

**The 14-th International Conference in Central Europe on
Computer Graphics, Visualization and Computer Vision 2006**

in co-operation with

EUROGRAPHICS

W S C G ' 2006

University of West Bohemia
Plzen
Czech Republic

January 31 – February 2, 2006

Short Papers Proceedings

Co-Chairs

**Joaquim Jorge, Technical University of Lisboa, Lisboa, Portugal
Vaclav Skala, University of West Bohemia, Plzen, Czech Republic**

Edited by

Joaquim Jorge, Vaclav Skala

WSCG'2006 Short Communication 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 80-86943-05-4*

WSCG 2006

International Programme Committee

Bartz,D. (Germany)	Pasko,A. (Japan)
Bekaert,P. (Belgium)	Peroche,B. (France)
Benes,B. (United States)	Post,F. (Netherlands)
Bengtsson,E. (Sweden)	Puppo,E. (Italy)
Bouatouch,K. (France)	Purgathofer,W. (Austria)
Brunnet,G. (Germany)	Rauterberg,M. (Netherlands)
Chen,M. (United Kingdom)	Rheingans,P. (United States)
Chrysanthou,Y. (Cyprus)	Rokita,P. (Poland)
Cohen-Or,D. (Israel)	Rossignac,J. (United States)
Coquillart,S. (France)	Rudomin,I. (Mexico)
Debelov,V. (Russia)	Sakas,G. (Germany)
Deussen,O. (Germany)	Sbert,M. (Spain)
du Buf,H. (Portugal)	Schaller,N. (United States)
Ertl,T. (Germany)	Schilling,A. (Germany)
Ferguson,S. (United Kingdom)	Schneider,B. (United States)
Groeller,E. (Austria)	Schumann,H. (Germany)
Hauser,H. (Austria)	Shamir,A. (Israel)
Hege,H. (Germany)	Slusallek,P. (Germany)
Jansen,F. (Netherlands)	Sochor,J. (Czech Republic)
Jorge,J. (Portugal)	Sumanta,P. (United States)
Kalra,P. (India)	Szirmay-Kalos,L. (Hungary)
Klein,R. (Germany)	Taubin,G. (United States)
Klosowski,J. (United States)	Teschner,M. (Germany)
Kobbelt,L. (Germany)	Velho,L. (Brazil)
Kruijff,E. (Germany)	Veltkamp,R. (Netherlands)
Lars,K. (Sweden)	Weiskopf,D. (Canada)
Magnor,M. (Germany)	Westermann,R. (Germany)
Moccozet,L. (Switzerland)	Wu,S. (Brazil)
Mudur,S. (Canada)	Wuethrich,C. (Germany)
Mueller,K. (United States)	Yamaguchi,F. (Japan)
Muller,H. (Germany)	Zara,J. (Czech Republic)
Myszkowski,K. (Germany)	Zemcik,P. (Czech Republic)
OSullivan,C. (Ireland)	

WSCG 2006 Board of Reviewers

Adamo-Villani,N. (United States)	Flaquer,J. (Spain)	Levy,B. (France)
Adzhiev,V. (United Kingdom)	Fleck,S. (Germany)	Lintu,A. (Germany)
Ammon,L. (Switzerland)	Francken,Y. (Belgium)	Linz,C. (Germany)
Andreadis,I. (Greece)	Gagalowicz,A. (France)	Lipus,B. (Slovenia)
Aran,M. (Turkey)	Galo,M. (Brazil)	Magalhaes,L. (Brazil)
Araujo,B. (Portugal)	Geraud,T. (France)	Magillo,P. (Italy)
Aspragathos,N. (Greece)	Giachetti,A. (Italy)	Magnor,M. (Germany)
Bartz,D. (Germany)	Giegel,J. (United States)	Mantler,S. (Austria)
Batagelo,H. (Brazil)	Groeller,E. (Austria)	McMenemy,K. (United Kingdom)
Battiato,S. (Italy)	Gudukbay,U. (Turkey)	Millan,E. (Mexico)
Bekaert,P. (Belgium)	Guerreiro,T. (Portugal)	Moccozet,L. (Switzerland)
Benes,B. (United States)	Habbecke,M. (Germany)	Molledo,L. (Italy)
Bengtsson,E. (Sweden)	Haber,T. (Belgium)	Montrucchio,B. (Italy)
Beyer,J. (Austria)	Hanak,I. (Czech Republic)	Mudur,S. (Canada)
Biber,P. (Germany)	Hast,A. (Sweden)	Mueller,K. (United States)
Bieri,H. (Switzerland)	Hauser,H. (Austria)	Muller,H. (Germany)
Bilbao,J. (Spain)	Havran,V. (Germany)	Multon,F. (France)
Biri,V. (France)	Hege,H. (Germany)	Myszkowski,K. (Germany)
Bischoff,S. (Germany)	Hernandez,B. (Mexico)	Nielsen,F. (Japan)
Borchani,M. (France)	Herzog,R. (Germany)	Novotny,M. (Austria)
Bottino,A. (Italy)	Hirschbach,H. (Germany)	O'Sullivan,C. (Ireland)
Bouatouch,K. (France)	Hornung,A. (Germany)	Pasko,A. (Japan)
Brodlie,K. (United Kingdom)	Chen,M. (United Kingdom)	Patel,D. (Austria)
Brunnet,G. (Germany)	Chrysanthou,Y. (Cyprus)	Patera,J. (Czech Republic)
Buehler,K. (Austria)	Isgro,F. (Italy)	Pedrini,H. (Brazil)
Cohen-Or,D. (Israel)	Jaillet,F. (France)	Perales,F. (Spain)
Coleman,S. (United Kingdom)	Janda,M. (Czech Republic)	Peroche,B. (France)
Coquillart,S. (France)	Jansen,F. (Netherlands)	Platis,N. (Greece)
Daniel,M. (France)	Jeschke,S. (Austria)	Plemenos,D. (France)
Danovaro,E. (Italy)	Jorge,J. (Portugal)	Porcu,M. (Italy)
de Aguiar,E. (Germany)	Jota,R. (Portugal)	Post,F. (Netherlands)
De Decker,B. (Belgium)	Kalra,P. (India)	Pratikakis,I. (Greece)
Debelov,V. (Russia)	Kavan,L. (Czech Republic)	Prikryl,J. (Czech Republic)
Del Rio,A. (Germany)	Keller,K. (United States)	Puig,A. (Spain)
Deussen,O. (Germany)	Kipfer,P. (Germany)	Puppo,E. (Italy)
Di Fiore,F. (Belgium)	Klein,K. (Germany)	Purgathofer,W. (Austria)
Diaz,M. (Mexico)	Klosowski,J. (United States)	Rauterberg,M. (Netherlands)
du Buf,H. (Portugal)	Kobbelt,L. (Germany)	Reaz,M. (Malaysia)
Duce,D. (United Kingdom)	Kolcun,A. (Czech Republic)	Reina,G. (Germany)
Erbacher,R. (United States)	Krüger,J. (Germany)	Renaud,C. (France)
Ertl,T. (Germany)	Kruijff,E. (Germany)	Revelles,J. (Spain)
Feito,F. (Spain)	Lanquetin,S. (France)	Ribelles,J. (Spain)
Felkel,P. (Czech Republic)	Lars,K. (Sweden)	Rodeiro,J. (Spain)
Ferguson,S. (United Kingdom)	Leon,A. (Spain)	Rojas-Sola,J. (Spain)
Fernandes,A. (Portugal)	Leopoldseder,S. (Austria)	Rokita,P. (Poland)

Rose,D. (Germany)
Rossignac,J. (United States)
Rudomin,I. (Mexico)
Sakas,G. (Germany)
Sanna,A. (Italy)
Sbert,M. (Spain)
Scateni,R. (Italy)
Segura,R. (Spain)
Shah,M. (United States)
Shamir,A. (Israel)
Schafhitzel,T. (Germany)
Schaller,N. (United States)
Scherzer,D. (Austria)
Schilling,A. (Germany)
Schneider,B. (United States)
Schneider,J. (Germany)
Scholz,V. (Germany)
Schumann,H. (Germany)
Sips,M. (Germany)
Sitte,R. (Australia)
Slusallek,P. (Germany)
Snoeyink,J. (United States)

Snoeyink,J. (United States)
Sochor,J. (Czech Republic)
Sojka,E. (Czech Republic)
Solis,A. (Mexico)
Sondershaus,R. (Germany)
Sporka,A. (Czech Republic)
Stephane,R. (France)
Stich,T. (Germany)
Strengert,M. (Germany)
Stroud,I. (Switzerland)
Stylianou,G. (Cyprus)
Sumanta,P. (United States)
Szekely,G. (Switzerland)
Szirmay-Kalos,L. (Hungary)
Tang,W. (United Kingdom)
Taubin,G. (United States)
Teschner,M. (Germany)
Theußl,T. (Austria)
Torres,J. (Spain)
Ulbricht,C. (Austria)
Van Laerhoven,T. (Belgium)
Vanecek,P. (Czech Republic)

Velho,L. (Brazil)
Veltkamp,R. (Netherlands)
Vergeest,J. (Netherlands)
Viola,I. (Austria)
VOLLRATH,J. (Germany)
Vuorimaa,P. (Finland)
Wan,T. (United Kingdom)
Weidlich,A. (Austria)
Weiskopf,D. (Canada)
Westermann,R. (Germany)
Wood,J. (United Kingdom)
Wu,S. (Brazil)
Wuethrich,C. (Germany)
Yilmaz,T. (Turkey)
Zach,C. (Austria)
Zachmann,G. (Germany)
Zalik,B. (Slovenia)
Zambal,S. (Austria)
Zara,J. (Czech Republic)
Zemcik,P. (Czech Republic)

WSCG 2006

Short Papers proceedings

ISBN 80-86943-05-4

Contents

(Additional files available on CD ROM version, only)

Paper code	Paper Title	Page
G67	Agrawal,A., Radhakrishna,M., Joshi,R.C.: Geometry-based Mapping and Rendering of Vector Data over LOD Phototextured 3D Terrain Models (India)	1
D11	Fonseca,F., Feijo,B., Dreux,M., Clua,E.: A Parallel Approach for Visualization of Relief Textures (Brazil)	9
B03	Bhattarai, D., Karki, B.: Visualization of Atomistic Simulation Data for Spatio-Temporal Information (United States)	17
G97	Ferreira,A., Vala,M., Pereira,J.A.M, Jorge,J.A., Paiva,A.: Calligraphic Interface for Management of an Agents Platform (Portugal)	25
A73	Ilmonen,T., Takala,T., Laitinen,J.: Soft Edges and Burning Things: Enhanced Real-Time Rendering of Particle Systems (Finland)	33
A97	Chambelland,J.-Ch., Daniel,M., Brun,J.-M.: An Iterative Method for Rational Pole Curve Fitting (France)	39
D29	Lobos,C., Hirschfeld-Kahler,N.: 3D NOffset Mixed-Element Mesh Generator Approach (Chile)	47
G53	Loke,R.E., Jansen,F.W., du Buf,H.: A Background-Priority Discrete Boundary Triangulation Method (Italy)	53
G17	Wundrak,S., Henn,T., Stork,A.: Dynamic Progressive Triangle-Quadrilateral Meshes (Germany) Additional files: G17-1.jpg (151KB), G17-2.avi (8,7MB)	61
C02	Heurtebise,X., Thon,S., Gesquiere,G.: Multiresolution Representation and Deformation of Wavelet-Based 3D Objects (France)	69
A37	Janney,P., Amur,H., Sridhar,G., Sridhar,V.: MV Number: Effective Key to Represent Images (India)	77
F41	Jaume-Capó,A. ,Varona,J. ,González-Hidalgo,M. ,Mas,R. ,Perales,F.: Automatic Human Body Modeling for Vision-Based Motion Capture (Spain)	81
F47	Le Garrec,J., Andriot,C., Merlhiot,X., Bidaud,P.: Virtual Grasping of Deformable Objects with Exact Contact Friction (France)	87
E47	Manresa-Yee,C., Varona,J., Perales,F.J.: Face-Based Perceptual Interface for Human-Computer Interaction (Spain)	93
E67	Veyret,M., Maisel,E.: Attention-Based Target Tracking for an Augmented Reality Application (France)	101
C03	Bleser,G., Wohlleber,C., Becker, M., Stricker, D.: Fast and Stable Tracking for Augmented Reality fusing Video and Inertial Data (Germany)	109

D61	Zhang,L., Li,L.: 3D Human Animation from 2D Monocular Data Based on Motion Trend Prediction (Australia)	117
B83	García,R., Urena,C., Revelles,J., Lastra,M., Montes,R.: Density Estimation Optimizations for Global Illumination (Spain)	125
H29	Martinelli,A.: A New Model for 3D Graphical Rendering (Italy)	133
E23	Papageorgiou,S., Aspragathos,N.: Rational Ruled Surfaces Construction by Interpolating Dual Unit Vectors Representing Lines. (Greece)	141
C13	Peng,E., Li,L.: Human Skeleton Modeling from 2D Uncalibrated Monocular Data (Australia)	149
E17	Tobler,R., Maierhofer,S.: A Mesh Data Structure for Rendering and Subdivision (Austria)	157
F83	Villard,P.F., Beuve,M. , Shariat,B.: An Approach to Convert 4D Geometry into a 4D CT Scan (France)	163
A13	Thomaszewski,B., Wacker,M.: Bending Models for Thin Flexible Objects (Germany)	171
G59	Xizhe,Z., Zhengxuan,W., Tianyang,L.: A Generalized Mandelbrot Set Based On Distance Ratio (China)	179

Geometry-based Mapping and Rendering of Vector Data over LOD Phototextured 3D Terrain Models

Anupam Agrawal
Indian Institute of Information
Technology,
Deoghat, Jhalwa,
India (211011), Allahabad, U.P.
anupam@iita.ac.in

M. Radhakrishna
Indian Institute of Information
Technology,
Deoghat, Jhalwa,
India (211011), Allahabad, U.P.
mkrishna@iita.ac.in

R.C. Joshi
Dept. of E&C Engineering,
Indian Institute of Technology,
India (247667), Roorkee, U.A.
joshifcc@iitr.ernet.in

ABSTRACT

Interactive three-dimensional (3D) visualization of very large-scale grid digital elevation models coupled with corresponding high-resolution remote-sensing phototexture images is a hard problem. The graphics load must be controlled by an adaptive view-dependent surface triangulation and by taking advantage of different levels of detail (LODs) using multiresolution modeling of terrain geometry. Furthermore, the display of vector data over the level of detail terrain models is a challenging task. In this case, rendering artifacts are likely to occur until vector data is mapped consistently and exactly to the current level-of-detail of terrain geometry. In our prior work, we have developed a view-dependent dynamic block-based LOD mesh simplification scheme and out-of-core management of large terrain data for real-time rendering on desktop PCs. In this paper, we have proposed a new rendering algorithm for the combined display of multiresolution 3D terrain and polyline vector data representing the geographical entities such as roads, state or country boundaries etc. Our algorithm for multiresolution modeling of vector data allows the system to adapt the visual mapping without rendering artifacts to the context and the user needs while maintaining interactive frame rates. The algorithms have been implemented using Visual C++ and OpenGL 3D API and successfully tested on different real-world terrain raster and vector data sets.

