the recognizable visual motor patterns common to two or more speech segments [Jac88]. A viseme is a visually contrastive unit usually associated with more than on speech segment. Speech segments represented by the same viseme are produced with similar visible speech articulatory movements and therefore are not visually distinguishable. Such segments are called homophenes. However a viseme representation can be affected by coarticulation. Coarticulation refers to the altering of the set of articulatory movements made in the production of one speech segment by those made in the production of an adjacent or nearby one [BP82]. Coarticulation can be classified as antecipatory or perseverative. Antecipatory coarticulation takes place when the articulatory movement of a given segment is influenced by the articulation of a following one. Perseverative coarticulation refers to the change in the articulation pattern of a given segment influenced by the production of a preceding one. Therefore visemes are dependent on the phonetic context of the acoustic production of the associated speech segment.

One strategy for speech synchronized facial animation aimed at reproducing the visible movements and associated coarticulation effects is based on context-dependent visemes [DMV06]. The central idea of the context-dependent viseme approach is the characterization of visemes not as visual representations of isolated speech segments, but as a composite including the influence of coarticulation. The main advantage of context-dependent visemes lies in its relatively simple way to handle antecipatory and perseverative coarticulation effects.

In this paper we assess the effective contribution to speechreading of context-dependent visemes

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.

approach. The contribution to speechreading is an objective measurement of speech synchronized facial animation quality. The benchmark procedure used is derived from the oral speech intelligibility test proposed by Sumby and Pollack [SP54]. The same test, with minor variations, has been widely applied to evaluate speech synchronized facial animation [BL98] [PO99] [Bes04] [MOC+05]. Essentially the intelligibility test involves the presentation of audio signals with, and without, supplementary visual information, conducted under different conditions of acoustic degradation.

2. RELATED WORK

Other strategies have been proposed to model coarticulation and visible articulatory movements in the context of facial animation. Pelachaud and colleagues [PBS96] specified a set of visemes based on the informal observations of Jeffers and Barley [JB71] and proposed a three-step algorithm to handle coarticulation that ranks visemes according to deformability. In this approach, the ideal lip shape of a deformable phoneme is influenced by the shape of a less deformable phoneme. Cohen and Massaro [CM93] implemented a coarticulation model based on Löfqvist's gestural theory of speech production [Lof90]. In this model, each speech segment is associated with a target and a dominance function. A weighted sum of dominance functions specifies the trajectory of the articulators. Le Goff and Benoît [LB96] [BL98] extended Cohen and Massaros's coarticulation model to get an n-continuous function. Revéret and colleagues [RBB00] adopted the Öhman's coarticulation model [Ohm67]. Albrecht et al. [AJS02] also extended the original Cohen and Massaro model to speed up the computation of coarticulation effects. Pelachaud [Pel02] used radial basis functions to model the trajectory of articulatory parameters. Beskow [Bes04] implemented and compared four coarticulation models: those of Cohen and Massaro, and Öhman's, as well as, two artificial neural network-based models. All the models performed equally well in the perceptual evaluation realized. All of these approaches use blending functions to model coarticulation.

In contrast, our approach does not define ideal targets and associated blending functions, but rather seeks to establish a set of context-dependent visemes that already incorporates the effects of coarticulation. The appropriate concatenation of context-dependent visemes reproduces the visible articulatory movements during speech production.

3. OUR APPROACH

Our approach characterizes visemes as transitions between context-dependent articulatory targets. The articulatory target of a speech segment is the distinctive conformation or posture of the vocal tract necessary for the acoustic production of the segment. Instead of treating speech segments in isolation, we consider the phonetic context of its production as well.

Articulatory targets

In order to identify context-dependent articulatory targets we measured and analysed the visible 3D trajectories of fiduciary points marked on the face of a real speaker during the speech production. The trajectories were measured from standard video sequences using stereovision photogrammetric techniques. The linguistic analysis was carried out for Brazilian Portuguese and based on a corpus of non-sense paroxytone words of the type $^{\prime}CV_1CV_2$ and diphthongs of the type $^{\prime}V_3V_2$, where C_1 = [p, t, k, f, s, ʃ, l, ʎ, ɣ(ɾ)], V_1 = [i, a, u], V_2 = [ɪ, ɐ, ʊ] and V_3 = [i, e, ɛ, a, ɔ, o, u], expressed with symbols of the International Phonetic Alphabet [IPA99]. As the alveolar tap [ɾ] never occurs in Portuguese at the beginning of a word, we chose, regarding [ɣ, ɾ], to analyze only the non-sense words of the type [ɣ]V_1[ɾ]V_2. More details about the reasoning behind the selection of this set of segments can be found in [DMV06].

A male speaker born and raised in the city of São Paulo, Brazil, was recorded producing the 102 stimuli composing the corpus. Figure 1 presents the location and labelling of the four fiduciary points, P_1, P_2 P_3 and P_4, for which 3D trajectories were measured and analysed.

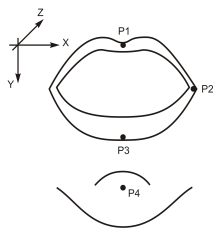


Figure 1: Fiduciary points.

The stationary points (derivative equal to zero) of the fiduciary point trajectories during segment production were considered as the segment articulatory target. The articulatory targets of a same segment in its various phonetic contexts in the corpus were then cluster analysed using the K-means clustering algorithm and the euclidian distance as a similarity criterion [Job92]. As result, a set of phonetic context-dependent articulatory targets was identified. Once the articulatory target clusters were established, an average value for the relative instants of realization of all the articulatory targets composing the cluster was adopted as the relative instant of

 ISBN 978-80-86943-98-5

realization of the representative articulatory target (centroid) of the cluster. The coordinates (x, y, z) and the timing of the articulatory target are used as the driving parameters of our facial animation system. Given the timed sequence of speech segments that compose an utterance, the fiduciary point trajectories were approximated by the smooth interpolation between the corresponding articulatory targets using a Hermite parametric cubic curve. These trajectories, which define visual motor patterns, and therefore visemes, are then used to drive a set of geometric transformation/deformation models that reproduce the rotation and translation of the temporomandibular joint on the 3D virtual face, as well as the behavior of the lips, such as protrusion, and opening width and height.

Speech-related facial movements

In order to control the dynamic behavior of a 3D synthetic face, the modeled fiduciary points movements were broken down into three components: a rigid body component associated with the mandible movement and two non-rigid components associated with the upper and lower lip movements.

3.1.1 Mandible movement

During speech, the mandible rotates and slides forward and backwards due to the temporomandibular joint. The temporomandibular joint, or TMJ, is the joint connecting the mandible to the temporal bones at both sides of the head. Figure 2 presents the typical behavior of the temporomandibular joint within the midsagittal plane during speech. The TMJ behavior can be modeled by the composition of a rotation around the center of the TMJ in rest position at initial time t_0 when the mouth is closed, followed by the translation of this center.

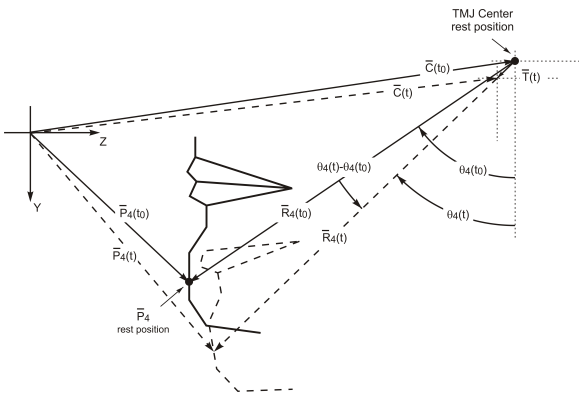


Figure 2: Temporomandibular joint behavior.

From the modeled trajectory of fiduciary point P_4, we calculate the rotation angle $q_4(t)$ and translation components $t_y(t)$ and $t_z(t)$ of the TMJ in the mid-sagittal plane xy using Equations 1 and 2, respectively.

$$q_4(t) = \frac{z_4(t) - c_z(t_0)}{y_4(t) - c_y(t_0)} \tag{1}$$

$$\begin{cases} t_y(t) = y_4(t) - c_y(t_0) - r_4 \cos(q4(t)) \\ t_z(t) = z_4(t) - c_z(t_0) - r_4 \sin(q4(t)) \end{cases} \tag{2}$$

In the above equations y_4 and z_4 are the coordinates y and z of the modeled trajectory of point P_4, and $c_y(t_0)$ and $c_z(t_0)$ are the y and z components of vector $\overline{C}(t)$, the TMJ center of rotation, at rest position ($t = t_0$). The radius r_4, the module of vector $\overline{R}_4(t)$, is a constant and is defined by the size of the mandible. It is possible to estimate r_4 in the position of rest by Equation 3.

$$r_4 = \sqrt{[y_4(t_0) - c_y(t_0)]^2 + [z_4(t_0) - c_z(t_0)]^2} \tag{3}$$

Figure 3: Vertices associated with the mandible.

To reproduce the movements of the mandible in the synthetic face, the rigid body transformations of rotation and translation described by TMJ behavior were applied to the polygonal vertices of the geometric model. The transformed vertices were those within the region of the face alongside the mandibular bone. More precisely, the vertices in the region below and including the lower lip and the lateral side of the face below an imaginary plane defined by TMJ rotation axis and the corners of the mouth. The lower bound of the mandibular region is defined by the neck. The white dots shown in Figure 3 show the vertices of the synthetic face submitted to these transformations. The rotation axis defined by the TMJ is represented in the figure by the white cylinder piercing the surface of the virtual face in front of the ears.

3.1.2 Lip movements

The movement of the fiduciary point P_3 located on the lower lip can be decomposed into two components. The first one is due to the mandible rotation and translation. The second one is the voluntary movement of the lower lip tissue necessary to produce specific speech gestures, such as lip protrusion. This first component is directly derived from the TMJ movement previously discussed. The

 ISBN 978-80-86943-98-5

second component is given by subtracting the movement of the TMJ from the trajectory of P_3.

Differently, the displacement of P_1, located on the upper lip, is only driven by the voluntary movement of the upper lip tissue.

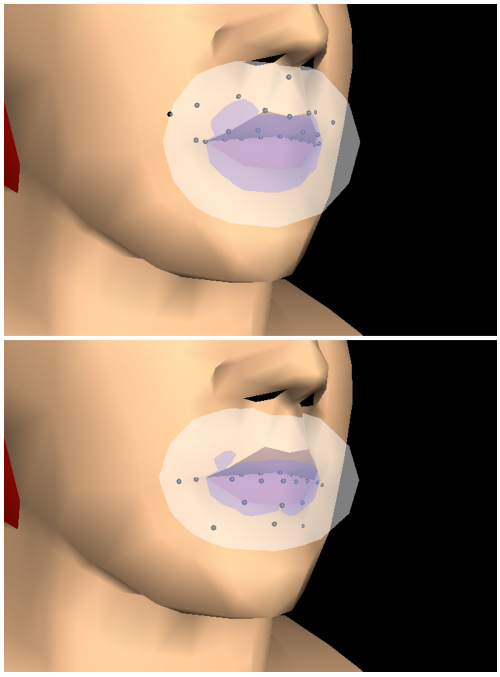


Figure 4: Vertices associated with the upper lip (above) and lower lip (below) movement.

The behavior of the upper and lower lips was mapped onto the synthetic face on the basis of three main considerations. First, the points on the geometric model corresponding to the fiduciary points must exactly follow the fiduciary point trajectories expressed by the viseme model. Second, during speech production, the skin tissue around the mouth, including the lips, suffers deformations primarily attributed to the sphincteral behavior of the *Orbicularis Oris* muscle, which has an elliptical constitution. Third, other muscles, which also influence the movement of the skin around the mouth, are distributed asymmetrically with respect to the horizontal plane.

Based on these considerations, a geometric model was derived to express the visible characteristics of the skin around the mouth during speech production. We approximated the region of influence of the *Orbicularis Oris* muscle with a spheroid. To accommodate for the asymmetrical characteristics of muscle insertions, the area around the mouth is divided into two regions, with the upper and lower regions influenced by the behavior of the upper and lower lips, respectively. Actually, each region of influence is defined by two spheroids: an internal and an external one. The external spheroid, which is merely a scaled instance of the internal one, defines the limits of influence of the lip behavior, while the internal one defines the points of maximum influence of that behavior. The influence of the lip behavior decays as one moves away from the surface of the internal spheroid, and ceases completely outside the external spheroid. The spheroids are expressed by Equation 4a (internal) and Equation 4b (external). The spheroids assume a Cartesian reference system centered in the mouth, with the same orientation as that of Figure 1.

$$\frac{x^2}{a^2}+\frac{y^2}{b_i^{\,2}}+\frac{z^2}{b_i^{\,2}}=\begin{cases}1 & \text{(4a)}\\ F_i^{\,2} & \text{(4b)}\end{cases} \qquad i=1,3$$

The parameters a, b_i and F_i, with i = 1, 3 (i = 1 upper lip; i = 3 lower lip), are defined by the face geometry: The parameter a is equal to half of the distance between the corners of the mouth, that is, half the distance between P_2 and its counterpart on the other side of the mouth; The parameter b_i is equal to the distance between the major axis of the spheroid and the fiduciary point P_i; the scale factor F_1 is defined by the distance from the upper lip to the bottom center edge of the nose (to limit the upper region of influence at the columella-labial junction); F_3 is defined to limit the lower region of influence to the point halfway between the midpoint of the cleft and the tip of the chin. Figure 4 shows the region of influence defined by the spheroids and the vertices of the 3D face model included in the lower and upper region.

Figure 5: The posture of the synthetic face during the production of /a/ (left side) and /u/ (right side).

Equation 5 gives the displacement $\Delta\overline{V}$ of a vertex inside a region of influence.

$$\Delta\overline{V} = R_i\left[D_i\ \Delta\overline{P}_2 + \left(1-D_i\right)\Delta\overline{P}_i\right] \qquad (5)$$

where $\Delta\overline{P}_2$ is the displacement of the fiduciary point P_2; $\Delta\overline{P}_i$ is the displacement of fiduciary point P_i, i =1,

 ISBN 978-80-86943-98-5

3 (i = 1 for the upper region of influence; and i = 3 for the lower one); $0 \le D_i \le 1$ is an interpolation factor given by Equation 6; and $0 \le R_i \le 1$ is an attenuation factor given by Equation 7.

$$D_i = \left[\cos\left(\frac{p\ d_2}{d_i + d_2} \right) + 1 \right] \Big/ 2 \qquad (6)$$

where d_2 is the distance between the vertex, whose displacement is being calculated, and the fiduciary point P_2 at rest position, and d_i is the distance between the vertex and fiduciary point P_i, i = 1,3, at rest position.

Depending on whether the vertex is inside or outside the internal spheroid, the fall-off factor R_i is calculated by:

$$R_i = \begin{cases} \cos\left((1 - S_i) \frac{p}{2} \right) & \textit{inside} \\ \cos\left(\frac{(S_i - 1)}{(F_i^2 - 1)} \frac{p}{2} \right) & \textit{outside} \end{cases} \qquad (7)$$

The factor S_i attenuates R_i as the location of the vertex moves away from the surface of the internal spheroid. S_i is obtained from the evaluation of the left side of Equation 4 at the vertex location.

To illustrate, Figure 5 presents snapshots showing the virtual face postures during the production of phonemes /a/ and /u/. Note the lip protrusion and rounding during /u/ and opening during /a/, in perfect agreement with real articulation.

4. EVALUATION

Method

The intelligibility test carried out to evaluate our approach consisted on the presentation of a set of 27 non-sense words or logatomes conveyed in a vehicle phrase uttered by a Brazilian male speaker. The adopted vehicle phrase had the following structure: “Eu falo <*logatome*>” (“I say <*logatome*>”). The vehicle phrase was devised as a mean to get the test subject's attention prior to the logatome utterance. The recorded audio was contaminated with noise in order to produce five different Signal-to-Noise-Ratio (SNR) conditions: 0dB, -6dB, -12dB, -18dB and -24dB.

The speaker was recorded in digital Mini-DV format using a JVC DV-GY500 camcorder and a Shure SM58 microphone. The camera was positioned to capture a front view of the speaker. The front view was chosen because it is the normal face-to-face conversation view. The recorded raw material was transferred to a video editing system iFinish V60, version 3.2, for further manipulation. After phrase segmentation, the audio was detached from the video. The audio-only files were used as the base material for the generation of new versions with different SNRs, resulting a total of 135 (27 x 5) audio files. The degradation of the acoustic signal was realized through the addition of noise.

The original, not degraded, audio signal was manually segmented at the speech segment level (phones). The resulting timed phonetic transcription was used to drive our facial animation system, which was configured to generate and store each frame of the animation in TGA format with the standard NTSC resolution of 720 x 468. Copies of the degraded audio files were then properly (re)synchronized with the corresponding real speaker video and with the facial animation frames. The material was packed in QuickTime “.mov” files with no audio compression and with video compression using the Soreson codec bundled in the iFinish system. As final result, 3 sets of 135 files were prepared for presentation: an audio-only, an audio+video, and an audio+animation set.

The resulting 405 files were organized for presentation in 15 groups, each containing the 27 different logatomes (and the associated vehicle phrase) with the same audio quality. The file contents of these groups were presented to the test subjects in a random order. Furthermore, in each group the presentation order of the 27 logatomes was also randomly varied.

A special software tool was implemented to present, collect and record the vote of the test subjects.

Corpus

The logatomes used in the intelligibility test were paroxytone logatomes of the type ˈCVCV, with C = /p, t, k, f, s, ʃ, l, ʎ, ɣ/, V = /i, a, u/. The logatomes formed by the concatenation of two CV contexts were chosen to stimulate the production of coarticulation effects during its utterance.

The consonant set was selected considering the following homophene grouping of the Brazilian consonants: /p, b, m/, /f, v/, /t, d, n/, /s, z/, /l/, /ʃ, ʒ/, /ʎ, ɲ/, /k, g/, /ɣ/, and /ɾ/. Consonants of a same homophene group are not visually distinguishable. With one exception, a representative consonant of each homophene group was considered in the intelligibility test. As the /ɾ/ does not occur at the beginning of a word in Portuguese, we decided to let it out of our evaluation set. The vowel set used is formed by the three extreme vowels of Brazilian Portuguese.

Noise Contamination

The audio material was originally recorded at 48 kHz sampling rate and 16 bits resolution. The audio files

 ISBN 978-80-86943-98-5

were then downsampled to 16 kHz sampling rate, and manually segmented using spectrogram analysis. The timing and phonetic identification of the segment was stored in a text file and used for controlling and speech synchronizing the facial animation.

The downsampled audio files were converted to float format, and normalized to the interval [-1.0, 1.0] after a division by 32768. Five versions of each vehicle phrase were prepared with SNRs of 0, -6, -12, -18 and -24 dB. A uniform distribution random noise was employed. The noise was added over the whole phrases, but the SNR level was calculated only considering the logatome. After the noise addition, the audio amplitude was renormalized to the [-1.0, 1.0] interval and again quantized to 16 bits resolution and packed into RIFF WAVE files.

Subjects

33 subjects were invited to participate in the intelligibility test. This group of individuals consisted of normal-hearing undergraduate and graduate students, and administrative personnel of our School. They had no prior knowledge of the test purpose or any involvement with the facial animation research. Their age ranged from 19 to 55 years.

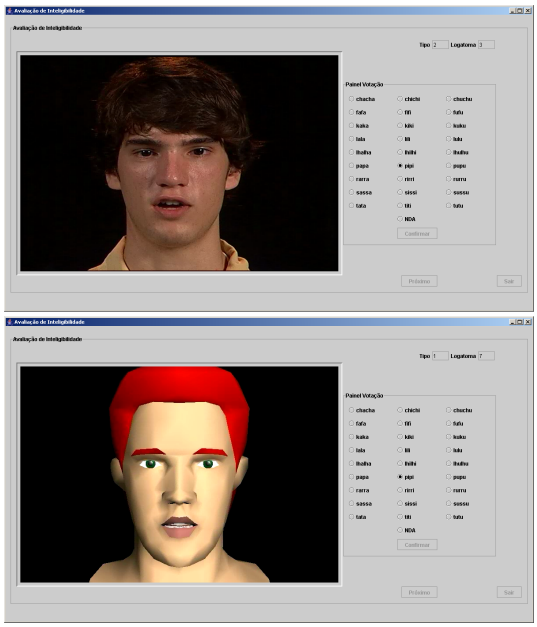


Figure 6: Screenshots of the voting tool showing above a frame of the real speaker video and below a frame of the facial animation.

Evaluation Environment

The intelligibility test was conducted on an isolated small dark room (3.5mx3.5m) with low ambient noise. A java application (j2sdk1.4.2_08) running on a Compaq Professional Workstation SP750 with an Intel Pentium III Xeon 733MHz processor was used to present the test material and collect the subject's responses. The workstation was equipped with a 21 inch Compaq X1100 color monitor. Figure 6 shows screenshots of the evaluation software. Each subject heard the audio material using a good quality headset plugged to the microcomputer audio card.

After presentation of a vehicle phrase, each subject was asked to indicate the understood logatome checking one of the 28 available options, consisting of the 27 logatomes and a NDA (none of the alternatives) option. Figure 7 shows the voting panel in detail. After the vote confirmation realized through the Confirm button (*Confirmar* in Portuguese), the subject had to click the Next button (*Próximo* in Portuguese) in order to start the next presentation.

Figure 7: Detail of the voting panel.

The subjects were strongly encouraged to guess even if they were in doubt between different logatome options, and only to choose the NDA (none of the alternative) option when they had absolutely no clue about the logatome presented or they had "heard" something else not provided as one of the logatome options. The subjects were informed that the logatomes were of the type CVCV and that different logatomes from those shown as options could be presented.

Results

The results of the intelligibility test are summarized in Figure 8. The figure shows the percentage of correct responses to the logatome stimulus. It can be seen that the visual information provided by both real and virtual speaker increases the speech intelligibility. This gain increases with the audio degradation, indicating that valuable and effective speech information is provided by the visual information. For SNR= -18 and -24 dB, the intelligibility gain provided by the inclusion of the facial animation is 25% and 10%, respectively, both

 ISBN 978-80-86943-98-5

with a statistic significance $p < 0.001$. The corresponding gain of the real speaker's video amounts to 41% and 44%, respectively. These scores are well above the score by chance of 3.7%. For SNR = -12 dB, -6 dB and 0 db there is no statistical difference between the audio-only and audio+animation conditions.

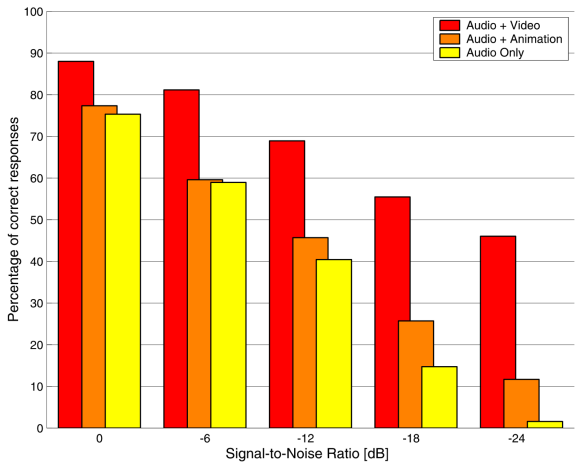


Figure 8: Percentage of correct responses.

A boxplot of the correct responses is shown in Figure 9. The 15 analysed cases are presented in the same order as in Figure 8. Each box has lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of the box to show the extent of the rest of the data. Outliers are data with values beyond the ends of the whiskers and are indicated by crosses.

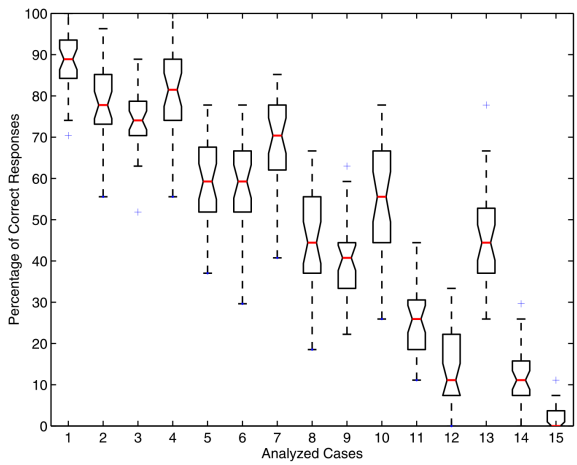


Figure 9: Boxplot of correct responses.

5. CONCLUSION & FUTURE WORK

The results so far show an improvement of the speech intelligibility due to the facial animation based on context-dependent visemes. The comparison with the real speaker video defines a reference to orient our future work. Although the context-dependent viseme approach helps speechreading, there is some room for improvement as it is not as effective as the real video. Some aspects of our approach still claim further elaboration. At least two aspects deserve special attention.

The first aspect is the refinement of the viseme model. The quality of the viseme representation in our approach is at some extent dependent on the number and distribution of the fiduciary points. Currently, the results are based on the analysis of only 4 points at normal video sampling rate (30 fps). It is expected that the analysis of more points in space and time will result in a better viseme representation.

The second aspect concerns the strategy used to manipulate the synthetic face. We used a pure geometric strategy to map the viseme representation given by the fiduciary point trajectories onto the virtual face, abstracting the existence of the skin and muscles and their biomechanical properties. An improvement of our approach could be potentially achieved by the use of a muscle-based approach to drive the facial animation. Of course, it is an open question if and in what extent a more complex muscle-based approach could improve speech intelligibility, specially because there are many biomechanical parameters whose values are not easy to determine but yet have to be properly tuned. To assess and compare results a version based on biomechanics simulation is currently being implemented. This version is based on the muscle-model approach proposed by Lee, Terzopoulos and Waters [LTW95]. We plan to test and compare in a near future the current version and the muscle-model version.

It is well known that viseme perception depends on talker, language, stimuli, subjects, response task, and environmental characteristics such as lighting, distance, and angle of observation. If all that is true for real visual speech, it gets worse in the context of facial animation. For facial animation, the coarticulation model and the strategy used to manipulate the virtual face also play an important role. It is not an easy task to compare and clearly identify the aspects that effectively contribute to the realism of speech synchronized facial animation. The systematic evaluation of the effective contribution to speechreading of a facial animation system seems to be an appropriate approach to the problem. In this paper we presented an evaluation of a facial animation system based on context-dependent visemes. The final result is a benchmark to be used for comparing developments using this and other approaches.

6. ACKNOWLEDGMENTS

The synthetic face used in this research is a modified version of the Miraface polygonal 3D face model

developed at MIRALab, University of Geneva, and published by ISO as MPEG-4 reference software.

7. REFERENCES

[AJS02] I. Albrecht, J. Haber and H.-P. Seidel. Speech Synchronization for Physics-Based Facial Animation, In *Proceedings of the 10th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision - WSCG '02*, Skala, V. (ed.), Union Agency, Plzen, Czech Republic, pp. 9-16, February 2002.

[Bes04] J. Beskow. Trainable Articulatory Control Models for Visual Speech Synthesis International. *Journal of Speech Technology*, 7(4), pp. 335-349, October 2004.

[BL98] C. Benoît and B. Le Goff. Audio-visual speech synthesis from French text: Eight years of models, designs and evaluation at the ICP. *Speech Communication*, 26(1-2), pp. 117-129, October 1998.

[BP82] A.-P. Benguerel and M.K. Pichora-Fuller. Coarticulation Effects in Lipreading. *Journal of Speech and Hearing Research*, Vol. 25, pp. 600-607, December 1982.

[CM93] M.M. Cohen and D.W. Massaro. Modeling Coarticulation in Synthetic Visual Speech. In *Models and Techniques in Computer Animation*, N. Magnenat-Thalmann and D. Thalmann (eds.), Springer-Verlag, pp. 139-156, 1993.

[DMV06] J.M. De Martino, L.P. Magalhães and F. Violaro. Facial Animation Based on Context-Dependent Visemes. *Computers and Graphics*, 30(6), December 2006 (accepted for publication).

[Jac88] P.L. Jackson. The Theoretical Minimal Unit for Visual Speech Perception: Visemes and Coarticulation. *The Volta Review*, 90(5), pp.99-115, 1998.

[JB71] J. Jeffers and M. Barley. *Speechreading (Lipreading)*, Charles C. Thomas Publisher, Springfield, Illinois, USA, 1971.

[Job92] J.D. Jobson. *Applied Multivariate Data Analysis – Volume II: Categorical and Multivariate Methods*, Springer-Verlag New York Inc., USA, 1992.

[IPA99] *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*, International Phonetic Association, Cambridge University Press, 1999.

[LB96] B. Le Goff and C. Benoît. A Text-To Audiovisual-Speech Synthesizer for French. In *Proceedings of the 4th International Conference on Spoken Language Processing - ICSLP '96*, Vol.4, pp. 2163-2166, October 1996.

[LTW95] Y. Lee, D. Terzopoulos and K. Walters. Realistic face modeling and facial animation. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques - SIGGRAPH '95*, pp. 55-62, 1995.

[Lof90] A. Löfqvist. Speech as Audible Gesture. In *Speech. Production and Speech Modeling*, W.J. Hardcastle and A. Marchal (eds.), Kluwer Academinc Publishers, pp. 289-322, 1990.

[MOC+05] D.W. Massaro, S. Ouni, M.M. Cohen and R. Clark. A Multilingual Embodied Conversational Agent. In *Proceedings of 38th Annual Hawaii International Conference on System Sciences - HICCS'05*, IEEE Computer Society, January 2-6 2005, CD-ROM 8 pages.

[Ohm67] S.E.G. Öhman. Numerical model of coarticulation. *The Journal of the Acoustical Society of America*, 41(2), pp. 310-320, 1967.

[Par72] Parke, F. I. *Computer generated animation of faces*. Master's thesis, University of Utah, June 1972.

[Pel02] C. Pelachaud. Visual Text-to-Speech. In *MPEG-4 Facial Animation*, I.S. Pandzic and R. Forchheimer (eds), John Wiley and Sons, pp. 125-140, 2002. ISBN 0-470-84465-5.

[PBS96] C. Pelachaud and N.I. Badler and M. Steedman. Generating Facial Expressions for Speech. *Cognitive Science*, 20(1), pp. 1-46, January-March 1996.

[PO99] I.S. Pandzic, J. Ostermann and D. Millen. User evaluation: Synthetic talking faces for interactive services. *The Visual Computer*, 15(7-8), pp. 330-340, November 1999.

[RBB00] L. Revéret, G. Bailly and P. Badin. Mother: A new generation of talking heads providing a flexible control for video-realistic speech animation. In *Proceedings of the 6th International Conference on Spoken Language Processing - ICSLP'00*, Beijing, China, ISCA, pp. 755-758, 2000.

[SP54] W.H. Sumby and I. Pollack. Visual Contribution to Speech Intelligibility in Noise, *The Journal of the Acoustical Society of America*, 26(2), pp. 212-215, March 1954.

 ISBN 978-80-86943-98-5

Improving Locality for Progressive Radiosity Algorithm: A Study based on the Blocking Transformation of the Scene.

J. R. Sanjurjo
Department of Electronics and Systems
Univertity of A Coruña
E–15071, A Coruña, Spain
josesan@udc.es

M. Amor
Department of Electronics and Systems
Univertity of A Coruña
E–15071, A Coruña, Spain
margamor@udc.es

M. Bóo
Department of Electronics and Computer Engineering
Univertity of Santiago de Compostela
E–15782, Santiago de Compostela, Spain
elmboo@usc.es

R. Doallo
Department of Electronics and Systems
Univertity of A Coruña
E–15071, A Coruña, Spain
doallo@udc.es

ABSTRACT

Efficient global illumination is an important challenge in computer graphics. The main problem of these algorithms is their associated execution time and storage requirements. This is a handicap for simulating large scenes and reducing these costs is a constant research objective of the computer graphics community.
In this paper we present a performance analysis of the progressive radiosity algorithm based on the blocking transformation of the scene according to a uniform partition. The data locality exploitation and the associated decrement in cache misses permit the reduction of the execution time of the algorithm. An extended analysis of the influence of the scene subdivision on the execution time requirements is presented. Because of our analysis, we conclude that important performance improvements in terms of execution time are obtained with this technique.

Keywords
Radiosity, data locality, scene partition, blocking transformation.

1. INTRODUCTION

The radiosity method [Coh93a] is a global illumination algorithm used to simulate light transfer in synthetic images. The main drawbacks of the radiosity method are its high storage requirements and long execution times [Sil94a]. Among the different radiosity algorithms we have focussed our analysis on the progressive radiosity algorithm [Coh88a, Coh93a], due to its simpler structure. Similar analyses can be performed on other radiosity strategies, e.g. for the hierarchy radiosity algorithm [Han91a].

In general, the performance of a code is influenced by the data reuse and data locality exploitation. It is considered data reuse when the next utilization of a data does not imply an access to the lower level of the memory hierarchy. In fact, there can be a wide performance gap between programs that are designed to optimize cache performance and those that are not, for example in the context of computer graphics [Cho06a].

In this work we present a theoretical data reuse analysis of the progressive radiosity algorithm. Based on the cache miss rate obtained, we have concluded that a data reuse technique can be employed to optimize the algorithm. Specifically, our optimization proposal is based on the blocking transformation of the scene through its uniform partition. This permits the optimization of data reutilisation and, with this, the acceleration of the algorithm.

Scene partitioning is a technique usually employed for distributed memory systems. On such platforms, the effective distribution of data among processors is very important. In some proposals the scene is replicated on all processors [Gar00a, Pad01a], but this limits the applicability of the solutions to scenes with low complexity. A valid method for complex scenes implies the distribution of the scene among the processors [Arn96a, Gib00a, Amo04a, Gue03a, San05a]. These proposals employ different methods for scene partition and distribution among processors.

Based on this idea, we work on one partitioning procedure [Gue03a, San05a] in our attempt to exploit and

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.

 ISBN 978-80-86943-98-5

analyze the data locality of the sequential algorithm. Scene partitioning implies different challenges. For example, the visibility technique employed to determine the visibility among elements associated with different partitions. This technique, as will be shown in the paper, also has an associated computational cost that has to be considered. We do not propose an accurate study of the cache behaviour, but an extended analysis of the benefits of the partitioning strategy. This analysis is validated with a set of test scenes.

The rest of this paper is organized as follows: in Section 2 we briefly introduce the progressive radiosity algorithm; in Section 3 we analyze the data reuse in the algorithm; in Section 4 we present the block transformation strategy based on a partitioning of the scene; experimental results are shown in Section 5; finally, in Section 6 we present the main conclusions.

2. THE PROGRESSIVE RADIOSITY ALGORITHM

The radiosity method is currently one of the most popular global illumination models in computer graphics. This method solves a global illumination problem expressed by the rendering equation [Kaj86a], simplified by considering only ideal diffuse surfaces. The resultant discrete radiosity equation is:

$$B_i = E_i + \rho_i \sum_{j=1}^{N} B_j F_{ij} \qquad (1)$$

where B_i is the radiosity of patch P_i, E_i is the emittance, ρ_i the diffuse reflectivity and N is the number of patches of the scene. The summation represents the contribution of the other patches of the scene and F_{ij} is the *form factor* between patches P_i and P_j. F_{ij} is an adimensional constant that only depends on the geometry of the scene and represents the proportion of the radiosity leaving patch P_i that is received by patch P_j. There is one radiosity equation per patch and one *form factor* among all pairs of patches. Therefore, interaction of light among the patches in the environment can be stated as a set of simultaneous equations:

$$\begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1N} \\ \rho_2 F_{21} & 1-\rho_1 F_{22} & \cdots & -\rho_1 F_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ \rho_N F_{N1} & -\rho_N F_{N2} & \cdots & 1-\rho_N F_{NN} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \end{bmatrix} \qquad (2)$$

where the matrix is called the *Interaction matrix*. Therefore, the radiosity equation system is liable to be very large with complexity $O(N^2)$.

Some algorithms, such as the progressive radiosity proposal [Coh88a], have been developed for the minimization of these computational requirements. The basic idea of the algorithm is to reduce the number of patches to be considered for each iteration and, with this, to reduce the number of computations.

```
1  while (not converged){
2        Pi = 0;
3        /* LOOP1: Select Patch with greatest ΔBiAi */
4        for ( P = scene→Patch_initial; P; P = P→next)
5              Pi = MaxPot(Pi, P);
6
7        /* LOOP2: Calculate (the first time that a patch
                 is shoot ) Patches interaction with Pi */
8        for ( P = scene→Patch_initial; P; P = P→next)
9              Interaction(P, Pi);
10
11       /* LOOP3: Shoot Radiosity from patch Pi */
12       for ( Pj = Pi →Interaction; Pj;
               Pj = Pi → Interaction→ next) {
13             Vij = Visibility(Pi, Pj);
14             Gij = Geometry(Pi, Pj);
15             Fij = Gij · Vij;
16             Radiosity(Pi, Pj);
17             }
18       }
```

Figure 1: Pseudocode for progressive radiosity algorithm.