Keywords

Digital terrain models, Multiresolution Modeling, Level-of-Detail Rendering, Vector and Raster Data.

1. INTRODUCTION

In conventional printed topographic maps, the real three-dimensional (3D) world is projected vertically onto a two-dimensional plane with symbolization of ground objects. On these maps, topography of the terrain is represented by contours, which are digitized and converted into grid digital elevation model (height map). Apart from contour information, the map consists of variety of other information including point features (e.g. buildings, trees etc.), line features (e.g. road networks, rivers etc.) and

polygon features (e.g. country boundaries, vegetation zone etc.). It is not easy to understand information on 2D topographic maps, as it demands some knowledge and skills in map reading.

3D rendering of the map provides information on geographical data about the shape of the terrain and location of other objects on a map quickly and easily. An interactive system with real-time rendering is useful in spatial support systems, virtual reality applications, real-time GIS and cartography.

Geographic data may be categorized as raster data and vector data. Raster data are analogous to a bit map or a regular 2D array where each array element contains a data value for a corresponding rectangular grid cell in the 2D plane. Common sizes of digital terrain raster data including height map and corresponding geo-referenced remote-sensing satellite imagery may consist of 1K*1K to 16K*16K or more grid cells. The interactive visualization of such large datasets has been a challenging problem. The main problem in real-time graphics is rendering efficiency [Ake02]. In order to get high rendering

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.

*Uj qtvEqo o wplecvkpu'rt qeggf lpi u'KDP': 2/: 8; 65/27/6
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press*

performance, one of the approaches is to reduce the scene complexity without leading to an inferior visual representation. Multiresolution models provide different levels of detail (LODs) representation of the modeled objects. In LOD scheme, the close regions are approximated more accurately than regions that are far away such that the resulting image is without any noticeable visual difference. In our prior work, we have developed a view-dependent dynamic block-based level-of-detail mesh simplification scheme and out-of-core management of large terrain data for real-time rendering on desktop PCs [Agr04a], [Agr04b].

Vector data represents one major category of geographic information and defines geometry as lists of 2D coordinates that form points, lines and polygons. Narrow linear features such as roads, railway lines etc. are usually not visible on a satellite image used in creating 3D phototextured views of the terrain. Other vector features such as state or country boundaries, property lines etc., usually used for logical demarcation, are also required to be overlaid on top of the 3D views with appropriate display properties. These vector features are separately digitized from corresponding topographic map.

Multiresolution level-of-detail terrain models, where underlying mesh geometry is changing for each frame, pose significant challenges in visual mapping of vector data in 3D without any rendering artifacts. Section 2 discusses related work and complications in displaying vector data over multiresolution terrain models. Section 3 briefly explains our view-dependent adaptive multiresolution mesh simplification framework, which is supporting real-time frame rates on desktop PCs on arbitrarily large digital terrain raster data. The proposed geometry-based mapping approach of polyline vector data for its integration with above multiresolution geometry modeling framework is explained in section 4. The approach smoothly adapts with our tile-based out-of-core management of terrain raster data. In section 5, we have shown the results of the proposed algorithm and performance analysis on a real-world terrain raster and vector data set. Finally section 6 gives conclusions and scope for future work.

2. RELATED WORK AND COMPLICATIONS IN VECTOR DATA DISPLAY

Display of 2D polyline vector data over 3D terrain geometry becomes relatively much simpler task if the underlying terrain geometry is static and is not changing with time i.e. the surface is being rendered at constant resolution [Agr98]. In this case, height values at points on vector data can be picked up from underlying geo-referenced DEM or height map. But

this approach has the drawback that it very much restricts the size of the terrain data. It is not feasible to render large terrain raster data sets of size 16K*16K or more at interactive frame rates even on a very high-end graphics workstation. Szenberg et al. [Sze97] describe a method of terrain visualization with polyline vector data such as transmission lines. However, the visualization scheme for terrain height field is not based on multiresolution modeling but combines the Z-buffer with the floating Horizon algorithm. Also results are shown on limited sized terrain data (512*512 size) only. Xiaoping et al. [Xia04] describe a method to render vector data on static terrain geometry. The actual size of the terrain data has not been reported in the paper.

In general, multiresolution modeling is necessary for representing large size geo-referenced surfaces in order to reduce their geometric complexity and to achieve real-time rendering [Lue03]. Relatively very less work has been reported in literature on displaying vector data over multiresolution terrain.

There are two options to rendering polyline vector data on multiresolution 3D mesh. One option is to convert the polyline data to a texture image layer and combine this polyline image layer with the primary terrain texture image layer (e.g. from a satellite/aerial photograph). The second option is to render the polyline data as separate 3D geometric primitives. We may call these two approaches as polyline-as-texture solution and polyline-as-geometry solution respectively. Both the approaches present a number of complications.

A simple polyline-as-texture solution is to rasterize the polyline into the primary texture image at the image's highest resolution and then to render the terrain in the standard way using mipmaps or other suitable filtering. However, this is a poor solution because when zoomed out on the 3D terrain, the user will find much of the polyline vector information filtered away especially if single pixel lines were used for the rasterized vector data. This could cause district borders to be nearly or completely filtered away when zoomed out to view an entire state. In fact, the abstract line's accuracy should be independent of the resolution of the primary imagery and also its visual representation's accuracy should not be limited to the texel resolution of the primary imagery. More complex polyline-as-texture approach is required to deal with above problems. Kersting et al. [Ker02] describe a texture-based rendering of vector data onto the level-of-detail terrain geometry. The method uses OpenGL P-buffer for rendering which allows to rasterize vector data within real-time so that on-demand generation of textures becomes practical.

The polyline-as-geometry approach allows more flexibility in polyline vector rendering as it supports interactive enabling and disabling of the display of different subsets of polyline data and interactive adjustment of their display properties. Douglass et al. [Dou99] describe a bottom-up LOD height-map rendering scheme by placing building objects over the terrain. In contrast to a top-down LOD approach, a bottom-up approach necessitates the entire model being available at the first step and therefore has higher memory and computational demands. Zachary et al. [Zac03] extend the approach of Douglass et al. to overlay vector data over multiresolution 3D terrain. It is important to note that any multiresolution modeling of vector data approach will depend on the underlying multiresolution level-of-detail terrain geometry mesh simplification scheme.

In this paper, we have proposed a new polyline-as-geometry approach, which integrates well with our dynamic multiresolution level-of-detail mesh simplification algorithm for real-time rendering. Displaying 2D polyline data on top of 3D terrain becomes challenging in our terrain visualization system for several reasons. First, we have used raster data tiling approach to handle terrain's 3D geometry and image data, as the same are too large to fit into primary memory. It requires dynamic paging of both the data types based on the current 3D view of the terrain. At any instant of time, only nine tiles, each of size 256×256 pixels, are kept in main memory based on viewer's location. Accordingly, the display of vector layer should also be limited to currently active nine tiles at a time. Second, the 2D polyline data should be treated independently from the raster data and therefore should be rendered as separate geometry by the graphics pipeline. This presents a challenge because our multiresolution LOD algorithm renders a 3D mesh whose constituent triangles of different patches are changing at nearly every frame. In order for the polyline vector data to appear overlaid on the 3D mesh, the rendered polyline geometry (height values on polyline points) must therefore also change at each frame. Fig. 1 shows the need of multiresolution modeling of vector data for geometry-based mapping.

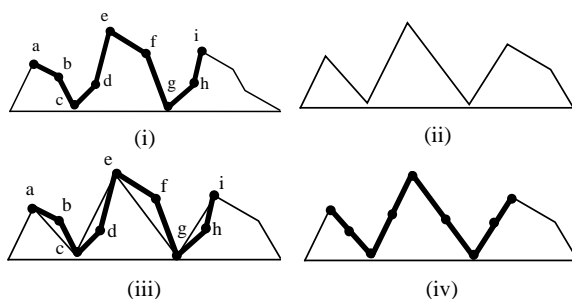


Figure 1. Geometry-based Mapping of Vector Data

Fig. 1(i) shows the 2D cross-sectional view of terrain where current geometry patch is rendered at full resolution and road polyline (bold line segments) tightly follows the terrain surface. Fig. 1(ii) shows the situation when the current patch is being rendered at lower resolution because of viewer position and screen resolution. If road polyline is displayed over the low-resolution terrain mesh without changing its geometry, then visual artifacts will be seen as shown in Fig. 1(iii). Hence, it is required to adjust height values on polyline points b, d, f, and h according to slopes of the triangles of lower resolution terrain patch (Fig. 1(iv)).

3. UNDERLYING MULTI-RESOLUTION FRAMEWORK FOR TERRAIN MODELS

We have developed a view-dependent dynamic block-based LOD modeling for mesh simplification and using tiled geospecific texture, to display the details of the high-resolution satellite imagery in real-time rendering [Agr04a]. The terrain geometry and texture data are organized in tiles of size 257×257 and 256×256 respectively. One pixel overlap is kept between adjacent geometry tiles to ensure proper stitching of tiles. At any instance of time, only nine tiles are kept in main memory. The viewer position is always assumed to be inside the centre tile. The algorithm efficiently handles out-of-core data by dynamic paging of terrain tiles between secondary storage and main memory. Each geometry tile data is organized with a quadtree with leaves corresponding to patches or blocks of size 17×17 (the size decided after experimentation) to speed up the view-frustum culling. Fig. 2 shows quad-tree based decomposition of the terrain geometry up to second level where the gray area indicates the current view-frustum seen by the camera.

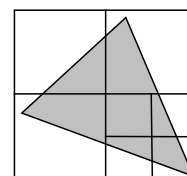


Figure 2. Quad-tree based Decomposition of Terrain Geometry

Multiresolution pyramid representation is used to define each terrain block. Fig. 3 shows the four pyramid levels of the height map block of size 17×17 . Considering the multiresolution representation for each patch, the algorithm employs a variable screen-space threshold to limit the maximum error of the projected image considering the terrain complexity, viewer distance and viewing direction as the viewer navigates the terrain. The

algorithm pre-computes a look-up table at terrain tile load time to decide the tessellation level of each block within view-frustum based on position of the camera from the block [Agr04b]. In our approach, a group of vertices are considered instead of single vertex for deciding whether to remove them or not. Hence CPU requirements are many times lower as compared to the other LOD mesh simplification algorithms, which work on individual vertices of the height field.

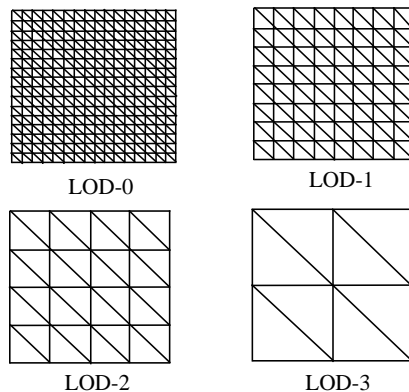


Figure 3. Multiresolution Modeling of Height Map

It is important to note that in a view-dependent framework, the resolution of adjacent patches might change at every frame. Hence, cracks occur on borders of adjacent patches of different levels of detail. In Fig. 4 (a), the circle shows the position of crack in tessellation with level difference one (right side patch is shown partially). Crack-filling methods usually involve creating additional triangles to fill in the gaps between patches, and/or modifying the geometry of one or other of the patches to produce a crack-free join. Fig. 4 (b) shows the modified geometry to remove the cracks where the dashed edges are excluded in triangulation and the bold edges are included. Similar procedure is followed to eliminate cracks when level difference between adjacent patches is two or three. Image draping over 3D mesh geometry is performed using texture mipmapping. The algorithm also handles the problem of texture seams between adjacent texture tiles [Agr04a].

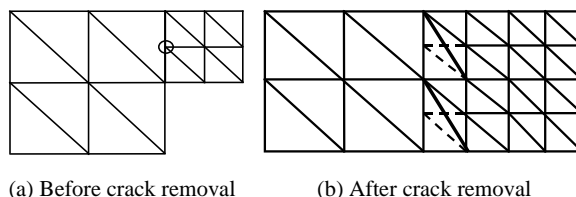


Figure 4. Removing Cracks between Adjacent Patches

To exploit the full performance of current GPUs hardware, transmission of large data chunks is advantageous. Graphics rendering can be accelerated through compact representations of polygonal meshes using data structures such as triangle strips and triangle fans. Using triangle strip primitive, it is possible to form a longer length of connected triangles as compared to triangle fan. Generating long triangle strips efficiently solves the CPU-to-card bandwidth problem and avoids redundant 3D vertex transformation and lighting (T&L) calculations. However in view-dependent meshing methods the underlying mesh is in a constant state of flux between view positions. This poses a significant hurdle to construct long triangle strips. Our triangle strip generation scheme for view-dependent dynamic multiresolution terrain shows significant improvement in rendering speed as compared to individual triangle-based and triangle-fan based rendering schemes [Agr05]. The snapshot of the triangulated height map is given in Fig. 5 using triangle-strip primitive.

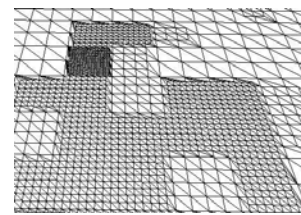


Figure 5. Wireframe View of Terrain Geometry

4. PROPOSED APPROACH TO RENDERING VECTOR DATA OVER MULTIREOLUTION TERRAIN MODELS

The proposed geometry-based mapping approach to rendering polyline vector data over the multi-resolution terrain consists of following four steps:

4.1 Vector Data Capture

The software TREND (acronym for **T**errain **R**endering) has the provision for interactively digitizing the polyline vector data. The user may open a geo-referenced map or image in a 2D display window and select option for vector digitization. The selected points on a polyline may be stored in a new vector file or may be appended at the end of an existing vector file. There are 'start' and 'end' options for polyline digitization so that user may capture different polylines in one session and save the same in a file as shown in Fig. 6.

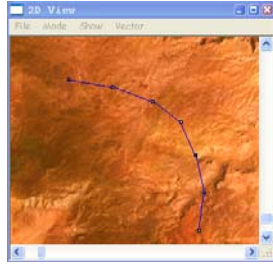


Figure 6. Digitized Polyline Vector Data

4.2 Vector Data Organization

As discussed in section 2, the display of vector data should be limited to current active nine tiles in memory at a given time instant. As the user moves over the terrain, these current nine tiles change through dynamic terrain paging scheme. Accordingly before storing the captured polyline vector data in a file, it is rasterized at highest geometry mesh resolution and has been divided into segments based on number of geometry tiles (each of size 257*257 pixels) it is passing through. A unique sequence number is also stored along with each point of the vector polyline. This helps in identifying multiple vector segments (part of same or different vector polylines) inside a tile and draws them appropriately.

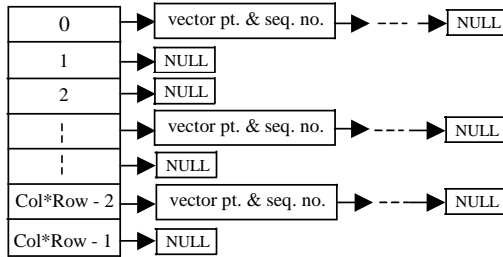


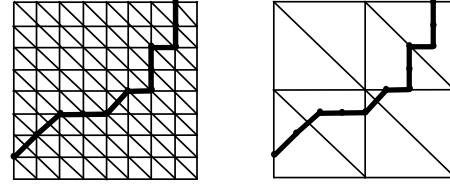
Figure 7. Tile-wise Organization of Vector Data

The software internally manages a dynamic data structure to store tile-wise vector segment details as shown in Fig. 7. When an existing vector file is loaded into memory, different vector segments are assigned to appropriate tile vector addresses. Based on the viewer position, vector data corresponding to current nine tiles is picked up for display.

4.3 Geometry-based Mapping and display

As discussed in section 3, the algorithm selects the resolution of a patch (size 17*17 pixels) using a lookup table based on its distance from the viewer. Fig. 5 shows the snapshot of a portion of displayed multiresolution mesh geometry. In our proposed approach, we map the vector data over different terrain geometry patches over which it is passing through. Height values of all the vector polyline points are picked up from the underlying geometry

patch. Now let us consider a situation when a patch is rendered at lower resolution with an overlay of a vector segment. Fig. 8(a) shows 1/4th portion of highest resolution patch (LOD-0) and Fig. 8(b) shows corresponding patch portion at lower resolution (LOD-2) both with an overlay of same vector segments.



(a) 1/4th portion LOD-0 patch (b) 1/4th portion LOD-2 patch

Figure 8. Geometry-based Mapping

When the patch is rendered at highest resolution, the vector polyline segments are displayed without any visual artifact (except in special case 1 discussed in subsection 4.4.1) because heights of all vector polyline points are matching with corresponding geometry patch heights (ref. Fig. 1(i)). However, when a patch is rendered at lower resolution, visual artifacts may occur in displaying vector data (ref. Fig. 1(iii)). In fact height values at all the vector points in Fig. 8(b) should be matched with the corresponding enclosing triangle slopes.

To compute correct height values, we first determine a plane passing through three vertices (x_i, y_i, z_i), $i=1,2,3$ of the enclosing triangle with following equation:

$$ax + by + cz + d = 0 \quad (1)$$

The height value z at vector point (x, y) may be computed by solving following determinant:

$$\begin{vmatrix} x - x_1 & y - y_1 & z - z_1 \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{vmatrix} = 0 \quad (2)$$

The height values at points on triangle edges may be computed through linear interpolation of heights at the end points. To minimize computational overhead, this process is applied only for those geometry patches through which the vector segment is passing on. The polyline vector points with updated height values will now tightly follow the terrain geometry.

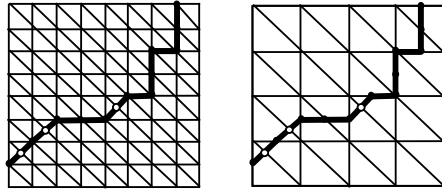
4.4 Dealing with Special Cases

The algorithm discussed in subsection 4.3 is required to be extended to deal with following two special cases otherwise the output of vector data rendering over multiresolution terrain will still suffer with some visual artifacts.

4.4.1 Case 1

When a vector segment crosses common diagonal of adjacent triangles considering patch resolution of different levels.

Fig. 9(a) shows a polyline vector drawn over $1/4^{\text{th}}$ of the full resolution patch (LOD-0) whereas Fig. 9(b) shows the same polyline vector drawn over a lower resolution patch (LOD-1) with level difference 1.



(a) $1/4^{\text{th}}$ portion LOD-0 patch (b) $1/4^{\text{th}}$ portion LOD-1 patch

Figure 9. Segments Crossing Common Diagonals

By careful examination of Figures 9(a) and 9(b), we find that a vector segment, which crosses a common diagonal of two adjacent triangles, may not be visible in some situations. Consider the case in Fig. 10. Assume that heights of points **a** & **c** are 100 and 110 respectively whereas heights of points **b** & **d** are 80 and 90 respectively. In this situation, the vector segment **bd** hides beneath the two triangular faces **abc** and **acd** of terrain geometry.

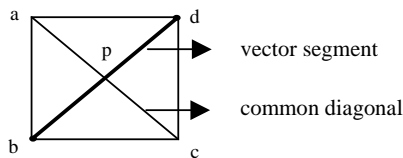


Figure 10. Vector Segment Crossing Common Diagonal

To deal with this situation, we should compute height 'h' at intersection point **p** considering heights at end points **a** and **c** and then render the vector segment **bd** as two sub-segments **bp** and **pd** with height 'h' at **p**. In Figures 9(a) and 9(b), the hollow circles show the positions where height values are to be computed at the intersection points and corresponding vector segment is to be drawn in two parts. We deal in similar manner when rendering of vector segments over two other lower resolution patches (LOD-2 and LOD-3).

4.4.2 Case 2

When a vector segment is passing through the crack-joining triangle at the boundary of two different resolution patches.

Recall that as shown in Fig. 4, the geometry of boundary triangles of higher resolution patch is altered to avoid cracks between adjacent patches of

different resolutions. The problem arises when LOD-1 patch is adjacent to LOD-2 (or LOD-3) patch as shown in Fig. 11. Here height at a vector point either inside the crack-filling triangle or on common boundary (shown as filled squares nodes) should be computed considering the modified vertices of the adapting or crack-filling triangle instead of vertices of enclosing triangle of current resolution patch.

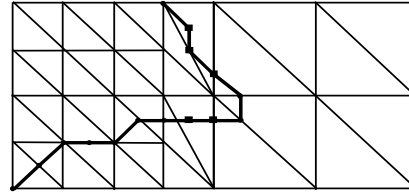


Figure 11. Vector Segments Passing through Crack-filling Triangles

We deal in similar manner when LOD-2 patch is adjacent to LOD-3 patch or when LOD-0 patch is adjacent to LOD-1 or LOD-2 or LOD-3 patch.

5. RESULTS AND PERFORMANCE ANALYSIS

The proposed algorithms have been implemented using Visual C++ and OpenGL 3D API for a Win32 environment. We have tested the software with 2K*4K terrain raster dataset of Grand Canyon and 16K*16K terrain data set of Puget Sound area obtained from Georgia Institute of Technology website. We have also generated the height map of Dehradun (India) area using digitized contours on Survey of India (SOI), India supplied topographic map. The corresponding geo-referenced IRS-1D FCC Satellite imagery has been used for image draping.

The software TREND provides a 2D window interface to digitize polyline vector data over the geo-referenced map or image in the background (Fig. 6). Figures 12(a) and 12(b) show the 3D wireframe display of terrain mesh geometry with overlay of polyline vector data and corresponding phototextured terrain view respectively using Grand Canyon dataset. Without multiresolution modeling of the polyline vector data, the visual artifacts are visible in vector data display.

Figures 13(a) and 13(b) show the views obtained after the proposed geometry-based mapping of polyline vector data over multiresolution 3D terrain as discussed in section 4.3. The visual appearance of the displayed vector data is now much improved. Fig. 14(a) shows the case when vector segments are crossing common diagonals of adjacent triangles. Fig. 14(b) shows visual artifact due to case 1 and Fig.

14(c) shows the result of algorithm proposed in subsection 4.4.1. The Fig. 15(a) shows the case when a vector segment crosses a crack-joining triangle. Fig. 15(b) shows visual artifact due to case 2 and Fig. 15(c) shows the result of algorithm proposed in subsection 4.4.2.

We have tested our software on a PC with Intel PIV 2.4 GHz CPU, 512 MB RAM, and Intel 82865G onboard Graphics Controller on 865GL motherboard. The performance of the algorithm on raster data is independent of size of terrain data as with the tiles indexing scheme, the algorithm only keeps 9 tiles active in main memory. The organization of the terrain data in tiles of defined size is required to be done only once on the same data set. For raster data, the number of frames rendered per second mainly depends on the complexity of the terrain (roughness) under the view-frustum and the user defined image quality metric (τ). Table 1 shows performance analysis of the algorithms without and with geometry-based mapping of polyline vector data overlay.

Terrain Rendering	Avg. no. of Triangles	Avg. Frames per Second
Full Resolution Raster Data (considering 9 tiles only)	1327104.00	1.72
View-frustum culled Surface	268120.70	7.18
Adaptive LOD algorithm for Raster Data ($\tau=4$)		
(a) using triangle list	20368.45	57.23
(b) using triangle fan	20368.44	74.11
(c) using triangle strip (indexed vertex array)	23733.79	130.79
Multiresolution modeling of Vector Data (total points: 573) and rendering using Triangle Strip as above	23698.74	92.76

Table 1. Performance Analysis

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a new geometry-based mapping approach to rendering map vector data over multiresolution 3D terrain models. The proposed algorithm maps polyline vector data consistently and exactly to the current level-of-detail of terrain geometry and thus minimizes the rendering

artifacts. Performance analysis results show that our combined multiresolution 3D terrain and polyline vector data display algorithm maintains real-time frame-rates on a desktop PC for large real-world terrain data sets.

Currently all vector polyline data is kept in a single file, but internally it is organized in the tile-wise order for all the terrain tiles. To reduce main memory overhead, the dynamic paging concept of tiled terrain raster data may be extended to vector data also. We may apply B-spline function to input points data to make smooth curved polyline features. Future work also includes extending the approach to display wider width polyline data (e.g. roads) with appropriate texture mapping.

7. ACKNOWLEDGMENTS

The research reported here has been partially supported by the Ministry of Human Resource Development (MHRD), Govt. of India under contract F.26-4/2002.TS.V (R&D Scheme).

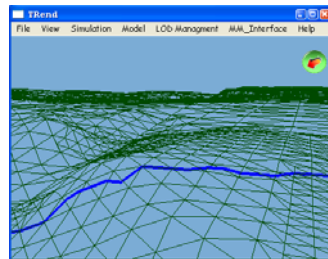
8. REFERENCES

- [Agr05] Agrawal, Anupam et al. An Approach to Improve Rendering Performance of Large Multiresolution Phototextured Terrain Models using Efficient Triangle Strip Generation. Paper presented at IEEE IGARSS-2005 (Proc. on DVD) held in Seoul, Korea, July 25-29, 2005.
- [Agr04a] Agrawal, Anupam et al. Dynamic Multiresolution Level-of-Detail Mesh Simplification for Real-time Rendering of Large Digital Terrain Models. Proc. IEEE INDICON-2004, IIT Kharagpur, India, pp. 278-282, Dec 20-22, 2004.
- [Agr04b] Agrawal, Anupam et al. TREND: Adaptive Real-time View-dependent Level-of-Detail-based Terrain Rendering. Proc. IT++: The Next Generation - the 39th Annual National Convention of Computer Society of India (CSI) held in Mumbai, pp. 146-157, Dec 1-4, 2004.
- [Agr98] Agrawal, Anupam et al. Delaunay Triangulation Based Surface Modeling and Three-Dimensional Visualization of Landforms. Journal of IETE Technical Review, pp. 425-433, 15(6), 1998.
- [Ake02] Akenine-Moller, T., and Haines, E. Real-Time Rendering. Ed.2. A.K. Peters, 2002.
- [Dou99] Douglass, D. et al. Real-Time Visualization of Scalably Large Collections of Heterogeneous Objects. Proc. IEEE Visualization'99, pp. 437-440, 1999.
- [Ker02] Kersting, O., and Dollner, J. Interactive 3D Visualization of Vector Data in GIS. Proc. of 10th ACM Int. Sym. on Advances in Geographic Information Systems, pp. 107-112, Nov. 2002.

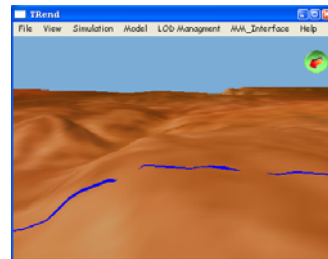
- [Lue03] Luebke, D. et al. Level-of-Detail for 3D Graphics. Morgan Kaufmann Pub., 2003.
- [Sze97] Szenberg, Flavio et al. An Algorithm for the Visualization of a Terrain with Objects. URL: http://www.tecgraf.pucrio.br/~szenberg/artigo_sib97/artigo_sib97.html.
- [Xia04] Xiaoping, R. and Yanmin, Z. Overlaying Vector Data On 3D Terrain. Proc. IEEE IGARSS

2004, Alaska, USA, pp. 4560-4563, Sept. 20-24, 2004.

- [Zac03] Zachary, W. et al. Rendering Vector Data over Global, Multiresolution 3D Terrain. Joint EUROGRAPHICS – IEEE TCV Symposium on Visualization, pp. 213-222, 2003.