The structure of the progressive radiosity algorithm is outlined in Figure 1. The key idea is to *shoot* in every step the energy of the patch P_i with the greatest unshot radiosity (lines 2 to 5). As a result of that shot, the other patches, P_j, may receive some new radiosity and patch P_i left without un-shot radiosity (lines 7 to 17).

Each shooting step can be viewed as multiplying the value of radiosity to be "shot" by a column of the *interaction matrix*. Therefore at that moment the *form factors* between the shot patch and the rest of the patches have to be calculated (lines 13 to 15). V_{ij} (line 13) is the visibility factor and takes the value 1 if P_i is visible from P_j, and 0 otherwise. In our implementation we have used a method based on directional techniques [Hai94a, Pad03a] to calculate the visibility term. G_{ij} (line 14) captures all the geometric terms. In our algorithm we have used the analytic method called *differential area to convex polygon* [Coh93a] to compute it.

As soon as the radiosity of a patch has been shot, another patch is selected. A patch may be reselected for the shooting procedure if it has received light from other patches. Therefore, the amount of radiosity the patch has received since the last time, ΔB_i, can be shot. The algorithm iterates until a desired tolerance is reached (line 1). Thus, the complexity would be $O(k \times N)$, k being the number of steps performed, with $k \ll N$.

3. DATA REUSE ANALYSIS IN PROGRESSIVE RADIOSITY ALGORITHM

In this section, we make a quantitative analysis of the cache behaviour for the progressive radiosity algorithm.

 ISBN 978-80-86943-98-5

The objective is to analyse the nature of the cache misses associated with the algorithm for a later optimization of the algorithm based on the data locality exploitation. As will be shown later in this section, the progressive radiosity algorithm requires that, for standard scene sizes and hierarchical memory structures, the same patch is retrieved multiple times from lower levels of the memory hierarchy.

As can observed in the pseudocode of Figure 1, the progressive algorithm consists of three main loops where read and write operations are performed on the sequential list of N patches of the scene. We will define the number of memory references (Nr) as the number of times a patch is referred. Taking into account this definition and assuming usual scene and memory sizes, the number of memory references (Nr) required by the different data structures to perform the calculation of radiosity for each loop is:

$$\begin{aligned} Nr_{Loop1} &= N \times k \\ Nr_{Loop2} &= N \times t \\ Nr_{Loop3} &= \overline{M} \times k \end{aligned} \tag{3}$$

where N is the number of patches, k the number of iterations, t the number of different patches that are shot, $t \leq k$, and being $\overline{M}$ the average number of accessed objects per iteration. Loop1 selects the patch P_i with maximum power and Loop2 the patches P_j that interact with it. Both loops present a sequential access nature. On the other hand, Loop3 exhibits a behaviour difficult to predict and exploit. This is due to the irregular structure of the algorithm employed to compute the visibility. In our case we have employed a directional technique that implies the access to potential candidates to include the interaction between two polygons. Taking into account this information, the total number of references is:

$$Nr = (N + \overline{M}) \times k + t \times N \tag{4}$$

As result, each patch has to be accessed multiple times and this leads to a large memory bandwith and a overhead of cache misses. As N is a very large number for usual scene sizes, the cache cannot contain all the blocks needed during the execution of the standard radiosity algorithm (code of Figure 1). Then, the replacement misses will occur in all loops (following the notation of [Gho99a] we call replacement misses capacity and conflict misses).

To clarify let us consider the example of Figures 2. This figure illustrates an interaction matrix for a given scene with $N = 16$ patches. In this figure, a symbol $\times$ indicates that there is an interaction between the patches associated with the row and the column. For example, Patch 0 interacts with patches $\{1,2,3,4,7,9,11,12,13,15\}$. We assume for this simple example a small cache with capacity of four data. We employ a fully associative1 cache because of its

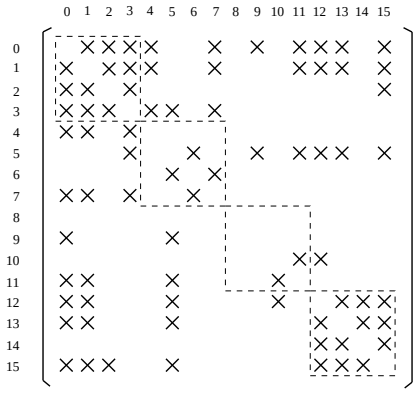


Figure 2: Interaction Matrix for a scene with 16 patches.

Address of reference	Hit/ /miss	Contents of cache blocks after reference			
		Block 0	Block 1	Block 2	Block 3
-	-	12	13	14	15
1	miss	1	13	14	15
2	miss	1	2	14	15
3	miss	1	2	3	15
4	miss	1	2	3	4
7	miss	7	2	3	4
9	miss	7	9	3	4
11	miss	7	9	11	4
12	miss	7	9	11	12
13	miss	13	9	11	12
15	miss	13	15	11	12

Table 1: Cache contents for the example of Figure 2: Standard progressive radiosity algorithm.

good performance. This simple example with a small cache permits the illustration of the low exploitation of the data locality when the available storage resources are not sufficient to store the full scene.

Let us suppose patch 0 is the patch with the greatest unshot radiosity. A snapshot of the data accesses for the progressive radiosity algorithm is shown in Table 1 . The first row shows the contents of cache after the execution of Loop2 and the rest of the table illustrates the series of references for Loop3. As an example, note that when patch 4 is referenced it replaces patch 15. This is due to the associative nature of the cache and we have choosen the replace method of the least recently used block. Therefore, this set of references generates 10 misses for 10 accesses. This simple example shows the low exploitation of the data locality for this system.

The performance of a code is highly influenced by data reuse and the available memory resources. The number of times a patch is referenced during the full execution of the algorithm constitutes a tool to evaluate the data transfer in the memory hierarchy for large scenes. Specifically, the redundancy access for the progressive algorithm can be measured using the following equation:

$$Ra = \frac{Nr}{N} = \frac{(N + \overline{M}) \times k + t \times N}{N} = k + \frac{\overline{M}}{N} \times k + t \tag{5}$$

For the scenes we have employed in our tests and for the convergence criteria we have employed (line 1 of code

 ISBN 978-80-86943-98-5

in Figure 1) we observe that $\overline{M} \simeq N^2/4$ and $t \simeq k$. The scenes employed in our tests have very different structures and sizes and, consequentially, we consider these approaches as representative of a generic case. Therefore:

$$Ra = \left(2 + \frac{N}{4}\right) k \tag{6}$$

Consequently, each patch is accessed an average of $\left(2+\frac{N}{4}\right)k$ times. Taking into account this large number of references, it is important to maximize the accesses to each patch while it is in the cache. Reducing the cache misses permits the reduction of the number of times the same patch have to be loaded in the cache and, consequently, the reduction of the execution time of the algorithm.

4. DATA LOCALITY OPTIMIZATION BASED ON BLOCKING TRANSFORMATION

Transforming the procedures of a program might improve the spatial and temporal locality of the data. In this section we present a method to improve the data reuse based on the data locality exploitation. To increase the reutilization of the data, a blocking transformation strategy based on the partitioning of the scene is performed. Specifically, we propose the blocking transformation of the code to reduce the cache misses via improving temporal locality exploitation. The goal is to maximize accesses to the data loaded into the cache before the data is replaced.

To ensure that the data being accessed can fit in the cache, the original code is changed to work on sub-scenes of a lower size. To do this, the whole scene data is read and the bounding volume, which contains the entire scene, is computed. This volume is divided into p uniform disjoint subspaces, where p is the number of partitions. The partitioning strategy implies three main challenges: first, the partitioning procedure to be employed, second, the visibility technique to determine the visibility among patches in different sub-scenes and finally the scheduling procedure to sequentially process the different partitions.

The partitioning strategy should assure resulting sub-scenes with a convex structure. This permits the simplification of the visibility determination between patches of the same sub-scene and between patches of different sub-scenes. Within the wide range of existing partitioning techniques, we have focused on static graph partitioning algorithms, since the progressive radiosity method (unlike, for instance, hierarchical radiosity method) does not use an evolving data structure. This way, it is not necessary to apply more complex dynamic graph partitioning methods. Within static techniques we have chosen geometric techniques, as we are dealing with geometric scenes and these techniques obtain an extremely fast partition based mainly on geometric information. A similar strategy was employed in [Amo04a] in the context of the parallelization of the progressive algorithm on distributed memory systems.

The version we have employed makes a geometric uniform partition of the input scene. It splits the scene domain into a set of disjoint subspaces of the same shape and size. This kind of partitioning is computed quickly and straightforwardly and, as will be shown later, it also provides high data locality. Taking into account that all subspaces are equal-sized, we cannot divide a scene for an odd number of sub-scenes into two halves and then, recursively, split one of the sub-scenes into two new halves, because the resulting subspaces would not be of equal size. So the implemented algorithm is not recursive and the number of splitting planes that are needed to split the scene along each coordinate axis is computed in advance. Finally, each partition has a disjoint set of polygons of the whole scene.

The visibility computation among patches is a challenge when the patches are in different sub-scenes. We have used a visibility mask technique [Arn96a, Gue03a] to solve this problem. This mask is a structure that stores all the visibility information encountered during the processing of a selected patch in its local sub-scene. This local visibility information can be employed for the computation of the other sub-scenes. In other words, the other sub-scenes do not require accessing the patch information associated to this scene, but to its visibility mask. This permits the reduction of data accesses with low locality.

The partitioning and visibility mask techniques have to be employed cautiously because both algorithms have associated computational costs that could result in an increment of the execution time of the algorithm. However, as will be shown in the result section, for a specific number of partitions these costs are completely overcome by the benefits associated with the data locality exploitation.

With respect to the new scheduling procedure, the general structure of the new algorithm is illustrated in Figure 3. The partitioning of the scene into sub-scenes, divides the calculations associated with each iteration. On one side, there is the gathering of local radiosity of the sub-scene, on the other side the gathering of remote radiosity coming from the rest of the scene. In each iteration, the sub-scenes are processed sequentially (line 2). For each sub-scene, the patch with the greatest unshot energy is chosen P_i^l (lines 4 to 7) and the patches in the same sub-scene that interact with it are identified (lines 9 to 11). After this, the local gathering of radiosity is computed following a similar procedure to the standard progressive radiosity algorithm but, as will be shown in the following, with slight modifications (lines 13 to 23).

 ISBN 978-80-86943-98-5

```
1  while (not converged){
2        for( sub-scene_l = scene→subcene_initial;
             sub-scene_l; sub-scene_l= sub-scene_l →next){
3
4            P_i^l = 0;
5            /* LOOP1: Select Patch with greatest ΔB_iA_i
                                   for subcene l */
6            for ( P = sub-scene_l →Patch_initial; P;
                 P = P→next)
7                P_i^l = MaxPot(P_i^l, P);
8
9            /* LOOP2: Calculate (the first time that a
             patch is shoot ) Patches interaction with P_i^l */
10           for ( P = sub-scene^l →Patch_initial; P;
                 P = P→next)
11               Interaction(P, P_i^l);
12
13           /* LOOP3: Shoot Radiosity from patch P_i^l */
14           for ( P_j = P_i^l →Interaction; P_j;
                 P_j = P_i^l → Interaction→ next) {
15               if ( P_i^l ∈ sub-scene_l )
16                 V_ij^local = Visibility_local(P_i^l, P_j);
17               else
18                 V_ij = Visibility_global(P_i^l, P_j)
19               G_ij = Geometry(P_i, P_j);
20               F_ij = G_ij · V_ij;
21               Radiosity(P_i, P_j);
22
23           }
24           if ( Mask(P_i^l) > 0 )
25             Gathering(P_i^l);
26       }
27 }
```

Figure 3: Pseudocode for progressive radiosity algorithm after blocking transformation.

Once the local radiosity is updated in the sub-scene its energy is propagated to the whole scene (lines 24 to 25). In this stage, the patch is copied to the sub-scenes of the adjacent partitions and the visibility information is calculated. As previously mentioned, we have employed the mask technique [Arn96a, Gue03a] that computes a visibility map per patch. This map stores the visibility of the patch with respect to the neighbouring scene, taking into account the occlusions associated with its local sub-scene.

Note that the shot patches of a sub-scene and their corresponding visibility masks are copied to the neighbour sub-scenes. This means that for the neighbour sub-scenes we have to process "local" and "copied" patches. This has to be taken into account for computing the gathering of radiosity inside each sub-scene (lines 13 to 23). In the case of a "copied" patch being processed, the gathering of energy is computed similarly but the non-local visibility information (line 18) is calculated by casting rays from patch to sub-scene through its visibility mask. The procedure follows for the adjacent sub-scenes with the same strategy, shooting patches are copied and visibility masks are computed/updated taking into account the occlusion information local to already processed sub-scene.

Address of reference	Hit/ /miss	Contents of cache blocks after reference			
		Block 0	Block 1	Block 2	Block 3
-	-	0	1	2	3
1	hit	0	1	2	3
2	hit	0	1	2	3
3	hit	0	1	2	3

Table 2: Cache contents for the example of Figure 2: Modified progressive radiosity algorithm.

This technique permits the evaluation of the progressive radiosity algorithm following a partitioning strategy. As will be shown in the following, benefits in terms of the cache misses reduction can be obtained. As an example, Table 2 shows the cache accesses after applying the blocking strategy to the example of Figure 2. In this example we consider that the scene was subdivided in four partitions, each sub-scene with 4 patches (indicated with dashed lines in the Figure 2). The first row of table shows the content of the cache after Loop2 and the remaining rows show the consecutive references for Loop3 associated with the local computations of the first sub-scene (patches 0, 1, 2, 3). The cache memory, with capacity for four data, stores the patches associated with the sub-scene and the local computations can be performed without cache misses. Note that for this example the set of references generates 3 hits for 3 accesses.

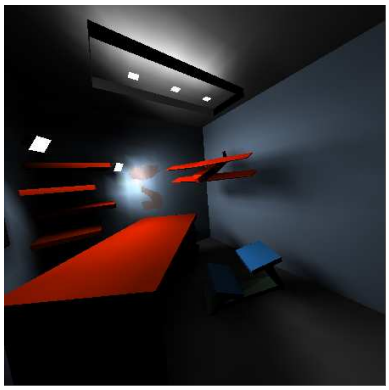
This simple example emphasizes the potential benefits that can be obtained with the partitioning technique. This way, if each sub-scene can be fully stored in the cache, the local computations can be performed without accessing the lower levels of the hierarchical memory. In more detail, let us compute the total number of memory references for the modified algorithm:

$$Nr = \sum_{l=1}^{p} \left(\underbrace{B^l \times k}_{t1} + \underbrace{B^l \times t}_{t2} + \underbrace{\overline{M^l} \times k}_{t3} \right) + (p-1) \times N_m \times k \tag{7}$$

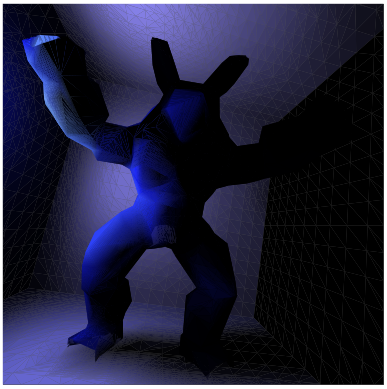
where p is the number of sub-scenes, B^l the number of patches in each sub-scene, $\overline{M^l}$ is the average number of accessed objects in each sub-scene per iteration and each mask consists of N_m patches. Note that the last term of equation is associated to the masks computation. But accordingly to its definition Nr is lower when the B^l patches fit into the cache level. In this case, once the sub-scene is stored in the cache for Loop1, the information is re-employed for the Loop2 and Loop3 without requiring more information. This means that terms $t2$ and $t3$ can be eliminated from the Nr equation. Therefore, the redundancy access is

$$Ra = \sum_{l=1}^{p} \frac{B^l \times k}{N} = k \tag{8}$$

 ISBN 978-80-86943-98-5



(a)



(b)

Figure 4: Illuminated scenes: (a) *room-model* and (b) *armadillo*.

Comparing this result with that obtained for the original algorithm (see Equation 6) we potentially have $\left(2+\frac{N}{4}\right)$ times less cache misses than with the original algorithm using the partitioning strategy. Note that the term associated to the masks computation is omitted as in this analysis we analyze only the redundancy access for the patches.

5. EXPERIMENTAL RESULTS

In this section, we present performance results and analysis of the proposed progressive radiosity method. We have analyzed the performance benefits of the partitioning strategy in terms of execution time and the reduction of cache misses. We have also verified that the quality of the final images is not affected by the techniques employed.

To evaluate our proposal we have used an Athlon 64 3.2 GHz with a 64 + 64 KB L1 cache, a 512 KB L2 cache and a 1 GB RAM. We have tested our proposal with different scenes. Here we include the result for three representative scenes, two of them depicted in Figure 4: *room-model* with 5306 patches (see Figure 4.a), *armadillo* with 3999 patches (see Figure 4.b) and *largeroom-model* with 7070 patches. The last scene, not included in the figure, is a larger *room-model* scene with more objects.

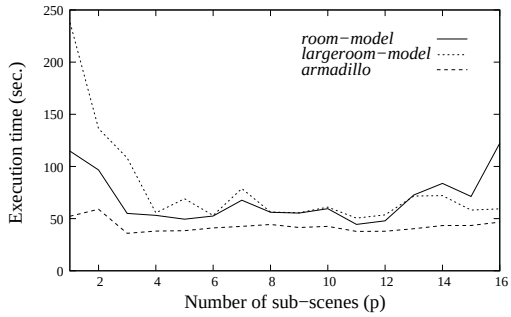


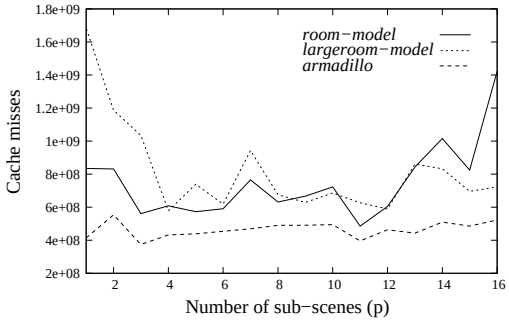
Figure 5: Execution time for *room-model*, *armadillo* and *largeroom-model*.

First, we evaluate performance in terms of execution time. The results for the test scenes on the *Athlon* system are depicted in Figure 5. No data for $p > 16$ are included as execution time increases in all cases. We observe that the partitioning strategy achieves a reduction in the required time when p is incremented for low p values, but the times increase for large p values. For example, we observe that for the *largeroom-model*, the execution time for the original algorithm is 238.63 seconds, while for the partitioned version with $p = 4$ this time is lowered to 55.54. Therefore, a high speedup of 4.30 is obtained for this example. This large reduction in the execution times is mainly due to two reasons: Firstly, the reduction in the cache misses due to data locality exploitation associated with the partitioning strategy and, secondly, the visibility mask technique, required to make the partitioning strategy viable, also has an inherent reduction in the computational time. This visibility technique implies an approximation to the visibility determination and, as a consequence, the reduction of the computational complexity.

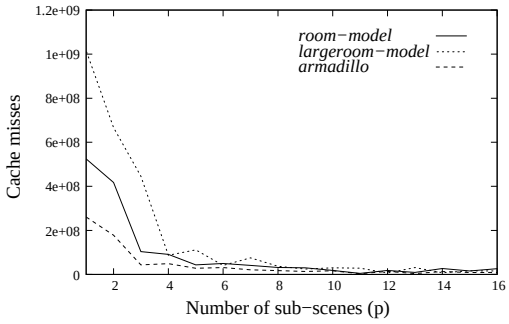
To verify the reduction in cache misses we have chosen a hardware measurement [Amm97a] that can accurately obtain a broad range of performance metrics for a program. Specifically, we have used the PAPI (*Performance Application Programming Interface*) [Moo01a] library that has the benefit of a cross-platform interface to the counters, allowing the maintenance of a common source for a wide variety of architectures. Figure 6 shows the number of cache misses for the *Athlon* system for our test scenes for the first level (see Figure 6.a) and for the second level (see Figure 6.b). For reasons of clarity the data miss rates are also depicted: for the first cache in Figure 7.a and for the second cache in Figure 7.b.

As can be observed in Figure 6.b and Figure 7.b the sub-scenes fit into the second cache level for the three scenes even for small p values. This can be deduced from the figures as increments in the p values imply reductions in the number of cache misses and the cache miss rate. This way, smaller sub-scenes fit into the memory and the data can be reutilized, reducing the

 ISBN 978-80-86943-98-5

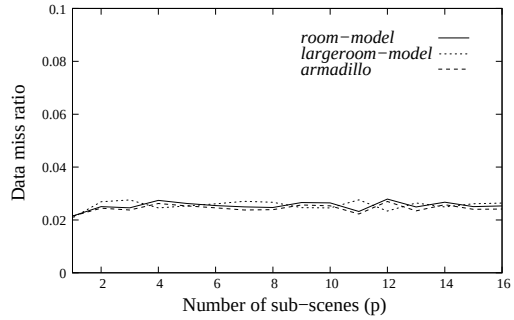


(a)

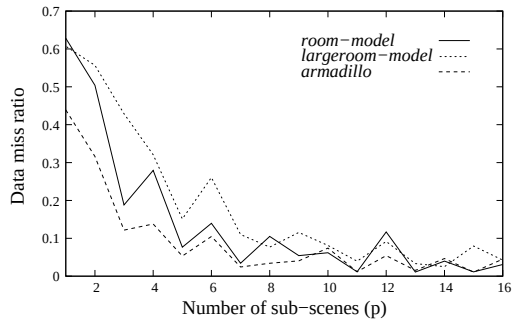


(b)

Figure 6: Cache misses (a) first level (b) second level.



(a)



(b)

Figure 7: Data miss ratio (a) first level (b) second level.

data traffic between memory levels. Note that, as can be observed in these figures, large p values do not eliminate the cache misses. These are compulsory misses, i.e. misses associated with the initial sub-scene loading each time a new sub-scene is processed.

With respect to the first cache level, Figure 7.a indicates that the cache miss rate is practically constant while Figure 6.a indicates that the number of cache misses decreases for intermediate p values. This means that there is a decrease of references for these p values. The reduction of loads and stores in the program is due to two reasons: the blocking helps register allocation and the utilization of visibility masks minimizes the accesses in Loop2.

The visibility masks technique has an important influence in the behaviour of the algorithm. This is due to the fact that, when a visibility mask of a patch has only zeros (line 23 of Figure 3), the patch is not further gathered to the other partitions. This contrasts with the original algorithm, in which interactions of this patch with all the patches of the scene would be tested, with the consequent increment in the cache misses. On the other hand, the utilization of visibility masks reduces the number of accesses to determine the visibility between two polygons. We have used a method based on directional techniques which calculates a list of potential candidates to occlude the interaction between two polygons. The idea is to remove from the list those patches which cannot be pierced by any ray cast between both polygons. Then, to determine the visibility between patches it is not necessary to access the list of patches that belongs to other sub-scenes and this permits a notable reduction of the accesses. At the same time, the computational requirements associated with the visibility mask technique are dependent on the p value. For low p values, the overhead associated with the visibility masks technique is clear, mainly for scenes with small number of triangles (see execution time values for two partitions of the *armadillo* in Figure 5). For large p values the computational requirements associated with the large number of patches that has to be "copied" to the other scenes can be prohibitive. Note that each time a patch is "copied" its visibility mask has to be computed/updated. As a result, an increment in number of sub-scenes implies an increment in the number of patches that have to be gathered to other sub-scenes.

In summary, the execution time of the modified algorithm is minimum for a given intermediate p value. The influence of the p value can be also deduced from Equation 7. For larger p values the last term of equation is larger while the t_3 is smaller ($\overline{M^l}$ is smaller). In consequence, the optimum p value depends on the scene, however, we have found in our tests that a $p \in [4,6]$ is a good partitioning value for the tests we have performed. Larger scenes would result in higher p values. With this optimum p value, the data locality is exploited because the sub-scenes fit into the second cache level. However, larger p values imply an increment in the computational

 ISBN 978-80-86943-98-5

complexity associated with the massive visibility masks computation and, as a consequence, an increment in the processing time.

Finally and with respect to the quality of the final images, we have not found any noticeable difference between the images generated with the standard algorithm and those generated with the modified algorithm. To further test the quality of the resulting images, we have used the mean residual error of the radiosity as the error metric. The numerical results indicate that the quality of the images is close to being the same for the original and the modified algorithms. For example, the result for the *largeroom-model* for the original algorithm is 0.000209 while for the partitioned version with $p = 4$ this error is 0.000240. In both cases the residual errors are very small. The visibility mask technique permits the elimination of computations and avoids testing the interaction between many pairs of patches. As an example, as was previously mentioned, a visibility mask with all zeros discards the interaction of the associated patch with any other patch in the following sub-scenes. However, although the number of visibility tests are reduced, no modifications are actually performed in the nature of the algorithm. Consequently, no changes in quality are expected.

6. CONCLUSIONS

In this paper we have proposed a technique for accelerating the progressive radiosity algorithm based on data locality exploitation. Specifically, we have employed a scene partitioning technique that permits the reduction of cache misses and, with this, the reduction of the execution time of the algorithm.

The simplicity of the partitioning strategy employed, permits an efficient partition of the scene with a very low computational complexity. On the other hand the visibility mask technique we have used allows the computation of the visibility information between patches in different sub-scenes. We have shown that this technique not only solves the problem of computing the visibility in a partitioned scene, but also reduces the visibility computation requirements without noticeable loss of quality in the final images.

Therefore, our modified progressive radiosity algorithm permits the obtaining of important reductions in the execution time with respect to the original algorithm, while keeping its quality properties. The benefits of our proposal have been experimentally evaluated for a set of representative scenes.

As future work, we plan to extend our analysis to a non-uniform partitioning strategy such as that proposed in [San05a]. We also plan to extend the analysis to other radiosity strategies, for example to the hierarchy radiosity algorithm [Han91a].

ACKNOWLEDGEMENTS

This work was partially supported by the Ministry of Science and Technology of Spain under contract MCYT TIN2004–07797–C02 and by the Secretaría I+D de Galicia (Spain) under the project PGIDT03TIC10502PR.

REFERENCES

[Amm97a] Ammons, G., Ball, T., and Larus, J. Exploiting Hardware Performance Counters with Flow and Context Sensitive Profiling. In *ACM SIGPLAn'97 Conf. Programming Language Design and Implementation (PLDI'97)*, pp. 85–96, 1997.

[Amo04a] Amor, M., Sanjurjo, J.R., Padrón, E.J., and Doallo, R. Progressive Radiosity Method on Clusters Using a New Clipping Algorithm. *Int. J. of High Performance Computing and Networking (IJHPCN)*, Vol. 1, Nos. 1/2/3, pp. 55–63, 2004.

[Arn96a] Arnaldi, B., Priol, T., Renambot, L., and Pueyo, X. Visibility Masks for Solving Complex Radiosity Computations on Multiprocessors. In *Proc. of the First Eurographics Workshop on Parallel Graphics and Visualization*, pp. 219–232, 1996.

[Cho06a] Choi, M.-H., Park, W.-C., Neelamkavil, F., Han, T.-D., and Kim, S.-D. An Effective Visibility Culling Method Based on Cache Block. *IEEE Transactions on Computer*, Vol. 55, No. 8, pp. 1024–1031, 2006.

[Coh88a] Cohen, M., Chen, S.E., Wallace, J.R., and Greenberg, D.P.. A Progressive Refinement Approach to Fast Radiosity Image Generation. *ACM Computer Graphics (Proc. of SIGGRAPH '88)*, Vol. 22, No. 4, pp. 31–40, 1988.

[Coh93a] Cohen, M. and Wallace, J.R. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, 1993.

[Gar00a] Garmann, R. Spatial Partitioning for Parallel Hierarchical Radiosity on Distributed Memory Architectures. In *Proc. of the Third Eurographics Workshop on Parallel Graphics and Visualization*, Girona, Spain, September 2000.

[Gho99a] Ghosh, S., Martonosi, M., and Malik, S. Cache Miss Equation: A Compiler Framework for Analyzing and Tuning Memory Behavior. *ACM Trans. Programming Languages and Systems*, Vol. 21, No. 4, pp. 703–746, 1999.

[Gib00a] Gibson, S., and Hubbold, R.J. A Perceptually-Driven Parallel Algorithm for Efficient Radiosity Simulation. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 6, No. 3, pp. 220–235, 2000.

[Gue03a] Guerra, A.P., Amor, M., Padrón, E.J., and Doallo, R. A High–Performance Progressive Radiosity Method Based on Scene Partitioning. *Lecture Notes in Computer Science*, Vol. 2565, pp. 537–548, 2003.

[Hai94a] Haines, E.A., and Wallace, J.R. Shaft Culling for Efficient Ray-Traced Radiosity. In *Proc. of Photorealistic Rendering in Computer Graphics (Second Eurographics Workshop on Rendering)*, pp. 122–138, 1994.

[Han91a] Hanrahan, P., Saltzman, D., and Aupperle, L. A Rapid Hierarchical Radiosity Algorithm. In *Computer Graphics (SIGGRAPH'91 Proc.)*, pp. 197–206, 1991.

[Kaj86a] Kajiya, J.T. The Rendering Equation. *Computer Graphics*, Vol. 20, pp. 143–150, 1986.

[Moo01a] Moore, S., and Smeds, N. Performance Tuning Using Hardware Counter Data. In *SuperComputing 2001*, 2001.

[Pad01a] Padrón, E., Amor, M., Touriño, J., and Doallo, R. Hierarchical Radiosity on Multicomputers: a Load–Balanced Approach. In *Proc. of the Tenth SIAM Conference on Parallel Processing for Scientific Computing*, 2001.

[Pad03a] Padrón, E.J., Troncoso, R., Amor, M., Sanjurjo, J.R., and Doallo, R. Estudio Comparativo de Algoritmos de Visibilidad en los Modelos de Iluminación Global. In *XII Congreso Español de Informática Gráfica (CEIG 2003)*, pp. 141–154, 2003.

[San05a] J.R. Sanjurjo, J.R., Amor, M., Bóo, M., and Doallo, R. Parallel Global Illumination Method Based on a Non-Uniform Partitioning of the Scene. In *Proc. of the 13th Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP'05)*, pp. 251–257, 2005.

[Sil94a] Sillion, F.X., and Puech, C. *Radiosity. Global Illumination.* Morgan Kaufman Publishers, Inc., 1994.

Robust Diffuse Final Gathering on the GPU

Tamás Umenhoffer and László Szirmay-Kalos

Dept. of Control Engineering and Information Technology
Budapest University of Technology and Economics
Magyar Tudósok krt. 2., H-1117, Hungary
szirmay@iit.bme.hu

ABSTRACT

This paper presents a method to obtain the approximate indirect diffuse reflection on a dynamic object, caused by a diffuse or a moderately glossy environment. Instead of tracing rays to find the incoming illumination, we look up the indirect illumination from a cube map rendered from the reference point that is in the vicinity of the object. However, to cope with the difference between the incoming illumination of the reference point and of the shaded points, we apply a correction that uses geometric information also stored in cube map texels. This geometric information is the distance between the reference point and the surface visible from a cube map texel. The method computes indirect illumination albeit approximately, but providing very pleasing visual quality. The method fits very well the GPU architecture, and can render these effects interactively. The primary application area of the proposed method is the introduction of diffuse interreflections in games.

Keywords: Global illumination, GPU programming.

1 INTRODUCTION

Final gathering, i.e. the computation of the reflection of the indirect illumination toward the eye, is one of the most time consuming steps of realistic rendering. According to the rendering equation, the irradiance of point $\vec{x}$ can be expressed by the following integral

$$I(\vec{x}) = \int_S L^{in}(\vec{x} \leftarrow \vec{y}) \cdot v(\vec{x},\vec{y}) \cdot \frac{\cos^+ \theta_{\vec{x}} \cdot \cos^+ \theta_{\vec{y}}}{|\vec{x}-\vec{y}|^2}\, dy,$$

where S is the set of surface points, $L^{in}(\vec{x} \leftarrow \vec{y})$ is the incoming radiance arriving at point $\vec{x}$ from point $\vec{y}$, v is the visibility indicator between two points, and $\theta_{\vec{x}}$ and $\theta_{\vec{y}}$ are the angles between the illumination direction and the surface normals at points at $\vec{x}$ and $\vec{y}$, respectively. When the angles get larger than 90 degrees, their cosine should be replaced by zero, which is indicated by superscript $+$.

The evaluation of this integral usually requires many sampling rays from each shaded point. Ray casting finds *illuminating points* $\vec{y}$ for *shaded point* $\vec{x}$ at different directions (Figure 1), and the radiance of these illumination points is inserted into a numerical quadrature approximating the rendering equation. In practice, number P of shaded points is over hundred thousands or millions, while number D of sample directions is about a hundred or a thousand to eliminate annoying sampling artifacts. On the other hand, in games and in real-time systems, rendering cannot take more than a few tens of milliseconds. This time does not allow tracing $P \cdot D$, i.e. a large number of rays.

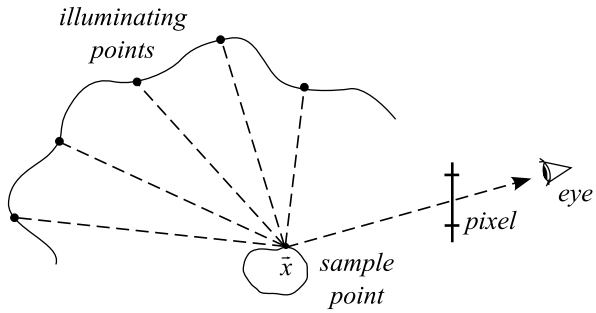


Figure 1: *Indirect illumination with sampling rays.*

To solve this complexity problem, we can exploit the fact that in games the dynamic objects are usually significantly smaller than their environment. Thus the global indirect illumination of the environment can be computed separately, since it is not really affected by the smaller dynamic objects. On the other hand, when the indirect illumination of dynamic objects is evaluated, their small size makes it possible to reuse illumination information obtained when shading its other points.

The first idea of this paper is to reuse illuminating points when the illumination on the dynamic object is computed. It means that first we obtain a set of virtual lights that represent the indirect illumination and calculate just the single reflection of these lights during final gathering (Figure 2). During this, we ignore complex self-shadowing of the dynamic object and apply just a simple test based on the normal vector and the illumination direction to determine whether the virtual light may illuminate the point.

This approach has two advantages. On the one hand, instead of tracing $P \cdot D$ rays, we solve the rendering problem by tracing only D rays. Assuming that the same set of illuminating points are visible from 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.

Plzen, Czech Republic.
Copyright UNION Agency – Science Press

 ISBN 978-80-86943-98-5

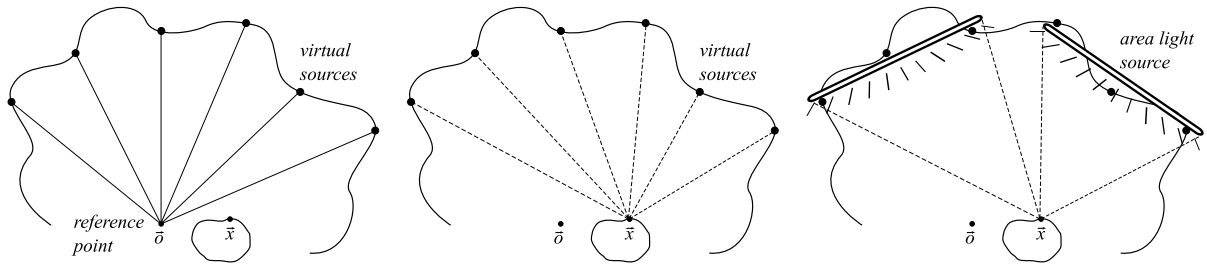


Figure 2: *The basic idea of the proposed method: first virtual lights sampled from reference point $\vec{o}$ are identified, then these point lights are grouped into large area lights. At shaded points $\vec{x}$ the illumination of a relatively small number of area lights is computed without visibility tests.*

shaded point, self-shadowing effects are ignored. However, while shadows are crucial for direct lighting, shadows from indirect lighting are not so visually important. Thus the user or the gamer finds this simplification acceptable.