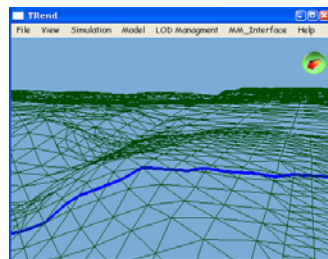


(a)

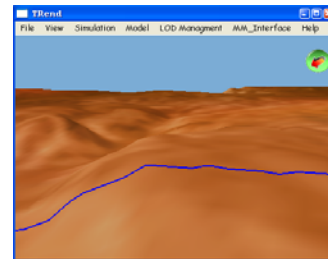


(b)

Figure 12. Display of Vector Data over LOD 3D Terrain (without Multiresolution Modeling)

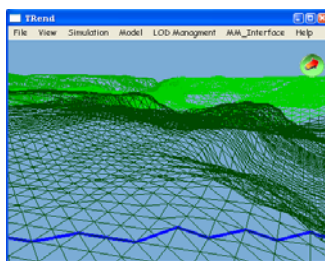


(a)

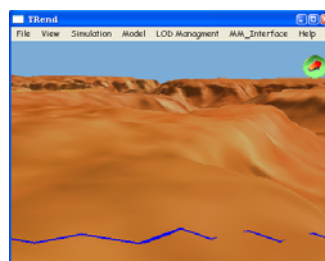


(b)

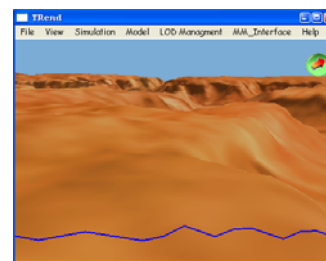
Figure 13. Display of Vector Data over LOD 3D Terrain (with Multiresolution Modeling)



(a)

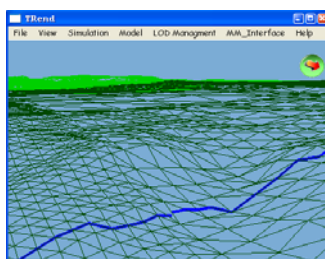


(b)

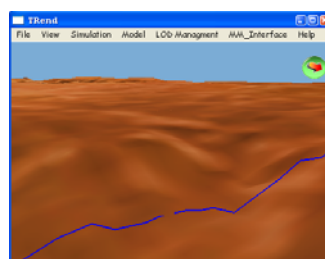


(c)

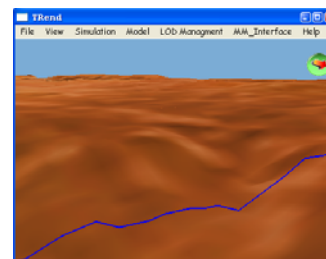
Figure 14. Handling Special Case 1



(a)



(b)



(c)

Figure 15. Handling Special Case 2

A Parallel Approach for Visualization of Relief Textures

Francisco Fonseca¹ Bruno Feijó¹ Marcelo Dreux² Esteban Clua¹

Catholic Univ. of Rio de Janeiro
R. Marquês de São Vicente, 225
Gávea, Rio de Janeiro, RJ
Brazil 22231-000

¹{ffonseca,bruno,esteban}@inf.puc-rio.br

²dreux@mec.puc-rio.br

ABSTRACT

With the continuous increase of processing power, the graphic hardware – also called Graphic Processor Unit (GPU) – is naturally assuming most part of the rendering pipeline, leaving the Central Processor Unit (CPU) with more idle time. In order to take advantage of this when rendering relief textures, the present work proposes two approaches for the mapping of relief textures. Both methods are fully implemented on the CPU leaving the GPU responsible for the per-pixel shading effects. These approaches allow the use of CPU idle time and/or multi-processed systems for the increase of real-time rendering quality and the inclusion of image-based representations.

Keywords

Relief Textures, Image-Based Rendering, Parallel Processing, Real-Time Rendering.

1 INTRODUCTION

In [Oli00], Oliveira introduces the concept of relief texture mapping as a technique to represent details of three-dimensional surfaces. While the traditional texture mapping technique [Cat74] does not consider view-motion parallax and, consequently, only reveals the two-dimensional nature of the texture, Oliveira's approach supports the parallax mechanism and permits the user to have a 3D texture perception. However, as it does not store sufficient geometric information about the details that are being simulated, the technique proposed by Oliveira does not allow the correct representation of non-diffuse surfaces. Moreover, the overhead introduced by the pre-warping step [Oli00] makes it difficult to be used in real-time rendering applications that require high

frame rates, such as games.

Some works have been proposed in order to improve the relief texture mapping technique. For instance, Fujita and Kanai [Fuj02] use the capability of the GPU programming to extend the relief texture mapping technique so that it can support per-pixel shading effects, such as normal mapping and reflection mapping. Although this approach achieves successful results by using shading effects, it does not appropriately work with high frame rate requirements.

Policarpo *et al.* [Pol05] implement the original relief texture mapping on the GPU. In their approach, the view direction is transformed to the texture space and a linear search is performed in order to find an intersection between the view direction and the virtual surface (represented by a depth map). Moreover, this process is improved through a binary search. Their approach complies with real-time requirements and supports per-pixel shading effects.

As the power of GPUs are rapidly increasing – in a faster pace than the power of CPUs – there is an inclination for them to assume almost the entire rendering pipeline processing work, leaving the CPU with more and more idle time. The objective of this work is to investigate the possibility of taking advantage of the CPU's idle time when rendering

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.

F[beg6b` ` ha Vtgbaf`cebWwWzF5A`+ #Z+), ' &Z#(Z
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

relief textures in order to obtain results as good as those obtained in the approach applied by Policarpo *et al* in [Pol05].

The contribution presented in this paper consists in two parallel approaches that optimize the processes involved in relief texture mapping. In some cases, the parallelization of the relief texture mapping algorithm represents an acceleration of up to 300% compared to the conventional technique.

2 RELIEF TEXTURE MAPPING

A relief texture is an image, obtained by an orthographic projection camera, which contains depth information. In a more formal way, a relief texture is a pair $\{i, K\}$, where i is a digital image and K is an orthographic projection camera model associated with i . Each color element of i is augmented in order to include a scalar value that represents the distance (depth), in the Euclidean space, between the sample and a reference entity. Since K is an orthographic projection camera model, the reference entity is the projection plane of K .

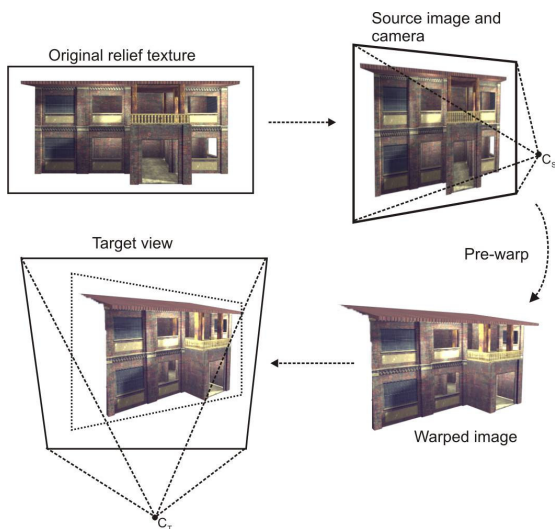


Figure 1. relief texture mapping process [Oli00].

At an implementation point of view, each element of i from a relief texture may be represented as a RGBA information, where RGB channels store color values while alpha channel stores depth values. The information about the camera model K may be represented as a 4x4 matrix.

In 1997, Leonard Mcmillan introduced the concept of three-dimensional image warping [Mcm97], which is the basis for the relief texture mapping technique. 3D image warping consists of geometric information that maps a $\{i_1, K\}$ source image with depth onto a i_2 target image, allowing a correct visualization of the i_1 image from several view points.

It is important to notice that 3D image warping may be interpreted as a composition of two two-dimensional transformations: a planar perspective transformation and a per-pixel shift in the direction of the epipole of the target view. Hence, the relief texture mapping can be seen as a factorization of a 3D image warping into these two transformations. That factorization proposed by Oliveira [Oli00] allows the planar perspective transformation, which consists essentially in a texture mapping operation, to be applied after the per-pixel shift (pre-warping). So, through the factorization, it is possible to benefit from the texture mapping implemented in hardware in order to make the final transformation (Figure 1). More details about that process may be obtained in Oliveira's Ph.D. thesis [Oli00].

2.1 Implementation

As defined in Section 2, a relief texture is composed of color, depth and camera information. However, with only such information it is not possible to capture effects that depend on the view point and illumination direction. A feasible solution to represent such effects is to use normal maps in conjunction with relief texture mapping.

Thus, normal information is merged with depth information in order to generate a normal map with depth, which is represented by an RGBA image, where the alpha channel stores a depth value and the color channels store the normal vector. The color information is stored in a conventional texture map, represented by an RGB image.

The implementation approach used to perform the relief texture mapping consists of a five-step algorithm, which may be represented by the diagram of Figure 2.

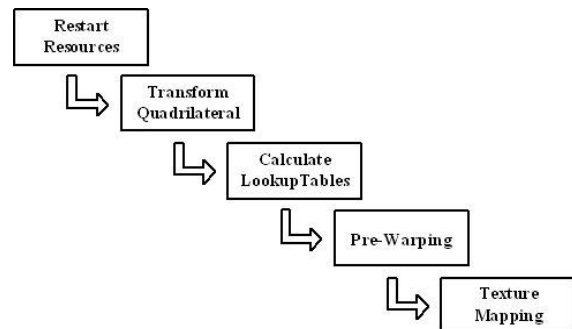


Figure 2. Diagram of the relief texture mapping algorithm.

The algorithm input data are: the current observer position p , a quadrilateral q and a relief texture $\{i_1, K_1\}$, where i_1 represents a normal map with depth information and a conventional texture map. Firstly, it is necessary to initialize image buffers and lookup tables. As the graphics application works in a loop fashion, in which the resultant image is updated

every frame, this step becomes necessary in order to avoid reading the information generated during previous executions.

When this step has finished, the parameters of the quadrilateral q are transformed according to the current viewing configuration. After that, some lookup tables are computed in order to avoid repetitive operations, hence optimizing part of the process.

The next step, called pre-warping, is the core of the algorithm and is the step that consumes more time and computational resources. This step is responsible for creating an output image i_2 that represents a partial visualization of the mapping from $\{i_1, K_1\}$ onto q viewed from p .

Thus, the content of the image i_2 may be stored into the GPU texture memory and, finally, it may be mapped onto quadrilateral q in order to produce a correct visualization. More details about this implementation may be obtained at Fonseca's MSc. dissertation [Fon04].

3 PARALLEL PROCESSING

In the context of real-time graphics applications, the rendering pipeline is usually divided into three conceptual stages: application, geometry and raster [Möl02]. Nowadays, both second and third stages are fully implemented in the graphics hardware, while the first stage is implemented on the CPU.

This division may also be used to represent the whole relief texture mapping computation. The diagram in Figure 3 illustrates the relief texture mapping pipeline stages according to the approach described in Section 2.1.

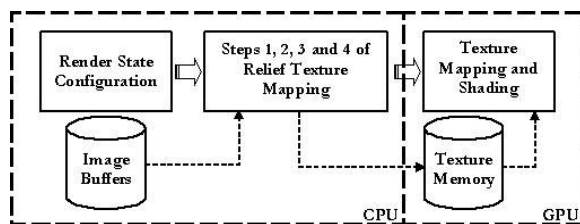


Figure 3. Conventional process for relief texture mapping.

In this diagram, the application stage is executed in a sequential fashion and comprehends all the steps performed on the CPU. The geometry and raster stages are implicitly represented as *texture mapping* and *shading* steps, both executed on the GPU.

By definition, the speed of a pipeline is determined by the slowest stage, independently how fast are other stages. In general, the slowest stage is known as the bottleneck.

According to Akenine-Möller & Haines [Möl02], the first step of a pipeline optimization process consists in locating the bottleneck. This localization is accomplished by a set of tests, such as those described in [Möl02].

Analyzing the algorithm proposed in the Section 2.1, the strongest candidate to be the bottleneck of the relief texture mapping process is the application stage, since all texels of the normal and color maps are processed every frame. For this reason, it is decided to make the referred tests to the application stage.

One way to verify if the application stage limits the rendering speed is to send data through the pipeline in such a way that the other stages perform little or no work. In OpenGL this can be achieved substituting every call to `glVertex3f` and `glNormal3f` by the call to `glColor3f`. By doing this, the work of sending data from the CPU remains unchanged, while the work of sending and receiving data on the geometry and raster stages are drastically reduced. If the performance does not improve, it is possible to affirm that the bottleneck is the application stage. The authors of the present paper applied this test to some input textures and verified that the application stage is actually the bottleneck.

In the light of the above mentioned considerations, the present authors propose the implementation of two parallel approaches for the relief texture mapping computation. The next section describes, for each approach, the rationale of the implementation and the proposed methodology.

3.1 Parallel Approach

With the intention of optimizing the CPU process, a CPU thread is created. This thread has the capability of running the four first steps of the relief texture mapping algorithm. With the aid of the Hyper-Threading technology [Mar03], this thread can be executed in parallel with the conventional CPU process and hence allowing a considerable time saving. The diagram of Figure 4 illustrates the new approach.

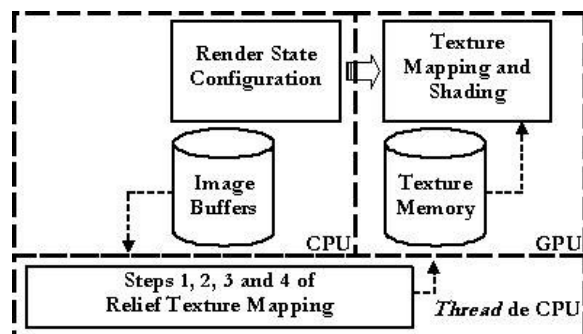


Figure 4. Parallel approach diagram.

In order to guarantee that the final result is correct, besides creating a CPU thread, it is necessary to synchronize it with the rest of the processes that are being executed. This synchronization is represented by the state machine illustrated in Figure 5.

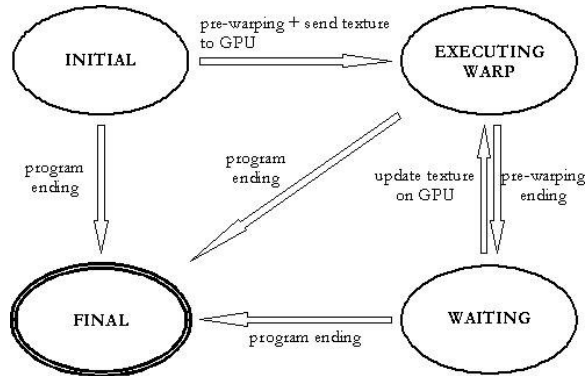


Figure 5. CPU thread state machine.

At the moment of the thread creation, the thread state is set to *initial*. The transition between the *initial* state and *executing warp* state is done by the first warping operation followed by the transmission of the resultant image to the GPU texture memory. During the rest of the execution, the state machine stays in a loop between the *execution warp* and *waiting* states, except at the end of the program, when the CPU thread goes to the *final* state. The pre-warping operation is performed during the *execution warp* state. When it finishes, it changes to *waiting* state. In this state, the resultant new image is sent to the texture memory and its content is refreshed. As soon as the texture memory is updated, the state is set to *executing warp* and it indicates that a new warping operation must be performed.

With the Hyper-Threading or similar technology, a processor may be exclusively assigned to the CPU thread, without affecting the rendering pipeline performance.