Unfortunately, this simplification alone cannot allow real time frame rates. The evaluation of the reflected radiance at a shaded point still requires the evaluation of the irradiance and the orientation angle, and the multiplication with the diffuse reflectance for all directions. Although the number of rays traced to obtain indirect illumination is reduced from $P \cdot D$ to D, the illumination formula must be evaluated $P \cdot D$ times. These computations would still need too much time.

In order to further increase the rendering speed, we propose to carry out as much computation globally for virtual lights, as possible. Intuitively, global computation means that the sets of virtual light sources are replaced by larger homogeneous area light sources. Since the total area of these lights is assumed to be visible, the reflected radiance can be analytically evaluated once for a whole set of virtual light sources.

2 PREVIOUS WORK

Methods for reusing illumination information for multiple points of a shaded object can be classified according to whether they encode the irradiance of the receivers [17] or consider the environment as a set of virtual light sources [6].

Environment mapping [2] has been originally proposed to render ideal mirrors in local illumination frameworks, but can also be applied for glossy and diffuse reflections as well. The usual trick is the convolution of the angular variation of the BRDF with the environment map during preprocessing [11, 7] to obtain the irradiance of surfaces of all possible orientations. This step enables us to determine the illumination of an arbitrarily oriented surface patch with a single environment map lookup during rendering. A fundamental problem of this approach is that the generated environment map correctly represents the direction dependent illumination only at a single point, the reference point of the object. For other points, the environment map is only an approximation, where the error depends on the ratio of the distances between the point of interest and the reference point, and between the point of interest and the surfaces composing the environment (see figure 4).

One possible solution is to use multiple environment maps [4, 18, 10], which can be compressed using spherical harmonics [11, 8] or wavelets [18]. For example, Greger et al. [4] calculate and store the direction dependent illumination in the vertices of a bi-level grid subdividing the object scene. During run-time, irradiance values of an arbitrary point are calculated by tri-linearly interpolating the values obtained from the neighboring grid vertices. They reported good results even with surprisingly coarse subdivisions. This means that the smooth irradiance function is especially suitable for interpolation. While Greger et al. used a precomputed radiosity solution to initialize the data structures, Mantiuk et al.[9] calculated these values during run-time using an iterative algorithm that simulates the multiple bounces of light. Unfortunately, the generation and compression of many environment maps require considerable time which is not available during real-time rendering. Thus most of this computation should be done during preprocessing, which imposes restrictions on dynamic scenes.

The idea of approximating the indirect illumination by a finite set of virtual lights was born in the context of global illumination algorithms, such as in instant radiosity [6], and has been used many times in Monte Carlo algorithms [16, 12, 13, 14].

For the computation of diffuse interreflections on the GPU, Dachsbacher [3] considered shadow map lexels as virtual lights, while Lazányi [5] assigned virtual lights to texels of an environment map. Indeed, if only two-bounce indirect illumination is considered, a shadow map lexel identifies the point which is directly illuminated by the light source. Such points may indirectly illuminate other points, so can be considered as virtual lights.

Virtual light source algorithms suppose that virtual lights are point sources, and thus use the point-to-point

ISBN 978-80-86943-98-5

form factor (also called geometry factor) when the irradiance is computed. Unfortunately, point-to-point form factor

$$\frac{\cos\theta_{\vec{x}} \cdot \cos\theta_{\vec{y}}}{|\vec{x}-\vec{y}|^2}$$

is numerically unstable since it goes to infinity when the virtual light gets close to the shaded point, which results in bright spikes making the position of the virtual lights clearly visible (see figures 4 and 5). Another problem is that not only the position of the virtual light source is required, but the normal vector is also needed to compute $\cos\theta_{\vec{y}}$, which doubles the required storage space and slows down the generation of the virtual light sources.

To address the numerical instability, in [5] virtual lights were supposed to be small disks, which made the geometry factor bounded. However, this approximation is still quite far from being precise when the shaded point is close to the virtual light and the shaded point is not in the normal direction from the virtual light source.

In this paper we address the numerical instability of virtual lights and eliminate the need of storing normal vectors at these lights. The basic idea is that we always consider four virtual lights that are close to each other and assume that the surface is roughly planar between them. Thus instead of point sources we have a set of homogeneous area light sources. Since self-shadowing is ignored, the reflection of these area light sources can be computed analytically, in a numerically stable way.

3 THE NEW ALGORITHM

The first step of the algorithm is the generation of virtual lights that may illuminate the given dynamic object. To find these points, the scene is rendered from reference point $\vec{o}$ of the dynamic object, and the resulting images are put into an environment map. Not only the radiance of the visible points is evaluated but the distance between the reference point and the visible surface is found and stored in the alpha channel of the generated cube map. In each texel of the environment map a potential virtual light source is visible.

In order to estimate the integral of the rendering equation, the set of surface points visible from the reference point is partitioned according to the texels of the environment map. The surface between four points visible from the texel corners is approximated by a quadrilateral, thus the environment is assumed to be a list of quadrilaterals $S_i, i = 1,\ldots,N$. After partitioning, the irradiance is expressed by the following sum:

$$I(\vec{x}) = \sum_{i=1}^{N} \int_{S_i} L(\vec{y}) \cdot \frac{\cos^+\theta_{\vec{x}} \cdot \cos^+\theta_{\vec{y}}}{|\vec{x}-\vec{y}|^2}\, dy.$$

Since $dy \cos^+\theta_{\vec{y}}/|\vec{x}-\vec{y}|^2$ is the projection of area dy onto the surface of a unit hemisphere placed around $\vec{x}$, and factor $\cos^+\theta_{\vec{x}}$ represents a further projection from the sphere onto the tangent plane, the irradiance integral can also be evaluated on the tangent plane of the surface at $\vec{x}$:

$$I(\vec{x}) = \sum_{i=1}^{N} \int_{P_i} L(\vec{y}_p)\, dy_p$$

where P_i is the double projection of the visible surface, first onto the hemisphere then onto the tangent plane. Let us assume that $L(\vec{y}_p)$ is linear. In this case the irradiance is the product of the area of P_i and the average of the radiances stored at the four corner texels:

$$I(\vec{x}) = |P_i| \cdot \frac{L_i^1 + L_i^2 + L_i^3 + L_i^4}{4}$$

Let us consider a single term of this sum representing the radiance reflected from S_i.

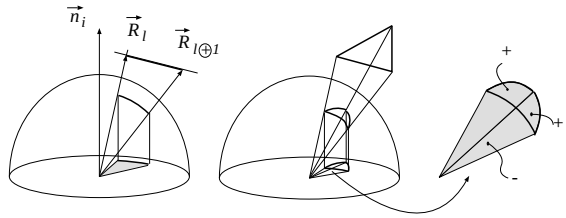


Figure 3: *Hemispherical projection of a planar polygon*

Area $|P_i|$ is in fact the polygon-to-point form factor [1]. Consider only one edge line of the polygon first, and two subsequent vertices, $\vec{R}_l$ and $\vec{R}_{l\oplus 1}$, on it (figure 3). The hemispherical projection of this line is a half great circle. Since the radius of this great circle is 1, the area of the sector formed by the projections of $\vec{R}_l$ and $\vec{R}_{l\oplus 1}$ and the center of the hemisphere is simply half the angle of $\vec{R}_l$ and $\vec{R}_{l\oplus 1}$. Projecting this sector orthographically onto the equatorial plane, an ellipse sector is generated, having the area of the great circle sector multiplied by the cosine of the angle of the surface normal $\vec{n}_i$ and the normal of the segment $(\vec{R}_l \times \vec{R}_{l\oplus 1})$.

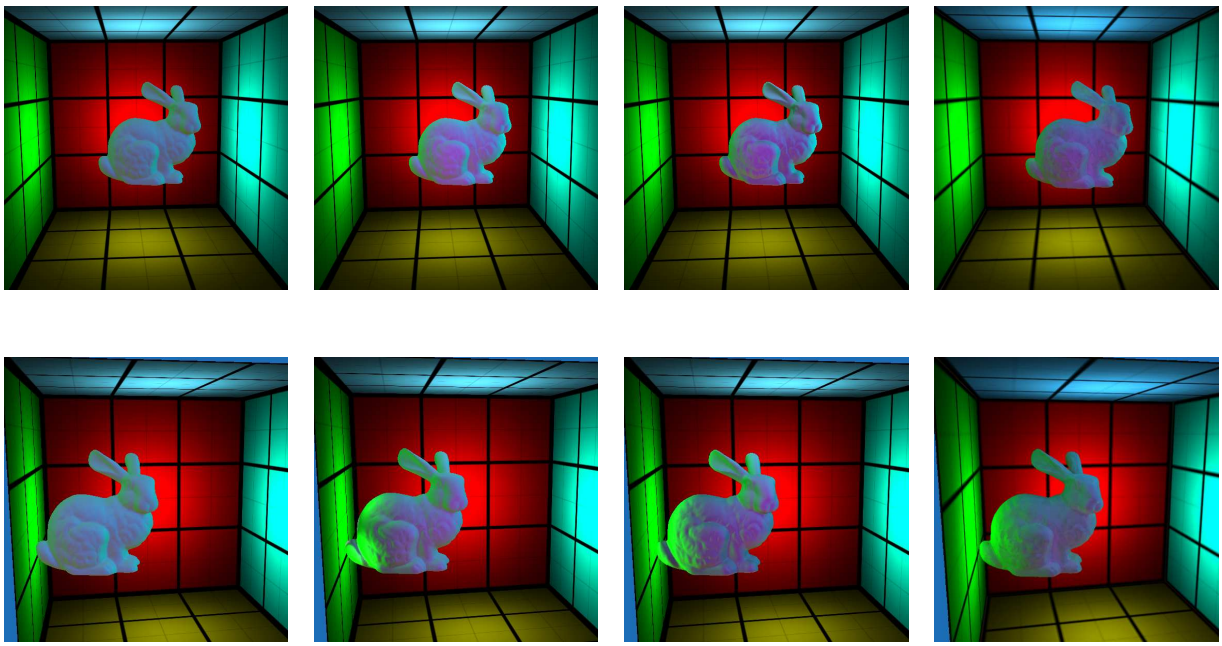
The area of the doubly projected polygon can be obtained by adding and subtracting the areas of the ellipse sectors of the different edges, as is demonstrated in figure 3, depending on whether the projections of vectors $\vec{R}_l$ and $\vec{R}_{l\oplus 1}$ follow each other clockwise when looking at them from the direction of the surface normal. This sign value provided by the dot product of the cross product of the two vertex vectors and the normal vector. Finally, the double projected polygon is a summation:

$$\sum_{l=0}^{L-1} \frac{1}{2} \cdot \mathrm{angle}(\vec{R}_l, \vec{R}_{l\oplus 1}) \cdot \left(\frac{(\vec{R}_l \times \vec{R}_{l\oplus 1})}{|\vec{R}_l \times \vec{R}_{l\oplus 1}|} \cdot \vec{n}_i \right). \qquad (1)$$

4 IMPLEMENTATION

The proposed algorithm first computes an environment cube map from the reference point and stores the radi-

ISBN 978-80-86943-98-5



600 FPS 60 FPS 11 FPS 30 sec

Figure 4: *The Stanford bunny model (about 25000 vertices and 50000 faces) in a colored box. On the right images diffuse reflections are calculated with only environment map convolution and one environment lookup. On the second column images point to point form factor while on the third column images polygon to point form factor is used. On the right column reference images rendered with Mental Ray are shown. The simple one environment map only produces pleasable results for points that are close to the environment reference point. The point to point form factor method can handle greater distances, but as the shaded surface gets close to the walls it gets unstable (see the bright green spikes on the bunny's tail and back). The polygon to point form factor has good results for any object positions. Configuration: 700x700 resolution on NVIDIA GeForce 7950 GX2 (SLI turned off), AMD Athlon64 Dual 4600+ processor.*

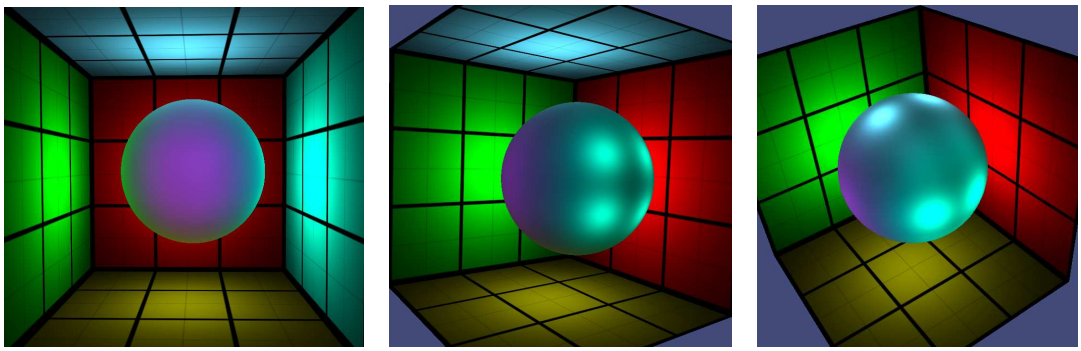
ance and distance values of the points visible in its pixels. We usually generate $6 \times 256 \times 256$ pixel resolution cube maps. Then the cube map is downsampled to have `M`$\times$`M` pixel resolution faces (`M` is 4 or even 2). One texel of the low-resolution cubemap represents an area light source. Note that both radiance and distance values are averaged, thus finally we have larger lights having the average radiance of the small lights and placed at their average position.

During final gathering for each texel in each cube face we should calculate the virtual light source's exact location and area and evaluate it's contribution to the reflected radiance. The following fragment shader code loops through the pixels of one cube face (in positive z direction), calculates pixel corner directions and calls the `getContr` function to evaluate texel contribution. The input parameters to this shader are: world space position (`pos`), world space normal (`N`), the reference point of the cubemap in world space (`referencePoint`), the diffuse coefficient (`kd`) and the cubemap sampler (`SmallEnvmap`). Our downsampled environment map had 4x4 pixel resolution(`M`).

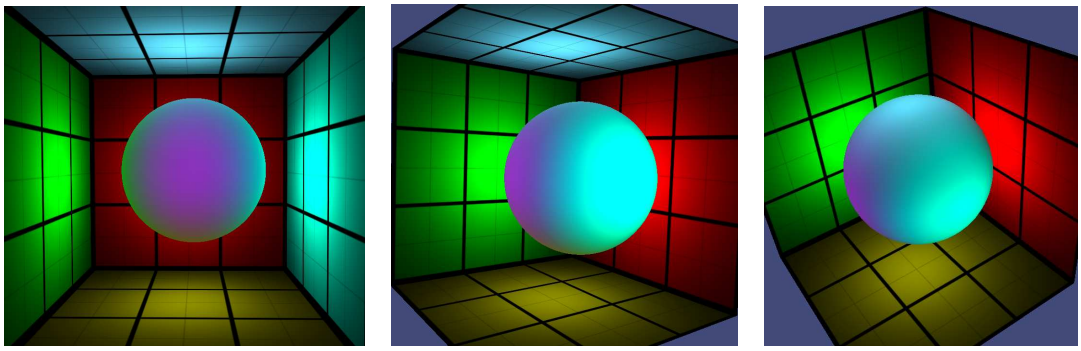
```
float4 DiffuseReflectance(
            float4 hPosition    : POSITION,
            float3 pos          : TEXCOORD0,
            float3 N            : TEXCOORD1,
            uniform samplerCUBE SmallEnvMap,
            uniform float3 referencePoint,
            uniform float kd
                          )  :COLOR
{
 pos -= referencePoint;
 N = normalize(N);
  //reflected radiance
 float4 I = 0;
  //texel corner directions
 float3 L1, L2, L3, L4;
 float3 L; //texel center direction
 float4 Le; //texel color
 float width = 1.0 / M; //width of one texel
 float dWidth = 2.0 * width; //double width

 for (int x = 0; x < M; x++)
  for (int y = 0; y < M; y++)
  {
    //calculate texel coordinates
   float2 p;
   p.x = x * width;
   p.y = y * width;
   p = 2.0 * p - 1.0; //to range [-1,1]
```

 ISBN 978-80-86943-98-5



150 FPS without SLI, 185 FPS with SLI turned on



32 FPS without SLI, 63 FPS with SLI turned on

Figure 5: *Sphere (about 2300 vertices and 2300 faces) in a colored box. Upper row with point to point form factor, bottom row with our polygon to point form factor. As the object gets closer to the walls the virtual light sources become visible in case of point to point form factor, while the polygon to point form factor method does not have this artifact. Configuration: 700x700 resolution on NVIDIA GeForce 7950 GX2 with SLI support, AMD Athlon64 Dual 4600+ processor.*

```
    //calculate texel corner directions
   L1 = float3(p.x, p.y, 1);
   L2 = float3(p.x + dWidth, p.y, 1);
   L3 = float3(p.x + dWidth, p.y + dWidth, 1);
   L4 = float3(p.x, p.y + dWidth, 1);
    //calculate texel center direction
   L = float3(p.x + width, p.y + width, 1);

    //read texel color
   Le = float4(texCUBE(SmallEnvMap, L).rgb,1);
    //get contribution from texel
   I += 0.5 * Le * getContr(L1, L2, L3, L4,
                            pos, N,
                            SmallEnvMap);
  }
  return kd * I;
}
```

The following code shows the `getContr` function which calculates the contribution of a single texel of the downsampled, low resolution cubemap `SmallEnvMap` to the illumination of the shaded point. It's input arguments are: the four pixel corner directions (`L1, L2, L3, L4`), the shaded point position and normal (`pos, N`) and the cubemap sampler (`SmallEnvMap`). This function calculates the exact position of pixel corners with the help of the stored distance values in the cubemap's alpha channel. Then evaluates the polygon-to-point form factor described above: calculates the four triangle areas given by the four edges of the texel and summs their signed values. The contribution from light sources located behind the tangent plane can be eliminated by ignoring the sums with negative values.

```
float4 getContr(float3 L1, float3 L2,
                float3 L3, float3 L4,
                float3 pos, float3 N,
                samplerCUBE SmallEnvMap)
{
  //texel corner distances and positions
  //from reference point
 float d;
 d = texCUBE(SmallEnvMap, L1).a;
 L1 = d * normalize(L1);
 d = texCUBE(SmallEnvMap, L2).a;
 L2 = d * normalize(L2);
 d = texCUBE(SmallEnvMap, L3).a;
 L3 = d * normalize(L3);
 d = texCUBE(SmallEnvMap, L4).a;
 L4 = d * normalize(L4);
```


ISBN 978-80-86943-98-5

```
 // corner directions from shaded point
 float3 r1 = normalize(L1 - pos);
 float3 r2 = normalize(L2 - pos);
 float3 r3 = normalize(L3 - pos);
 float3 r4 = normalize(L4 - pos);

 //calculate projected triangle areas
 float3 crossP = cross(r1, r2);
 float r = length(crossP);
 float dd = dot(r1, r2);
 float tri1 = acos(dd) * dot(crossP / r, N);

 crossP = cross(r2, r3);
 r = length(crossP);
 dd = dot(r1,r2);
 float tri2 = acos(dd) * dot(crossP / r, N);

 crossP = cross(r3, r4);
 r = length(crossP);
 dd = dot(r1,r2);
 float tri3 = acos(dd) * dot(crossP / r, N);

 crossP = cross(r4, r1);
 r = length(crossP);
 dd = dot(r1,r2);
 float tri4= acos(dd) * dot(crossP / r, N);

 //summation of triangle areas
 return max(tri1 + tri2 + tri3 + tri4, 0);
}
```

5 RESULTS

In order to demonstrate the results, we took a simple environment consisting of a colored cubic room. The first set of pictures shows the Stanford bunny model inside the room (figure 4). The images of the first column were rendered by the traditional environment mapping technique for diffuse materials where a precalculated convolution enables us to determine the irradiance at the reference point with a single lookup. This method has correct results only at the reference point and cannot deal with the position of the object, thus the bunny looks similar everywhere. The second column shows the point to point form factor. Although this method have pleasing results for points other than the reference point, it has artifacts for points too close to the virtual light sources. The right column shows our polygon-to-point form factor method which has correct results for arbitrary point positions. The difference between point to point and polygon-to-point factors can clearly be seen in the second set of pictures (figure 5). The first row shows point to point while the second row shows polygon-to-point form factor results. While the two methods show similar results when the object is in the center of the room there is a significant improvement in case of the polygon-to-point factor method when the sphere gets close to the walls.

The third set of pictures (figure 6) shows the Stanford Buddha model in the room with point to point (left) and polygon-to-point form factors (right). The images on the left show visual errors at surface points near the environment (see the keystone of the statue).

In figure 7 the Buddha model is placed in a dimly lit hallway. This environment was implemented in an actual game engine called OGRE. The scene has per pixel shading and reflections too.

In figure 8 the Buddha was placed in an other complex environment: a scientific laboratory. These pictures were also taken from an OGRE implementation.

6 CONCLUSIONS

This paper presented a GPU based method for computing diffuse reflections of the incoming radiance stored in environment maps. The environment map is considered as a definition of large area light sources whose reflections are obtained analytically without checking self-shadowing. The presented method runs in real-time and provides visually pleasing results.

ACKNOWLEDGEMENTS

This work has been supported by OTKA (T042735), GameTools FP6 (IST-2-004363) project, and by the National Office for Research and Technology (Hungary).

REFERENCES

[1] D.R. Baum, H.E. Rushmeier, and J.M. Winget. Improving radiosity solutions through the use of analytically determined form-factors. *CComputer Graphics (SIGGRAPH '89 Proceedings)*, pages 325–334, 1989.

[2] J. F. Blinn and M. E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547, 1976.

[3] Carsten Dachsbacher and Marc Stamminger. Reflective shadow maps. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 203–231, New York, NY, USA, 2005. ACM Press.

[4] Gene Greger, Peter Shirley, Philip M. Hubbard, and Donald P. Greenberg. The irradiance volume. *IEEE Computer Graphics and Applications*, 18(2):32–43, March/April 1998.

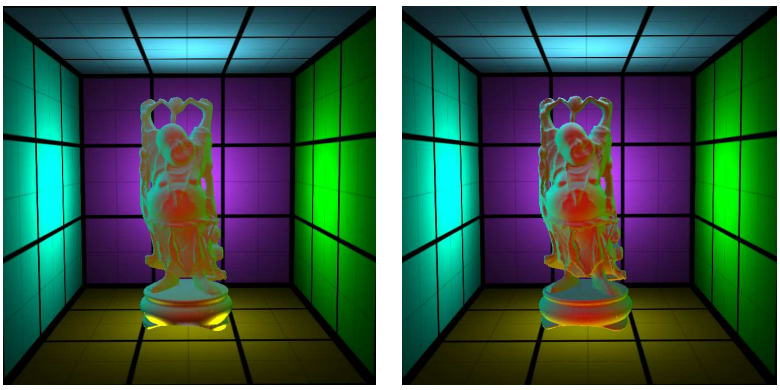
[5] László Szirmay Kalos and István Lazányi. Indirect diffuse and glossy illumination on the gpu. In *SCCG 2006*, pages 29–35, 2006.

[6] A. Keller. Instant radiosity. In *SIGGRAPH '97 Proceedings*, pages 49–55, 1997.

[7] Gary King. Real-time computation of dynamic irradiance environment maps. In Parr M., editor, *GPU Gems II*, pages 167–170. Addison-Wesley, 2005.

[8] A.W. Kristensen, T. Akenine-Moller, and H.W. Jensen. Precomputed local radiance transfer for real-time lighting design. In *SIGGRAPH 2005*, 2005.

 ISBN 978-80-86943-98-5



40 FPS without SLI
85 FPS with SLI turned on

7 FPS without SLI
19 FPS with SLI turned on

Figure 6: *Diffuse Stanford Buddha (about 35000 vertices and 67000 faces) in a box. Left image with point to point form factor, right image with our polygon-to-point form factor method. Configuration: 700x700 resolution on NVIDIA GeForce 7950 GX2 with SLI support, AMD Athlon64 Dual 4600+ processor.*

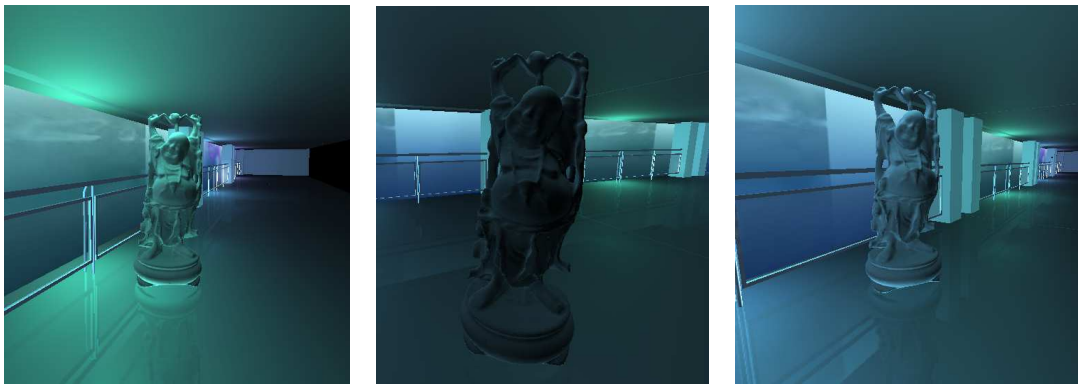


Figure 7: *Screen shots from Buddha in the hallway (the whole scene with the Buddha is about 38000 vertices and 70000 faces). OGRE implementation. 12 FPS, 800x600 resolution on NVIDIA GeForce 7800GT, AMD Athlon 64 3500+ processor.*

[9] R. Mantiuk, S. Pattanaik, and K. Myszkowski. Cube-map data structure for interactive global illumination computation in dynamic diffuse environments. In *International Conference on Computer Vision and Graphics*, pages 530–538, 2002.

[10] Mangesh Nijasure, Sumanta Pattanaik, and Vineet Goel. Real-time global illumination on the GPU. *Journal of Graphics Tools*, 2004. To appear.

[11] R. Ramamoorthi and P. Hanrahan. An efficient representation for irrandiance environment maps. *SIGGRAPH 2001*, pages 497–500, 2001.

[12] P. Shirley, C. Wang, and K. Zimmerman. Monte Carlo techniques for direct lighting calculations. *ACM Transactions on Graphics*, 15(1):1–36, 1996.

[13] I. Wald, C. Benthin, and P. Slussalek. Interactive global illumination in complex and highly occluded environments. In *14th Eurographics Symposium on Rendering*, pages 74–81, 2003.

[14] B. Walter, S. Fernandez, A. Arbree, K. Bala, M. Donikian, and D. P. Greenberg. Lightcuts: A scalable approach to illumination. In *SIGGRAPH 2005*, 2005.

[15] G. Ward. Adaptive shadow testing for ray tracing. In *Rendering Workshop '94*, pages 11–20, 1994.

[16] G. J. Ward. The RADIANCE lighting simulation and rendering system. *Computer Graphics*, 28(4):459–472, 1994.

[17] K. Zhou, Y. Hu, S. Lin, B. Guo, and H.-Y. Shum. Precomputed shadow fields for dynamic scenes. In *SIGGRAPH 2005*, 2005.

ISBN 978-80-86943-98-5

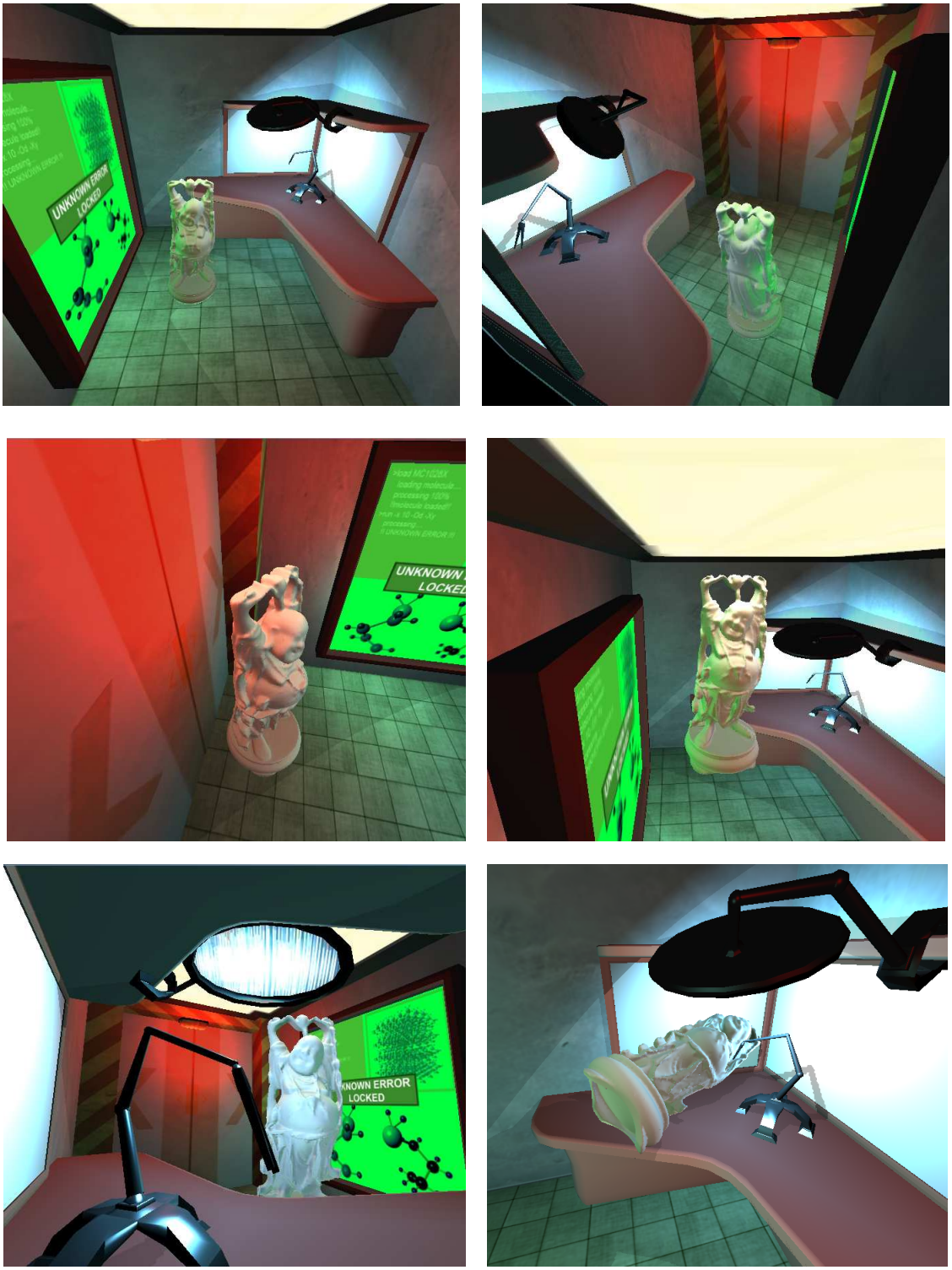


Figure 8: *Screen shots from Buddha in a laboratory (the whole scene with the Buddha is about 38000 vertices and 70000 faces). OGRE implementation. 15 FPS, 800x600 resolution on NVIDIA GeForce 7800GT, AMD Athlon 64 3500+ processor.*

 ISBN 978-80-86943-98-5

Perception Motivated Hybrid Approach to Tone Mapping

Martin Čadík
Department of Computer Science and Engineering, Czech Technical University in Prague
Karlovo náměstí 13, 121 35 Prague, Czech Republic
cadikm@fel.cvut.cz

ABSTRACT

We propose a general hybrid approach to the issue of reproduction of high dynamic range images on devices with limited dynamic range. Our approach is based on combination of arbitrary global and local tone mapping operators. Recent perceptual studies concerning the reproduction of HDR images have shown high importance of preservation of overall image attributes. Motivated by these findings, we apply the global method first to reproduce overall image attributes correctly. At the same time, an enhancement map is constructed to guide a local operator to the critical areas that deserve enhancement. Based on the choice of involved methods and on the manner of construction of an enhancement map, we show that our approach is general and can be easily tailored to miscellaneous goals of tone mapping. An implementation of proposed hybrid tone mapping produces good results, it is easy to implement, fast to compute and it is comfortably scalable, if desired. These qualities nominate our approach for utilization in time-critical HDR applications like interactive visualizations, modern computer games, HDR image viewers, HDR mobile devices applications, etc.

Keywords: Tone mapping, HDRI, dynamic range reduction.

1 INTRODUCTION

Merits of high dynamic range imaging (HDRI) are currently widely recognized in computer graphics, high-quality photography, computer vision, etc. However, HDRI becomes popular in interactive and real-time applications as well. Data visualization, computer games and other interactive applications gain new qualities thanks to HDRI. The reproduction of high dynamic (HDR) data on the low dynamic (LDR) output devices requires the reduction of dynamic range, commonly referred to as a tone mapping.

Many so-called tone mapping operators were proposed in history [Dev02, Rei05]. We can classify the existing approaches according to the transformation they apply to convert input luminances to the output values. *Global* tone mapping methods apply a tone reproduction curve - e.g. a function. Therefore, they transform particular value of the input luminance to one specific output value. *Local* tone mapping operators may on the other hand reproduce particular input luminance to different output values depending on the surrounding pixels.

Although many sophisticated local tone mapping operators were published, these are typically not very generic approaches and just a few of these methods is suitable for interactive and computationally weak applications. Even worse, hardly any of them can be marked as general and scalable. Besides, recent perceptual studies concerning the reproduction of HDR images [Yos05, Led05, Cad06, Cad07] have shown high importance of preservation of global image attributes (overall brightness, overall contrast).

Generally speaking, global methods reproduce overall image attributes well, they are fast to compute, and easy to implement, but may wash away important details. Local approaches excel in reproduction of local contrast (details), but they are computationally intensive and may reproduce overall image attributes poorly, see Figure 1. Motivated by the mentioned findings we present a fast and simple yet powerful general hybrid approach to tone mapping issue. This approach takes advantages of both global and local tone mapping approaches to overcome mentioned limitations. Moreover, since the aims of tone mapping can differ among particular applications, the proposed approach can be easily tailored to the miscellaneous goals.

The paper is organized as follows. In Section 2, we overview the previous work and we focus particularly on linear tone mapping methods. In Section 3, we introduce and describe generally the new hybrid tone mapping idea. In Section 4, we present two exemplary implementations of hybrid tone mapping approach and show the results. Finally, in Section 5, we conclude and give suggestions for future work.

2 RELATED WORK

The areas of high dynamic range imaging and the tone mapping are currently quite complex. Refer to the state of the art by Devlin [Dev02] or to the book by Reinhard et al. [Rei05] for an overview. Since we concentrate mainly on interactive applications and computationally

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.

Figure 1: Global methods reproduce overall image attributes well, but wash away important details (left). Local approaches excel in reproduction of details, but may reproduce overall image attributes poorly (right).

efficient methods, we survey briefly the time-dependent methods suitable for interactive applications.

2.1 Time-dependent Tone Mapping

Adaptation mechanisms of human visual system show time-dependency. Human needs some time to adapt to low luminance levels after entering the dark cinema from the sunlit street, for example. However, involvement of time-dependency is profitable also for non-perceptual applications of tone mapping, since it avoids flickering and other time-dependent artifacts.

Probably the first time-dependent tone mapping method was presented by Pattanaik et al. [Pat00]. The method is based on statical color model by Hunt that is extended of time adaptation. Authors use global s-shaped curve for mimicking the response of both direct and inverse models of visual perception.

Durand and Dorsey [Dur00] used the global tone mapping operator by Ferwerda et al. [Fer96] for interactive tone mapping. Authors model visual adaptation course over the time for rods and cones.

Fairchild and Johnson [Fai03] adapted the iCAM model [Fai02] to account for time-dependent adaptation effects. The time-dependency provide two exponential filters that modify adaptation level. The used model is local (filtering using wide gaussian curve) and therefore computationally intensive. Since the filtering kernel is very large, the properties of the iCAM outputs resemble global tone mapping results.

Ledda [Led04] proposed strictly local time-dependent approach. The method is based on initial effort by Pattanaik, but adds local processing using bilateral filter. Time-dependency is modeled using exponential filters for rods and cones.