The transition from *initial* state to the *executing warp* state occurs during the execution of the main draw function, performed once a frame. An algorithm that describes this process is presented below.

```

algorithm draw()
1 Configure the OpenGL API rendering state;
2 Activate vertex program;
3 Activate fragment program;
4 If it is the first execution of draw then
5   Start resources;
6   Transform the quadrilateral;
7   Initialize lookup tables;
8   Perform pre-warping;
9   Send processed texture to the GPU;
  
```

```

10 stateThread ← EXECUTING_WARP;
11 else
12   If stateThread = WAITING then
13     Update texture on the GPU;
14     stateThread ← EXECUTING_WARP;
15   end-if
16 end-if
17 Draw quadrilateral with the processed texture;
18 Deactivate fragment program;
19 Deactivate vertex program;
end
  
```

At line 1, the OpenGL API rendering state is configured. This configuration consists in enabling operations such as blending, culling, and depth tests. In lines 2 and 3 the vertex and fragment programs are enabled in order to calculate per-pixel illumination. The first time that the draw function is executed, the four first steps of the relief texture mapping algorithm (lines 5 to 8) are performed and the resultant texture from the warping is sent to the graphics card, through the OpenGL `glTexImage2D` function. Once the texture is sent to the graphics card, the state changes from *initial* to *executing warp* (line 10). After the second execution of the draw function, it is necessary to verify (in each cycle) if the current state of the CPU thread is *waiting*. In that case, it means that the thread has executed a warping operation and, consequently, the texture in the graphics card must be updated (line 13). So, the update is achieved by the OpenGL `glTexSubImage2D` function. After that, the CPU thread changes from *waiting* to *executing warp* state and a new warping operation must be performed. Finally, at line 17, the quadrilateral with the mapped texture is drawn and, in lines 18 and 19 the fragment and vertex programs are deactivated. It is important to notice that during every execution of the draw function, the CPU thread is executed in parallel.

The remaining state transitions are described by the `threadCPU` algorithm.

```

algorithm threadCPU()
1 While true do
2   If stateThread = EXECUTING_WARP then
3     Start resources;
4     Transform the quadrilateral;
5     Initialize lookup tables;
6     Perform pre-warping;
7     stateThread ← WAITING;
8   else if stateThread = FINAL then
9     return;
10  end-if
11 end-while
end
  
```


The algorithm is basically a loop: line 1 guarantees that the thread is being executed until its current state is modified to the *final* state. During this loop, if the current state is *executing warp*, the four first steps of the relief texture mapping algorithm are executed (lines 3 to 6) and the state is assigned to waiting (line 7), which indicates that a texture updating operation must be performed. When the state is equals to *final*, the `threadCPU` function ends.

3.2 Multi-Threaded Approach

Besides the previous approach, the use of Hyper-Threading technology allows the elaboration of a parallel process for different parts of input data. More specifically, it is possible to divide the $\{i_1, K_1\}$ input texture into two parts and to simultaneously execute the four first steps of the relief texture mapping algorithm for each part. Consequently, a post-processing becomes necessary in order to unify the resultant textures into one unique texture that will be mapped into quadrilateral q . The diagram in Figure 6 illustrates that process.

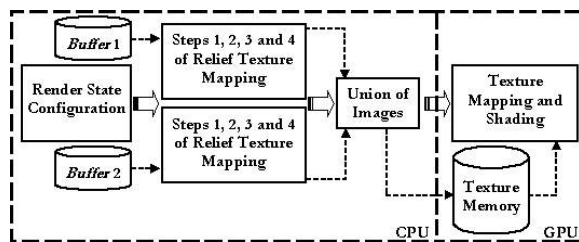


Figure 6. Multi-Threaded process diagram.

Firstly, it is necessary to divide the image buffer into two different parts. It is important to notice that the image buffer consists of a normal map, a depth map and a color map used as input data to the algorithm. Although it is not obvious, both resultant images of the division process must have the same dimension as the original one. For instance, the left half of the first resultant image will be filled with the left half of the original image content, while the right part of the resultant image will be filled with invalid values (i.e. values that represent background information, as illustrated in Figure 7). This is necessary because during the pre-warping process some texels could exceed the limits of the plane that supports the image. Consequently, the resultant images of the pre-warping process would contain an incomplete vision of the representation (see Figure 8).

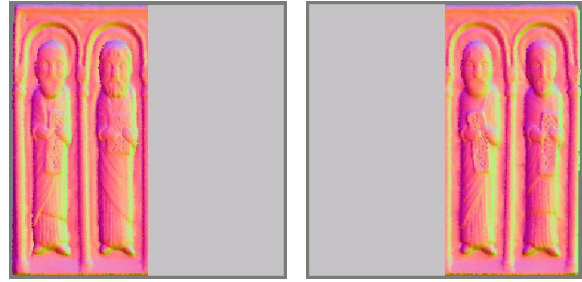


Figure 7. Illustration of the division process of the normal map.



Figure 8. The incomplete vision of the represented surface occurs because texels that exceed the limits of the support plane during the warping are not considered.

During the union process of both resultant images, some parts of the result may have overlapping texels in regions where the limits of the support plane of the image were exceeded. The algorithm will correct this problem discarding the texels that have invalid values in the places where this overlapping occurs.

The `draw` function, described in Section 3.1, must be modified in order to correspond to that new approach:

```

algorithm draw()
1  Configure the OpenGL API rendering state;
2  Activate vertex program;
3  Activate fragment program;
4  If it is the first execution of draw then
5    Execute the four first steps for left half;
6    Execute the four first steps for right half;
7    Unify the resultant images;
8    Send processed texture to the GPU;
9    stateThread1 ← EXECUTING_WARP;
10   stateThread2 ← EXECUTING_WARP;
11 else
12   If stateThread1 = WAITING and
       stateThread2 = WAITING then
13     Unify the resultant images;
14     Update texture on the GPU;

```

```

15   stateThread1 ← EXECUTING_WARP;
16   stateThread2 ← EXECUTING_WARP;
17   end-if
18 end-if
19 Draw quadrilateral with the processed texture;
20 Deactivate fragment program;
21 Deactivate vertex program;
end

```

The differences from the function described in Section 3.1 are basically located between lines 5 and 16. The first time that the `draw` function is executed the four first steps of the relief texture mapping algorithm (lines 5 and 6) are performed for both halves of the original image. Once the warping process is concluded, the union of the resultant images may be executed. After this, the resultant texture from that union is sent to the graphics card. As soon as the texture is sent, the transition from the *initial* state to the *executing warp* state can be made (lines 9 and 10) for each thread. In cases where the `draw` function is executed again, it is necessary to test if the current state of the CPU, for both halves, is *waiting*. It means that the tread completed a full warping operation and, consequently, the result of each warping may be unified so that the texture in the graphics card may be updated (line 14). Following it, the state of each CPU thread changes from *waiting* to *executing warp* and a new warping operation must be performed.

4 RESULTS

This section presents statistics of elapsed time of the implemented algorithms as well as the obtained visual results. These measures were made with an Intel Pentium IV PC with 2.66 GHz and 512 Mb of memory RAM and a graphics card GeForce FX 5600 with 256 Mb of video memory.

Three samples have been used, as shown in Figure 9 at the end of the section. Table 1 presents some specifications for the used samples such as, resolution (in pixels) and the number of texels with invalid depth values (i.e. texels that represent background information).

Table 2 presents the frame rate for each one of the implemented approaches. The sequential approach (S) represents the conventional technique of relief texture mapping, while the parallel (P) and the multi-threaded (M) approaches are the algorithms proposed in this work.

Sample	Resolution	Invalid Texels
1	256x256	3800 (5,80%)
2	256x256	3223 (4,92%)
3	256x256	30484 (46,51%)

Table 1. Samples information.

	Sample1		Sample2		Sample3	
	FPS	SD	FPS	SD	FPS	SD
S	24.18	0.86	24.93	1.54	38.24	1.86
P	97.97	4.50	96.45	5.01	103.64	4.88
M	33.14	6.02	31.09	7.67	34.51	8.21

Table 2. Frames per second average (FPS) and standard deviation (SD) for sequential (S), parallel (P) and multi-threaded (M) approaches.

Analyzing Table 2, it is possible to conclude that the parallel approach is better than the sequential one. However, the sequential approach is the more stable method in relation to the frame rate average. It may be verified by the small dispersion of the data presented by the obtained standard deviation.

Only in the case of sample 3 the multi-threaded approach is worse than the sequential one. There are two points that must be considered in order to explain that fact. The first one refers to the sample 3, which is a special kind of sample, where the number of invalid texels is large (see Table 1) and, therefore, the pre-warping step effort is inferior in relation to samples 1 and 2, since only valid texels need to be processed. The second point refers to the multi-threaded approach, which has an overhead of three processes being executed in a two processor system. So, it is possible to conclude that the inherent overhead related to the multi-threaded approach was superior to the gain obtained by the optimization of the pre-warping step in the case of the sample 3, demonstrating the inferiority of the multi-threaded approach in relation to the sequential one in this specific case.

It is important to notice that high frame rates per second obtained by the parallel approach do not reflect the number of times that a warping operation is being performed. These rates refer to the number of times that the images are being rendering per second, independently if a new warping operation is being executed or not. It occurs because the warping process is being executed in parallel to the rest of the pipeline and, consequently, sometimes it is possible to notice a progressive update in the rendered image when there are many camera movements.

The superiority of the parallel approach in relation to the multi-threaded one was already expected, since the latter consumes a reasonable time during the realization of the texture union operation.

The obtained visual results are presented in Figure 10, which are the same for all approaches.



Figure 9. Samples 1 (left image), 2 (right image) and 3 (bellow image).

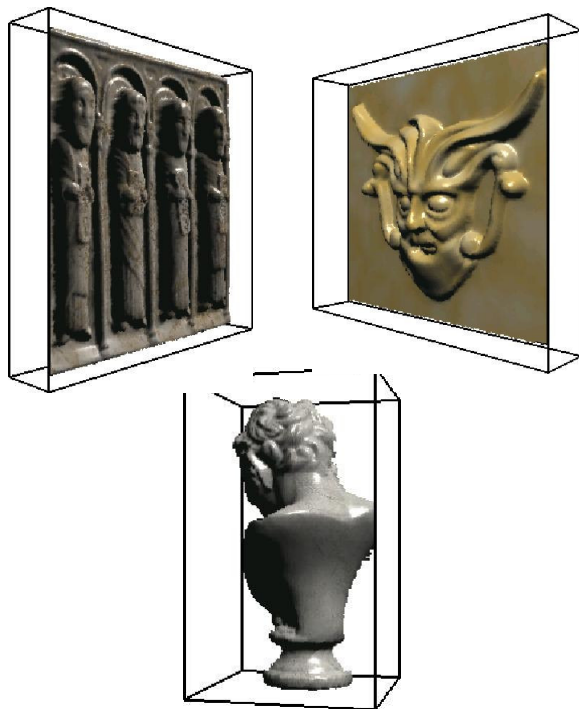


Figure 10. Visual results for sample 1, sample 2 and sample 3.

5 CONCLUSIONS AND FUTURE WORKS

In order to optimize the relief texture mapping process, two new approaches have been implemented:

- **Parallel.** In the parallel approach a CPU thread was created with the objective of resolving the first four steps of the relief texture mapping algorithm. Such thread is executed in parallel with the

main CPU process, using for this propose the Hyper-Threading technology.

- **Multi-Threaded:** In this method, the input texture is divided into two parts with the objective of simultaneously running the four first steps of the relief texture mapping algorithm for each part. It becomes necessary a post-processing to unify both results into a single resultant image.

It is possible to conclude that the parallelization of the relief texture mapping considerably speeds up the process in comparison to the conventional methods (up to 300%), i.e., with the parallelization the relief texture mapping may be implemented in real-time on the CPU and thus allowing the GPU to be used only for shading calculations. The performance and image quality of the proposed approach are similar to the ones obtained by Policarpo et al [Pol05].

In relation to future works, two possible optimizations could be incorporated: utilization of a pipeline of multi-processors in the relief texture mapping and a variant to the multi-threaded approach that processes rows and columns in parallel.

Furthermore, a better analysis of the warping per second rate could be done in relation to the frame per second rate (where wps corresponds to the number of warpings that are being performed per second). This analysis could allow a more efficient implementation of the process (with less warping operations), in the case of the wps to be larger than the fps rate.

Finally, an implementation that uses many CPU processors could be developed in order to take more advantages of the described techniques.

6 ACKNOWLEDGEMENTS

The authors would like to thank Fabio Policarpo for his help during the development of this work. The first author thanks CAPES for the scholarship used during the development of this work. Moreover, the second author thanks the support for the research projects under the following contracts: CNPq Grant PQ No. 305982/2003-6, SEPIN-CNPQ-FINEP No. 01.02.0235.00 (Ref. 2425/02) and FINEP No. 01.04.0945.00 (Ref. 3110/04).

7 REFERENCES

- [Cat74] Catmull, E. *A Subdivision for Computer Display of Curved Surfaces*. December 1974. Thesis (Ph.D in Computer Science). Department of Computer Science, University of Utha, Utha, 1974.
- [Fon04] Fonseca, F.M. A. *Relief Textures using Per-Pixel Illumination and Parallel Processing*. January 2004. Dissertation (MSc in Computer Science). Department of Computer Science,

- Catholic University of Rio de Janeiro, Rio de Janeiro, 2004 (in portuguese).
- [Fuj02] Fujita, M and Kanai, T. Hardware-Assisted Relief Texture Mapping. In: European Association for Computer Graphics (EUROGRAPHICS). 2002. *Proceedings of the annual conference of the European Association for Computer Graphics* Saarbrücken, Germany, 2002.
- [Mar03] Marr, D. T. *et al.* Hyper-Threading Technology Architecture and Microarchitecture. *Intel Technology Journal*. v. 6, n. 1, p. 4-152, fev. 2003.
- [Mcm97] Mcmillan, L. *An Image-Based Approach to Three-Dimensional Computer Graphics*. April 1997. Thesis (Ph.D in Computer Science). Department of Computer Science, University of North Carolina, Chapel Hill, 1997.
- [Möl02] Akenine-Möller, T.; Haines, E. *Real-Time Rendering*. 2. ed. Massachusetts: A K Peters, 2002. 482 p.
- [Oli00] Oliveira, M. M. de. *Relief Texture Mapping*. March 2000. Thesis (Ph.D in Computer Science). Department of Computer Science, University of North Carolina, Chapel Hill, 1997.
- [Pol05] Policarpo, F.; Oliveira, M. M.; Comba, J. L. D. *Real-time relief mapping on arbitrary polygonal surfaces*. Proceedings of the Symposium on Interactive 3D Graphics and Games, Washington, District of Columbia, 2005.

Visualization of Atomistic Simulation Data for Spatio-Temporal Information

Dipesh Bhattarai and Bijaya B. Karki

Department of Computer Science

Louisiana State University

Baton Rouge, LA 70803, USA

ABSTRACT

We have proposed a scheme to support interactive visualization at space-time multiresolution of the atomistic simulation data. In this scheme we have adopted two perspectives that differ in their purposes and in the way they process and render the data. First, the complete or nearly complete dataset is rendered using animation, particle-pathline and color-mapped-dimension techniques to achieve an overall idea of the spatio-temporal behavior of the atomic system under consideration. Second, additional data are generated on the fly and analyzed/visualized using a combined graph-theoretic and statistical approach to gain better and more detailed insight into the desired spatio-temporal information. It is also shown that the proposed approach can greatly assist us to better understand various important atomistic (molecular) properties and processes including bond-breaking/reconstruction, radial distribution, atomic coordination, clustering, structural stability, defects and diffusion.