The above surveyed time-dependent approaches apply either global curves and thus they destroy subtle details or they apply local methods and thus they are computationally demanding. Moreover, interactive applications often need to do some sort of load balancing, however there is an unanswered question, how to scale the time-dependent methods properly.

2.2 Linear Tone Mapping

As we have noticed, global methods reproduce overall image attributes well, see Figure 1. The group of global methods comprise a subset of linear tone mapping curves. Despite of the simplicity of *linear* tone mapping curves, the approaches utilizing linear (or close-to linear) mapping have many advantages that deserve our attention.

Since the linear methods scale image intensities by a constant (scale factor), they do not change scene contrasts for display. This is probably the reason why these methods show [Cad06, Cad07] to perform well in perceptual reproduction of the overall image attributes.

Linear tone mapping methods transform the input HDR image to the output image values using the scale factor, $L_d = m \cdot L_w$, where m is the *scale factor*, L_w is the input luminance, and L_d is the output value in the interval of [0, 1].

The simplest linear approach is the *maximum luminance* mapping, where we map the maximal input luminance to the maximal output value (e.g. to 1): $m = \frac{1}{L_{wmax}}$, where L_{wmax} is the maximal input luminance. Since the maximal luminance is usually enormous in case of HDR images, this approach produces typically too dark and valueless results. *Mean value* mapping approach gives more reasonable outputs by mapping the *average* input luminance to the average output scale: $m = 0.5 \cdot \frac{1}{L_{avg}}$, where L_{avg} is the average input luminance.

Ward's contrast based scale factor [War94] focuses on the preservation of *perceived contrast*. The computation of the scaling factor is based on Blackwell's [CIE81] psychophysical contrast sensitivity model. Almost the same principle of contrast preservation is exploited also in the work of Ferwerda et al. [Fer96].

Another linear approach was introduced by Neumann et al. [Neu98]. They propose the minimum *information loss* method that tries to mimic the photographer's practice to lose a minimum amount of information. The method automatically selects ideal clipping interval to obtain a minimum of detail-lost areas. The automatic selection of the interval is done by means of logarithmic image histogram.

Mapping using *s-shaped curve* [Pat00] is formally not a linear approach, but practically it can produce results that are very close to the linearly mapped results. S-shaped curves resemble transfer curves of classical photographic media.

 ISBN 978-80-86943-98-5

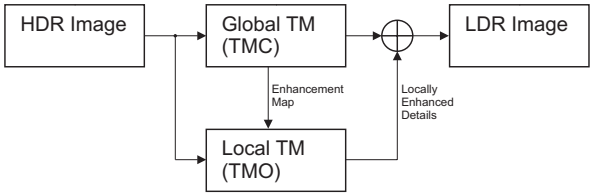


Figure 2: Overview of the hybrid tone mapping approach.

3 HYBRID TONE MAPPING

We propose a general idea of new hybrid tone mapping approach in this section, while two exemplary implementations are described in the next section. We do not concern the inherent physically-induced issues of real output devices (gamma correction, etc.), in this paper.[1] In the further text, we assume (without the loss of generality) that the output device range is linear and is limited to the interval of [0, 1]. Moreover, we do not concern the reproduction of colors here. All of our proposed methods aim to reduce dynamic range of the input image and the color information is left untouched. However, there is no obstacle to involve any method with a specific color processing into hybrid tone mapping approach if desired.

The overview of general hybrid tone mapping framework is shown in Figure 2. The process consists of three steps. First, the input high dynamic range image is transformed using global tone mapping curve. This mapping produces the base of the output low dynamic range image and (if necessary) outputs desired values for the construction of an *enhancement map*. In the second step, the enhancement map is constructed considering the aim of the tone mapping (see Figures 3, 6). Finally, the enhancement map is used to guide a local tone mapping method to reconstruct subtle details. The locally enhanced details are then added back to the globally transformed image to improve the final image while maintaining good overall reproduction of image attributes. Due to the enhancement map, the local tone mapping method is applied merely to the critical areas of the original image and therefore a lot of computational resources are saved.

3.1 Construction of Enhancement Map

Besides the choice of the involved methods, the manner of the construction of the enhancement map is an essential heart of the hybrid tone mapping. The enhancement map therefore has a profound effect on the properties of the hybrid tone mapping approach as a whole.

The enhancement map is generally a map of float numbers with the same dimensions as the original HDR image. In the examples shown in section 4, the enhancement map is constructed using a sort of thresholding of input luminance values. However, there is no obstacle to construct the enhancement map in a different way (for example based on input gradients, etc.). When we use a linear transform function in a global part of hybrid mapping, the construction of the enhancement map is very simple and effective. Since we know the analytical equation of the transfer curve, the clipping values are then directly known and the enhancement map is constructed effortlessly by thresholding. For the other approaches (s-shaped curves, log scaling, etc.), we propose to use the clip values of 5th and 95th luminance percentiles. However, the manner of the enhancement map construction can reflect a special goal of implemented method, as we show in section 4.3.

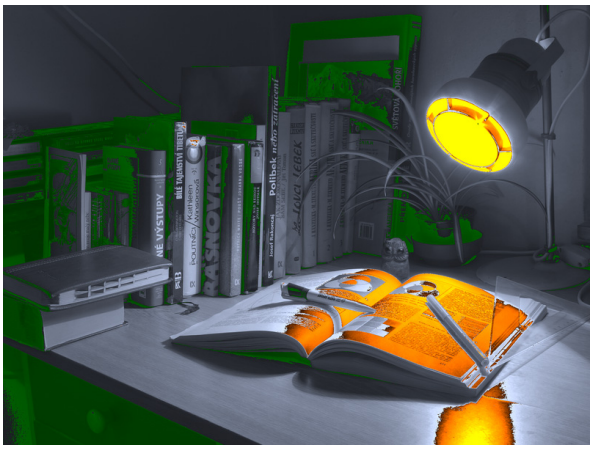


Figure 3: Example of an enhancement map (orange and green areas) for perceptually plausible hybrid approach. The luminances in orange-colored areas are potentially clipped due to linear mapping, while the details in dark parts, green-colored areas, are lost due to insufficient numerical precision.

Due to the abrupt changes of the global mapping to the local mapping (and vice versa), artifacts might appear on the borders of the enhancement map. To avoid the artifacts, we propose to construct and apply the enhancement map as follows (we show the process for only one clipping value L_{clip}):

$$EM(x,y) = \begin{cases} 0 & if\ TMC(L_w) < L_{clip} \\ \min(1, TMC(L_w) - L_{clip}) & otherwise, \end{cases}$$

where $EM(x,y)$ is the enhancement mask value, $TMC()$ is a global tone mapping method, L_w is the input luminance, and L_{clip} is the clipping luminance. The final output value L_d is then computed as a weighted sum of a global method and a local method outputs: $L_d(x,y) = EM(x,y) \cdot TMO(L_w) + (1 - EM(x,y)) \cdot TMC(L_w)$, where $TMO()$ is the involved local method.

If the computational time is not an issue, we can involve more sophisticated criteria to the construction of an enhancement map, e.g. the human visual system properties. Using the visual attention model, for example, we can pass the computational resources to the visually important areas of the image. Another possi-

[1] Nevertheless, all the tone mapping results presented in this paper have been gamma corrected finally, using the value of $\gamma = 2.2$ as usual.

 ISBN 978-80-86943-98-5

bility is the usage of contrast sensitivity function (CSF) during the construction of the enhancement map. In this case, the effort of local method will be directed to the areas, where the detail is (at least potentially) visible for a human observer.

4 USE CASES AND RESULTS

The proposed idea of hybrid tone mapping is general – virtually any combination of existing (and potentially forthcoming) methods is possible. However, the choice of the involved methods has to reflect the intent of the resultant combination (f.e. an aesthetic view, a cognitively rich depiction, or a perceptually plausible reproduction). A combination of methods that is excellent in reproducing details can fail miserably when we aim in reproducing the perceptual experience of an observer. In this section, we show two different examples of the hybrid approach: perceptually plausible approach and cognitively rich approach. We show and discuss actual outputs of these methods and we also exhibit the performance values comparing to original local methods.

4.1 Fast Perceptually Plausible Approach

For such an interactive applications where the perceptually convincing reproduction is desired and where the computational resources are limited (e.g. in computer games), we propose fast and simple implementation of hybrid tone mapping approach as follows.

At the post of global method, we use the linear mapping by Ward [War94]. This method was proven to give reasonable results for natural scenes [Cad06, Cad07], and the computational demands of the method are minimal. Since the global part is purely linear, we can construct the enhancement map directly by thresholding of input luminance values. The exact clipping values are known: $L_{clipLO} = 0$ and $L_{clipHI} = 1$ for an original method. We can modify the approach by shifting the transform curve if desired (to allow the user to adjust brightness or contrast), but even in this case, the clipping values are easily found analytically. Pixels with luminance values outside of the linear interval would be clipped and therefore the information would be lost there. Therefore, these pixels form the enhancement map.

Having the enhancement map, we run the bilateral filtering method [Dur02] just on the areas marked in the map. Bilateral filter separates the original luminance map to the base layer and the detail layer. We use the detail layer to enhance the result of the global tone mapping method. For acceleration of the local filtering, we utilize graphics hardware (GPU) [Fia06]. Figure 4 compares the transforms of the original methods and the hybrid approach and Figure 5 shows the results in the form of output images.

4.2 Time-Dependent Hybrid Mapping

It is usually advantageous to model the course of visual adaptation over the time for interactive applications. Time-dependency of tone mapping is twofold profitable: it increases the perceptual quality on one hand, and it also avoids temporal image artifacts on the other hand.

The way of implementation of time-dependency is influenced by the goal of the whole tone mapping method and it is not necessary to realize it at all, in some cases. In accordance with other authors [Dur00, Pat00] we use an exponential decay function in our perceptually plausible hybrid approach (described in Section 4.1) to model the light adaptation. We omit the simulation of long-term dark adaptation due to its subtle and slow effect and due to efficiency reasons. We modulate the adaptation level $L_{a(w)}$ in the Ward's method [War94] by the exponential function for smooth transitions when tone mapping a dynamic environment:

$\frac{dL_{a(w)}}{dt} \approx \frac{L_{a(w)} - L_{a(w)(t)}}{\tau}$,

where $L_{a(w)}$ is the visual adaptation for static image, $L_{a(w)}(t)$ is the actual adaptation and $\tau = 0.1$ is a time constant that mimics the speed of adaptation.

Similar approach is amenable in many other hybrid tone mapping implementations, since we can usually smoothen the response of particular parameter of involved global tone mapping method. If the computational cost is the main limitation, the time-dependency may be omitted temporarily with reasonable loss of reproduction quality.

4.3 Cognitive Approach

As an example of cognitive (e.g. detail-oriented) hybrid tone mapping approach, we propose the combination of histogram adjustment global tone mapping operator [War97] enhanced by locally applied bilateral filtering [Dur02].

The histogram adjustment method grants most of the available device contrast to the areas of abundant luminance values in the input HDR image. Generally speaking, large areas in an input image are given more contrast (thus subtle details present at these areas may become visible) at the expense of tiny areas. This advanced 'distribution of contrast' is achieved thanks to cumulative function derived from formerly constructed image histogram. The cumulative function is then used as a tone reproduction curve to transform input luminance to output values.

In accordance with the choice of involved methods, we propose to construct also the *enhancement map* seeking the same goal of cognitively rich (detailed) output image. We detect the areas of small local contrast on the chart of cumulative function constructed in the previous step – these areas represent pixels, where the detail is potentially vanished. We use the second derivative of cumulative function for this detection (note that

 ISBN 978-80-86943-98-5

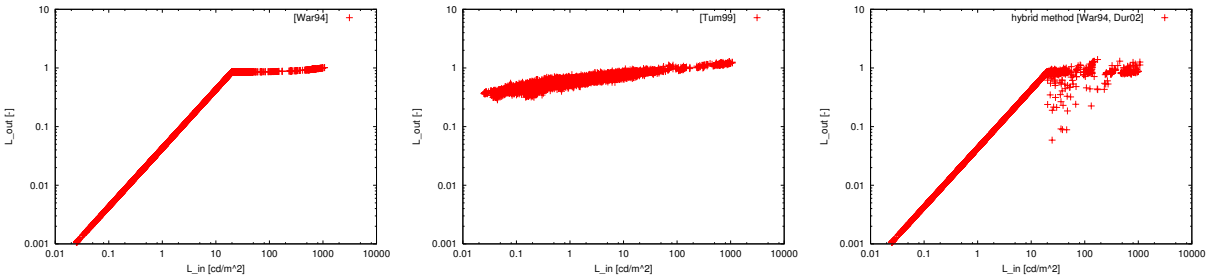


Figure 4: Tone mapping transforms, left: global method [War94] maps input HDR values via linear function – note the clipping of high luminances, middle: local method [Tum99] applies different transform to different pixels – the reproduction of overall image attributes is poor, right: hybrid approach [War94, Dur02] combines merits of both the global and local approaches.

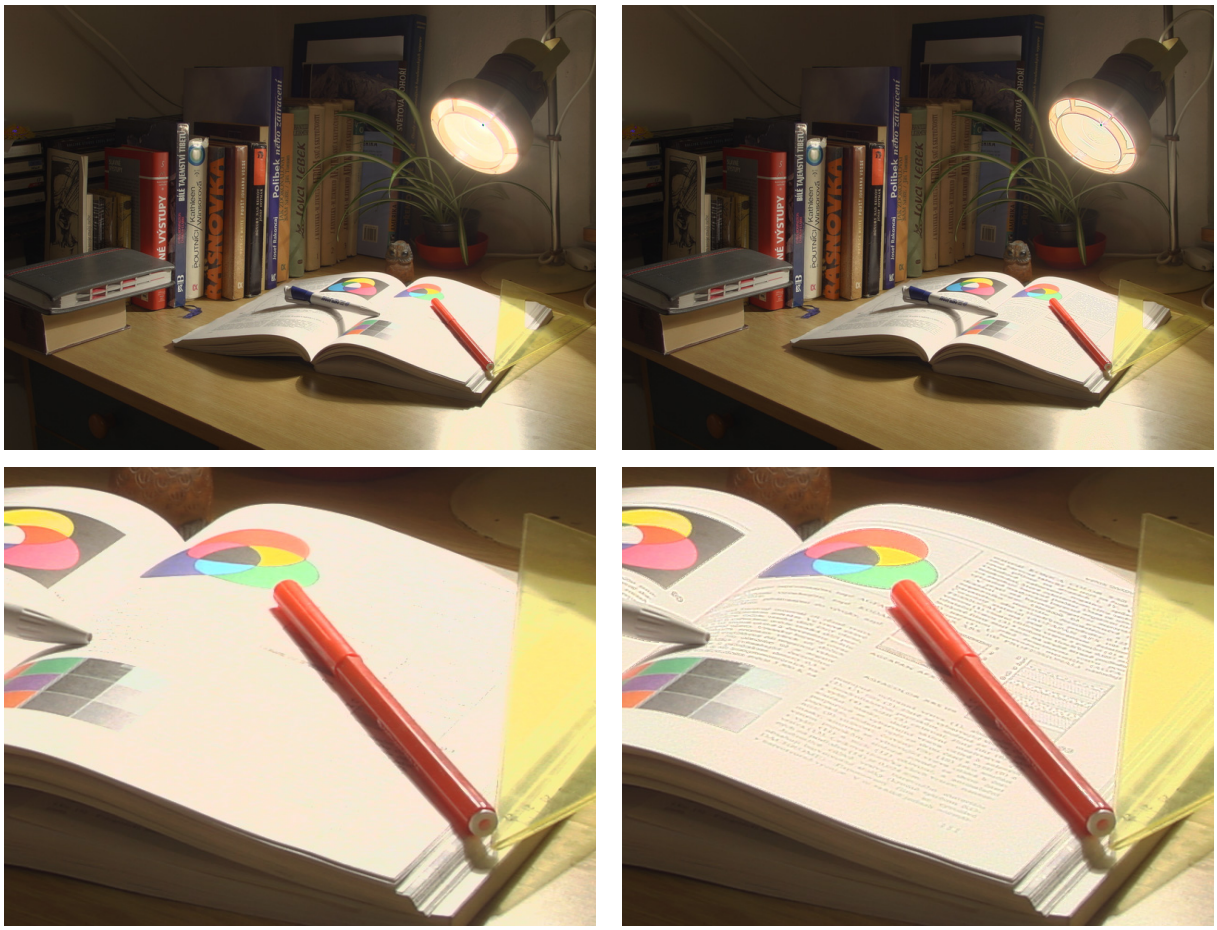


Figure 5: top left: result of purely global method [War94] exhibits well reproduced overall contrast, however shows the lack of subtle details, top right: result of the new hybrid approach [War94, Dur02] preserves the overall contrast accurately, and adds the lost details. Bottom: close-ups of the book, note the reproduced details in the hybrid approach result (bottom right).

 ISBN 978-80-86943-98-5

Perceptual method (Sec. 4.1)		Cognitive method (Sec. 4.3)	
Enhancement map [% of image pixels]	Speedup [-]	Enhancement map [% of image pixels]	Speedup [-]
1.4e-3%	**118,5**	0.132 %	**41,74**

Table 1: Comparison of performances of two different implementations of hybrid tone mapping (average results over 10 HDR images). The speedup value shows the acceleration of hybrid approach against the original, completely local approach.

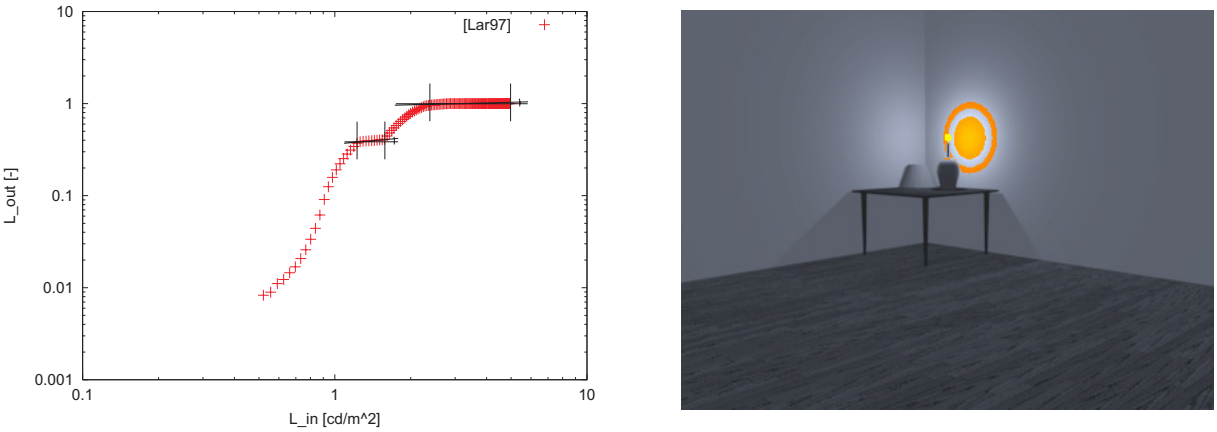


Figure 6: Cognitive hybrid approach. Left: cumulative function [War97] with detected areas for local enhancement. We use the threshold of 0.1 for detection, e.g. $L'_{out} < 0.1$. Right: corresponding enhancement map constructed from the cumulative function.

the the derivative is never a negative number in our case). Sections that show low slope (low values of the second derivation) are selected to construct the enhancement map (see Figure 6) – each pixel with the luminance value within the boundary of a selected section becomes a member of enhancement map.

Finally, we apply the local trilateral filtering method [Cho03] on the original input pixels that are present (or masked) in enhancement map. Similarly to the bilateral filter, the trilateral filter produces blurred image, but preserves significant luminance edges. We obtain local details by dividing the original image by the blurred image. Finally, we enrich the result of histogram adjustment method by these details. See Figure 7 (bottom line) where two renditions of one input image are shown to compare the two presented hybrid implementations.

4.4 Performance Results

Since one of the goals of the hybrid approach is the reduction of computational complexity, we present here the numerical performance results, see the Table 1[2]. The table shows average values measured at the group of 10 input HDR images. The perception-targeted fast hybrid approach is on average 3 times faster than the cognitive method and more that 118 times faster than the original bilateral filtering method. The reported speedup is gained thanks to the enhancement map. Since the enhancement map contains usually just a small portion of the original image pixels (as shown in the table), the time-demanding local approach is applied locally, to the small (necessary) part of the image.

Generally speaking, our technique places very small additional load to the system leaving large space for other computations. This is very profitable in interactive applications like the computer games, etc. However, note that besides the performance improvement, hybrid tone mapping can enhance the quality of the output image as well (see Figure 5).

In the imminent future, we can expect the need of dynamic range reduction even on various portable devices and on small and computationally elementary machines. The hybrid tone mapping will be reasonable in this case as well, thanks to its *good scalability*. If we face the lack of computational power, we can modify (soften) the criteria of the construction of the enhancement map. Depending on these criteria, we are able to continuously balance the computational load spanning from the complete locally enhanced method up to the factual omitting of the local enhancement (e.g. purely global tone reproduction). Finally, the other possibility to decrease the time consumption is to omit the time-dependent processing, as noted in Section 4.2.

[2] The performance values strongly depend on the selection and the implementation of the involved methods and we therefore present the average values for an overview.

 ISBN 978-80-86943-98-5

Figure 7: Further results of the new hybrid approach. Top left: pure global method [War94] without enhancement washes away details seen through the window. Top right: hybrid approach [War94, Dur02] enhances the image (note the revival of the birch twigs) without affecting the overall image attributes. Bottom left: hybrid approach [War94, Dur02] – perceptually plausible reproduction of well-known Memorial church image, bottom right: cognitive rendition of the same input image by hybrid combination of histogram adjustment and trilateral filtering [War97, Cho03].

5 CONCLUSIONS AND FUTURE WORK

We presented a novel hybrid approach to the issue of reproduction of high dynamic range images on devices with limited dynamic range of luminance (e.g. tone mapping issue). In our approach, we combine results of arbitrary global and local tone mapping operators. Recent perceptual studies concerning the reproduction of HDR images have shown high importance of preservation of overall image attributes. Motivated by these studies, we apply the global method first to reproduce overall image attributes correctly. At the same time, we construct an enhancement map to guide a local operator to the critical areas that deserve enhancement.

We do not invent another complex tone mapping method, but we rather propose a general framework that utilizes already known ideas and combines existing and potentially forthcoming methods. We have shown that the presented hybrid approach can be easily tailored to miscellaneous potential goals of tone mapping (e.g. to get perceptually plausible images, to get detail-rich depictions, etc.). Our experiences indicate that an implementation of proposed hybrid tone mapping approach typically produces reasonable results, it is easy to implement, fast to compute and it is comfortably scalable, if desired. These qualities nominate our approach for utilization in time-critical HDR applications like interactive visualizations, modern computer games, HDR image viewers on mobile devices, etc.

The perception of image attributes depends partially on the semantics of the input image or scene. Therefore every, even a subtle modification of an image can affect the quality of reproduction of an attribute (in both positive and negative sense). In the future, we will conduct subjective perceptual experiments to uncover and to quantify the effect of particular local enhancement method (in relation to the manner of enhancement map construction) on the quality of reproduction of image attributes.

ACKNOWLEDGEMENTS

This work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research programs MSM 6840770014 and LC-06008 (Center for Computer Graphics). Thanks to Jiří Bittner for proof-reading the manuscript.

REFERENCES

[Cad06] M. Cadik, M. Wimmer, L. Neumann, and A. Artusi. Image attributes and quality for evaluation of tone mapping operators. In *Proceedings of Pacific Graphics 2006*, pages 35– 44, Taipei, Taiwan, 2006. National Taiwan University Press.

[Cad07] M. Cadik, M. Wimmer, L. Neumann, and A. Artusi. Evaluation of HDR tone mapping methods using essential perceptual attributes. *Submitted to Journal of Visual Communication and Image Representation*, 2007.

[Cho03] P. Choudhury and J. Tumblin. The trilateral filter for high contrast images and meshes. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 186–196. Eurographics Association, 2003.

[CIE81] CIE. *An Analytical Model for Describing the Influence of Lighting Parameters upon Visual Performance*, volume 1: Technical Foundations, CIE 19/2.1. International Organization for Standardization, 1981.

[Dev02] K. Devlin, A. Chalmers, A. Wilkie, and W. Purgathofer. STAR: Tone reproduction and physically based spectral rendering. In D. Fellner and R. Scopignio, editors, *State of the Art Reports, Eurographics 2002*, pages 101–123. The Eurographics Association, September 2002.

[Dur00] F. Durand and J. Dorsey. Interactive tone mapping. In *Proceedings of the Eurographics Workshop on Rendering*. Springer Verlag, 2000. Held in Brno, Czech Republic.

[Dur02] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 257–266. ACM Press, 2002.

[Fai02] M. D. Fairchild and G. M. Johnson. The iCAM framework for image appearance, image differences, and image quality. In *IS&T/SID Color Imaging*, 2002.

[Fai03] M. D. Fairchild and G. M. Johnson. Image appearance modeling. In *SPIE/IS&T Electronic Imaging Conference*, 2003.

[Fer96] J. A. Ferwerda, S. N. Pattanaik, P. Shirley, and D. P. Greenberg. A model of visual adaptation for realistic image synthesis. *Computer Graphics*, 30(Annual Conference Series):249–258, 1996.

[Fia06] O. Fialka and M. Cadik. FFT and convolution performance in image filtering on GPU. In *Tenth International Conference on Information Visualisation*, pages 609– 614, Los Alamitos, 2006. IEEE Computer Society Press.

[Led04] P. Ledda, L. P. Santos, and A. Chalmers. A local model of eye adaptation for high dynamic range images. In *AFRIGRAPH '04: Proceedings of the 3rd international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 151–160, New York, NY, USA, 2004. ACM Press.

[Led05] P. Ledda, A. Chalmers, T. Troscianko, and H. Seetzen. Evaluation of tone mapping operators using a high dynamic range display. In *ACM SIGGRAPH 2005, LA*. ACM Press, August 2005.

[Neu98] L. Neumann, K. Matkovic, and W. Purgathofer. Automatic exposure in computer graphics based on the minimum information loss principle. In *Proceedings of Computer Graphics International (CGI '98)*, pages 666– 679. IEEE Computer Society Press, 1998.

[Pat00] S. Pattanaik, J. Tumblin, H. Yee, and D. Greenberg. Time-dependent visual adaptation for fast, realistic image display. In *SIGGRAPH 2000 Conference Proceedings*, Annual Conference Series, pages 47– 54. ACM SIGGRAPH, Addison Wesley, 2000.

[Rei05] E. Reinhard, G. Ward, S. Pattanaik, and P. Debevec. *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting*. Morgan Kaufmann, 2005.

[Tum99] J. Tumblin and G. Turk. Low curvature image simplifiers (LCIS). In *SIGGRAPH 99 Conference Proceedings*, Annual Conference Series, pages 83–90. Addison Wesley, 1999.

[War94] G. Ward. A contrast-based scalefactor for luminance display. *Graphics Gems IV*, pages 415– 421, 1994.

[War97] G. Ward, H. Rushmeier, and C. Piatko. A visibility matching tone reproduction operator for high dynamic range scenes. *IEEE Transactions on Visualization and Computer Graphics*, 3(4):291–306, 1997.

[Yos05] A. Yoshida, V. Blanz, K. Myszkowski, and H. P. Seidel. Perceptual evaluation of tone mapping operators with real-world scenes. *Human Vision & Electronic Imaging X, SPIE*, 2005.

Automated 3D scan multi-view registration based on rotation estimation

Huaxin Wang, Joris S. M. Vergeest, Yu Song, Tjamme Wiegers
CADE, DE, Industrial Design Eng.
Delft University of Technology
Landbergstraat 15
2628 CE, Delft
{huaxin.wang, j.s.m.vergeest, y.song, t.wiegers}@tudelft.nl

ABSTRACT

A new approach is presented for automatic registration of a sequence of 3D scanned range images into a complete model. Provided that the user follows the proposed scanning strategy, the difficulty to register the split views into a complete model is alleviated and success rate of automatic registration is improved by an adaptive forward estimation based on the extracted rotation axes and angles of the object between each scanning steps. The basic idea of this improvement is that when each of the scan differs slightly from its previous one in a small rotation about an approximately constant axis, the rotation axes and angle in the previous scanning step can be used as the first estimation of the rotation in the next step, so as to avoid manual coarse initial alignment. Formulation of the principle is provided and experimental results demonstrated.

Keywords

3D scan, automatic registration, estimation

1. INTRODUCTION

In industrial design, despite of the ubiquitous application of CAD digital modeling, traditional hand-made clay models remain as an intuitive approach to demonstrate and polish design concepts. As digital models are required for further machined prototyping and manufacturing purposes, fast digitalization of these physical models for import into CAD systems helps to build a seamless design process encompassing both physical and virtual modeling approaches.

In our recent project *Synthetic Environments for Design*, digitalization of physical models into the virtual environment for further simulations and operations is also an interesting approach. This is where the 3D scanning technique finds an application among others including reverse engineering of shapes for improvement or adaptation of the shape design of products which are in or out of production.

In the process of 3D digitalization of physical models, registration (matching and merging) of the different slices of the model surface scanned from different view angles is perceived as a bottleneck, as it requires tedious pre-processing to identify and provide initial matching points, which a product designer is typically unwilling to follow, and even economically impossible for bigger or complexer models that require registration of typically dozens to hundreds of scanned views. In the literature, a lot of attempts to automate the registration process have been reported, yet unfortunately, full automation of the registration of 3D scan data is, in general, still an unsolved problem [Ver01].

Nevertheless, supplementary data provided either by human assistance or additional devices, can help the registration algorithm -- normally an improved variant of ICP (iterative closest point) -- to succeed.

Besides manual selection of the initial matching points, other possible sources of supplementary data for automated registration were explored, the most significant approaches among which are: making use of the shape feature or geometric characteristics (spin image, normal vectors, etc.) [Ben01] [Joh01] [Pin01] [Sch01] [Sha01]; exploiting the scanning sequence information [Pin01]; global registration making use of overall consistency of all matches [Hub01]; improving the algorithm to find corresponding points, and exclude outlier points of the two shells to be registered [Che01] [Mas01] [Liu01] [Liu02] [Zha01].

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.

Summarizing all the available approaches, [Pla01] attempted to unify the terminologies used in this field and build a theoretical framework based on the taxonomy of the variety of methods, for which a thorough survey and comparison can be found in [Mat01]. Most of the above mentioned methods tried to attack the problem from two aspects: point matching and/or shell pair matching. Our proposed method won't try to replace the available registration algorithms, but to make use of the information acquired with the registration algorithms to improve their rate of success and thus can be combined with them.

The previous studies suggested that data about the initial relative or absolute poses can be introduced to reduce the number of wrong pair-wise matches (hence the space of solution to seek in), improve the robustness of the whole process and speed up the calculation for real-time interactive operation. Especially, a new approach in which a large series of 3D scans is made, where subsequent views differ in a small rotation about an approximately constant axis is proposed [Ver01]. Based on this assumption and statistical analysis of extracted rotation axis in previous scans, estimation of further rotation and translation can be used as a hint to rotate the next view into a better position that nearly snaps on the previous registered views.

In this paper, following and extending the above ideas, we try to further improve the success rate of automatic registration based on current available registration algorithms, starting with the attempt to acquire a quantitative impression of how far the assumption of nearly constant rotation axis of the user operation can stand, and how efficient this method can be.

Detailed description of the problem and geometric formulation of the solution are provided in section 2. After that, numerical results of some experiment examples are presented and analyzed. Our current comments and suggestions of possible extensions and improvements to this approach can be found at the end.

2. Theory

2.1. Principle

Because of the relative movement between the scanned object and the scanner caused by rotating the object or moving the scanner around the object, there is always relative displacement of the next scanned view (as point cloud and polygon shell data after processing) from where it should be registered with the previously scanned pieces. If there is a way to restore the position of the next scanned view to where it should be, the registration is done.

This relative movement can be decomposed into a pure rotation along an axis and a translation of the rotation axis. First, we consider the case in which the object is rotated without translation.

At the beginning, the object is scanned at certain position to get the first view. Then it's rotated by a certain angle θ along a rotation axis a and scanned to get a second view. If the second scanned view is rotated by angle θ along a in the opposite direction, theoretically it is registered with the first scanned view. So if there is a way to know the exact value of θ and a, registration can be done. As the general practice, these data can be acquired by using a digitally controlled rotation table or certain controlling or sensing apparatus with 3D position measurement. As the project is targeted at Small to Medium Business (SMB) applications, we're interested in less expensive automated solutions suitable for routine freehand operations by novice users.

On the other hand, it's a common experience that when the position difference of two scanned views are sufficiently small, currently available ICP code and the registration algorithms in commercial 3D CAD software (such as RapidForm of INUS) can present good result, even without manual selection of matching points. Yet scanning with very small position steps is very time consuming and things are even worse when it comes to the registration of the large number of shell data acquired.

So we now propose to aim at a balance between some efforts required from the inexperienced users, versus the chance to achieve automated registration.

The user is asked to rotate the object about a roughly constant axis, first by relatively small angles. Since the overlap of two subsequent views is big, the chance to obtain a successful automatic registration of the first two views with the current available CAD software is expected to be high. The rotation angle and axis of the object between these 2 scans can be extracted from the registration process. Then the user is guided to rotate the object further in the same direction with a steadily bigger angle step by the almost fixed rotation axis. Based the previously extracted rotation angle and axis, the next rotation can be estimated by extrapolation to rotate the shells to be registered into a better initial position for the registration algorithm to succeed.

By proceeding the scanning in sequence, 360 degree views of the object can be obtained and at the same time automatically registered. Normally it's still insufficient to obtain a digitization of the entire surface of the object, so optionally another (or even more) sequence of views rotated about a different

 ISBN 978-80-86943-98-5

axis should be taken. The user should rotate the object in roughly the same direction between each consecutive scan, unless a change of direction is necessary to cover the whole surface of the object.

The method to extract rotation angles and axes is further developed in the next section.

2.2. Geometric formulation of registration

The scanning and registration procedure is illustrated in Fig 1. For simplicity of illustration, a simple triangle surface representing one of the many surfaces that match between two consecutive scan views on a digitized shell of certain 3D shape is used. In the process of registering different scan views, this basic relationship still applies, though some of the surfaces on the two shells to be registered won't match with each other.

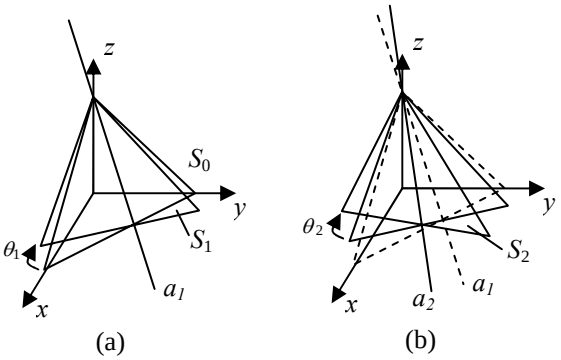


Fig 1. The consecutive scan and registration of surfaces

S_0 is one the scanned surface of the object in the starting position. Then, the object is rotated with an angle θ_1 about the axis a_1 in the direction shown as the arrow labeled θ_1, and a second scan is made in which the original surface S_0 is rotated to the position of surface S_1, as shown in Fig 1 (a).

For an ideal registration of S_0 to S_1, the resulting S_0 should be rotated to the exact position of S_1. The transform matrix $\mathbf{T}_1$ got from the registration process can thus be taken as the same for rotation θ_1.

In case the registration algorithm fails, the user can be requested to revert and do a smaller rotation to retry. We assume that given a sufficiently small rotation, the general registration algorithms should succeed, thus a $\mathbf{T}_1$ can always be got.

Then the user is asked to do the next scan with the object rotated in the same way as in the previous step. The next rotation angle θ_2 and axis a_2 might be different from θ_1 and a_1, because a manual operation can't be accurate, as shown in Fig 1 (b).

If we can know the rotation θ_2 about a_2 done by the user and apply it to the scanned data S_1, it can be registered to S_2. Normally this information can be acquired by position tracking hardware facilities such as a digitally controlled / readout turn-table. Nevertheless, we can take the approximation that θ_2 is repeating the similar angle of θ_1, and a_2 is the similar as a_1. If S_1 is rotated about axis a_1 with angle θ_1, it will be roughly aligned with S_2, whereafter within a tolerable error range for the automatic registration to work thus the requirement of a complex position tracking hardware facility is released. If the registration fails, possibly it's because the rotation angle θ_1 is too big or too small as a guess value for θ_2. First we try to increase the guess of rotation angle by multiplying a ratio $R > 1$ to the previous angle repeatedly, until registration succeeds, or an upper limit is reached. Then we try to decrease the guess of rotation angle by multiplying a ratio $R < 1$ to the previous angle repeatedly, until registration succeeds, or a lower limit is reached. If registration can't be done in this way, then the rotation axis a_2 might be too different from a_1, so the process reverts to how $\mathbf{T}_1$ was decided.

2.3. Extraction of rotation angle and axis

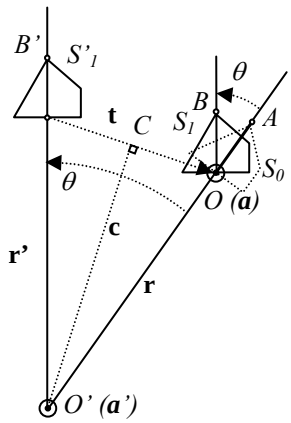


Fig 2. Effective rotation and translation in the registration procedure

Fig 2 shows a 2D projected view of two 3D scanned shells S_0 and S_1, where S_0 is to be registered to S_1. Shell S_0 at position A is rotated about axis $\boldsymbol{a}$ (located at point O, pointing out of the paper surface) with an angle θ to S_1 at position B. The vector $\mathbf{r}$ points from the world coordinate origin O' to O, which is one point on the rotation axis $\boldsymbol{a}$ such that $\mathbf{r}$ is perpendicular to $\boldsymbol{a}$.

The process of registration is equal to the problem to seek for a transform matrix $\mathbf{T}$, such that for the corresponding points on S_1 and S_0, in the least square error sense, $\mathbf{T}S_0 = S_1$, where

 ISBN 978-80-86943-98-5

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \ \mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad (1)$$

where **R** is the 3x3 rotation matrix, and **t** is the translation vector $[t_x, t_y, t_z]^T$. In general, **T** specifies a rotation **R** around an axis passing the coordinate origin followed by a translation along vector **t**.

T is a pure rotation if and only if the rotation axis ***a*** passes though *O'*, the origin of the world coordinate system. The transform matrix **T** of the registration process in effect rotates S_0 about the axis ***a***' with the same angle θ, to S'_1 at position *B'*, then translate it along vector **t** to position *B*. ***a***' is parallel with ***a*** and passes through the world coordinate origin *O*'.

The rotation angle θ and direction vector **k** of rotation axis ***a'*** can be extracted from **T** with the following formulae [Cra01]:

$$\theta = \arccos\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right)$$

$$\mathbf{k} = \frac{1}{2\sin\theta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \qquad (2a,b)$$

Vector **t** is perpendicular to vector axis ***a'*** and its length $|\mathbf{t}| = 2|\mathbf{r}||\sin(\theta/2)|$.

The absolute position of axis ***a*** can be derived as follows. Let $\mathbf{r}' = \mathbf{T}\cdot\mathbf{r}$, then $\mathbf{t} = \mathbf{r} - \mathbf{r}'$. Let point *C* be defined as the mid-point of *OB'*, then the line *O'C* is perpendicular to **t**. The direction vector of *O'C* can be represented by $\mathbf{k}\times\mathbf{t}$. Therefore we get:

$$O'C = \mathbf{c} = ½\ |\mathbf{t}|/\tan(\theta/2)\ (\mathbf{k}\times\mathbf{t})/|\mathbf{k}\times\mathbf{t}| \qquad (3a)$$

$$\mathbf{r} = \mathbf{c} + \mathbf{t}/2 \qquad (3b)$$

Since ***a*** is parallel to **k**, the points on the rotation axis ***a*** can be represented as $\mathbf{r} + p\mathbf{k}$ where *p* is any real number. Any points $\mathbf{a}_i$ on the rotation axis ***a*** will satisfy the equation $\mathbf{T}\mathbf{a}_i = \mathbf{a}_i$, i.e. $\mathbf{a}_i$ are the eigenvectors of **T** corresponding to the eigenvalue 1.

Besides rotation, there can also be a small translation $\mathbf{t_a}$ between two consecutive scans. The translation vector $\mathbf{t}_a$ can be decomposed into 2 components:

$$\mathbf{t_a} = \mathbf{t_n} + \mathbf{t_t} \qquad (4)$$

where $\mathbf{t_t}$ is perpendicular to the rotation axis ***a***, and $\mathbf{t_n}$ is parallel with ***a***. $\mathbf{t_t}$ is in effect contained in the translation vector **t** in the above formulation as a shift of the rotation axis location, while the effect of $\mathbf{t_n}$ is currently ignored. If it's beyond certain limit that defeats the registration algorithm, the user can be prompted to redo the scanning with a smaller translation movement.

2.4. Algorithm of rotation estimation

Based on the approach described in section 2.1-2.2, an adaptive automatic registration algorithm utilizing rotation estimation can be devised. Among the several possible sequences to pre-rotate the scanned shells to provide a roughly registered initial position for the easiness of automatic registration, we chose for simplicity to transform the previously registered scanned views to be aligned with the newly scanned data in each step. The detailed procedures are listed below:

1. Import scan data as shell S_i.

2. Rotate the object with a small angle to ensure that the rotation amount fits in the tolerance range of the registration algorithm.

3. Import the next scan data as shell S_{i+1}.

4. Fix S_{i+1} in place and register S_i to S_{i+1}. The transform matrix T_i of the registration operation is got.

5. Check if the registration is OK.

5.1. Not OK: Ask the user to rotate back the object then redo the rotation with a smaller angle and goto 3.

5.2. OK: Merge S_i with S_{i+1} and name it as S_{i+1}.

6. Extract the rotation angle θ_i and axis $\boldsymbol{a}_i$ from T_i.

7. Ask the user to rotate the object in the same direction as the previous rotation in step 2.

Although the same rotation angle as the previous step makes the automatic registration more reliable, for most cases a gradually bigger angle is preferred to speed up the scan process, while a slowing down preparation with gradually smaller angles is suggested (Fig 3) if the user plans to rotate the object about another distinct rotation axis after the next few scan steps in order to cover the complete surface of the object. In Fig 3, the positive and negative half axis is used to denote rotation angle θ about different rotation axes.

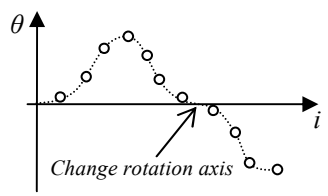


Fig 3. Change of rotation angle and rotation axis

8. Import scan data as S_{i+2}.

 ISBN 978-80-86943-98-5

9. Assuming the user rotated the object roughly about the same axis as he did in the previous step, θ_i can be used as the initial guess value of the rotation angle in step 7. Use θ_i and $\boldsymbol{a}_i$ as initial estimation of the rotation, set $\theta_g = \theta_i$.

10. Set magnification ratio $R = 1$;

11. Rotate S_{i+1} about axis $\boldsymbol{a}_i$ with angle $\theta_g \cdot R$ to be S'_{i+1}. Shell S'_{i+1} is now expected to be roughly aligned with S_{i+2} after this operation.

12. Try to register S'_{i+1} to S_{i+2}. Get the rotation angle θ'_{i+1} and axis $\boldsymbol{a}'_{i+1}$ from the total transform matrix including pre-rotation in step 11 and registration transform.

13. Check if registration is OK?

13.1. Not OK:

13.1.1. If $R >= 1$, enlarge R.

13.1.2. If $R < 1$, shrink R.

13.1.3. If $R >$ preset upper limit, set $R < 1$.

13.1.4. If $R <$ preset lower limit, set $i = i+1$, goto step 5.1.

13.2. OK: Set $\theta_{i+1}=\theta'_{i+1}$ $\boldsymbol{a}_{i+1}=\boldsymbol{a}'_{i+1}$, $i = i+1$, repeat from step 7.

3. EXPERIMENTS

3.1. Setup

The experiment setup consists of a dark color supporting surface (a table with black covering) or holder (dark color gloves for hand-held scanning or a pedestal to fix the object on the supporting surface for easier single person operation), a Minolta Vivid 700 3D laser scanner, a desktop computer running Vivid scanner controlling and data acquisition software Polygon Editing Tools, and INUS Tech RapidForm 2006. The object to be scanned was simply placed on the supporting surface or holder (as shown in Fig 10) in a moderate distance to the scanner to fit the object image inside the view finder frame. A set of C++ code is written to drive RapidForm 2006 through its COM automation API to implement the proposed automatic registration approach.

3.2. Scan

Because of the current software automation limitations, a simplified procedure similar to that described in 2.4 was followed in the experiments, in which all the scans were done in a batch before registration and stored in a sequence of data files. Then these data files were imported into one RapidForm model file and automatic registration was carried on by the user code in the same sequence of the scan. This simplified procedure differs from the proposed full one in that it either succeeds, or fails at certain step, without backtracking and re-doing feasibility. Nevertheless, the efficiency of the rotation estimation algorithm can still be demonstrated despite the limitations of the simplifications.

Time used for preparations like hardware / software adjustment, test of scan parameters for best output, etc. ranges from 0.5 to 1 hour for an experienced operator familiar with the hardware system and scan procedures.

Time used for each scan averages in 40 to 70 seconds for an experience operator.

As a comparison, general manual registration process for each pair of scanned shells takes about 1 to 3 extra minutes to select 3 matching points, which are once too often difficult to choose thus a new scan with proper markers attached to the object is required to provide matching points, otherwise completely impossible for objects with few visible feature points.

3.3. Registration

RapidForm API Function *RFMeshTools. RegisterShells()* was called in step 4 of section 2.4. The 4x4 transform matrix *RFTMatrix.m* returned by this API was taken as T_i.

3.4. Results

Test 1: Verification of the axis extraction

Several steps of rotation are simulated by rotating one shell in RapidForm several times to create a model file containing a set of purely rotated shells and the axis extraction algorithm was verified from registration result of this model file. The errors of the extracted rotation angles are acceptable. For ideal registrations, the extracted rotation axes should be theoretically the same, while those as results of non-ideal registrations are shown in the following Tab 1, and illustrated in Fig 4.

Tab 1. Rotation angle and axis extraction results

a) Angle

Step	Ideal (deg)	Extracted (deg)
1	5	5.002890
2	10	10.003064
3	10	10.001035

b) Axis

Step	Ideal	Extracted
1	(0, 1, 0)	(0.000492, 1.000000, -0.000316)
2	(0, 1, 0)	(0.000061, 1.000000, -0.000422)
3	(0, 1, 0)	(0.001556, 0.999999, -0.000375)

 ISBN 978-80-86943-98-5

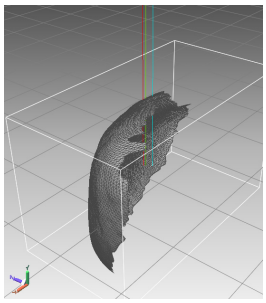
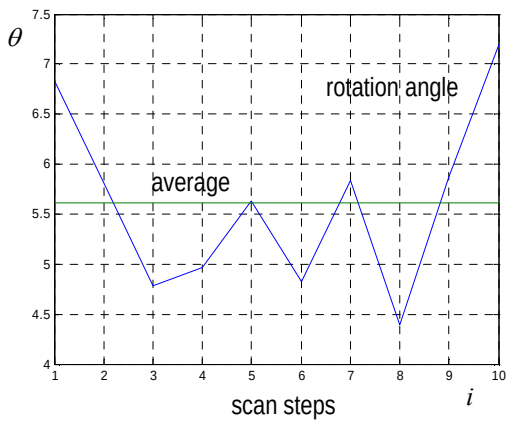


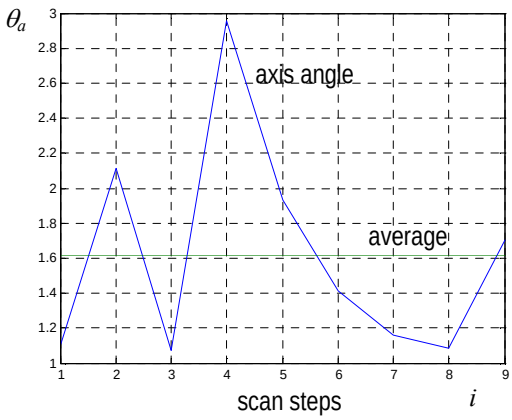
Fig 4. The directions of the rotation axes are almost the same, but the origins of rotation axes have small vertical shifts (along the rotation axis) and apparent horizontal shifts (in *xoz* plane).

Test 2: Accuracy of manual rotation

To investigate the accuracy of the user's manual rotation to maintain the same rotation axis and angle between each scan, 10 scans were made with manual rotation of almost the same axis and angle on the table. The rotation axes and angles were extracted as approximated estimation of the rotation motion done by the user. Results are illustrated in Fig 5 and Fig 6, which provide an intuitive verification of how well the presumption of constant rotation axis can be achieved manually on a supporting surface.



a) Rotation angle and average angle (degree)



b) The angle (degree) between the estimated approximate rotation axis and the first axis, and the average

Fig 5. Manual rotation on the table

3D illustration of the approximate rotation axes extracted from the registration process of a scanned model rotated by hand on the table is presented to give a visual impression of the variation of location and direction of the rotation axes as an empirical indication of the stability of the estimation method.

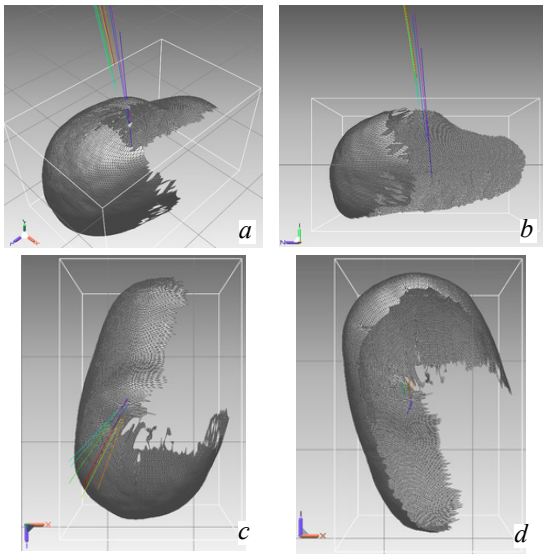


Fig 6. Approximate rotation axes for manual rotation on table *a*) isometric *b*) back *c*) top *d*) bottom view

Test 3: Comparison of rotation angles of registrations with and without pre-rotation

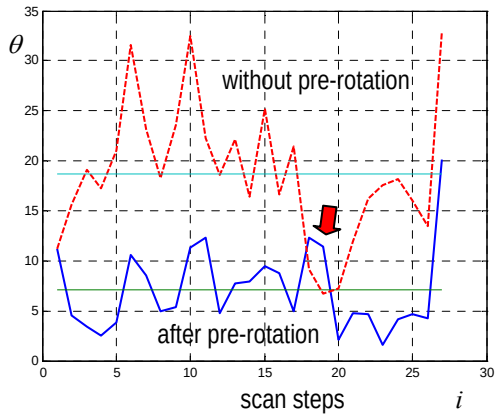


Fig 7. Rotation angle of registration without pre-rotation (upper line) **and after pre-rotation** (lower line), (arrow points to region of rotation axis change)

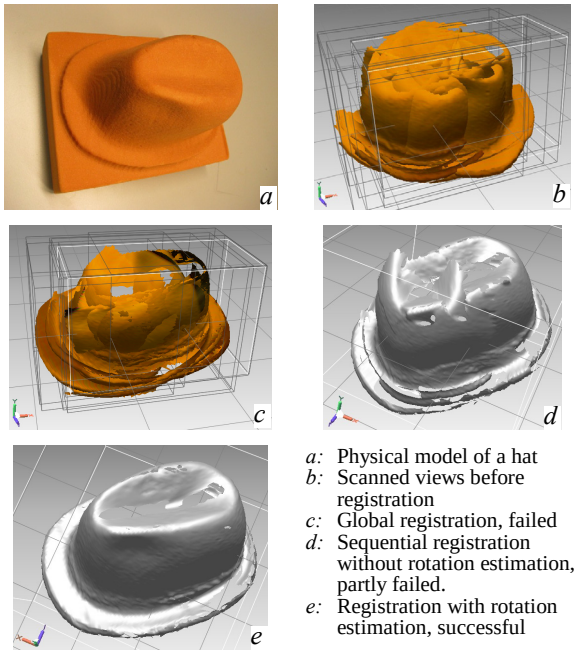
The rotation angles as returned results of registrations are compared in Fig 7, for the case without pre-rotation and the one with pre-rotation based on estimated approximate rotation motion in the previous step. It shows that the rotation angles of registrations are apparently decreased by the pre-rotation. This means the estimation and pre-rotation

 ISBN 978-80-86943-98-5

did help to make the registration easier by remarkably reducing the necessary amount of rotation to do the registration.

Test 4: Comparison of registration failure without and with the help of rotation estimation

One of the examples is shown in Fig 8 in which the automatic registration is successful with rotation estimation based pre-rotation, while both the RapidForm global registration and the sequential pairwise registration failed. The failure seems caused by the geometrical similarity of different parts of the shape.



a: Physical model of a hat
b: Scanned views before registration
c: Global registration, failed
d: Sequential registration without rotation estimation, partly failed.
e: Registration with rotation estimation, successful

Fig 8. With and without rotation estimation

Test 5: Complete surface scan

An example of successful registration of complete surface scans of an object is demonstrated in Fig 9. The rotation axes and angles were changed as suggested in section 2.4, step 7, in order to cover the complete surface of the object, as shown in Fig 10. Totally 28 scans were done with rotation angles varying from 5 to 35 degrees, while the rotation angles for registration after pre-rotation estimation ranges from as low as 1.6 degree up to 20 degrees (Fig 7).

The extracted rotation axes group in two different direction clusters which correspond to those in the scanning process. But note that the relative angles between the vertical rotation axes (the first rotation direction) and the object is not correctly drawn because of relative movement of the shells later in another different direction in the registration process.

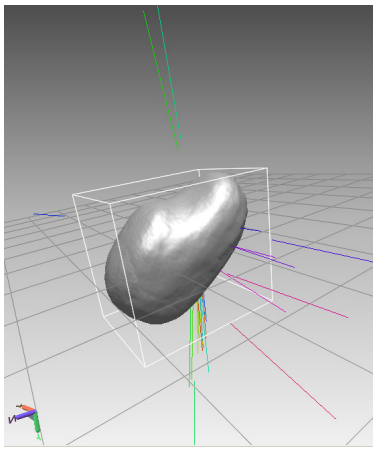


Fig 9. Complete surface registration and rotation axes

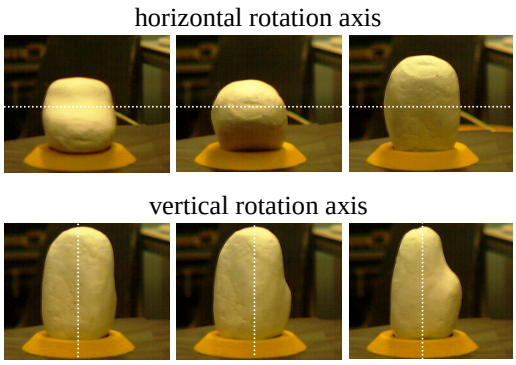


Fig 10. Rotation axis direction change

4. CONCLUSION

4.1. Advantages

The following advantages of the proposed approach can be observed in the theoretical formulation and experiment results:

a. Automatic batch registration is successful for more cases, especially for those objects with many similar geometric features for which automatic registration algorithms are doomed to fail without help of rotation information.

b. No complex hardware device is required to record the rotation angles. For the same reason it's also suitable for cases where such device is not applicable, e.g. scanning of non-movable objects.

c. No sticker marker is required to be placed on the object to help choosing of matching points for registration. Thus the scanned texture images are not disturbed by the markers and can be retained for later texture mapping of the 3D model.

 ISBN 978-80-86943-98-5

4.2. Limitations

a. Can't work well with hand-held rotation raised above the supporting surface.

b. Scanning should be kept in sequence thus inserting scans to patch a region is difficult for automatic registration. To insert a scan in to the sequence, the user must rotate step by step back to the position where the desired scan is to be made.

c. Cumulative registration error in such sequential pair-wise registration approaches may degrade the global registration result and should be taken into consideration. An extra global fine registration step should improve the result further.

4.3. Possible future extensions

Registration result for the hand-held rotation case needs to be improved for more casual applications of 3D scan. The proposed method may be combined with methods like feature point extraction and matching to improve the rotation axis estimation.

Currently, the judgment of whether the registration of each step is successful is done by visual inspection of the user. Fully automation of the registration process requires a method to evaluate the registration validity and accuracy. It may also help in the rotation angle estimation step (section 2.4, step 10) to choose the best magnification ratio *R*.

5. ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of the Dutch Innovation Oriented Research Program 'Integrated Product Creation and Realization (IOP-IPCR)' of the Dutch Ministry of Economic Affairs.

6. REFERENCES

[Ben01] Bendels, G. H., Degener, P., Wahl, R., Koertgen, M., Klein, R. Image-Based Registration of 3D-Range Data Using Feature Surface Elements, The 5th Int. Symp. Virtual Reality, Archaeology and Cultural Heritage (VAST 2004).

[Che01] Chen, Y., Medioni, G. G. Object modelling by registration of multiple range images, Image and Vision Comp. 10(3), pp 145-155, 1992.

[Cra01] Craig, J. J. Introduction to robotics: mechanics and control. Addison-Wesley Publishing Company, Inc., 1989, Edition 2, p52.

[Hub01] Huber, D. F., Hebert, M. Fully Automatic Registration of Multiple 3D Data Sets, Image and Vision Computing Vol. 23, 2003, pp. 637 – 650.

[Joh01] Johnson, A. E., Hebert, M. Using spin images for efficient object recognition in cluttered 3D scenes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21, 5, 1999, pp433- 449.

[Liu01] Liu Y. Improving ICP with easy implementation for free-form surface matching, Pattern Recognition, Vol. 37, No. 2, February 2004, pp. 211-226, http://dx.doi.org/10.1016/S0031-3203(03)00239-5.

[Liu02] Liu, Y. Automatic 3d free form shape matching using the graduated assignment algorithm, Pattern Recognition, Vol. 38, Issue 10, October 2005, Pages 1615-1631, http://dx.doi.org/10.1016/j.patcog.2005.01.008.

[Mas01] Masuda, T., Yokoya, N. A robust method for registration and segmentation of multiple range images. Comp. Vision and Image Under. 61(3), pp 295-307, May 1995.

[Mat01] Matabosch, C., Salvi, J., Fofi, D., Meriaudeau, F. Range Image Fusion for Industrial Inspection, SPIE Electronic Imaging-Machine Vision Applications in Industrial Inspection XIII, San José, USA, January 2005 http://vision.u-bourgogne.fr/le2i/user_data/publications/matabosch05b.pdf

[Pin01] Pingi, P., Fasano, A., Cignoni, P., Montani, C., Scopigno, R. Exploiting the scanning sequence for automatic registration of large sets of range maps, Computer Graphics Forum, 24:3, Sept. 2005, p. 517.

[Pla01] Planitz, B.M. et al. The correspondence framework for 3D surface matching algorithms. Computer Vision and Image Understanding, 2005.

[Sha01] Sharp, G. C., Lee, S. W., Wehe, D. K. Invariant features and the registration of rigid bodies. Proc. IEEE Int. Conf. on Robotics and Autom., pp 932-937, 1999.

[Sch01] Schuetz, C. J., Huegli, H. T. Multi-Feature Matching Algorithm for Free-Form 3D Surface Registration, International Conference on Pattern Recognition, Vol. 14 No. 2, 1998, pages 982-984.

[Ver01] Vergeest, J., Song, Y. ICP fitness analysis for 3D shape data matching. Technical Report, Delft University of Technology, 19 January 2006.

[Zha01] Zhang, Z. Y. Iterative point matching for registration of free-form curves and surfaces. Int. J. of Computer Vision, 13(2), pp 119-15, Oct. 1994

An Evolutionary Strategy for Free Form Feature Identification in 3D CAD Models

T.R.Langerak, J.S.M.Vergeest, H.Wang, Y.Song
Delft University of Technology
Landbergstraat 15
2628 CE Delft
The Netherlands

t.r.langerak@tudelft.nl

ABSTRACT

Nowadays, most 3D CAD systems support the use of form features. The main advantage of form features is that they provide parametric, high-level support for shape manipulation. When the parametric information of a shape is not available, it can be retrieved using a feature recognition procedure. In this procedure, a target shape is recognized as an instance of one or more features in a pre-defined feature library. The speed and accuracy of the feature recognition procedure significantly improve when the target feature type is known. This information can be provided by a user, but we propose a new method that identifies the feature type automatically. This method uses an evolutionary algorithm to find the optimal feature type. The algorithm randomly generates a population of instances of every type of feature available in the feature library. From this initial population of feature instances, successive populations are generated using the principle of natural selection. The algorithm was tested for different settings and was found to correctly identify features in between one and three minutes.

Keywords

Feature recognition, evolutionary algorithms, feature library

1. INTRODUCTION

One of the trends in geometric modeling in the past decades has been to bring shape manipulation routines to a higher level, meaning that a modeler is able to perform a certain shape operation with less effort. A means to do this is by using *form features*. A form feature is a parameterized shape (part) that can be manipulated in pre-defined ways by adapting parameter values. Many, if not all, currently used geometric modeling applications use form features.

Information on form features is stored and maintained as a model is built up, to ensure that form features are available at every step of the modeling process and always lead to a valid result. However, there are situations in which the form feature information is not available, such as when a form feature is an unintended result of other modeling operations or when there is a domain change, e.g. when using different modeling applications. Form feature information is also not available when raw geometrical data is imported, such as when data is obtained from physical objects by a 3D measuring device. In these cases, form feature information can be retrieved through the process of *form feature recognition*. As a special version of shape recognition, form feature recognition does not only retrieve shape information, but also a semantic understanding of the (possible) parametric constitution of the shape. In the practical sense, feature recognition means matching a target shape to the information that is available in a *feature library*, the collection of available features.

In the past decades, many algorithms have been developed for feature recognition in the domain of machining features. These algorithms are mostly applied to communication between Computer-Aided Design applications and Computer-Aided Manufacturing applications, but are not useful for free form modeling applications.

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.

Efforts are being made to extend feature recognition to the free form domain, but many of the techniques used for machining features can not, or only in highly adapted forms, be used for free form features. Some methods for free form feature recognition have been developed ([Tho99], [Song05], [Lan05], [Pal05]), but the results are few and preliminary.

More so than for machining features, the recognition of free form features is aided by user input. Because free form features are more complex than machining features, computation times are unavoidably larger. Extra information that is given by the user can help in reducing the computation time. Also, the set of theoretically correct solutions is larger than for machining features but the user is typically interested in only one or a few results. User input is needed to select the desired output from the set of feasible outputs.

One of the pieces of information that lead to more efficient algorithms is the type of the feature to be recognized. When the feature type of the target shape is known, the solution space of the problem is known and the search strategy can be adapted. This information can easily be asked of the user when the feature library is small and distinct, but when the number of features to choose from is large and when the distinction between features is small, then providing the right feature type requires an expert user. In this paper we propose a method for retrieving the feature type of a target shape automatically. The method makes use of the well-known principle of evolutionary computation.

Evolutionary computation is a popular technique that is based on the principle of natural selection. It applies 'survival of the fittest' to populations of possible solutions to a problem, in order to evolve towards an optimal solution. Survival of the fittest is a biological mechanism in which organisms with certain genetic elements have a higher chance to procreate and are therefore more likely to pass on these genetic elements to a next generation. Evolutionary algorithms have been applied to many computational issues in the past. Good results have been reported, but part of the popularity of evolutionary algorithms can only be explained by their elegance. The only application of evolutionary algorithms to feature recognition known to the authors is that of Pal et al. [Pal05]. However, this method makes far-going assumptions on the input data and does not result in a parametric description of the recognized feature.

2. PREVIOUS WORK

In the past two decades, research after form feature recognition has been fueled by its application in Computer-Aided Process Planning (CAPP), linking design to manufacturing. Many different techniques for form feature recognition have been developed, of which good overviews can be found in ([Shah01], [Sub95]). Among the techniques that have been developed are graph-based feature recognition ([Jos88], [Chu90], [Flo89]), hint-based feature recognition ([Van93], [Van94]), volume decomposition ([Woo82], [Kim92], [Sak95], [Sak96]) and neural networks ([Pra92], [Nez97]). Graph-based feature recognition expresses target shapes in the form of connectivity graphs and then uses graph-based heuristics to search these graphs for features. Hint-based feature recognition uses a two-step approach to feature recognition; in the first step, hints of possible features are retrieved and in the second step these hints are processed and combined. Volume decomposition expresses target shapes in the form of a tree in which each node is defined as the difference between its parent and its parent's convex hull. Features can be found in the leafs of the tree.

More recently, techniques have been proposed for free form feature recognition. Thompson et al [Tho99] proposed a method to recognize features from point clouds that are the result of scanning real objects. This was extended by Song et al. [Song05], who implemented a method that recognizes features by fitting a template feature to the point cloud data. Once an optimal configuration is found, the parameter types and values of the recognized feature match that of the template feature. They also show how the template feature can be used to manipulate the recognized feature. Recently, Langerak et al. [Lan05] proposed a method to recognize styling features in polyhedral data. This method uses surface curvature analysis to locate features and offers a sketch-based manipulation tool for the identified feature.

There is literature on evolutionary computation in abundance. Goldberg [Gol89] and Davis [Dav91] are good early references on the subject. A more recent overview of the actual problems in evolutionary computation is given by Ghosh and Tsutsui [Gho02].

Evolutionary algorithms have been recently applied to feature recognition by Pal et al. [Pal05]. Their method extracts features from collections of feature surfaces by generating a population of individuals with a random number of surfaces from the given collection. Consecutive generations inherit feature surfaces, which thus accumulate to form an entire, recognized feature. They report good results, but their method is only applicable to polyhedral inputs and assumes it to be know what surfaces belong to a feature. In addition, their method is not able to handle feature interference.

 ISBN 978-80-86943-98-5

3. METHOD OUTLINE

Feature identification can be seen as a minimization problem of a shape similarity function in the parametric domain of a feature. The minimization problem is a well-known problem. Several methods for solving it have been proposed in the past, but as the solution space is different for each problem there is no generic method that works for all problems. A short overview and implementations can be found in Press et al. [Pre02]. A popular method for the minimization problem is the conjugate gradient method, which bases its search strategy on the derivative of the fitness function. This method is not applicable to the feature identification problem, because a derivative for the shape similarity function can not be computed. Other minimization methods, such as the direction-set method or the simplex method are slow and inefficient. We therefore choose to apply an evolutionary strategy to feature identification, which does not rely on derivative information and can be, depending on its implementation, relatively fast.

Evolutionary algorithms solve a problem by regarding some element of the problem as an individual, onto which the mechanisms of evolution are applied. Each individual has a genetic mockup that reflects the variables that play a role in the problem that is to be solved. When an individual procreates, its genetic constitution is inherited by an individual in the next generation. The chance that an individual procreates is derived from its fitness, which is a value that indicates how well the individual represents a solution to the problem. How well an evolutionary algorithm performs depends on the shape of the solution space of the problem, of course, but also on the quality of the genetic mechanism.

In the case of feature identification, populations are formed by individual features instances. The shape similarity between a feature instance and the target shape serves as the fitness function in the evolutionary computation. Because the appearance of feature instance is determined by its parametric configuration, it is only natural to consider parameters to be a feature's genes. In this section we will first describe the constitution of a feature instance in more detail, and then define the mechanisms that render successive feature populations.

3.1 A Free Form Feature Definition

Because the evolution of a feature is managed through its parameters, in defining a feature there is a strong emphasis on the parametric constitution of a feature. However, because all parameters need to be treated similarly in an evolutionary computation procedure, we use a two-layered system in which one layer defines the amount of parametric influence, whereas the second layer, which we call the *parameter mapping*, defines the nature of the influence. As an added benefit, this allows for a generic feature definition system, in which the parameter mapping can be made domain-independent.

As a means to represent features, we use nurbs patches. Parametric influence is defined through the nurbs control points, so as to mirror a natural genetic structure, where each gene has a limited global influence. The following definition is used:

A free form feature is a parametric description of a shape, $\{E, P, \mu\}$*, containing at least*

- *a set of nurbs control points* $E = \{e_1 = (x_1, y_1, z_1), \ldots, e_n = (x_n, y_n, z_n)\}$ each with a basic state $e_i^0 = (x_i^0, y_i^0, z_i^0)$, $1 \le i \le n$
- *a parameter set* $P = \{p_1, \ldots, p_m\}$, $1 \le j \le m$ and $p_j = \{name_j, value_j\}$
- *a parameter mapping, defined as the set* $\mu = \bigcup_{i=1}^{n} \bigcup_{j=1}^{m} \{\mu_j^i (e_i \in E, p_j \in P)\}$*, where* μ_j^i *is a function with the type* $(e, p) \to e$ *that defines the influence of parameter* p_j *on element* e_i*. The configuration of each element can be computed by combining all the mappings defined on the element, so that:*

$$e_i = \mu_1^i(\mu_2^i(\ldots(\mu_{m-1}^i(\mu_m^i(e_i^0, p_m), p_{m-1})\ldots), p_2), p_1)$$

In our implementation, we express the nurbs control points in homogenous coordinates and define the parameter influence in the form of 4 by 4 transformation matrices. Each function μ_j^i consist of two parts: a transformation matrix and a function mask matrix that defines how parameter values influence the elements of the transformation matrix. For example, when a control point e has a basic state $e^0 = (0,0,0)$, then a parameter that defines a vertical movement (in the z-direction) is defined by a transformation matrix $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ and a function mask matrix:

 ISBN 978-80-86943-98-5

$$\begin{pmatrix} const & const & const & const \\ const & const & const & const \\ const & const & const & var \\ const & const & const & const \end{pmatrix},$$

where $const(x, p) = x$ and $var(x,p) = x \cdot p$.

This definition allows for orientation and location of a feature to be regarded as a parameter of the feature as well, by using the cosine and sine functions in the function mask matrices. Although traditionally orientation and location are not considered to be part of the parametric space of a feature, they do play an important role in the identification of a feature. Location and rotation settings do alter the shape representation of a feature, and there is therefore no theoretical objection to regarding them as parameters. This enables orientation and location to be treated identical to other parameters. They do, however, differ from other parameters in that they have identical parameter mappings for all nurbs control points.

Our implementation allows for both rigid body control of the feature as well as deformation, but for the sake of simplicity in this paper it is assumed that only rigid body transformation is used.

3.2 Evolutionary Computation

The goal of the feature identification procedure is to find the type of a feature that best matches a certain target shape. It may be that only part of the target shape represents a feature. The more 'material' there is that does not belong to the feature, the higher the computation time will be and the less accurate the result. In our implementation we therefore provide a mechanism for a user to select a region of interest. This mechanism asks a user to place a bounding box roughly at the location of the feature. Only the shape data that is contained by this bounding box is used in the feature identification procedure. As will be shown in section 4, this mechanism is not needed to guarantee the correctness of the feature identification, it is merely a means to bring back the computation time.

At the start of the evolutionary computation, a population of features is generated. Each feature is a copy of a randomly chosen feature from the feature library, so the initial feature population consists of a mix of all the features that are present in the feature library. The features are initiated with random parameter values in the domain [-1000, 1000]. This domain was used with the assumption that, in practical cases, no parameter values outside the domain are used. If necessary, the scale can be adapted by using larger values in the transformation matrix. The size of the population is variable and influences the success of the identification procedure. In section 4, it is shown how different population sizes affect the success of the procedure.

From this initial population, successive populations are generated using the following steps:

1. The fitness $f()$ of each individual is computed as the mean directed Hausdorff distance between the feature and the target shape. The directed Hausdorff distance is defined as the smallest maximal distance between two shapes. Because the Hausdorff distance is most efficiently computed on point sets, we use point set samples of both target shape and feature shape to compute the fitness.
2. The individuals in the population are ranked by fitness, so that $f(F^0) \geq \ldots \geq f(F^n)$, where the F 's are the features in the population, f is the fitness function and n the population size. If the terminating conditions are met, the algorithm terminates. For brevity, we will denote $f(F^o)$ as f^0 .
3. For each of the n individuals in the next generation, two parents (a 'mother' and a 'father') are selected. An individual feature can be parent to more than one individual in the next generation. The chance that an individual is selected to be a parent is determined by a one-sided Gaussian distribution of the fitness: $P(f^{parent}) = \frac{2}{\sigma\sqrt{2\pi}} e^{\frac{-(f^{parent})}{2\sigma^2}}$

   The standard deviation σ determines how fast the chance to procreate decreases with the fitness and is also known as the *selection size*. Individuals with a low fitness are less likely to become parents to a next generation.
4. For each individual in the new generation, parameters are randomly copied from one of the two parents. To prevent the degrading of the gene pool, the collection of available genes, in a population is subject to mutation. The chance that mutation occurs is indicated as the *mutation probability*. When mutation does occur, the value of the copied gene is distorted by Gaussian noise with a standard deviation that is also known as the *mutation* rate. The two parents may be of different types and thus have a different number of parameters. In this case, without loss of generality, assume that $\left|P^{mother}\right| \geq \left|P^{father}\right|$.