Keywords

Scientific visualization, Molecular dynamics, Space-time dataset, Cluster analysis

1. INTRODUCTION

Adoption of visualization approach towards better understanding of a given real material system at atomic (molecular) scale has drawn much attention over last several years [Hum96, Kokaji99, Kokaji03, Crysm, Sharma03]. One major challenge is how to extract as much spatio-temporal information as possible from highly accurate data produced by computationally intensive quantum mechanical simulations. The first principles molecular dynamics (FPMD) algorithm based on interatomic interactions derived within the quantum mechanical framework (such as density functional theory) can typically deals with several tens to several hundreds of atoms [Codes]. Typical simulation times are in picoseconds, which translate to a few thousands of FPMD steps. Thus, the simulation data are three-

dimensional (discrete degrees of freedom defining the positions of atoms) and time-dependent in the nature.

In this paper, we propose an effective-efficient scheme of visualization of the atomistic simulation data at space-time multiresolution. To the best of our knowledge, no such systematic approach has previously been proposed although a large number of atomistic (molecular) visualization studies have been reported over the years (see the Related Work Section). Here, we only deal with data sets produced by FPMD simulations, which are generally much smaller than the data sets produced by the classical molecular dynamics (MD) simulations [sharma03] based on empirical or simplified interatomic interaction potential model. However, given the high accuracy of the FPMD data, it is important to extract detailed quantitative and qualitative information hidden in the data. For this purpose, instead of just focusing on direct rendering of the given data, additional data (containing more quantitative information) that usually have to be extracted by some other means are extracted and rendered on the fly. This allows us to gain better insight at multi space-time details of the data (in other words, to extract the global as well local spatio-temporal behavior) for important information such as bonding, pair correlation, coordination, structural units and complex clustering, structural stabilities, defects, diffusion and other dynamic processes.

In the past, the first-principles simulations were mostly performed for crystalline systems in which

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.

SHORT COMMUNICATION proceedings ISBN 80-86943-03-8

*WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.*

Copyright UNION Agency – Science Press

the symmetry constrains the simulation size (often unit cells with periodic boundary conditions are used) as well as the atomic configuration (a few free positional parameters need to be considered explicitly). For example, a widely studied material, MgO in its conventional simple cubic structure requires only 8 atoms whose positions are fixed by the cubic symmetry. Now due to the rapid advances in the computational resources/capabilities, the computational studies of defective crystals (such as ones containing vacancies, impurities, dislocations or grain boundaries) and disordered systems such as liquids are increasingly becoming common and precise [Chel01, Karki05, Stix05, Alfe05]. Such parallel simulations often require much larger atomic systems and the crystal symmetry is violated partially or completely. For example, the data for the liquid MgO used in this study were obtained for the atomic system consisting of 216 atoms and a total of 648 positional degrees of freedom need to be explicitly taken into account [Karki05].

The atomic configurations of our interest are represented by the simulation-produced time-varying 3D data. Additional attributes such as charge density, energy, forces, stress- and strain-fields are often associated with these atomistic data. Here, our proposed multi space-time resolution approach deals only with the positional data for constituent atoms as a function of time. This approach is expected to represent a more complete visualization than what currently exist and is also expected to be of an increased importance, particularly in the case of FPMD simulation data sets to extract a variety of structural and dynamical information in both qualitative and quantitative manner. Such knowledge is highly desirable as it forms a basis for understanding different macroscopic properties of materials of substantial scientific and technological importance.

The paper is organized as follows: The Section 2 contains a quick review of previous related work. We present our approach in the Section 3 and explain the proposed visualization model with the details of various rendering algorithms incorporated into it in the Section 4. Important implementation and performance details are presented in the Section 5. Finally, the Section 6 concludes with future directions.

2. RELATED WORK

A lot of work has been done in atomistic (molecular) visualization and analysis for which numerous applications, both commercial and public domain, currently exist. The common examples include VMD [Hum96], Molscript [Krau91], XcrysDen [Kokaji99, Kokaji03], CrystalMaker [Crysm] and amiraMol [amira]. Both Visual Molecular Dynamics (VMD), which were originally designed for biomolecular systems can now be used for general purpose.

XCrysDen can visualize crystalline and molecular structures with ability to superimpose the charge densities on them. Similarly, commercial packages such as CrystalMaker and amiraMol support similar level of interactive 3D visualization, the former being able to handle up to a couple of billions of atoms. Besides, examples of the specific-purpose programs are also abundant [Li05], which include Rasmol, AtomEye, Aviz, Atomsviewer, gOpenMol, VASP DataViewer, PyMD, a few to name. Majority of these applications exploit 3D computer graphics suitable for personal computers and some also support immersive and interactive visualization requiring specialized hardware such as CAVE or Immersadesk system.

In general, the existing visualization systems share many features given below:

- A wide variety of representation (rendering and coloring) modes such as balls, points-and-lines, balls-and-sticks, space-fill and polyhedra
- Real-time manipulations such as rotation, translation, scaling and selection
- Measurement of distances, angles, and dihedrals
- Crystalline properties (e.g., switching between primitive and conventional cell settings, displaying the Wigner-Seitz cell and Brillouin-zone)
- Animations (particularly, for MD simulation trajectories) imported either from files or from a direct connection to a running MD simulation
- Support for a variety of database files from a variety of simulation and experimental sources.

To the best of our knowledge, these visualization systems have a little, if not at all, capability to visualize data at multiple-time and length scales. Several of them are designed to be of general purpose and for that very reason they come short of fulfilling various domain specific requirements. Given a high accuracy of the spatio-temporal atomistic data, our objective here is to extract as much information (both visual and statistical) as possible. For instance, we want to visualize coordination, clustering behavior and diffusion, both at local and global scale and varying with time.

3. VISUALIZATION APPROACH

In a typical molecular dynamics (MD) simulation [Allen87], initially a system is defined by a group of properties that represent atomic types, atomic positions, and constraints that have to be satisfied. Let S be the collection of $\{P, T, C\}$ describing the system, where $P = \{p \sqcap X \sqcap R^3\}$ (X demarcates the simulation box) is the set of the atomic positions in the system, $T = \{t \sqcap Types\}$ is the set of the atomic types, and C is the set of constraints, which have to

be satisfied by the system. As the simulation progresses, during each intermediate step, a new system configuration is generated, i.e. over each MD step. Collection of S 's over the simulation time duration describes the system dynamics, i.e., $D = \{S_i\}$. We approach the atomistic visualization problem in the following three ways.

3.1 Spatial Proximity Analysis

Given a snapshot of the system dynamics, S_i , it is possible to gather information about the spatial proximity of individual atoms to each other, and their collective tendency to form a structure. The structure can range from a regular polyhedral unit to a more complex cluster. In order to fully detect and characterize such structures, one should be able to visualize spatial variations in bond-lengths, bond-angles and coordination environment. It is likely that the bonding and coordination environment for a given atom type is not uniform throughout the system. Radial distribution function, coordination number and compositional disorder are computed and used in subsequent clustering. Also, polyhedral structures may vary in their types and degrees of distortion, and in their arrangements; whether they exist in isolation or form a network. Clusters of different sizes and shapes are computed and visualized.

3.2 Temporal Proximity Analysis

As the simulation progresses, the constituent atoms move in different directions to different extents. A common way of representing the dynamics is animation, which renders the atoms in 3D space as a function of time. Another way is quantification and visualization of the extent of a constituent atom's drift from its initial position or mean position to give further insight into the dynamics of atoms at different times. Temporal clustering for individual atom is then quantified by finding a sphere which is just big enough to enclose all the positions of the atom in 3D space. Center of such a sphere is the centroid of the set of positions an atom occupies during the simulation, and radius is the distance to the farthest position from the center. We also use path-lines - the trajectories of individual atoms in time - to gather information about the dynamics of the individual atoms.

3.3 Spatio-Temporal Analysis

The spatial proximity analysis or temporal proximity analysis alone does not show the complete picture of the system structure and dynamics. It is necessary to integrate together the two types of analyses to visualize how the proximity relationships among the atoms (spatial information) vary over time (temporal information) using animation. For instance, an animated system shows the formation and breakage

of bonds and their variation over the time thereby giving overall visual perception of the dynamics. Other properties such as pair correlation, coordination environment, structural units and clustering are also animated. The animation in this way, however, gives an instantaneous view of the dynamics. The MD simulation represents a statistical ensemble. So, it is preferable to visualize the time-averaged and finite-time-span behavior of such an ensemble. For instance, it is interesting to see whether a given structural unit is stable during the whole simulation period. If not, we better know what percentage of time and how many times, the structure is on the scene.

4. VISUALIZATION MODEL

Our visualization model adopts different rendering and analysis techniques [Sch97] and hence it consists of several modules to support the three-types of analyses introduced earlier in the Section 3. The current model represents a major step towards these endeavors. It is worth to mention two important points related to how the model deals with data. First, some modules directly render the given positional data whereas others render the new data, which are extracted from the original data on the fly. Second, some modules render the complete or significant amount of the data at a given instant whereas others render a subset of the data (local in space and time). Thus, our various visualization/analysis modules can broadly be put into two categories, namely, *complete data rendering* and *local/extracted data rendering*. For demonstration purposes, we use the FPMD simulation data for MgO and MgSiO₃, which are among the most widely studied materials due to their substantial geophysical significance for the Earth's mantle. Liquid phases and defective systems are analyzed using the 3D positional data for the systems containing 64 and 216 atoms for MgO, and 80 and 160 atoms MgSiO₃ [Karki05, Stix05]. Data per simulation have been collected over up to 5000 FPMD steps.

4.1 Complete Data Rendering

Animation

Animation is a widely used technique to render any type of time-dependent data for a rapid navigation through the data. This gives a quick overview of the spatio-temporal behavior of the system. Our model renders atoms at their respective positions for one FPMD step at a time using spheres of different radii and colors differentiating their atomic types. Animation is supported in the ball representation (showing only atoms), ball-stick representation (showing inter-atomic bonds and their formation/breakage) and polyhedral forms. The bonds are color coded according to their lengths

whereas the polyhedra are color-coded according to the degree of the distortion from their ideal structure. In addition, animation is also coupled with rendering of several other spatio-characterization properties such as radial distribution function, coordination, displacement data, structural units and clusters.

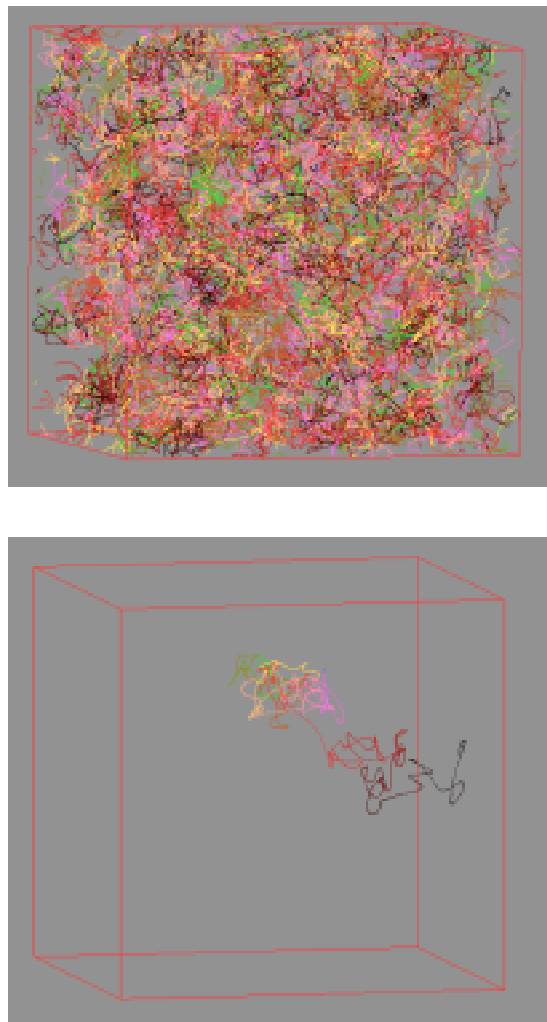


Figure 1: Pathlines of all 64 atoms (top) and the pathline of one selected atom (bottom) of the liquid MgO at 4000 K. The bottom clearly shows that the particle wanders in different regions as time elapses.

Pathlines

The complete trajectories of all or selected atoms are displayed as pathlines (Figure 1). The pathline technique is widely used to represent the flow of particles [Sch97]. As an atom moves within the supercell satisfying the periodic boundary conditions, its path is traced using a color-coded line. The pixel color at the position is varied according to the elapsed time. This allows us to easily visualize the history and extent of the atomic movement.

Color-mapped dimension

We propose a color-mapped dimension scheme to reduce the four-dimensional space-time data into two-dimensional representation. In this scheme, a given atomic position or displacement is mapped to a color with its red-component representing the x-coordinate, green-component representing the y-coordinate and blue-component representing the z-coordinate. The screen coordinates, horizontal and vertical screen axes, are used for representing simulation time step and the atoms, respectively. The pixels along some vertical line (parallel to the y-axis) correspond to different atoms at a given time whereas the pixels along some horizontal line (parallel to the x-axis) correspond to the positions or displacements of a particular atom as a function of time. The individual pixels displayed on the screen thus represent individual 4D space-time data points. Unlike the direct 3D particles' rendering, this does not suffer from any occlusion. However, there is no direct visual representation of the real 3D arrangement of atoms. Here, we place all Mg atoms in the lower half and O atoms in the upper half. The other way is to arrange the atoms in the form of 8-atoms unit cells, which are put together to form the supercell. For example, a combination of 2x2x2 unit cells makes up a 64-atoms supercell whereas 3x3x3 arrangement determines a 216-atom supercell.



Figure 2: Color-mapped dimension representation of the atomic displacements relative to the perfect crystalline positions calculated over 3000 time steps (every 10 steps are skipped) across the solid-to-liquid phase transition in a 64-atom MgO system.

The purpose here is to visualize the differences between the consecutive atomic configurations or displacements corresponding to different phases and/or conditions. Figure 2 renders the atomic displacement data during the time period in which the solid-to-liquid phase transition takes place in MgO. We clearly see the contrast in the brightness of the pixels' colors between the left (solid phase) and right (liquid phase) portions of the image. The pixels' intensities and their variations for the liquid phase are much larger than those for the solid phase.