 ISBN 978-80-86943-98-5

Then $\left|P^{child}\right|$ receives a random value between $\left|P^{mother}\right|$ and $\left|P^{father}\right|$ and $p_i = p_i^{mother}$ if $i > \left|P^{father}\right|$.

5. When a new population of features is generated, the algorithm proceeds at step one.

While going through an iterative process, the algorithm keeps track of the ancestry of a feature. As in natural evolution, each individual feature can be placed in a family tree and by backtracking it can be determined from what ancestors an individual stems. Instead of backtracking for each individual, this information is passed on and added to when new individuals are created.

The algorithm terminates when one of the following conditions are met:

1. More than 75% of the ancestors of the 10% fittest individuals in a population have the same feature type. This type is returned as the result of the feature identification.
2. The fitness of the fittest individual drops below a certain threshold. In this case, it is very likely that a large majority of the ancestors of this feature have the same type, so the type that occurs the most in the feature's ancestral history is returned as the identified feature type.
3. The fitness of the fittest individual in the population is lower than or equal to that of the previous generation. In this case, the algorithm is likely to be stuck in a local minimum. As a best guess, the feature type that occurs in 50% of the ancestors of the 10% fittest individuals is returned as the identified feature type. If no such feature exists, the algorithm fails.

4. RESULTS AND ANALYSIS

The algorithm was tested on a feature library that consists of 8 features (see figure 1). This library contains features that have been the subject of past work on features (Bump, Ridge) and other features known from the modeling world (Step, Blend) as well as newly constructed features that we expect to have a significantly different search space than the other features (Cross, Wave, Crown). The cross feature is a combination of two ridge features, with two perpendicular regions of leveled geometry. The Wave feature is a special version of the ridge feature, with an added parameter that causes the ridge to lean over. The Crown feature is a special version of the Bump feature that does not only have a parameter for the central bump, but for the middle and the corner of its edges as well. Finally, a Plane feature without parameters was included to test if the algorithm is able to distinguish cases of other features with small parameter values.

Each feature is defined using a 5 by 5 patch of nurbs control points. Each feature has a height parameter that defines a translation of the control points in the z-direction. The translation in the x- and y-direction is combined in the parameter *radius* (for the Bump and Crown feature) or defined separately as the parameter *width* and the parameter *length*

Feature identification starts with asking the user to place a bounding box on a target shape to determine the rough position and orientation of the feature. The bounding box also indicates what parts of the target shape assumingly belong to the feature to be identified.

4.1 Testing on Artificial Target Shapes

Because testing the algorithm on a large number of target shapes is very time-intensive, we simulated target shapes by instantiating a random feature from the library and setting its parameter values to random values.

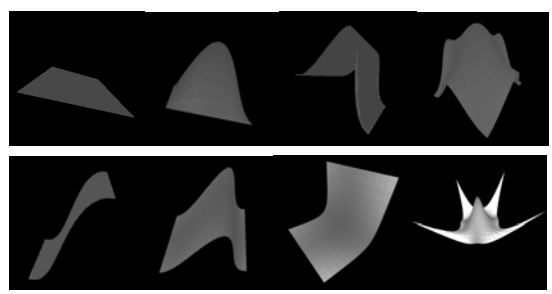


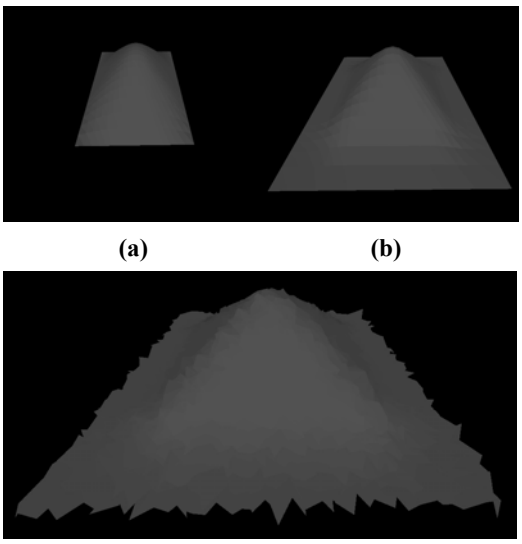
Figure 1: From the top left to bottom right: Plane, Bump, Ridge, Cross, Step, Wave, Blend and Crown feature.

Possible 'excess' material was simulated by creating two extra layers of control points around the control points used for the feature. The entire feature was converted to a triangular mesh with 2500 triangles and the points of the mesh were distorted by Gaussian noise. We conducted 100 tests for each combination of five different population sizes and five different selection sizes (which indicate the standard deviation of the Gaussian curve used for parent selection), amounting to 2500 tests in total. The results are given in Table 1.

The selection size is expressed as a percentage of the total population, and determines how selective the fitness values are. It indicates the percentage of

 ISBN 978-80-86943-98-5

individuals for which the chance of being selected as a parent is within two standard deviations of that of a prefect individual. The best result occurs for a selection size of 10%. The algorithm was implemented in C++ and tested on a computer with a 3GHz Pentium processor and 1GB RAM. The average computation times for the different population sizes were 58, 86, 112, 139 and 165 seconds for the respective population sizes. As the initial orientation and location of the feature are assumed to be given by the user, the 6 parameter values for translation and rotation are set to zero. To correct for the inaccuracy of the initial estimation of the user, Gaussian noise was added to the parameter values. The variables mutation probability and mutation rate did not have a significant influence on the success of the feature identification or the computation time.



(a) **(b)**

(c)

Figure 2: (a) A target feature, randomly chosen from the feature library (b) with added 'material' (c) distorted with Gaussian noise

Feature type *(# of params)*	Correctly identified	Incorrectly identified	Not identified	Total
Plane *(6)*	308 (96%)	13 (4%)	1 (0%)	**322**
Bump *(8)*	291 (94%)	14 (5%)	3 (1%)	**308**
Ridge *(9)*	274 (88%)	32 (10%)	5 (2%)	**311**
Cross *(9)*	324 (96%)	8 (2%)	4 (1%)	**336**
Step *(8)*	296 (95%)	14 (5%)	1(0%)	**311**
Wave *(10)*	212 (69%)	73 (24%)	23 (7%)	**308**
Blend *(8)*	301 (96%)	9 (3%)	2 (1%)	**312**
Crown *(10)*	284 (97%)	4 (1%)	4 (1%)	**292**
Total	**2290 (92%)**	**167 (7%)**	**43 (2%)**	**2500**

Table 2: Results of the feature identification per feature type

A possible explanation for this is that the algorithm converges to a solution very fast (in on average 2.41 generations). The effect of mutation rate and probability is more long term and therefore not apparent in our tests. The effect of the initial inaccuracy of the translational and rotational parameters was significant, but very small. This is probably due to the large size of the domain in which parameter values are instantiated.

Table 2 shows, for each feature type, the number of features that were correctly or incorrectly recognized or for which the algorithm failed. The performance of the algorithm for the wave feature is considerably worse than for the other feature types, because it is often recognized as a ridge feature. This is no surprise, because the wave feature is a special version of the ridge feature, with one added parameter.

Population size/ Selection size	1000	1500	2000	2500	3000	Total
5%	78	91	96	100	100	**463**
10%	83	96	100	100	100	**479**
20%	82	94	95	99	100	**470**
30%	76	88	88	97	100	**449**
40%	62	83	89	93	100	**429**
Total	**381 (76.2%)**	**452 (90.4%)**	**468 (93.6%)**	**489 (98.8%)**	**500 (100%)**	**2290**

Table 1: Number of successful feature identifications per population and selection size

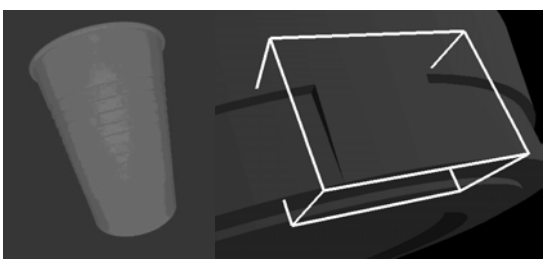
 ISBN 978-80-86943-98-5

If this parameter has a small value, the wave feature behaves almost identical to the ridge feature. This also explains the slightly worse performance for the ridge feature.

From the results it can be concluded that the method correctly identifies a large portion of the test shapes, and that the computation time of the method is reasonable. Feature identification cannot be compared to other methods since no similar method exists, known to the authors, in current literature.

4.2 An Application Example

To demonstrate that the algorithm also works in non-artificial situations, we show an application example in which the algorithm is applied to a CAD model of a plastic coffee cup (see figure 3a). One of the features in this model, a wedge on the edge of the bottom of the cup, is shown in figure 3b, indicated by the user with a bounding box. To successfully be able to recognize the wedge-shaped feature, a new feature was added to the library, as is shown in figure 4. With this addition, the library consisted of 9 features.



(a) **(b)**

Figure 3: (a) CAD model of a coffee cup (b) A region of interest

At the start of the procedure, the user was asked to place a bounding box at the region of interest, as shown in figure 3b. The bounding box contained 113 polygons. From there on, a feature identification procedure was started. The feature identification used a feature population of 3000 and a selection size of 10%. For the identification, a mutation rate and probability of respectively 0.1 and 10% were used. The feature identification procedure was repeated 10 times. In all repetitions of the procedure, the feature was correctly recognized as a wedge feature. In 8 of these repetitions, three generations were needed to come to a correct identification, which cost on average 172 seconds. In two cases, only two generations were needed, in on average 118 seconds. In all cases, the feature identification procedure terminated because 75% of the ancestors of the top feature in the last generation were of the type Wedge.

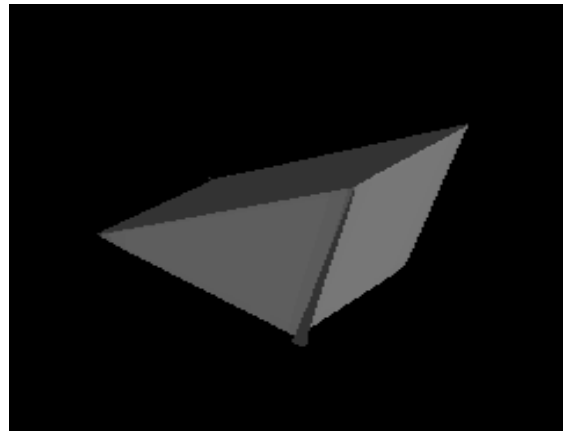


Figure 4: A wedge feature

5. CONCLUSIONS AND FURTHER WORK

We have proposed and implemented a method for feature identification using an evolutionary algorithm. The algorithm is successful in retrieving the feature type of a target shape and its computation time is reasonable. The algorithm was tested on a small feature library, but there is no theoretical limitation for using it on a larger, even interactive feature library. We intend to use the feature identification in a broader free form feature support system, in which users define and compose their own features, which can then be recognized in target shapes of their choice. We are currently completing an extension of the presented techniques to a free form feature recognition algorithm. In this procedure, the feature type is assumed to be known, but the parameter values are not. The free form feature recognition algorithm retrieves the parameter values in an evolutionary process that is similar to that presented in this paper.

An even more advanced application of the presented evolutionary technique that we are working on is in feature type construction. In this procedure, the target shape does not match a feature type that is available in the feature library. The shape is analyzed and a new feature type is automatically constructed that fits the target shape. To obtain a mechanism that is flexible enough to automatically construct a feature type, the genes are no longer formed by the parameters, but by the elements of the parameter mappings, increasing the number of genes from m to mn, where m is the number of parameters and n the number of nurbs control points. Also, the principle of mutation is extended to allow new individuals to have more or less genes than their parents.

The proposed feature identification method can be applied to retrieve the feature type prior to any (free form) feature recognition system. In future work it

 ISBN 978-80-86943-98-5

will be investigated if other user-input can be automated, such as the location of a feature or an estimation of its parameter values. When more parts of the work with form features is automated, an even higher-level approach to feature recognition can be implemented that in turn allows a user to more easily manipulate shapes.

6. REFERENCES

[Chu90] Chuang, S. H., Henderson, M.R., Three-dimensional shape pattern recognition using vertex classification and vertex-edge graph Computer-Aided Design, Vol. 22, No. 6, pp. 377-387, 1990.

[Dav91] Davis, L, editor, Handbook of genetic algorithms, New York, Van Nostrand Reuinhold, 1991.

[Flo89] De Floriani, L., Feature Extraction from Boundary Models of Three-Dimensional Objects, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 11, No. 8, pp.785-798, 1989.

[Gho02] Ghosh, A. and Tsutsui, S., editors, Advances in Evolutionary Computing, Natural Computing Series, Springer, 2002.

[Gol89] Goldberg, D.E., Genetic algorithms in search, optimization and machine learning, Addison-Wesley, Reading, Massachusetts, 1989.

[Jos88] Joshi, S., Chang, T.C., Graph-based heuristics for recognition of machined features, Computer-Aided Design, Vol. 20, No.2, pp.58-66, 1988.

[Kim92] Kim, Y.S., Recognition of form features using convex decomposition, Computer-Aided Design, Vol. 24, No. 9, pp. 461-476, 1992.

[Lan05] Langerak, T.R., Vergeest, J.S.M., Song, Y., Parameterising styling lines for reverse design using free form shape analysis.", Proceedings of IDETC/CIE, ASME 2005.

[Nez97] Nezis, K., Vosniakos, G., Recognizing 2½D shape features using a neural network and heuristics, Computer-Aided Design, Vol. 29, No. 7, pp. 523-539, 1997.

[Pal05] Pal, P., Tigga, A.M., Kumar, A., Feature extraction from large CAD databases using genetic algorithm, Computer-Aided Design, Vol. 37, No. 5, pp 545-558, 2005.

[Pra92] Prabhakar, S., Henderson, M.R., Automatic form-feature recognition using neural-network-based techniques on boundary representations of solid models, Computer-Aided Design, Vol. 24, No. 7, pp. 381-393, 1992.

[Pre02] Press, W.H., Vetterling, W.T., Teukolsky, S.A., Flannery, B.P., Numerical Recipes in C++: the art of scientific computing, Cambridge University Press, 2002.

[Sak95] Sakurai, H., Volume decomposition and feature recognition: part 1 – polyhedral objects, Computer-Aided Design, Vol. 27, No. 11, pp. 833-843, 1995.

[Sak96] Sakurai, H., Dave, P., Volume decomposition and feature recognition: part 2 – curved objects, Computer-Aided Design, Vol. 28, No. 6, pp. 517-537, 1996.

[Shah01] Shah, J.J., Anderson, D., Kim, Y.S., Joshi, S., A discourse on geometric feature recognition from CAD models, Journal of computing and information science in engineering, Vol 1, pp. 41-51, 2001.

[Song05] Song, Y., Vergeest, J.S.M., Bronsvoort, W.F., Fitting and manipulating freeform shapes using templates, Journal of computing and information science in engineering, Vol 5, No. 2, pp. 86-95, 2005.

[Sub95] Subrahmanyam, S., Wozny, M., An overview of automatic feature recognition techniques for computer-aided process planning, Computers in industry, Vol. 26, pp. 1-21, 1995.

[Tho99] Thompson, W.B., Owen, J.C., de St. Germain, H.J., Stark, S.R., Henderson, T.C., Feature-based reverse engineering of mechanical parts, IEEE Transactions on robotics and automation, Vol. 15, No. 1, 1999.

[Van93] Vandenbrande, J.H., Requicha, A., Spatial reasoning for the automatic recognition of machinable features in solid models, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 15, No. 12, pp 1269-1285, 1993.

[Van94] Vandenbrande, J.H., Requicha, A., Geometric computation for the recognition of spatially interacting machining features, In: Advances in feature based manufacturing, Shah, J., Mäntylä, M, Nau, D., eds. Elsevier Science, New York, pp. 39-63, 1994.

[Woo82] Woo, T., Feature extraction by volume decomposition, Proc. Conf. CAD/CAM Technology in Mechanical Engineering, Cambridge, MA, USA, March 1982.

 ISBN 978-80-86943-98-5

Interactive Distributed Translucent Volume Rendering

Balázs Domonkos Balázs Csébfalvi
Department of Control Engineering and Information Technology
Budapest University of Technology and Economics
Magyar tudósok krt. 2.
H-1117, Budapest, Hungary
domonkos@ik.bme.hu, cseb@iit.bme.hu

ABSTRACT

Translucent volume rendering is a robust and efficient direct volume-rendering technique for capturing optical effects, like subsurface scattering, translucency, and volumetric shadows. However, due to the limited computing and memory resources of the recent consumer graphics hardware, high-resolution volume data can still hardly be interactively visualized by this method. In this paper we present the theoretical aspects and implementation details of a parallelization scheme for translucent volume rendering. Our method is a three-pass parallel rendering algorithm with parallel compositing, based on object-space distribution of the data among the rendering nodes. In the first pass the 2D shadow maps are computed and sent to the effected nodes. In the second pass the nodes render their associated subvolumes by sequential translucent volume rendering. The generated framelets are then visualized by a dedicated display node in the third pass.

Keywords: Volume Rendering, Volumetric Shadows, Parallel and Distributed Graphics

1 INTRODUCTION

Using traditional direct volume visualization, the classical volume-rendering integral is numerically computed by evaluating finite number of samples along the viewing rays [Lev88]. Optical properties, like color and opacity are assigned to the samples by mapping the density and optionally the gradient magnitude with a transfer function. The color samples are shaded according to the normalized direction of the estimated gradient, which is treated as a normal of an isosurface. In a nearly homogeneous region, however, the variation of the densities is presumably due to the noisy data acquisition. Therefore the gradient estimation yields stochastic normal directions in the originally homogeneous regions. As there are no well-defined isosurfaces in these regions, the evaluation of a local shading model is not physically plausible. This problem is usually avoided by modulating the opacities by the gradient magnitude [Lev88], which enhances the well defined isosurfaces contained in the volume. Another drawback of the classical direct volume rendering model is that it relies on accurately estimated gradient directions. However, the gradient is usually calculated from quantized density values, so it can represent only a limited number of surface normals. Furthermore, the ideal gradient estimation cannot be efficiently implemented, therefore it is only approximated in practical applications. Because of these two reasons, images rendered by the traditional direct volume-rendering approach typically contain staircase artifacts.

Translucent volume rendering [KPHE02], which is based on a fundamentally different optical model, does not rely on estimated gradients at all. In this case, the colors are also assigned to the samples by a transfer function, but they are not shaded by evaluating an explicit local shading model. Instead, the color of each sample is multiplied by the intensity of an attenuated light ray coming from the light source into the given sample position. Furthermore, with a Gaussian perturbation, this approach can also be used for a rough approximation of forward scattering. Despite its robustness and optical modeling potential, the literature on translucent volume rendering is relatively narrow. In this paper we aim at an efficient parallel implementation scheme for translucent volume rendering of large-scale volumetric data sets. For the classical direct volume rendering of high-resolution data sets several researchers proposed efficient hierarchical data distribution and parallelization schemes. According to our best knowledge, however, a parallel translucent volume rendering algorithm has not been published yet.

In Section 2 the previous work related to distributed and parallel volume rendering is reviewed. In Section 3 we briefly overview the traditional sequential implementation of translucent volume rendering on a single GPU. Our parallelization scheme and its implementation are presented in Section 4 and Section 5 respectively. Images generated by our algorithm and the performance measurements are reported in Section 6,

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 2007
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

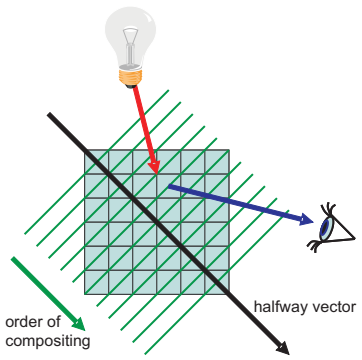


Figure 1: Translucent volume rendering on a single GPU using back-to-front compositing.

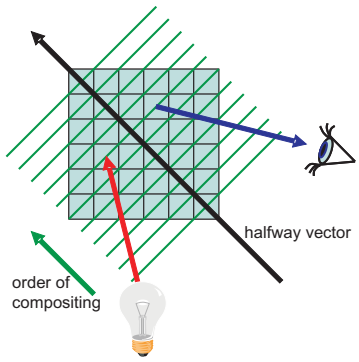


Figure 2: Translucent volume rendering on a single GPU using front-to-back compositing.

while in Section 7, we summarize the contribution of this paper.

2 RELATED WORK

There are four fundamentally different approaches for direct volume rendering: ray casting [Lev88], splatting [Wes90], shear-warp factorization [LL94], and texture mapping [CCF94, WE98]. The sequential or single-processor implementations of these methods are usually used to visualize data sets of moderate resolution. Practical data sets, however, continue to drastically increase in size, therefore different parallelization and data distribution schemes have been proposed for all the four basic algorithms.

Ray casting, which can produce the highest quality of rendered images, has been implemented on different architectures using either image-space or object-space partitioning [LY96, MPH93, MPHK94, BIPS00, PTT98, RPS99]. Similarly, the classical object-order splatting technique has also been adapted to multiprocessor environments [Elv92, JG95, LWM97]. The shear-warp algorithm, which had been originally proposed as a fast software implementation of direct volume rendering, was parallelized on an SGI system [Lac96]. As most of the recent consumer graphics cards supports 3D texture mapping, the texture-slicing approach became one of the most popular volume-rendering techniques. Nevertheless, its most important drawback is that due to limited texture memory large-scale volume data cannot be rendered without swapping subvolumes between the main memory and the local texture memory [GWGS02]. In this case, the bottleneck is the bandwidth of data transferring rather than the performance of the GPU. Therefore implementations for parallel architectures have been proposed by several researchers to overcome this limitation [KMM01, MHE01, GPH04].

The previously published parallel systems usually support only the classical volume-rendering model. However, a more sophisticated optical model, which includes volumetric shadows and forward scattering, requires a more complicated communication between the parallel processing units. In this paper we adapt the sequential translucent volume rendering method to parallel computing nodes. We demonstrate that, using a static data distribution scheme and parallel image compositing with high-speed communication channels, intermediate image data can be efficiently transferred between the nodes. As a consequence, the communication overhead is negligible compared to the rendering cost.

3 TRANSLUCENT VOLUME RENDERING

Translucent volume rendering can be efficiently implemented exploiting the 3D texture mapping capability of recent graphics cards. Unlike traditional slice-based direct volume rendering techniques, in this case volumetric shadows are calculated simultaneously with the compositing of the resampling slices. In order to avoid the computation of a 3D shadow map, the slicing is performed perpendicular to the halfway vector between the viewing direction and the direction of the light source as illustrated in Figure 1. For each pixel covered by the projection of a slice, the intensity of the light has to be determined, which reaches the intersection point between the slice and the corresponding viewing ray. Therefore each slice is simultaneously projected onto a plane perpendicular to the direction of the light source. In this way, a 2D shadow map can be maintained, which corresponds to the current stage of compositing. Whenever a pixel is processed in the pixel shader it is determined where the corresponding intersection point is projected onto the current shadow map and its color is modulated accordingly. As the resolution of the shadow map is the same as the resolution of the frame buffer, accurate volumetric shadows can be calculated by this method. If the angle between the viewing direction $\mathbf{V}$ and the direction of the light source $\mathbf{L}$ is less than 90 degrees then the halfway vector $\mathbf{H}$ is calculated as $\mathbf{H} = (\mathbf{L}+\mathbf{V})/2$ and a back-to-front compositing is performed (see Figure 1). Otherwise the halfway vector is

 ISBN 978-80-86943-98-5

calculated as $\mathbf{H} = (\mathbf{L} - \mathbf{V})/2$ and a front-to-back compositing is performed (see Figure 2).

4 DISTRIBUTED TRANSLUCENT VOLUME RENDERING

As our major goal is to interactively render large-scale data sets using the translucent shading model, the basic algorithm is adapted to a parallel computing environment. Parallelization schemes can be classified according to the type of entities, which are simultaneously processed. Single-threaded software renderers take graphics primitives one after another and the pixels corresponding to these primitives are also processed sequentially. In contrast, recent graphics cards have multiple graphics pipelines, therefore more vertices and pixels can be processed at the same time. Pixel-based parallelization can also be performed when multiple graphics cards are used for creating tiles of the overall output image and the rendering queue is branched into multiple pipes. On the other hand, when the data is divided in an initialization step, multiple subsets of graphics primitives and subvolumes can be processed at the same time.

In our case the original volumetric data is decomposed into subvolume blocks using axis-aligned subdivision. These blocks are distributed among the computing nodes. This static data distribution scheme is more favorable than pixel-level partitioning, because of two reasons. (1) To create an equivalent rendering model, initial shadow maps are needed to start the rendering of a subset of the volume. This inter-node shadow communication can be more easily performed using a fixed object-space subdivision rather than image-space decomposition, which deals with non-axis-aligned connection surfaces. (2) Furthermore the nodes can efficiently render only subvolumes of moderate resolution without swapping because of their limited texture memory.

This object-parallel approach needs compositing the subsets of pixels corresponding to different subvolumes. This procedure involves processing of alpha and depth values for each pixel, therefore compositing can be a bottleneck of the overall rendering system and unsuitable for interactive applications. However, when the compositing is also done in parallel, interactive compositing is possible. There are several algorithms providing parallel image compositing on multiprocessor architectures including direct send, parallel pipeline [LRN96], and binary swap [MPHK94].

In our approach a three-pass, object-parallel algorithm was used with parallel pipeline compositing. It performs the following steps:

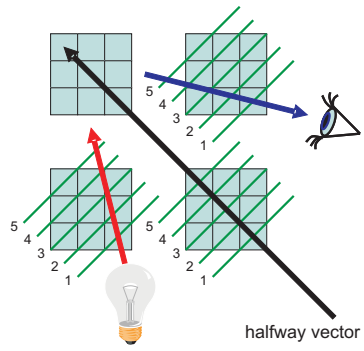


Figure 3: Parallel calculation of 2D shadow maps on the separate nodes. The numbers represent the different time steps.

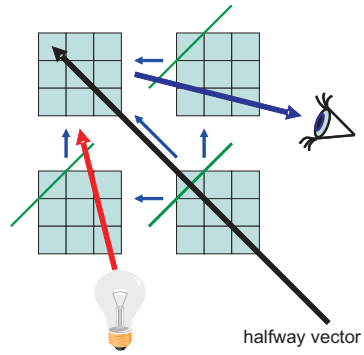


Figure 4: Sharing the 2D shadow maps with the effected nodes for parallel compositing.

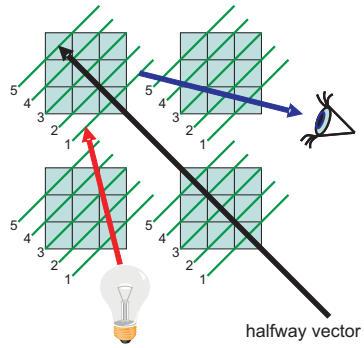


Figure 5: Parallel translucent volume rendering on the separate nodes. The numbers represent the different time steps.

1. pass: Each node computes its 2D shadow map and shares it with the effected nodes for parallel compositing.

2. pass: After compositing the received 2D shadow maps, each node performs translucent volume rendering as in the basic algorithm. The images of subvolumes are shared among all nodes.

3. pass: The portions of the final image are also composited in parallel and sent to the display node.

The first step is necessary, because each node needs an initial shadow map in order to start its effective rendering process. This map comes from the composited

 ISBN 978-80-86943-98-5

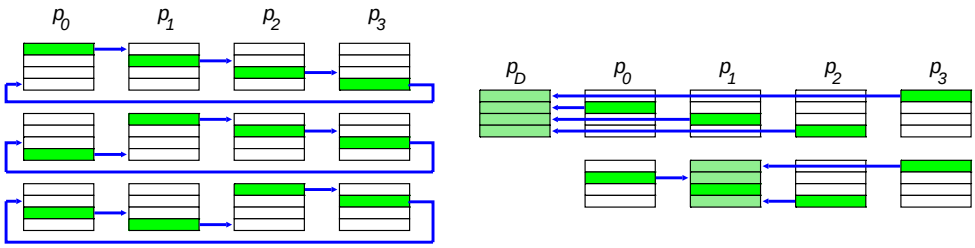


Figure 6: Parallel pipeline algorithm on distributed memory architectures. Left: framelet transfer for four compositing processes performed in $N-1$ steps, where N is the number of compositors. Right: collecting final framelets for an external (display) process or for an internal process (compositing shadow maps). This is performed in one step [LRN96].

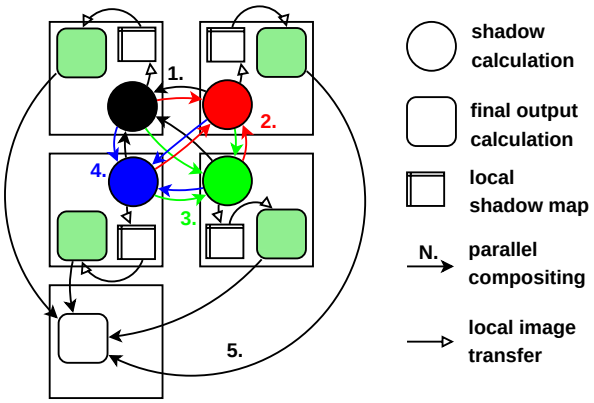


Figure 7: Parallel translucent volume rendering scheme for four rendering-compositing nodes and a composting-only display node. Shadow map calculation requires four parallel compositing contexts while a single context is needed for final image compositing. The participants of the final image context are fixed, while the members of the shadow maps are dynamically calculated based on the orientation of the volume. Note that the filled arrows point from the rendering sources to the destinations.

```
global p_i, N

function parallel_pipeline(i, target, frame) {
  def f_k ← frame.sub_image(0,k*frame.height/N,
          frame.width, (k+1)*frame.height/N−1)
  p_next ← p_{i+1} mod N
  p_prev ← p_{i−1} mod N

  for j ← 0, ..., N−2
    f_send ← f_{i−j} mod N
    f_recv ← f_{i−j−1} mod N
    send f_send → p_next
    receive f_recv ← p_prev
    compose f_recv with f_send

  if i ≠ target
    send f_i →p_target
  else
    for j ← 0, ..., N−1 (j ≠ i)
      receive f_j ← p_j
}
```

Listing 1: Pseudo-code of the parallel pipeline algorithm on distributed memory architectures for process p_i. The variables are coded as follows: $p_0 \ldots p_{N-1}$ are the compositing processors, $f_0 \ldots f_{N-1}$ are the image framelets, and *target* is the index of the target process.

shadow maps produced by the nodes associated to the covering subvolumes. Rendering these maps can be performed very efficiently on the graphics hardware, since only one multiplication has to be executed per pixel in the pixel shader code. Moreover, the nodes can simultaneously generate the shadow maps of their corresponding subvolumes, as illustrated in Figure 3, without waiting for each other. After having the shadow maps calculated they are shared with the effected nodes for parallel compositing (see Figure 4). In the second step, the nodes first have to composite the received images to produce an initial shadow map for the rendering. Depth information is not required here, as only an accumulated light attenuation needs to be evaluated for each pixel. Afterwards the nodes simultaneously perform the traditional translucent volume rendering for their assigned subvolumes (see Figure 5). The resulting framelets are split up, composited simultaneously, and the portions are sent to a dedicated node, which is responsible for displaying. In this third step depth-sorting is necessary before compositing, since the alpha-blending evaluation is order-dependent.

5 IMPLEMENTATION

We implemented our algorithm on a GPU cluster, a shared memory parallel rendering and compositing environment using the ParaComp[1] library. This software solution uses the parallel pipeline compositing algorithm consisting of two parts (see Figure 6 and Listing 1). The images to be composited are divided into

[1] Parallel Compositing library specified for multiprocessor systems by Hewlett-Packard in collaboration with Computational Engineering International

 ISBN 978-80-86943-98-5

N framelets, which is the number of the compositing processes. In the first part of the algorithm these framelets flow around through each node in $N-1$ steps, each consisting of a compositing and a communication stage. After $N-1$ steps each processor will have a fully composited portion of the final frame. The framelets are collected for an external display node or for an internal node in the second part in one step. The clear benefit of this compositing scheme is that the amount of data transferred on the network is independent of the number of compositing processes.

The detailed scheme of our algorithm is illustrated in Figure 7. Each rendering-compositing node being responsible for a subvolume has two separate processes and a local shadow buffer. The shadow calculation process renders the shadow image of the corresponding subvolume and belongs to multiple compositing contexts. The number of these contexts equals to the number of rendering nodes, which is four in Figure 7. Each compositing context has an explicit target process: the one which will use the composited shadow image. The other processes provide framelets to the context if the subvolumes of them occlude the subvolume corresponding to the target process. The composited shadow map is the input of another process of the node, which performs the final output calculation. The final image is composited in an additional context similarly in parallel – this is the fifth context in Figure 7. The parallel image compositing is illustrated with filled arrows while the empty arrows stand for direct local image transfer.

It follows that $N+1$ frames have to be composited applying N parallel rendering-compositing nodes. According to this scheme, for interactive rendering the algorithm could be optimal when the rendering and compositing time of the shadow map calculation equals to the time required for producing the final output.

Node-level aspects

As a bottleneck of our algorithm could be the efficiency of the sequential translucent volume rendering performed on the separate nodes, we optimized the fragment programs corresponding to this pass (see Listing 2). In order to perform high-precision compositing, we use alternating floating-point render targets or ping-pong buffering [KPHE02]. The previous state of the frame buffer is stored in a 2D texture denoted by `frameBuffer`. In the first step the color of the current pixel (`colorIn`) is read from this 2D texture. As the opacity channel represents the accumulated opacity, we can terminate the execution of the fragment program if it is already higher than a predefined threshold. This is similar to the well-known early ray termination [DH92].

The volumetric data is loaded into the texture memory as a 3D density array. The trilinearly interpolated density values are used for addressing a look-up table representing the current transfer function. Using this post-classification approach, the transfer function can be interactively modified on the fly. If the opacity of a sample (`color.a`) read from the look-up table is less than a predefined threshold, its contribution to the current pixel is negligible. Therefore the execution is terminated in this case as well. For shading the colors assigned to the samples, the attenuation of the light coming from the light source to the given sample has to be calculated. This information is stored in a 2D texture map denoted by `shadowMap`. However, to read the appropriate pixel of the shadow map, the object-space position of each sample has to be projected onto an image plane perpendicular to the direction of the light source. This task can be shared between the vertex shader and the pixel shader. The vertex shader performs the projection for the vertices of the proxy geometry and the result is stored in texture coordinates assigned to the vertices according to Listing 3. First the vertex positions are transformed by the model-view and projection transformations. The role of `TexTrans` is to calculate the normalized 3D texture coordinates from the position of the given vertex. Transformations `LightView` and `LightProj` are initialized according to the position of the light source, and they are used to map the vertex position onto an image plane perpendicular to the direction of the light source.

It is assumed that the graphics hardware performs perspectively correct interpolation of the texture coordinates. However, `IN.TEX1` in the pixel shader represents the homogeneous coordinates of a sample point projected onto an image plane perpendicular to the direction of the light source. To calculate the corresponding Cartesian screen coordinates, a homogeneous division has to be performed in the pixel shader and afterwards the appropriate pixel position in the shadow map is calculated by translation and scaling operations. From this pixel position the intensity of the attenuated light is read (`light`) and the color of the current sample is modulated by this value. The rest of the fragment program is just responsible for the usual evaluation of front-to-back compositing. The fragment program for the back-to-front evaluation is almost the same, only the compositing operations are different.

For the simultaneous compositing of the shadow map another pixel shader is used. In each iteration step, first the current resampling slice is projected onto an image plane perpendicular to the viewing direction, and processed by the previously described fragment program. Afterwards the same slice is projected onto an image plane perpendicular to the direction of the light source, in order to update the shadow map for the next iteration (see Listing 4). Similarly to the previous case the volume data is available in a 3D texture map, while the transfer function is stored in a look-up table represented by a 1D texture. Here we also use alternating

 ISBN 978-80-86943-98-5

```
fragout main(vf30 IN,
  uniform float halfWidth,
  uniform float halfHeight,
  uniform sampler3D data,
  uniform sampler1D transferFunction,
  uniform samplerRECT frameBuffer,
  uniform samplerRECT shadowMap)
{
  fragout OUT;

  float4 colorIn =
    f4texRECT(frameBuffer, IN.WPOS.xy);
  if(colorIn.a > 0.99) discard;

  float4 color =
    f4tex1D(transferFunction,
            f4tex3D(data, IN.TEX0).a);
  if(color.a < 0.01) discard;

  float4 position = IN.TEX1;
  position.xy /= position.w;
  position.x = (position.x+1.0)*halfWidth;
  position.y = (position.y+1.0)*halfHeight;
  float light = f4texRECT(shadowMap,
                          position.xy).a;

  float4 colorOut = colorIn;
  if(light > 0.01) colorOut.rgb += color.rgb
    * (color.a * (1.0 - colorIn.a) * light);
  colorOut.a += color.a - colorIn.a * color.a;
  OUT.col = colorOut;

  return OUT;
}
```

Listing 2: Front-to-back compositing fragment shader code.

```
struct OUTPUT
{
  float4 HPosition     : POSITION;
  float4 TCoords0      : TEXCOORD0;
  float4 TCoords1      : TEXCOORD1;
};

OUTPUT main(float4 Position : POSITION,
  uniform float4x4 Proj,
  uniform float4x4 View,
  uniform float4x4 LightProj,
  uniform float4x4 LightView,
  uniform float4x4 TexTrans
  )
{
  OUTPUT output;

  float4 position  = mul(View, Position);
  output.HPosition = mul(Proj, position);
  output.TCoords0  = mul(TexTrans, Position);
  float4 light     = mul(LightView, Position);
  output.TCoords1  = mul(LightProj, light);

  return output;
}
```

Listing 3: Common vertex shader code of the three passes.

```
fragout main(vf30 IN,
  uniform sampler3D data,
  uniform sampler1D transferFunction,
  uniform samplerRECT shadowMap)
{
  fragout OUT;

  float transparency =
    f4texRECT(shadowMap, IN.WPOS.xy).a;
  if(transparency < 0.01) discard;

  float alpha =
    f4tex1D(transferFunction,
            f4tex3D(data, IN.TEX0).a).a;
  if(alpha < 0.01) discard;

  OUT.col.a = transparency * (1.0 - alpha);
  return OUT;
}
```

Listing 4: Shadow map compositing fragment shader code.

floating-point render targets for high-precision compositing. Therefore the previous state of the shadow map is obtained as an input 2D texture by the fragment program. In fact, the shadow map contains a transparency value in each pixel, which is first read by a 2D texture fetch. If this value is already less than a small predefined threshold, the current sample practically does not contribute to the given pixel, therefore the execution of the program is terminated. Otherwise the opacity of the current sample is read from the look-up table. If this opacity value is under the threshold, the sample practically does not have an influence onto the shadow map in this case either, so the execution is terminated as well. Generally, the transparency of the sample is calculated and it is multiplied by the previous value of the given pixel. This fragment program is used in the rendering pass and in the first initialization pass as well, when each node calculates the shadow map of its associated subvolume. Note that it is simpler than the one applied for the frame buffer compositing, therefore it is performed more efficiently. That is the reason why the cost of the initialization pass is relatively low compared to that of the second rendering pass.

6 RESULTS

For our experiments we used a Hewlett-Packard "Scalable Visualization Array" consisting of five computing nodes. Each node has a dual-core AMD Opteron 246 processor, an nVidia Quadro FX3450 graphics controller, and an InfiniBand network adapter.

To illustrate the scalability of our volume rendering system, configurations of one up to four rendering nodes were investigated for four different data sets. Each setup also contained an additional display node. The data sets were visualized on a viewport of resolution 512^2. The volumes were rotated around two

 ISBN 978-80-86943-98-5

Number of nodes	Average frame rate	Average shadow fps	Shadow overhead
1	8.15	-	0%
2	9.67	13.53	28.0%
3	11.02	15.73	29.4%
4	12.67	18.29	30.1%

Table 1: Scalability results using the human head data set (256x256x159). Average frame rate, average shadow frame rate for all compositing contexts, and relative shadow computation and communication overhead.

Number of nodes	Average frame rate	Average shadow fps	Shadow overhead
1	6.12	-	0%
2	5.73	19.59	29.3%
3	6.74	21.60	31.2%
4	7.67	21.87	35.1%

Table 2: Scalability results using the frog data set (500x470x136).

Number of nodes	Average frame rate	Average shadow fps	Shadow overhead
1	4.62	-	0%
2	5.08	17.54	29.0%
3	7.35	25.29	29.1%
4	9.71	27.56	35.2%

Table 3: Scalability results using the Christmas tree data set (512x499x512).

Number of nodes	Average frame rate	Average shadow fps	Shadow overhead
1	7.78	-	0%
2	8.68	35.03	24.8%
3	10.03	35.70	28.1%
4	11.95	40.99	29.1%

Table 4: Scalability results using the stag beetle data set (416x416x247).

axes, in order to shuffle continuously the sorting order of the subvolumes. The average output frame rates, the average shadow calculation frame rates, and the relative shadow overheads are shown in Tables 1-4. The shadow calculation overhead is the ratio of the shadow rendering-compositing time and the overall rendering time. Note that the efficiency of the 2-node distribution might appear to be poor compared to the 1-node configuration. This illusory retrogression comes from the incoming shadow calculation overhead, which is not present in the 1-node setup. According to our measurements, the shadow computation overhead is around 30% including the communication on our hardware, however it is apparently growing by increasing the number of the partner nodes. The images of medical data sets rendered using 4 rendering-compositing nodes are shown in Figure 8.

7 CONCLUSION AND FUTURE WORK

In this paper we have demonstrated that using an object-space partitioning and high-speed communication channels between the rendering nodes, parallel volume rendering can be efficiently implemented on distributed memory architectures using even a more sophisticated optical model, which also takes light attenuation in translucent materials into account. Although our algorithm requires an initialization step to calculate, transfer, and composite the 2D shadow maps for each subvolume, due to the simple order-independent compositing operation this step can be efficiently performed. Furthermore, the high-speed channels are exploited to handle the additionally required communication overhead, which is theoretically not limited by the computing nodes when the compositing is also performed in parallel. Because of these two reasons the performance of our parallel translucent volume-rendering system is not set back by the additional computation and communication, therefore it can be improved by increasing the number of rendering nodes.

In our future work, we intend to further improve our algorithm. In the rendering phase each node has to wait until it receives all the necessary shadow maps. However, the nodes processing subvolumes shadowed by many other subvolumes have to wait longer, while the nodes corresponding to subvolumes closer to the light source can start rendering earlier. In order to improve load balancing, the calculation of consecutive frames can overlap, such that a node can render a framelet belonging to the next frame, while another node is still calculating a framelet belonging to the current frame.

ACKNOWLEDGEMENTS

This work has been supported by the Postdoctoral Fellowship Program of the Hungarian Ministry of Education, Hewlett-Packard, and the Hungarian National Office for Research and Technology.

REFERENCES

[BIPS00] C. Bajaj, I. Ihm, S. Park, and D. Song. Compression-Based Ray Casting of Very Large Volume Data in Distributed Environments. In *Proceedings of Fourth International Conference on High Performance Computing in the Asia-Pacific Region*, pages 720–725, 2000.

[CCF94] B. Cabral, N. Cam, and J. Foran. Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware. In *Proceedings of IEEE Symposium on Volume Visualization*, pages 91–98, 1994.

[DH92] J. Danskin and P. Hanrahan. Fast Algorithms for Volume Ray Tracing. In *Proceedings of Workshop on Volume Visualization*, pages 91–98, 1992.

[Elv92] T. Elvins. Volume Rendering on a Distributed Memory Parallel Computer. In *Proceedings of IEEE Visualization*, pages 93–98, 1992.

[GPH04] C. Gribble, S. G. Parker, and C. Hansen. Interactive volume rendering of large datasets using the silicon graphics Onyx4 visualization system. *TR No. UUCS-04-003, University of Utah School of Computing*, 2004.

[GWGS02] S. Guthe, M. Wand, J. Gonser, and W. Straßer. Interactive Rendering of Large Volume Data Sets. In *Proceedings of IEEE Visualization 2002*, pages 45–52, 2002.

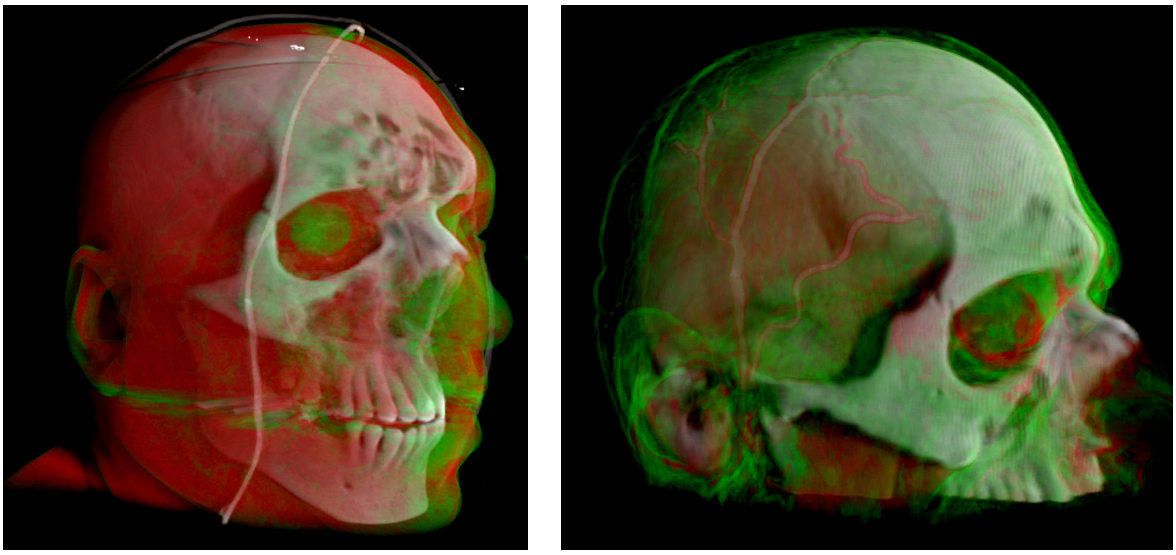


Figure 8: Images of medical data sets rendered by our parallel translucent volume rendering system.

[JG95] G. Johnson and J. Genetti. Medical Diagnosis using the Cray T3D. In *Proceedings of Parallel Rendering Symposium*, pages 70–77, 1995.

[KMM01] J. Kniss, P. McCormick, and A. McPherson. Interactive Texture-Based Volume Rendering for Large Data Sets. *IEEE Computer Graphics and Applications*, 21(4):52–61, 2001.

[KPHE02] J. Kniss, S. Premoze, C. Hansen, and D. Ebert. Interactive translucent volume rendering and procedural modeling. In *Proceedings of IEEE Visualization*, pages 109–116, 2002.

[Lac96] P. Lacroute. Analysis of a Parallel Volume Rendering System Based on the Shear-Warp Factorization. *IEEE Transactions on Visualization and Computer Graphics*, 2(3):218–231, 1996.

[Lev88] M. Levoy. Display of Surfaces from Volume Data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988.

[LL94] P. Lacroute and M. Levoy. Fast Volume Rendering using a Shear-Warp Factorization of the Viewing Transformation. *Computer Graphics (Proceedings of SIGGRAPH '94)*, pages 451–457, 1994.

[LRN96] Tong-Yee Lee, C. S. Raghavendra, and John B. Nicholas. Image Composition Schemes for Sort-Last Polygon Rendering on 2D Mesh Multicomputers. *IEEE Transactions on Visualization and Computer Graphics*, 2(3):202–217, 1996.

[LWM97] P. P. Li, S. Whitman, and R. Mendoza. ParVox - A Parallel Splatting Volume Rendering System for Distributed Visualization. In *Proceedings of IEEE Symposium on Parallel Rendering*, pages 7–14, 1997.

[LY96] A. Law and R. Yagel. Multi-frame Thrashless Ray Casting With Advancing Ray-Front. In *Proceedings of Graphics Interface*, pages 70–77, 1996.

[MHE01] M. Magallon, M. Hopf, and T. Ertl. Parallel volume rendering using PC graphics hardware. In *Proceedings of Ninth Pacific Conference on Computer Graphics and Applications*, pages 384–389, 2001.

[MPH93] K. L. Ma, J. S. Painter, and C. D. Hansen. A Data Distributed, Parallel Algorithm for Ray-Traced Volume Rendering. In *Proceedings of Parallel Rendering Symposium*, pages 15–22, 1993.

[MPHK94] K. L. Ma, J. S. Painter, C. D. Hansen, and M. F. Krogh. Parallel Volume Rendering using Binary-Swap Compositing. *IEEE Computer Graphics and Applications*, 14(4):59–68, 1994.

[PTT98] M. E. Palmer, B. Totty, and S. Taylor. Ray Casting on Shared-Memory Architectures: Memory-Hierarchy Considerations in Volume Rendering. *IEEE Concurrency*, 6(1):20–35, 1998.

[RPS99] H. Ray, H. Pfister, and D. Silver. Ray Casting Architectures for Volume Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):210–223, 1999.

[WE98] R. Westermann and T. Ertl. Efficiently Using Graphics Hardware in Volume Rendering Applications. *Computer Graphics (Proceedings of SIGGRAPH '98)*, pages 169–176, 1998.

[Wes90] L. Westover. Footprint Evaluation for Volume Rendering. *Computer Graphics (Proceedings of SIGGRAPH '90)*, pages 144–153, 1990.

Smooth Subdivision Surfaces over Multiple Meshes

Przemyslaw Musialski
Bauhaus-University Weimar
Weimar, Germany
musialsk@uni-weimar.de

Robert F. Tobler
VRVis Research Center
Vienna, Austria
rft@vrvis.at

Stefan Maierhofer
VRVis Research Center
Vienna, Austria
sm@vrvis.at

ABSTRACT

Standard subdivision rules, such as Catmull-Clark and Loop allow the creation of smooth surfaces with C^2 continuity over almost the whole domain except at extraordinary vertices. Normally, the subdivision schemes are limited to one mesh and need special rules for handling the boundaries of the domain. This issue leads to complications in straightforward approaches to compose objects out of multiple joined meshes. We propose a new method for stitching control meshes at common faces. The new approach uses the stitching information to smoothly subdivide the meshes across the stitching edges, while maintaining the meshes as separate units in memory. This makes it possible to compose large, complex geometries using simple components, without the necessity to subdivide the complete mesh down to the same detail level.

Keywords: subdivision surfaces, level of detail, multiple meshes

1 INTRODUCTION

Complex objects are difficult to model and render with conventional, manual modeling techniques. In order to facilitate such tasks, different methods for modeling and representing objects have been developed over the years in Computer Graphics. Using the common boundary representation in 3D space, one possibility to reduce complexity in more sophisticated shapes is to create objects composed of several simpler pieces.

Combining this idea with the subdivision surface approach results in a method for generation of smooth and complex geometry over multiple polygonal meshes. Since these objects can be defined through piecewise simple, primitive geometric elements, procedural generation of complex objects by rule based composition of the basic shapes is possible [17].

In this paper we present a method for subdividing multiple, loosely joined meshes, which we call *stitched meshes*. The complete set of all stitched meshes serves as the control domain for further subdivision to achieve smooth geometric shapes. By using the standard Catmull-Clark scheme as described by DeRose [3] we are able to create meshes with C^2 continuity over almost the complete surface and C^1 continuity at extraordinary vertices respectively.

The advantage of this method is the fact that problem regions such as the boundaries of single meshes do not have to be treated separately by using modified subdivision rules but can be handled with the standard rules for interior regions. At the same time all partial meshes of the common surface remain independent of each other, and must not be subdivided to the same level.

In section 2 we will to give a brief overview of subdivision surfaces and point out the main problems that arise when multiple subdivided surfaces need to be joined. In sections 3 and 4 we will describe the generalized subdivision and our method based on joining multiple meshes on the coarsest subdivision level. Here we will introduce *stitch faces*, which serve as a connection between different meshes and discuss the *subdivision to different levels* as a level–of–detail approach. Finally we will present some results of our method applied on models composed of several meshes.

2 RELATED WORK

Mesh subdivision is a technique for generating smooth surfaces that has been introduced quite some time ago by Catmull and Clark [2] and Doo and Sabin [4]. For a long time the theoretical foundation of the subdivision process was not as thorough as for other mod-

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.

 ISBN 978-80-86943-98-5

eling techniques such as BSplines and the more general NURBS, and thus it took a while for subdivision methods to become widely known and used. Recently this has been rectified by the introduction of methods to analyse and evaluate subdivision surfaces at any point [12],[14], a method for extending subdivision surfaces for emulating NURBS [13], the addition of normal control to subdivision surfaces [1], and a method to closely approximate Catmull-Clark subdivision surfaces using BSpline patches [11]. A number of other extensions to subdivision surfaces [3], [7], [8], [5], [6] have established them as the modeling tool of choice for generating topologically complex, smooth surfaces.

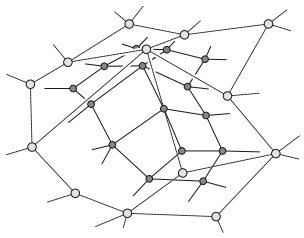


Figure 1: A step in the Catmull-Clark subdivision scheme.

The main goal of most subdivision surface techniques is the use of recursive refinement to obtain smooth surfaces out of arbitrary polygonal input. One significant disadvantage is their undefined behavior on the boundaries of geometric domains. In order to create complex shapes often several meshes have to be joined at their borders to gain satisfying visual results with sufficient complexity. Here, often continuity problems arise since smoothly joining the meshes at the boundaries is a non-trivial task. To overcome these drawbacks Biermann *et al.* [1] introduced a method for controlling the boundaries and their normals in order to join independent meshes, but his approach involves again different rules for special cases.

Our approach uses standard Catmull-Clark rules which are applied to multiple meshes, by joining the meshes to a virtual single mesh.

3 GENERALIZED SUBDIVISION SURFACES

The standard subdivision process starts out with a mesh $M^{(0)}$ composed of vertices, edges, and faces that serves as the base for a sequence of refined meshes $M^{(0)}, M^{(1)}, M^{(2)}, \ldots$ which converges to a limit surface, called the subdivision surface.

The process for generating submesh $M^{(n+1)}$ of a specific mesh $M^{(n)}$ in the sequence can be split up into two operations. The first operation, which we will call *mesh refinement*, is the logical introduction of all the vertices in the submesh. This operation yields all the connectivity information for the vertices of the submesh without specifying the positions of these newly introduced vertices. The second operation, which we will call *vertex placement*, is the computation of the actual vertex positions. Standard subdivision schemes use specific rules for generating the new vertex positions, that ensure that the limit surface of the subdivision process satisfies certain continuity constraints, e.g. C^1 or C^2 continuity.

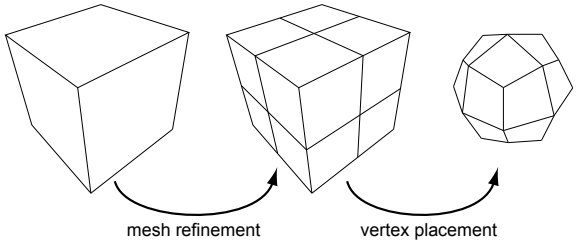


Figure 2: Generalized subdivision.

To obtain maximum flexibility in generating subdivision surfaces, we propose to separate the two operations of *mesh refinement* and *vertex placement*, and make it possible to independently specify both of these operations.

4 SUBDIVISION OVER MULTIPLE MESHES

General subdivision rules ensure C^2 continuity between interior mesh polygons and C^1 at extraordinary vertices. At the boundary of a mesh different subdivision rules have to be applied to achieve considerable smoothness. To connect two meshes at their borders, both pieces have to have exactly the same border definition. Unfortunately, depending on the chosen subdivision scheme, it is not always straightforward to do as mentioned in section 2.

To handle these problems we suggest to stitch meshes at selected faces (referred as *stitch faces*) prior to the subdivision process with the effect that standard interior subdivision rules can be applied at the borders to a neighbor mesh. Furthermore, since the connection between the meshes is established basically on the logical level, only the *vertex placement* step has to be applied over multiple meshes — the topological subdivision proceeds independently.

To achieve the connection, only small amounts of information about the neighbors have to be held in each mesh additionally. Since control meshes are rather sparse and easy to control, common borders between two can be constructed quite easily. Moreover, due to this simplicity, the stitch faces can be either user selected or automatically generated by routines to construct complex geometry as proposed by Tobler *et al.* in [17].

 ISBN 978-80-86943-98-5

Nevertheless, each of the partial meshes remain independent from any higher located common control surface and can be rendered, edited or subdivided separately.

4.1 Stitch Faces

A logical connection between two meshes can be established by defining *stitch faces* (Figure 3). Any of the adjacent meshes has to contain such a face pointing to its neighbor. Corresponding stitch faces have to be identical and tangent to each other; only the vertex order must be reversed. These regions can be seen as shared polygons which contain connection data bidirectionally, such that a way from a mesh to its neighbor and *vice versa* can be found in order to obtain all needed geometric information.

For the connection the following information about the neighbor mesh is kept in each stitch face:

- *reference to the neighbor mesh*
  via the reference the whole structure of the neighbor can be accessed
- *stitch face index in the neighbor mesh*
  with this index the corresponding face within the neighbor mesh can be identified
- *face–vertex–index in the stitched face*
  with this index the offset in the vertex order in the face can be specified, such that geometrically corresponding vertices are matched to each other

Stitch faces can be simple defined by adding the auxiliary information to particular elements i.e. in form of a hashtable, that uses the face-index as a key.

To keep track of the connection, all edges around the stitch face are marked. Whenever a marked edge is encountered during processing of a face, the respective

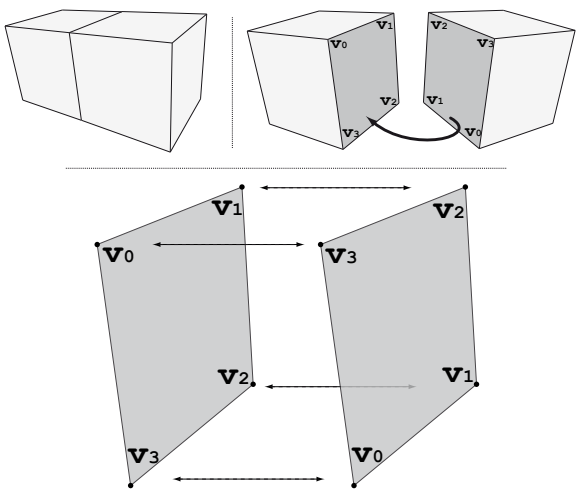


Figure 3: Top: Multiple stitched meshes, Bottom: Corresponding stitch faces

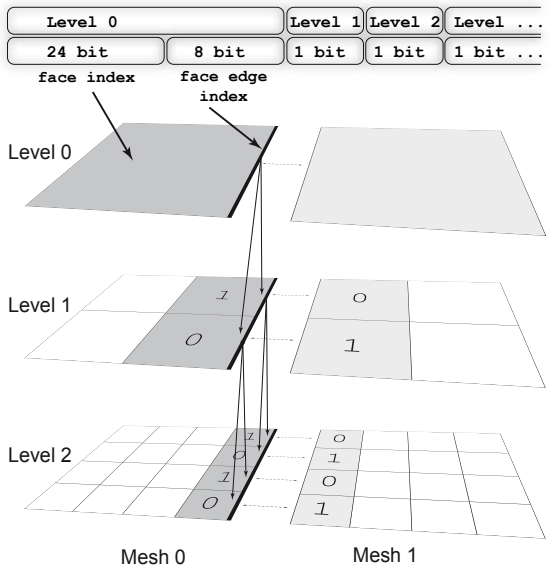


Figure 4: Encoding the subsequent face location path along a stitch edge.

stitch face can be accessed. Through the provided reference to an adjacent mesh and a face–vertex–index, a corresponding face in the neighbor mesh can be exactly detected. Now a standard subdivision rule can be applied at border edges and vertices and the geometric information behind the boundary is retrieved from the neighbor. The same procedure has to be applied on the opposite mesh, which finally produces a continuous surface as if there were no junction, yet both pieces are only loosely connected with a single reference between the control meshes.

4.2 Further Subdivision Steps

Subdivision rules are based on recursive refinement of given initial control meshes. Control meshes in our case contain connection information about each other, but these are not propagated to the following subdivision steps in the same manner. In fact, the topology of the subdivided meshes changes in comparison to its parent, since elements previously defined as stitch faces disappear in the subsequent mesh. The stitch faces have been introduced to simplify modeling the original base mesh and to keep an unambiguous connection between meshes as described in the section above, but they do not serve any geometric purpose. In the subdivided meshes, each stitch face is not present anymore and results in a topological hole. Since each hole fits exactly to a corresponding one in the neighbor mesh, the common surface defined by the subdivision process is continuous.

Due to this fact only the control mesh holds the necessary connection to its neighbors but the successive generated submeshes do not. We overcome this problem by keeping the whole subdivision hierarchy acces-

 ISBN 978-80-86943-98-5

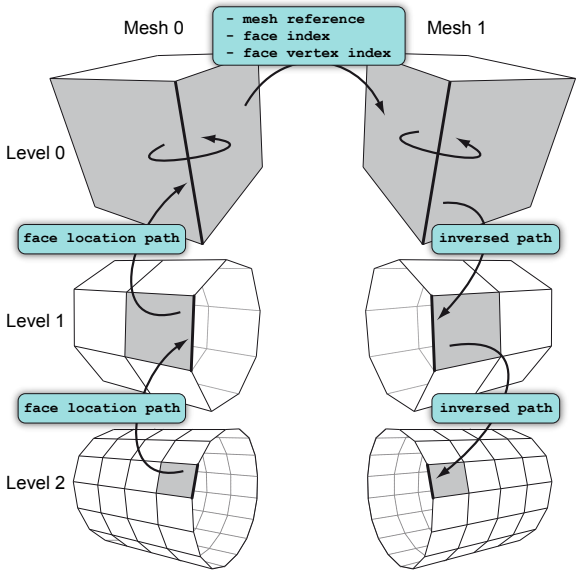


Figure 5: Traversing the subdivision hierarchy along the face location path.

sible in storage and by marking the edges around each hole as *stitch edges* (bold painted edges in Figure 4,5 and 6).

Since we are using Catmull-Clark rules, each quadrangle is successively subdivided into four pieces. In our approach corresponding faces of two stitched meshes touch each other at a single edge. Therefore at each subdivision level a single bit can be used to encode the location of a sub–edge along a parent edge. Concatenating the bits along the subdivision hierarchy, a path to each face along the stitch edge can be defined. The only exception provide the elements in the base level, where the face index and corresponding face–edge–index indicates an non–ambiguous stitch edge. In a memory conservative implementation it is not necessary to store this path, as it can always be computed on the fly. Figure 4 shows how this path is constructed.

For any face along the stitch edge this unique path allows to locate it in the hierarchy and furthermore, because of mirror symmetry, it is easily possible to locate the corresponding face in the subdivision hierarchy of the neighbor mesh by exchanging the level–zero information and simple bitwise inversion of the path as shown in Figures 4 and 5.

It should be mentioned that other subdivision methods, like i.e. Loop [15] can also be fitted in this or similar scheme.

4.3 Recursive Stitched Subdivision

The stitched subdivision method described above performs with two joined objects. Indeed there is no limitation in the number of adjacent elements which could be affected. Since every border face can hold as many unique paths as its edge count, more meshes could be connected to it. The procedure then recursively proceeds by moving from one mesh to the next as shown in Figure 6. In the showed case the top face of mesh M_2 holds the unique paths at the particular edges stitched to the two neighbors M_1 and M_3. In the corner with all three meshes the algorithm retrieves first the stitch face from M_2 to M_3, but since it is again a stitch face it proceeds recursively to next mesh in the same manner. The two stitch faces in M_2 and respectively one per M_1 and M_3 are removed after the first subdivision step.

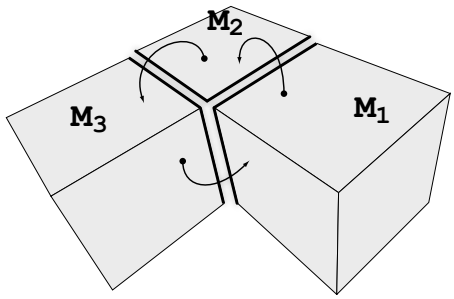


Figure 6: Subdivision over three meshes.

4.4 Subdivision to different levels

As mentioned, due to the two phase nature of our subdivision method, only the vertex placement operation needs access to neighboring meshes in a stitched mesh. As with all subdivision methods, the vertex positions of a new level are computed from the positions of the elements in the previous subdivision level. Thus a subdivision step of a single mesh can be performed, while all neighboring stitched meshes remain not subdivided (see Figure 7). This, of course is possible with only one step difference between adjacent meshes, which is still sufficient for a level–of–detail approach. In a rendering application this can be exploited as follows: only subdivide meshes that are currently viewed in a close up to a certain level, and remove subdivision levels that are not visible or not in the near field anymore (see Figure 15).

To solve the problem of discontinuity at a junction of two (or more) connected meshes at different resolutions, we propose to force the border vertices of the higher sampled mesh to their positions of the lower level. The newly generated edge–vertices in the higher level can be linearly placed in the middle of each edge, which produces T–triangles (see Figure 16, 17). While the meshes are basically joined, indeed, during the rendering small micro holes in the surface can still appear at the T–junctions. This issue can be enforced by introducing zero–area–triangles at the T–junctions similar as proposed in [10].

 ISBN 978-80-86943-98-5

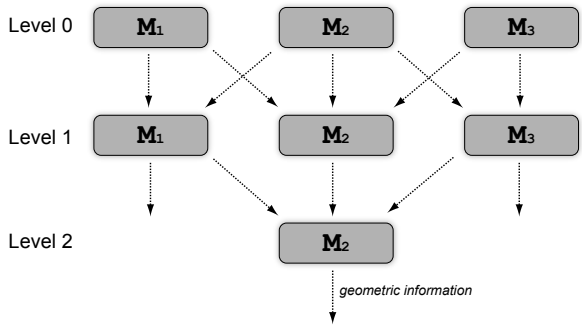


Figure 7: Different subdivision levels and geometric information flow.

5 RESULTS

We have compared the performance of our method on two scenes showing the same model. In the first one we used the model of a triple–cross built out of one coherent mesh. In the second scene we used a model composed of five pieces as shown in Figure 10.

All observed subdivision timings are presented in Figures 8 and 9 and were measured on a Intel Centrino Duo 1.83 Mhz machine with 1GB main memory and 2MB CPU–Cache.

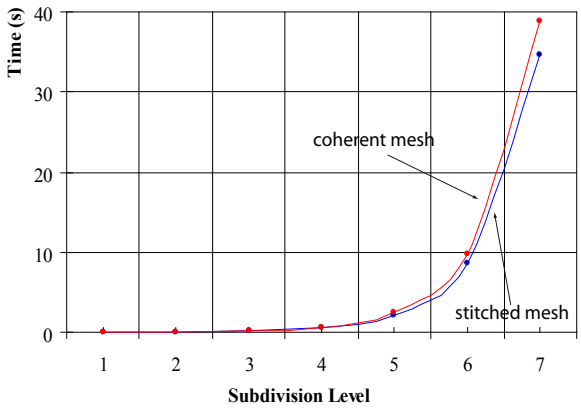
Figure 9 shows the comparison of the run times of both models. One can observe, that our approach has nearly exactly the same performance as the classical one of subdividing a complete mesh. Indeed, the stitched meshes were computed even faster in higher subdivision steps, probably due to cache effects.

	$M_1 \ldots M_4$		M_5		M_Σ	
	time(s)	space(kb)	time(s)	space(kb)	time(s)	space(kb)
$M^{(1)}$	0,002	1	0,005	1	0,014	5
$M^{(2)}$	0,007	2	0,013	4	0,042	12
$M^{(3)}$	0,025	8	0,049	16	0,148	48
$M^{(4)}$	0,092	31	0,185	63	0,563	187
$M^{(5)}$	0,362	124	0,725	249	2,183	745
$M^{(6)}$	1,443	494	2,865	989	8,657	2.965
$M^{(7)}$	5,765	1.972	11,518	3.944	34,560	11.832
Σ	7,696	2.633	15,36	5.266	46,167	15.798

Figure 8: Runtime and memory usage of the partial meshes $M_1 \ldots M_5$ and their sum M_Σ (see model in Figure 10).

6 CONCLUSION AND FUTURE WORK

We have implemented a new method for stitched meshes, that makes it possible to perform the subdivision of large, complex geometric objects in a piecewise fashion, while retaining the continuity properties of standard subdivision. Other necessary tasks like i.e. vertex normal interpolation or texture coordinate mapping as well as crease edges handling [3] can be facilitated by this method just as in standard subdivision. This is possible, because at the stitches the access to any necessary neighborhood information takes place just like in a standard mesh.



M_Σ	0,014	0,042	0,148	0,563	2,183	8,657	34,560
M	0,010	0,040	0,150	0,603	2,409	9,703	38,882

Figure 9: Runtime comparison between partial meshes M_Σ and coherent mesh M in seconds.

This new approach has been shown to have no significant performance impact on the actual subdivision process while opening up a number of optimization possibilities for level-of-detail rendering.

Additionally, the new method is optimally suited as a basis for procedural generation of complex objects. In procedural based subdivision, the stitch-faces correspond to symbols which are used in a L-system grammar that builds more complex objects.

We are currently working on a system for vegetation generation, that combines these two approaches in order to generate highly detailed plants models as well as on a GPU–supported implementation of this approach, which will make a real-time subdivision at higher resolutions possible.

ACKNOWLEDGMENTS

Most of this work has been done at the VRVis research center in Vienna, Austria, which is partially founded by the Austrian government research program Kplus. Parts of this work have also been funded by the Austrian Science Fund (FWF) under project P17260-N04.

REFERENCES

[1] Henning Biermann, Adi Levin, and Denis Zorin. Piecewise smooth subdivision surfaces with normal control. In *SIGGRAPH '00: Proceedings of the 27th*

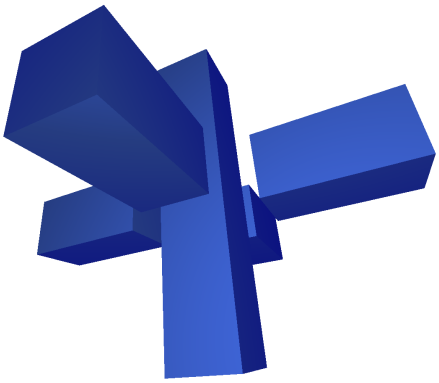


Figure 10: Test model at base level, composed of five meshes. We refer to them as $M_1 \ldots M_4$ to the four small pieces (each composed of six faces), M_5 to the bigger one (composed of 14 faces) and M_Σ to whole object.

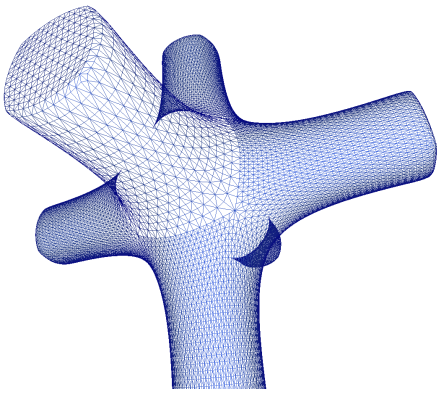


Figure 11: Stitched model from Figure 10 subdivided partially to 4th and 5th level.

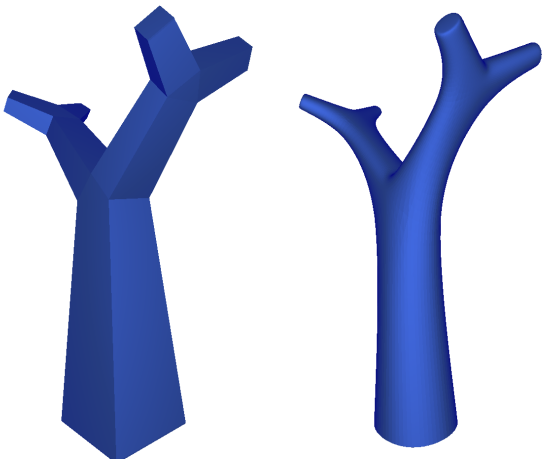


Figure 12: Tree trunk model composed of several simple meshes.