4.2 Local/Extracted Data Rendering

Radial distribution function

We compute the radial distribution functions (RDF), both partial and total, $g_{\alpha\beta}(r)$ and $g(r)$, and structure factor, $S(q)$ to examine the microstructures of the simulated system [Chel01, Allen87]. The RDF is the probability of finding another atom at a distance r from a specified atom. Computation of RDF is not straightforward due to the supercell structure and its periodicity, so the algorithm is presented here (Figure 3). The RDF information is often used to compute other structural properties, for instance, the positions of the first peak and the first minimum after the peak are used in the coordination and cluster analysis. Figure 4 shows the total RDF, $g(r)$, of the liquid MgO, which saturates to the unity after the first peak, indicating the absence of any long-range correlation among atoms – a characteristic feature of the liquid phase. On the other hand, the RDF of solid MgO shows several well-defined peaks. The structure factor (SF) is also calculated by taking the Fourier transform of RDF.

```

For each Simulation Step
  For i := 1 to na - 1
    For j := 1 to na
      dist := d(pi(t), pj(t))
      idx := dist/dr
      ht(i)t(j)[idx] := ht(i)t(j)[idx]+1
    End
  End
End
v := a3
pf := v / (4πnt) ;
For i := 1 to ns
  For j := 1 to ns
    f := pf/ninj
    For k := 1 to nd
      ppcfijk := f*hijk / (rhk2 dr) ;
    End
  End
End
For k := 1 to nd
  pcfk := 0.0 ;
  For i := 1 to ns
    For j := 1 to ns
      ppcfk := ppcfk + ninj ppcfijk / n ;
    End
  End
End

```

Figure 3: A pseudocode for the radial distribution function calculation. Here, n_a is the total number of atoms, n_i and n_j are the number of species i and j .

Coordination environment

The simplest and most common parameter, which characterizes the local structural information, is the coordination number. It indicates the average number of particles of species β around a particle of species α

within a sphere of radius r . It is simply the integral over the corresponding radial distribution function [Chel01, Allen87]:

$$C_{\alpha\beta} = 4\pi\alpha\beta\int_0^{r_{\min}} r^2 g_{\alpha\beta}(r) dr$$

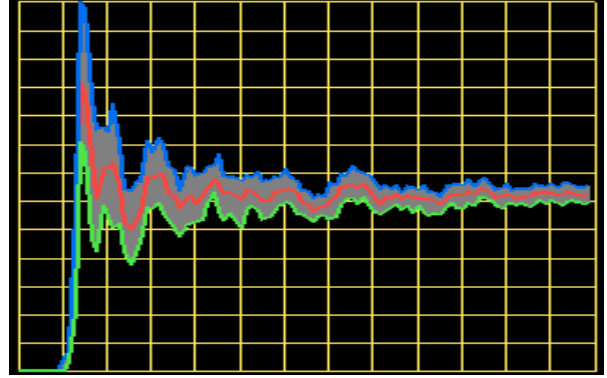


Figure 4: Plot of the RDF for the liquid MgO system as a function of distance, r (Blue: upper bound, Green: lower bound, Red: time averaged over 5000 steps)

Here, α is the number density and β is the concentration (N_β/N) of atoms of type β . Since we are interested only at the nearest neighbors, the cutoff is taken to be the first minimum (r_{\min}) of the corresponding partial radial distribution function. We visualize the coordination as a function of space (per atom basis) in the form of simple bonding or polyhedral representation. The individual atoms or polyhedra are color-coded according to the calculated coordination numbers. As shown in Figure 5, the coordination environment is not uniform throughout the system. The coordination environment also gives substantial insight into the compositional disorder in the structure, i.e., the degree and nature of the mixing of homogeneous (like-atoms) and heterogeneous (unlike-atoms) bonding.

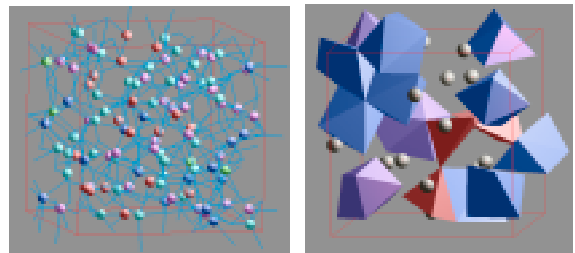


Figure 5: Mg-O coordination in the 216-atoms liquid MgO (left) and Si-O coordination in the 80-atoms liquid MgSiO₃ (right). Their space-time averaged values are 4.8 and 6.1, respectively. In MgO, 108 Mg atoms are shown with five green (coordination greater than 6), 16 blue (the six-fold coordination), 33 pink (five-fold coordination), 36 cyan (four-fold coordination) and 17 orange spheres (coordination smaller than 4). In MgSiO₃, the 16 Si-polyhedra are shown with 10 blue SiO₆ (octahedra), 3 purple SiO₇ units and 3 red SiO₅ and SiO₄ units. The polyhedra are constructed using the qhull algorithm [Barb96].

Cluster analysis

We adopt the following convention to obtain insight into the internal cluster structure: If we can traverse through a group of atoms following path from one atom to the another atom such that distance between any two consecutive atoms on the path are within a specified cutoff distance, then we say they are related in some fashion to form a cluster, which connects successive the nearest neighbor (NN) atoms [Jarvis73], as shown in Figure 6. For example, the relevant cutoff distance could be r_{crystal} (crystalline bond-length), r_{peak} or r_{min} (distance to the first peak or the first minimum of RDF).

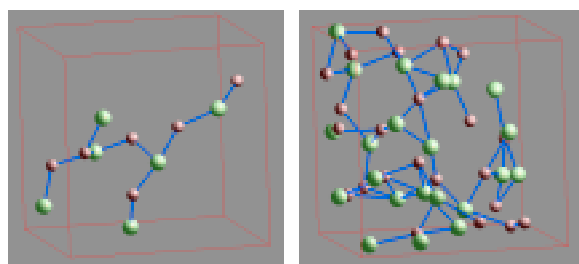


Figure 6: Nearest-neighbor (NN) clusters formed starting any atom in the cluster for different cutoff distances used: r_{peak} (left) and r_{crystal} (right). Their size and shape characterize the structure at different length scales.

```

For i := 1 to na
  For j := i+1 to na
    For all a ∈ alisti
      If dist(i,a) < dm marka := 1
    End
    For all a ∈ alistj
      If d(j,a) < dm and marka=1
        Then cn.add(a)
      End
    For all a ∈ cn
      For all b ∈ cn
        If a!b and dist(a, b) < dm
          Then (a,b):=BONDED
        End
      End
    End
  End
End

```

Figure 6: Algorithm for common-neighbor (CN) analysis. Here, n_a is the total number of atoms. The minimum distance d_m is usually determined from RDF: the minimum after the first peak.

Common-neighbor (CN) analysis [Honey87, Blais84] is also used to analyze the cluster structures (Figure 7). It represents a way of decomposing the radial distribution function, $g(r)$, according to the environment of the pairs. In this analysis, a pair is considered bonded if they are within the r_{min} distance of each other. For the purpose of cluster analysis, we represent the system as a graph $G = (V, E)$, where the atoms are the vertices, V , and the distance between two vertices, V_1 and V_2 , is the weight for the edge, E , between those two atoms. The graph is stored in the form of an adjacency list. Given a vertex, V , only those vertices $V' \in G$, which are within some maximum distance ($4r_{\text{min}}$) and hence contribute to the analysis, are kept in the adjacency list of V .

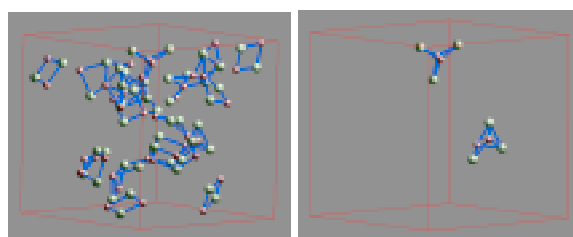


Figure 7: Common-neighbor (CN) clusters in the 216 atoms liquid MgO using r_{min} as the cutoff distance. Short chain-like structures for the pairs with two-CN's can be seen. Three-CN's which are always absent in the solid phase exist in the liquid phase. Their number, size and shape are found to vary greatly with time. The higher-order-CN's occasionally appears during the simulation.

Spheres of atomic movement

Visualization of the extent of the atomic movement or displacement during the simulation is useful to characterize the local and overall behavior of the dynamics of the given atomic system. We extract the several sets of atomic displacement data. They represent the differences of the atomic configuration at a given instant from the initial (perfect crystalline or some random) or previous or next configuration. The other relevant differences are those involving the farthest positions the atoms reach or the mean atomic positions during the whole simulation period. In each case, the magnitude of the displacement is represented by the size of the sphere located at each atomic position whereas the direction of the displacement is represented by a line segment pointing away from the surface of the sphere. Figure 8 shows the displacement patterns induced by point (vacancy) defects. The size of the spheres differs among different atomic species (shown by different colors) and, in the case of MgSiO_3 , among the atoms of the same type (spheres of the same color). The lines are also oriented along various directions. The atomic displacement visualization thus implies that the defect effect is isotropic in MgO whereas it is anisotropic in MgSiO_3 . We have also visualized the

liquid data to show that the diffusion process is anisotropic and non-uniform to different extents at different pressure and temperature conditions.

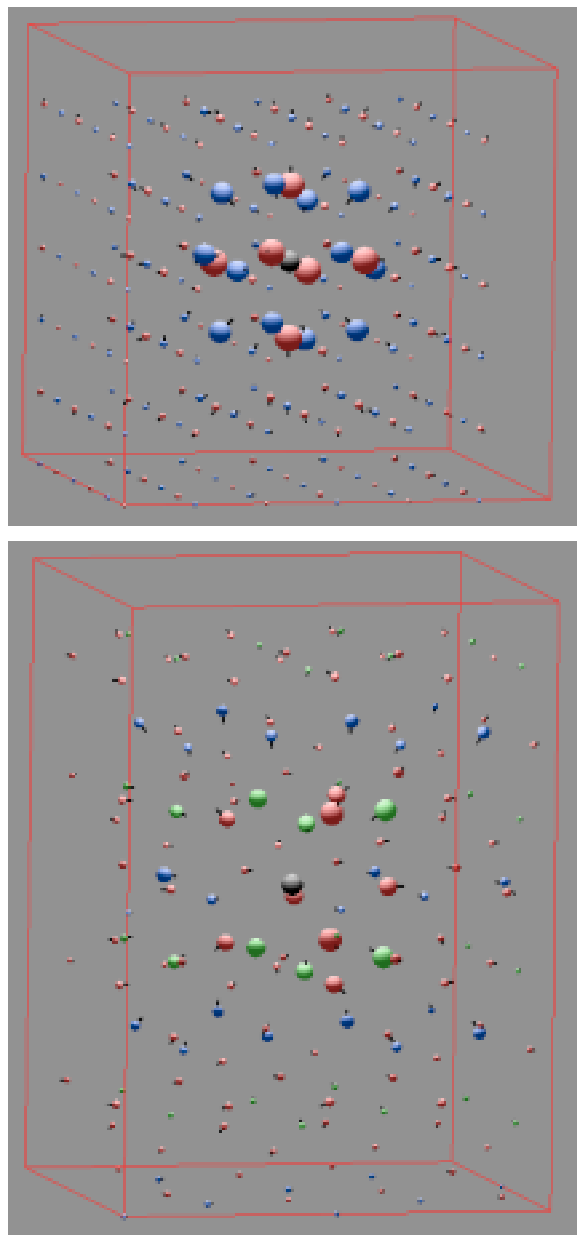


Figure 8: Visualization of atomic displacement data for defective 216-atom MgO (top) and 160-atom MgSiO₃ (bottom). The dark grey sphere indicates the vacancy site from where the Mg atom is removed in the both systems. Other spheres, blue, red and green, represent the extents of movement (or displacement relative to the perfect crystal positions) for Mg, O and Si atoms, respectively.

5. IMPLEMENTATION

The proposed space-time multiresolution visualization scheme is implemented in a modular

fashion (Figure 9) so it is easy to add additional modules whenever needed. In essence, the input configuration data (consisting of a given time series over 5000 steps of atomic 3D positions) are read into the memory. The data are used to generate additional data on the fly, which exploit the use of specific data-structures (e.g., adjacency lists) and dynamic memory allocation. Some datasets are generated dynamically and some are kept in the memory throughout the visualization process. For instance, for the Common Neighbor analysis, it is computationally as well as memory-wise prohibitive to compute all the clusters for all 5000 or so steps beforehand. Moreover, the clusters are dependent on the critical distance used for the analysis. The application lets the user interactively change the critical distance or the time step to study the cluster formation. On the other hand, the averaged RDF is computed once to obtain the r_{peak} and r_{min} values for their use later by other modules.

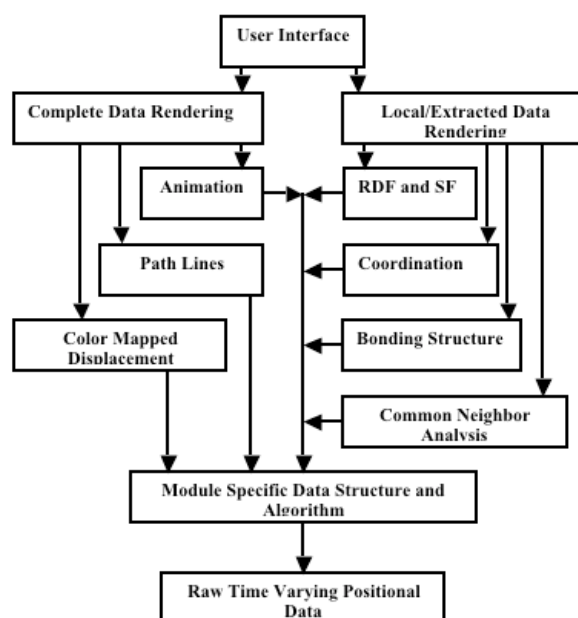


Figure 9: Various visualization modules and their relationship among each other.

Implementation is done using C, OpenGL, GLUT and GLUI. Performance measurements conducted on a Windows XP PC with 3.2 GHz Pentium IV processor, 1 GB RAM and 128 MB NVIDIA GeForce FX 5200 with 128 MB graphics card show that the interactive frame rates are achievable for all system sizes (64, 80, 160 and 216 atoms) considered in this study. The timings (which include the computation time and subsequent rendering time) are calculated for various modules for the 216-atoms system. The RDF module takes about 0.15 seconds per time step. The CN module takes 0.14, 0.18 and

0.25 seconds per time step for the cutoff distance equal to r_{peak} , r_{crystal} , and r_{min} , respectively. The NN-based cluster analysis module takes about 0.09 seconds per step. The time is about 0.02 seconds to draw the spheres of the atomic movement, and the time is even smaller for the execution of several other modules.