Figure 13: A simple hand model composed of 14 meshes.

Figure 14: Hand model subdivided without stitching information.

annual conference on Computer graphics and interactive techniques, pages 113–120, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[2] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10:350–355, September 1978.

[3] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 85–94, New York, NY, USA, 1998. ACM Press.

[4] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10:356–360, September 1978.

[5] Igor Guskov, Wim Sweldens, and Peter Schröder. Multiresolution signal processing for meshes. In Alyn Rockwood, editor, *Proceedings of the Conference on Computer Graphics (Siggraph99)*, pages 325–334, N.Y., August8–13 1999. ACM Press.

[6] Igor Guskov, Kiril Vidimče, Wim Sweldens, and Peter Schröder. Normal meshes. In *SIGGRAPH '00:*

 ISBN 978-80-86943-98-5

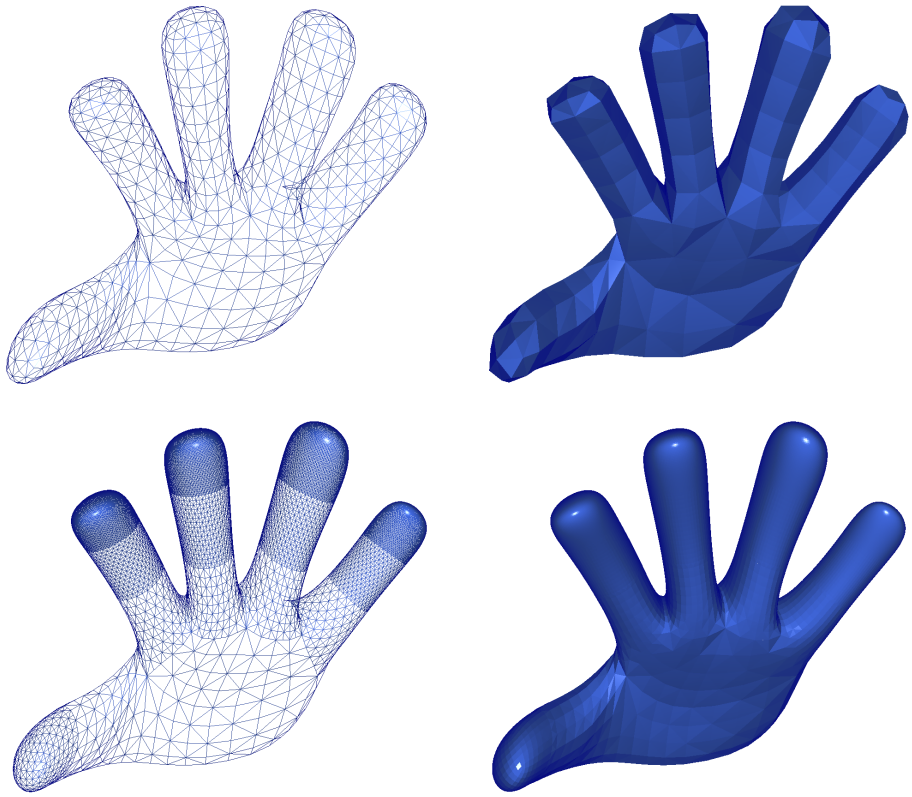


Figure 15: A simple hand model. Top: uniform subdivision. Bottom: multi–resolution subdivision with forced border vertices.

Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pages 95–102, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[7] Leif Kobbelt. $\sqrt{3}$ - subdivision. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 103–112, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[8] Aaron Lee, Henry Moreton, and Hugues Hoppe. Displaced subdivision surfaces. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 85–94, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[9] Charles Loop. Triangle mesh subdivision with bounded curvature and the convex hull property. Technical Report MSR-TR-2001-24, Microsoft Research (MSR), January 2001.

[10] Frank Losasso and Hugues Hoppe. Geometry clipmaps: terrain rendering using nested regular grids. *ACM Trans. Graph.*, 23(3):769–776, 2004.

[11] Jörg Peters. Patching catmull-clark meshes. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 255–258, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[12] Ulrich Reif. A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Geometric Design*, 12(2):153–174, 1995.

[13] Thomas W. Sederberg, Jianmin Zheng, David Sewell, and Malcolm Sabin. Non-uniform recursive subdivision surfaces. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 387–394, New York, NY, USA, 1998. ACM Press.

[14] Jos Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 395–404, New York, NY, USA, 1998. ACM Press.

[15] Jos Stam and Charles T. Loop. Quad/triangle subdivision. *Comput. Graph. Forum*, 22(1):79–86, 2003.

[16] Robert F. Tobler and Stefan Maierhofer. A mesh data structure for rendering and subdivision. In *WSCG 2006 - Short communication Proceedings, ISBN 80-86943-05-4*, pages 157–162, 2006.

[17] Robert F. Tobler, Stefan Maierhofer, and Alexander Wilkie. A multiresolution mesh generation approach for procedural definition of complex geometry. In *SMI '02: Proceedings of the Shape Modeling International 2002 (SMI'02)*, page 35, Washington, DC, USA, 2002. IEEE Computer Society.

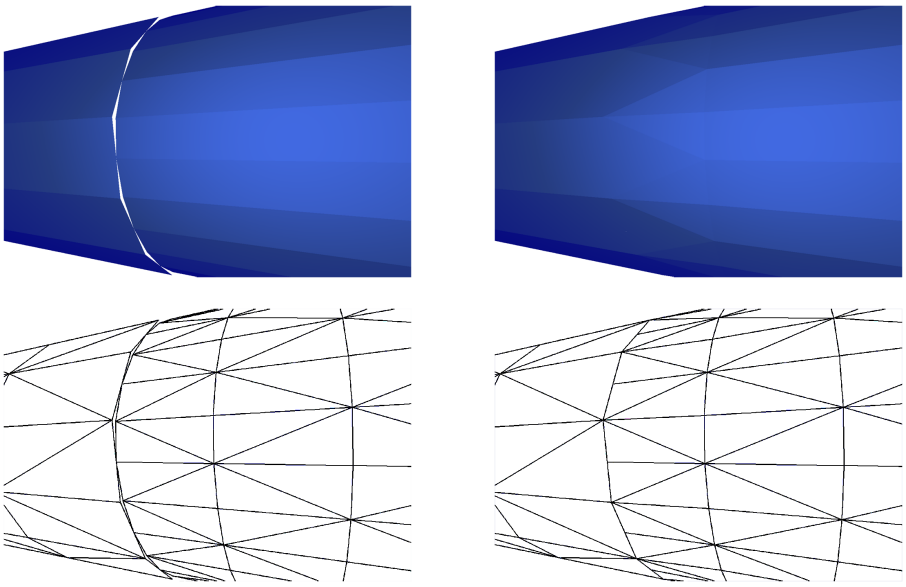


Figure 16: A junction between two meshes at different LODs. Left: without adjustment, Right: with forced border vertices.

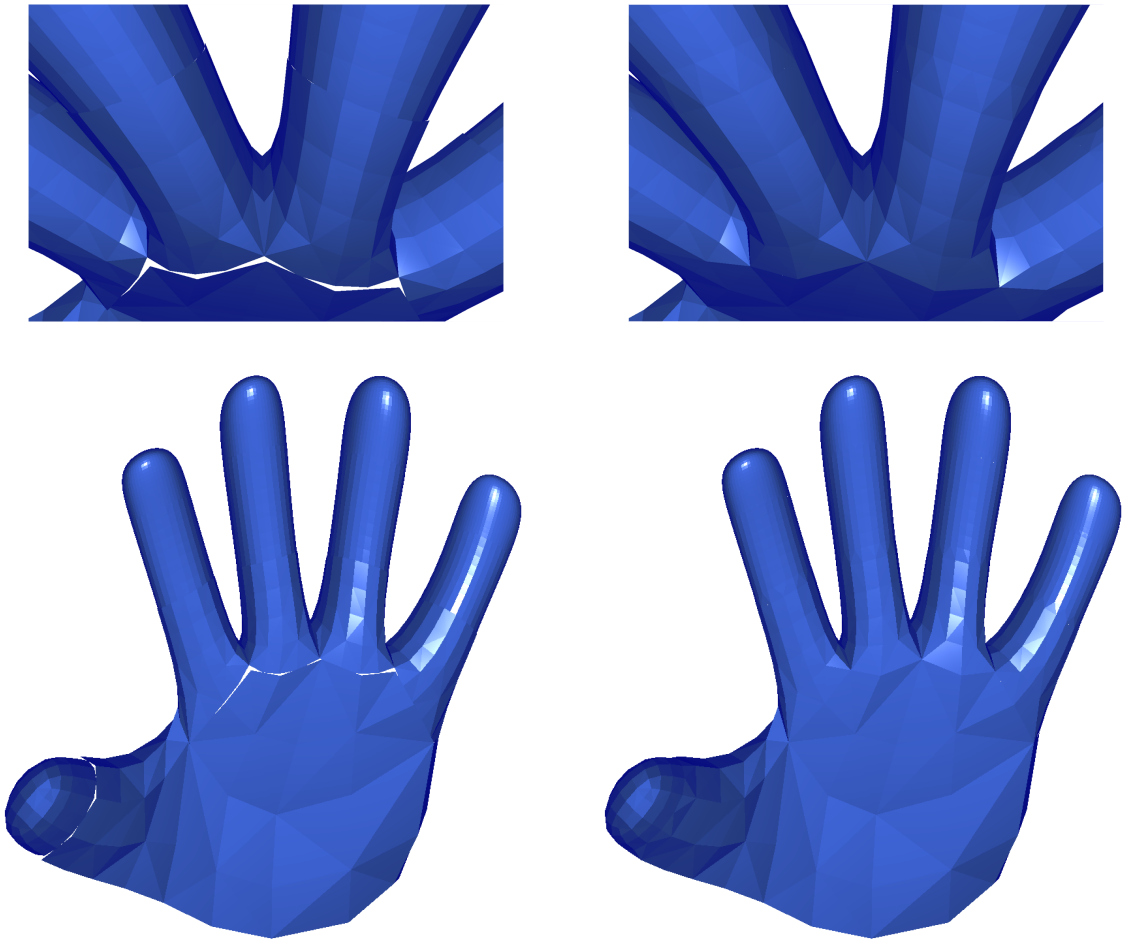


Figure 17: Hand model at different LODs. Left: without junction adjustment, right: with forced border vertices.

 ISBN 978-80-86943-98-5

A new Image Interpolation Technique using Exponential B-Spline

Raghubansh B. Gupta
Dongseo University
San 69-1, Jurye 2-Dong
Sasang-gu
617-716, Busan, S.Korea
raghubansh_b@yahoo.com

Byung Gook Lee
Dongseo University
San 69-1, Jurye 2-Dong
Sasang-gu
617-716, Busan, S.Korea
lbg@dongseo.ac.kr

Joon Jae Lee
Dongseo University
San 69-1, Jurye 2-Dong
Sasang-gu
617-716, Busan, S.Korea
jjlee@dongseo.ac.kr

ABSTRACT

In this paper, we propose a new interpolation technique using exponential B-spline, which is super-set of the B-spline. An interpolation kernel of exponential B-spline was proposed using IIR based technique by Unser. As another approach, this paper presents an exponential B-spline interpolation kernel using simple mathematics based on Fourier approximation. A high signal to noise ratio can be achieved because exponential B-spline parameters can be set depending on the signal characteristics. The analysis of these interpolated kernels shows they have better performance in high and low frequency components as compared to other conventional nearest neighbor, linear, spline based methods.

Keywords

Exponential B-spline, Interpolation, Image zooming.

1. INTRODUCTION

Nowadays, interpolation technique has a large usage in the field of computer vision, digital photography, multimedia and electronic publishing for generating preview images. In image compression, digital zooming, computed tomography (CT), magnetic resonance imaging (MRI), image reconstruction requires interpolation to approximate the discrete data to get the enhanced image, hence there is a need of good interpolatory scheme that can efficiently restore the signal and can help to reduce the cost of systems.

Interpolation is a method of constructing new data points from a discrete set which are being fitted in the continuous curves and then sampling at a higher rate interpolates the given data. In the early years, simple algorithms, such as nearest neighbor or linear interpolation, were used for sampling. After the introduction of sinc function a revolutionary idea was born in the field of interpolation because of its acceptance as a ideal interpolation function. However, this ideal interpolator has an infinite impulse response (IIR) and is not suitable for local interpolation with finite impulse response (FIR).

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.

The B-spline functions because of its close resemblance with the sinc function were being started to use prominently as an interpolation function. The term spline is used to refer to a wide class of piecewise polynomial function jointed at certain continuity points called as knots. Until now, in the spline family, extensive research is being done for polynomial spline [Uns99a]. However, the exponential splines are more general representation of these polynomial splines [Dah87a]. In the present work, the continuous exponential function is derived at equally spaced knots using truncated power functions and for the formulation of the exponential interpolated kernel this approximation function is convolved with Fourier approximation of the sampled exponential E-spline function [Leh99a]. The calculation of polynomial B-splines is a particular case, when the parameters of the exponents are set to be zero. The exponential B-spline interpolation function is derived for symmetric case taking different exponential parameter in consideration. The spatial and the frequency domain characteristics of the exponential spline interpolation function is discussed. Analyzing these characteristics it can be said exponential B-spline are less band-limited functions passing some of the high frequency components and reducing some of the low frequency

 ISBN 978-80-86943-98-5

components. This characteristic is useful for edge enhancement in various types of images. These spline-based algorithms have been found to be quite advantageous for image processing and medical imaging, because of restoring the high frequency components, especially in the context of high-quality interpolation

The paper is organized as follows. Section II starts with a brief review of B-spline and shows how this B-spline's can be linked to the extension of exponential splines. The exponential B-spline framework is ideally suited for performing some basic continuous-time signal processing operations in the discrete domain. Further characteristic of the spline is shown by analyzing it in the spatial and Fourier domain. Section III presents the results after applying the kernel of the exponential B-spline on one and two dimensional data followed by conclusion.

2. EXPONENTIAL B-SPLINES

2.1 Exponential B-splines Function

A continuous basic spline can be obtained by the convolution of $w^i(x), i = 1,..., n$ [Hsi78a],[Asa01a].

$$\beta^n(x) = (w^1 * w^2 * \cdots * w^n)(x) \qquad (1)$$

Where $*$ denotes the convolution operator. These weights are defined in the domain $x \in [-\frac{1}{2}, \frac{1}{2})$ and 0 otherwise for centered splines [Uns05b]. For shifted spline the domain is $x \in [0,1)$. In the case of polynomial splines, weight $w^i(x)$ is a rectangular function of height 1 and in the case of exponential B-splines the weight being exponential in nature are given as $w^i(x) = e^{\alpha_i x}$ with $\alpha_i \in C$ is the parameter of the exponential function. The exponential parameter vector is given as $\vec{\alpha}_n = (\alpha_i,...., \alpha_n)^T$. Continuing from the B-spline the exponential B-spline can be obtained by convolving the weights. Also, the exponential B-spline can also be expressed as [Dah87a]:

$$\beta^{\vec{\alpha}_n}(x) = \sum_{k \in Z} d^{\vec{\alpha}_n}[k] \rho^{\vec{\alpha}_n}(x-k) \qquad (2)$$

Where the single and multiple discrete differences function are given by

$$d^{\alpha_1}[k] = \delta[k] - e^{\alpha_1}\delta[k-1]$$

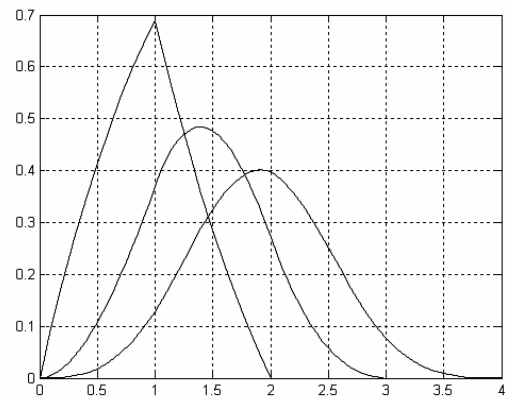


Figure 1. n-*th* order Exponential B-spline with n varying from 2.

$$d^{\vec{\alpha}_n}[k] = d^{\alpha_1} * d^{\alpha_2} * \cdots * d^{\alpha_n}[k] \qquad (3)$$

respectively, with $\delta[k]$ being the kronecker's delta function and $\rho^{\vec{\alpha}_n}(\bullet)$ is the continuous exponential truncated power function which can be expressed as:

$$\rho^{\alpha_1}(x) = 1_+(x) e^{\alpha_1 x}$$

And further can be recursively calculated using,

$$\rho^{\vec{\alpha}_n}(x) = (\rho^{\alpha_1} * \rho^{\alpha_2} * \cdots \rho^{\alpha_n})(x). \qquad (4)$$

Considering linear (n=2) case, we have vector $\vec{\alpha}_2 = (\alpha_1, \alpha_2)^T$ and from equation (3),

$$d^{\vec{\alpha}_2}[k] = d^{\alpha_1} * d^{\alpha_2}[k],$$

$$d^{\vec{\alpha}_2}[k] = \delta[k] - (e^{\alpha_1} + e^{\alpha_2})\delta[k-1] + e^{\alpha_1+\alpha_2}\delta[k-2].$$

And the truncated exponential from equation (4) can be simplified as :

$$\rho^{\vec{\alpha}_2}(x) = 1_+(x)\left\{\frac{e^{\alpha_1 x}}{\alpha_2 - \alpha_1} + \frac{e^{\alpha_2 x}}{\alpha_1 - \alpha_2}\right\}$$

where $1_+(x) = \begin{cases} 1, x \geq 0 \\ 0, otherwise \end{cases}$

Therefore from equation (2),

$$\beta^{\vec{\alpha}_2}(x) = d^{\vec{\alpha}_2}[k] * \rho^{\vec{\alpha}_2}(x-k)$$

Hence,

$$\beta^{\vec{\alpha}_2}(x) = \begin{cases} \dfrac{e^{\alpha_2 x} - e^{\alpha_1 x}}{\alpha_2 - \alpha_1} & 0 \leq x < 1, \\ \dfrac{e^{\alpha_2 x - \alpha_2 + \alpha_1} - e^{\alpha_1 x + \alpha_2 - \alpha_1}}{\alpha_2 - \alpha_1} & 1 \leq x < 2. \end{cases} \qquad (5)$$

 ISBN 978-80-86943-98-5

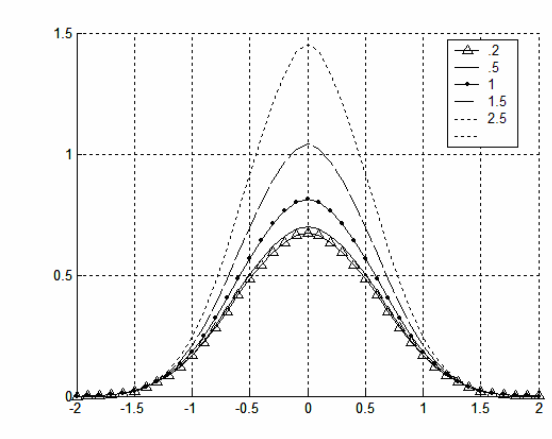


Figure 2. 4-*th* order Exponential B-spline for (α, α,-α, -α) where α = .2, .5, 1,1.5,2.5,3.

By similarly expanding equation (2), the piecewise function for the exponential B-splines of order n is given by the expression:

$$\sum_{k=1}^{n} d^{\vec{\alpha}_n}[k]x^k = \prod_{k=1}^{n}(1-e^{\alpha_k}x),$$

$$\rho^{\vec{\alpha}_n}(x) = 1_+(x)\sum_{i=1}^{n}\left\{\frac{e^{\alpha_i x}}{\prod_{j\neq i}(\alpha_j-\alpha_i)}\right\}.$$

The 4-th order exponential B-spline function considering exponential parameters (α, α, -α, -α) are symmetric for real values of α as shown in Fig. 2.

2.2 Interpolation Kernel Function

For interpolation, the steps involved must reconstruct a one dimensional continuous signal $s(x)$, from its discrete samples $s(k)$ with $s, x \in R$ and $k \in \mathrm{N}$. Thus the value at the position x must be estimated from its discrete neighbors. This can be formally described as the convolution of the discrete samples with the continuous impulse response given by

$$s(x) = \sum_k s(k)\cdot h(x-k) \qquad (6)$$

The 1-D ideal interpolation equals to the multiplication with a rectangular function in the Fourier domain and it can be realized by convolution with the sinc function in the spatial domain.

$$h_{ideal}(x) = \frac{\sin(\pi x)}{\pi x} = \sin c(x) \qquad (7)$$

Some fundamental properties of any interpolator can be derived from this ideal interpolation function. Therefore

$$h(0) \equiv 1,\ h(x) \equiv 0, |x| = 1,2. \qquad (8)$$

These zero crossings guarantee that the signal is not modified, avoids smoothing and preserves high frequencies.

As interpolating the discrete signal is equivalent to sampling the continuous signal. Due to aliasing of the high frequency in the lower ones, It is important to examine not only the continuous interpolation function $h(x)$ but also interpolated function $h(k)$ i.e., the sum of all samples should be one for any displacement $0 \le d < 1$

$$\sum_{k=-\infty}^{\infty} h(d+k) \equiv 1 \qquad (9)$$

This means that for any displacement d the summation of the function h should be unity. That's why the mean amplitude of the signal remains unaffected if the signal is resampled or interpolated.

2.3 Exponential B-spline Interpolation

To create an interpolating exponential B-spline kernel, the exponential B-spline approximator is applied to different sets of samples $t(k)$. Since the exponential B-spline kernel is symmetrical and separable, the reconstruction yields

$$s(x) = \sum_k t(k)\cdot h(x-k) \qquad (10)$$

with h being the basis function. The general case (10) reduces to equation (6) if the samples are taken from the given data: $t(k) \equiv s(k)$. Here, the $t(k)$ must be derived from the given data $s(k)$ in such a way that the resulting curve interpolates the given data.

So,

$$s(k) = \sum_{m=k-2}^{m=k+2} t(m)\cdot h(x-m) \qquad (11)$$

The function $h(x)$ is exponential B-spline function for n=4 case, so

$$h(x) = \beta(x)$$

Therefore,

$$s(k) = \beta(1)t(k-1) + \beta(0)t(k) + \beta(-1)t(k+1)$$

ignoring edge effects, results in a set of equations to solve

$$S = BT$$

The coefficient T can be evaluated by multiplying the known data points S with the inverse of the tridiagonal matrix B

 ISBN 978-80-86943-98-5

$$T = B^{-1}S \tag{12}$$

To simplify its analytical derivation, the interpolated image $s(x)$ and the data samples $s(k)$, calling them u and v, respectively. From equation (11) we obtain,

$$v = t * B$$
$$v(x) = t(x) * (\beta(1)\delta(x-1) + \beta(0)\delta(x) + \beta(-1)\delta(x+1)) \tag{13}$$

In frequency domain for symmetric case the equation reduces to

$$V(f) = T(f) \cdot B(f) \tag{14}$$

Inversion of equation (14) yields,

$$T(f) = V(f) \cdot \frac{1}{\beta(0) + 2\beta(1)\cos(2\pi f)}$$

As $G(f) = \dfrac{1}{\beta(0) + 2\beta(1)\cos(2\pi f)}$ is even so Taking Fourier series approximation we get

$$T(f) = V(f) \cdot (a_0 + \sum_{n=1}^{\infty} a_n \cos(2\pi f))$$

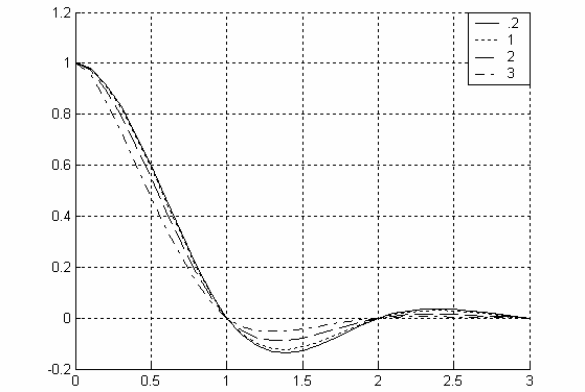


Figure 3. 4-*th* order Exponential B-spline interpolation kernel with (α, α,-α, -α).

where,

$$a_0 = \frac{1}{T}\int_0^T G(f)df$$

$$a_n = \frac{1}{T}\int_0^T G(f)\cos(2\pi f)df$$

Taking back in the spatial domain

$$t(x) = v(x) * (\frac{a_0}{2}\delta(x) + \sum_{n=1}^{\infty} a_n(\delta(x-n) + \delta(x+n)))$$

Hence equation (10) can be written as

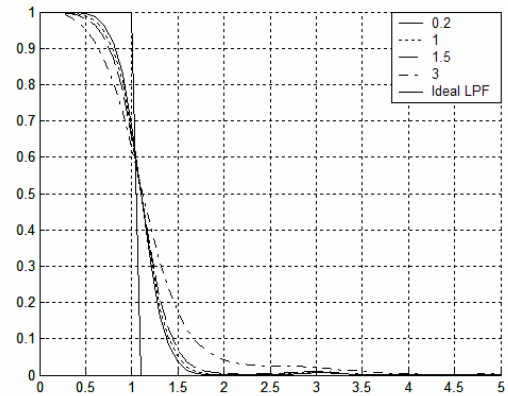
$$u = t * h = v * h_{Espline} \tag{15}$$

Finally with (15) we obtain,

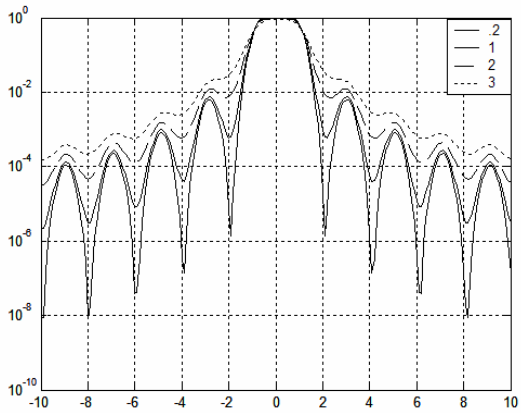
$$h_{Espline}(x) = \beta(x) * (\frac{a_0}{2}\delta(x) + \sum_{n=1}^{\infty} a_n(\delta(x-n) + \delta(x+n)))$$

The interpolation kernel is symmetric, passes through the integer points. Fig. 3 shows the interpolation kernel for only [0,3], moreover the kernel is symmetric around x=0.

The Fourier domain response of the interpolation kernel is shown in Fig. 4 for different values of exponential parameters. As the value of alpha is changed the filter response deviates from the ideal low pass filter. However, the interpolation kernel is band limited passing the high frequency components near the cut-off frequency, which can be used to preserve the edge information in the images. With the increase in value of alpha, the transition part of the filter decreases the magnitude of the low frequency and increases the number and magnitude of the high frequency components.



(a)



(b)

Figure 4. (a) Fourier domain magnitude plot and, (b) log plot of Exponential B-spline for (α, α,-α, -α) with different values of α.

 ISBN 978-80-86943-98-5

Exponential parameter α	Energy distributed between -3 to 3	Energy distributed between -2 to 2
0.2	0.999800	0.998423
1	0.999941	0.998964
2	0.999991	0.999711
3	1.000000	0.999961
4	1.000000	0.999997

Table 1. The percentage energy distribution in exponential B-spline interpolation kernel for different α after truncation.

The energy distribution in exponential B-spline interpolation kernel is minimum for lower value of exponential parameter. As the value of these parameter increases, energy decreases in both between -2 to 2 and -3 to 3 which is shown in Table 1.

Fig. 5 plots the sum of sampled interpolation kernel from equation (9) as a function of displacement *d*. The summation is done after truncating the kernel from -3 to 3. It is clear that for alpha closer to 1.2, the sum of sampled interpolated kernels is closer to 1, hence the value close to 1.2 give better interpolation. This can be verified with PSNR given in table 2.

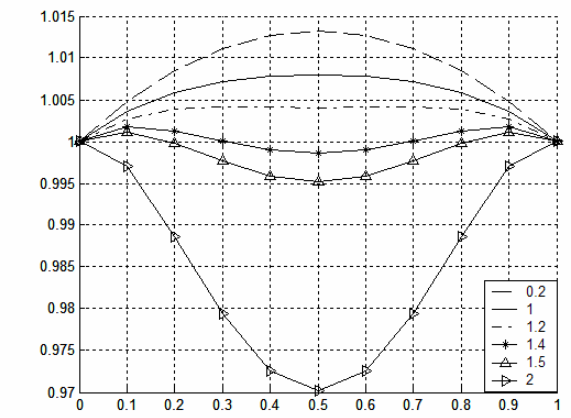


Figure 5. Sum of sampled interpolation kernels as a function of the displacement for different α.

3. SIMULATION AND RESULT

Here we have examined the exponential spline interpolation kernel with different 1-D and 2-D data's. The exponential B-spline interpolation function derived is experimented with unit step function. It is shown in Fig. 6 that Exponential B-spline results in less oscillation for higher values of exponential parameter. Similarly Fig. 7 and Fig. 8 show the interpolated Lena image by factor 2. In the case of the expansion using B-spline, some ringing effect near the edges can be observed, however these ringing effect is diminished by using the Exponential B-spline.

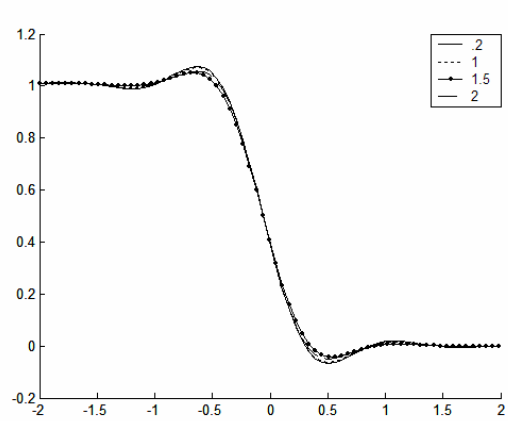


Figure 6. Interpolation of the discrete Step using exponential and polynomial B-splines functions for different α.

This Exponential B-spline interpolation is compared with the other conventional methods (nearest neighbor, bilinear, and B-spline). In the set of 2-D experiments with original Lena, Barbara and other gray scaled image exponential B-spline has performed better results. For calculating peak signal to noise ratio (PSNR), initially we have filtered and downsampled the image using wavelet decomposition with Daubechies's 9-7 tap filter and then the image is interpolated using exponential B-spline interpolation where PSNR=20log10(255/RMSE) and RMSE is root mean square error between the original signal and interpolated signal. The PSNR calculation is shown in Table 2, the shaded part shows the maximum PSNR values.

4. CONCLUSION

This paper is focused on interpolation methods using Exponential B-spline, the method is easy for calculation of Exponential B-spline (where the calculation of B-spline is a particular case). Continuing from the Exponential B-spline approximation to the derivation of Exponential B-spline interpolation this paper provided a unified framework for the theoretical analysis and performance of both approximation and interpolation kernels. We have applied this analysis to specific case that involves piecewise exponential B-spline approximation having exponential parameter (α, α, -α, -α). This interpolation kernel can be used for images containing high as well as low frequency components for different values of α. The Exponential B-spline interpolation is performed on various 2-D and 1-D examples. Experiments shows that as the value of *α*

 ISBN 978-80-86943-98-5

is increased the ringing and oscillatory behavior can be reduced at the edges.

Further the best performance of the kernel is near α=1.2 which is shown by PSNR in Table 2. The further improvement is to use some adaptive algorithm for image interpolation based on changing values of exponential parameter.

5. ACKNOWLEDGEMENT

This work was supported in part by grant No. R01-2004-000-10851-0 from Ministry of Science and Technology and by the BK21 program of the Ministry of Education, Korea

6. REFERENCES

[Uns99a] Unser, M.: Splines: A Perfect Fit for Signal and Image Processing. IEEE Signal Processing Magazine, pp. 22-38, Nov.1999.

[Dah87a] Dahmen, W., Micchelli, C. A.: On Theory and Application of Exponential Splines Academic Press New York, 37-46, 1987.

[Leh99a] Lehmann T. M., Gonner C., Spitzer K., Survey: Interpolation Methods in Medical Image Processing. IEEE transactions on medical imaging Vol. 18, No. 11, pp. 1049-1075, Nov.1999.

[Uns05b] Unser, M., Blu, T.: Cardinal Exponential Splines: Part I Theory and Filtering Algorithms. IEEE transactions on signal processing Vol. 53, No. 4, pp.1425-1438, April 2005.

[Hsi78a] Hsieh S. Hou, Harry C. Andrews: Cubic Splines for image Interpolation and digital Filtering, IEEE transactions on acoustics, speech, and signal processing, Vol. Assp-26, No.6, pp.508-517, December 1978.

[Asa01a] Asahi T., Ichige K., Ishii R.., Fast computation of Exponential B-Splines, Proceedings of International Technical Conference on Circuits/Systems, Computers and Communications, pp. 1296-1299, July 2001.

Image	Linear	B-spline	Exponential B-spline							
			α = 0.2	α= 0.5	α= 1	α= 1.2	α=1.3	α=1.4	α=1.5	α =2
Lena	30.49	31.34	31.37	31.41	31.54	31.55	31.53	31.46	31.33	29.31
Gold-hill	29.60	29.84	29.86	29.89	29.98	30.00	29.95	29.87	29.54	29.60
Pepper	29.85	30.88	30.90	30.92	30.98	30.97	30.94	30.88	30.77	29.20
Barbara	24.29	24.63	24.64	24.65	24.67	24.66	24.65	24.63	24.59	23.95
Boat	27.70	28.25	28.27	28.29	28.38	28.40	28.39	28.35	28.28	26.98

Table 2. PSNR calculation of different interpolation technique on gray images.

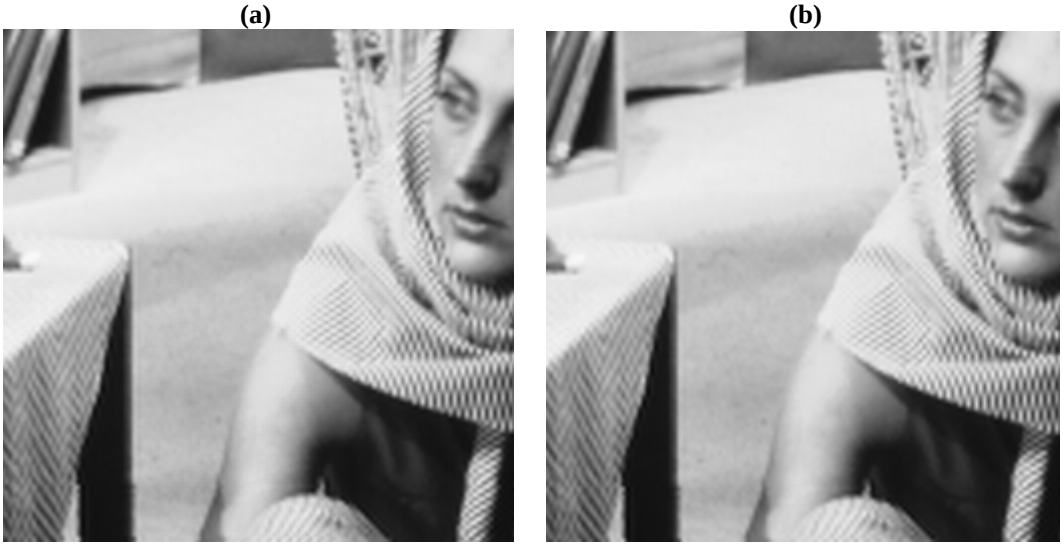
(a)

(b)

(c) **(d)**

Figure 7. (a)Original Lena Image (256*256), (b) Linear interpolated image (256*256), (c) Exponential B-spline interpolated image (256*256) with α=0.2, and (d) with α =1.2.

(a) **(b)**



(c) **(d)**

Figure 8. (a) Original Barbara Image (256*256), (b) Linear interpolated image (256*256), (c) Exponential B-spline interpolated image (256*256) with α=0.2, and (d) with α =1.2.

 ISBN 978-80-86943-98-5