6. CONCLUSIONS

We have proposed an efficient visualization scheme to interactively explore the atomistic simulation data for the detailed spatio-temporal information for the structure and dynamics of a real material system. It adopts an application-based approach, which integrates several existing rendering and graph-theory techniques to fulfill the specific visualization needs imposed by the sophisticated first-principles materials modeling. In particular, it supports an interactive visualization of the original and on-the-fly-extracted data to gain insight into the atomic coordination, clustering and diffusion under different conditions and their variation with the time. In many cases, the physical relevance (atomic arrangement) needs to be rendered correctly, whereas, in other cases, only the information needs to be graphically rendered. There is still a lot to do with quantification and visualization of various interesting properties and processes, which are represented by simulation data. We plan to extend our approach by taking into account the additional data such as electron density. Interactivity and memory are expected to be the major issues in extension of the proposed scheme to the case of larger atomic systems.

7. ACKNOWLEDGMENTS

This work is supported by the NSF Career (EAR 0347204) and ITR (ATM 0426601) grants.

8. REFERENCES

- [Alfe05] D. Alfe, Melting curve of MgO from first principles simulations, *Physical Review Letters*, 94, 235701, 2005.
- [Amira] <http://www.amiravis.com/mol>
- [Allen87] M. J. Allen and D. J. Tildesley, *Computer simulation of liquids* (Oxford: Oxford University Press, 1987).
- [Blais84] E. Blaisten-Barojas. Structural effects of three-body interactions on atomic microclusters. *Kinam*, 6A: 71, 1984.
- [Barb96] C. B. Barber, D. P. Dobkin, and H. T. Huhdanpaa, The quickhull algorithm for convex hulls, *ACM Transactions on Mathematical Software*, 22(4): 469-483, 1996.
- [Chel01] James R. Chelikowsky, Jeffrey J. Derby, Vitaliy V. Godlevsky, Manish Jain, and J. Y. Raty. Ab initio simulations of liquid semiconductors. *Journal of Physics C*, 13 (R817), 2001.
- [Codes] FPMD packages: PWScf (www.pwscf.org) and VASP (cms.mpi.univie.ac.at/vasp).
- [Crysm] <http://www.crystallmaker.com>
- [Honey87] J. D. Honeycutt and H. C. Andersen. Molecular dynamics study of melting and freezing of small Lennard-Jones clusters. *Journal of Physics and Chemistry*, 91:4950-4950, 1987.
- [Hum96] W. Humphrey, A. Dalke, and K. Schulten. VMD: Visual Molecular Dynamics. *Journal of Molecular Graphics*, 14: 33-38, 1996.
- [Jarvis73] R. A. Jarvis and Edward A. Patrick. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, C-22: 615-624, November 1973.
- [Karki05] B.B. Karki, D. Bhattarai and L. Stixrude, First-principles calculations of structural, dynamical and electronic properties of liquid MgO, *Physical Review B*, 2005 (submitted).
- [Kokaji99] A. Kokaji. XcrysDen-a new program for displaying crystalline structures and electron densities. *Journal of Molecular Modeling*, 17: 176-179, 1999.
- [Kokaji03] A. Kokaji, Computer graphics and graphical user interfaces as tools in simulations of matter at the atomic scale, *Computational Material Science*, 2: 155-168, 2003.
- [Krau91] P. J. Kraulis, Molscript – a program to produce both detailed and schematic pict of protein structures, *Journal of Applied Crystallography*, 24, 946-950, 1991.
- [Li05] J. Li, Atomistic visualization, *Handbook of Materials Modeling*, S. Yip (ed.), 2005 Springer, 1051-1068.
- [Sch97] W. Schroeder, K. Martin and B. Lorensen, *Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*, Prentice Hall, 1997.
- [Sharma03] A. Sharma, A. Haas, A. Nakano, R. K. Kalia, P. Vashista, S. Kodiyalam, P. Miller, W. Zhao, X. Liu, T. J. Campbell, Immersive and interactive exploration of billion-atom systems, *Proceedings of IEEE Virtual Reality Conference*, 12, 85-95, 2003.
- [Stix05] L. Stixrude and B.B. Karki, Structure and freezing of MgSiO₃ liquid in Earth's mantle, *Science*, 310, 297-299, 2005.

A Calligraphic Interface for Managing Agents

Alfredo Ferreira
alfredo.ferreira@inesc-id.pt

Marco Vala
marco.vala@tagus.ist.utl.pt

J. A. Madeiras Pereira
jap@inesc-id.pt

Joaquim A. Jorge
jorgej@acm.org

Ana Paiva
ana.paiva@inesc-id.pt

Department of Information Systems and Computer Engineering
INESC-ID / IST / Technical University of Lisbon
R. Alves Redol, 9, 1000-029 Lisboa, Portugal

ABSTRACT

Despite the considerable work on agent frameworks, user interfaces to manage these are mostly script based. Even though some solutions provide graphical interfaces to build agent worlds these are quite limited and overly dependent on textual input. Recently, calligraphic systems using sketch-based and pen-input have emerged as a viable alternative to conventional direct-manipulation interfaces in a wide range of areas, such as user interface design or mechanical systems simulation. In this paper, we present a preliminary approach to a calligraphic interface for managing agents. It recognizes gestures drawn by users allowing them to create and manage agent worlds flexibly and efficiently using a concise language.

Keywords: Calligraphic Interfaces, Sketch Based Modeling, Agent Modeling Tools, Agent Frameworks

1 INTRODUCTION

Many definitions have been proposed to describe "software agents" or simply "agents". Russell and Norvig [RN03] define agent as "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors". Maes [Mae95] adds that "autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed". And we could add several other agent definitions that focus on particular domains.

Agents are used in a large set of areas with a large set of purposes. They are often seen as a programming methodology and, in that sense, contribute to the appearance of several agent frameworks. But, as these frameworks are becoming increasingly widespread, they lack tools to efficiently create and manage agents.

Recently, some agent frameworks have been including graphical tools to aid in the creation of agent-based applications. These tools are, however, quite limited

and dependent on script-based languages and/or textual input. Moreover, they are not really user friendly since they focus more on the agents of the world being created than on the user's task of creating the world.

Thus, we try to take advantage of the way developers sketch agent-based worlds in sheets of paper before creating them, and we propose a novel approach based on a calligraphic interface. The tool we present in this paper, called `editION`, allows users to sketch the agent-based world directly on the computer, reducing scripting and textual input.

After a short discussion of related work, we will present an overview of our approach to sketch-based management of agents, describing the proposed architecture. Then we focus on the creation of agent-based worlds using sketches, presenting our symbol recognition methodology and describing the user interaction with our tool. Finally we present some conclusions and suggest directions for future work.

2 RELATED WORK

Usually agent frameworks do not offer native tools to create agent-based applications. Among positive exceptions are ZEUS Agent Building Toolkit [NNLC99] and AgentFactory [Col01] which have tools to generate starting scripts for creating agents (see Figure 1). Agent Academy [MKSA03] has a tool to parameterize and launch agent-based applications using previously defined agent types. Agent Society Configuration Manager and Launcher (ASCML) [BPB⁺05] is a tool for the Java Agent Development Framework (JADE) which facilitates the configuration and deployment of agent so-

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.

SHORT COMMUNICATION proceedings
ISBN 80-86943-05-4
WSCG'2006, January 30 – February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

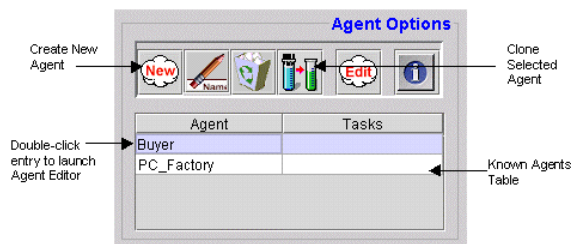


Figure 1: Zeus agent building toolkit

cieties. NetLogo [Wil99] is a programmable modeling environment, featuring hundreds of independent agents, where modelers can give instructions using text-based input (see figure 2).

But all these tools lack some interactivity. Most use a script language and a command interpreter that parses and executes scripts. Even the solutions which provide graphical interfaces are quite limited and dependent on textual input. Moreover, the mouse based interaction in these graphical interfaces mostly relies on drag-and-drop and menu navigation techniques, taking no advantage of the latest¹ user interaction techniques, such as calligraphic interfaces.

Although no calligraphic interfaces had been devised for agent frameworks, several experimental sketch-based systems were developed in recent years for a number of different areas, such as interface design, mechanical systems simulation or control systems analysis. SILK [LM01] is an interactive tool to sketch interfaces using an electronic pad and stylus. Designers can use SILK to quickly sketch the user interface and, when they are satisfied with the early prototype, produce a complete and operational interface. A similar tool, JavaSketchIt, was presented by Caetano *et al.* [CGFJ02] and generates a Java interface based on hand-drawn compositions of simple geometric shapes. The JavaSketchIt evaluation concluded that users consider their sketch-based system more comfortable, natural and intuitive to use than traditional mouse-based tools.

Alvarado and Davies [AD01] developed ASSIST, a program that produces simple 2D mechanical devices from hand-drawn sketches. This system performs real-time interpretation, as the sketch is being created, using

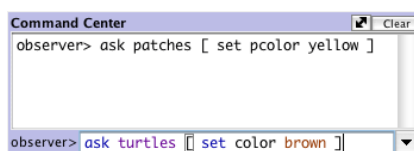


Figure 2: Netlogo command center.

¹ Some of these techniques are not quite novel, but only recently have been adopted by the mainstream manufacturers. An example of this is the pen-based interaction, proposed by Ivan Sutherland [Sut64] almost forty years before being available to the general public.

a set of heuristics to construct a recognition graph containing the likely interpretation of the sketch and selects the best one based on both contextual information and user feedback.

Hong and Landay [HL00] developed a Java toolkit to support the creation of pen-based applications. Using this framework, Lin *et al.* [LNHL00] created DENIM, a sketch-based system that helps web site designers in the early stages of the design process. SketchySPICE [HL00] is another program developed using SATIN. It consists in a calligraphic interface for SPICE² where users can draw simple circuits gates in two distinct modes. In *immediate* mode recognized sketches are immediately replaced by its formal symbol, while in *deferred* mode recognized objects are left sketchy but the recognized symbols are drawn translucently behind the sketch, in order to give some feedback to users.

More recently, Kara and Stahovich [KS04] presented the SIM-U-SKETCH, an experimental sketch-based interface for Matlab's Simulink software package³. With this tool users can sketch functional Simulink models and interact with them, modifying existing objects or add new ones. SIM-U-SKETCH was designed to allow users to draw as they would do on paper, with no constraints imposed by the recognition engine. To that end, this system employs a *recognize on demand* strategy in which the users have to explicitly indicate whenever they want the sketch to be interpreted.

Despite their apparent similarity, distinct approaches and strategies are used in the systems referred above. We studied the advantages and drawbacks of these methodologies and used them to devise a novel calligraphic interface to create agent-based worlds in the context of an agent framework.

3 OVERVIEW

This paper introduces editION, a tool to create agent-based worlds on top of the ION agent framework. Users sketch world elements or commands in order to create and control the agent-based world. Since editION works closely together with the ION framework, it will be able to provide immediate visual feedback to users of what is happening in the world.

The architecture of the proposed solution can be divided in two distinct parts: the ION framework, which handles the agent world, and the editION tool, that provides management capabilities to users. In figure 3 we depict a block diagram of such architecture.

² SPICE is a circuit CAD tool developed at University of California at Berkeley

³ Simulink is an add-on package for analyzing feedback control system and other similar dynamic systems.

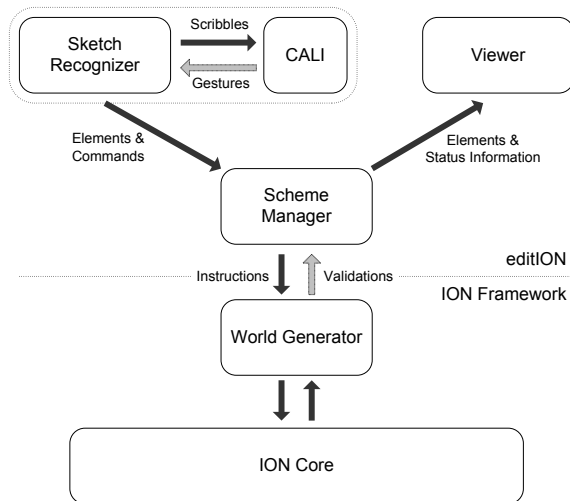


Figure 3: Architecture

3.1 The editION tool

The editION tool includes a calligraphic interface where users sketch the agent world and visual feedback is given. Unlike SketchySPICE, where users can select in which mode they draw, only the *immediate* mode makes sense in editION. Therefore, the sketch is interpreted and validated as it is being drawn, allowing on-the-fly creation instead of having to draw the entire world prior to its creation.

Scribbles drawn by users are processed by the Sketch Recognizer module, which uses CALI [FPJ02] library to recognize them as shapes or commands. Sketch Recognizer then identifies if they are an element of the world or a command, sending the corresponding information to the Scheme Manager module.

The Scheme Manager module can be considered as the core of editION. It handles the diagram representing the agent world, performing syntactic and basic semantic validation. To that end, it applies a set of predefined grammatical rules to each change to guarantee the correctness of the scheme. The scheme is stored as a directed graph, in which the nodes represent the elements and the edges represent the connectors. This way, common graph manipulation techniques could be used to manage the scheme and navigate through it.

The Viewer encapsulates the output details of our approach. It is responsible for providing visual feedback to the user, by selecting the graphical shapes displayed for each element and its properties according to the current state. Based on information received from the scheme manager, the viewer determines which shape must be drawn, its position and color. Moreover, it manages the way messages are shown to users and how long they remain in the screen.

3.2 The ION framework

The ION framework [AV05] is yet another agent framework. However, unlike most agent frameworks, which mainly look at agents that will enrich pre-existent virtual or real environments, the ION framework is also concerned with the creation and the simulation of the environment itself⁴. For the purpose of this paper we will only briefly describe the World Generator and the representation model within the ION core.

The World Generator is a bridging layer. It receives instructions from editION and tries to execute these instructions in the ION core. Depending on the outcome, it also sends feedback which might be useful for further semantic validation.

The ION core manages the world model which is populated by several entities. Entities have properties that store relevant information about the entity, actions to access and modify the environment, and relations with other entities. These relations also have properties that keep information about the role played by the entity in the relation.

The previous representation model allows us to create and simulate different worlds. Imagine, as an example, a small world with a blue object, a red object and a dog that likes red objects and grabs them all the time. Using the ION core, the objects are represented by entities with a single property, its color. The dog is an agent, represented by an entity, which has two actions: look for objects, and grab objects. These actions would be the agent's sensors and actuators. The dog will sense the environment for red objects (using the "look for objects" action) and will act in the environment grabbing the red object (using the "grab objects" action). The dog could even remember the objects that were grabbed before, if we create a relation between the dog and those objects.

4 SKETCHING AGENT-BASED WORLDS

Developers often start by drawing agent-based worlds on paper. Then, they generally use script-based tools to specify the world in the framework. Even when these tools have graphical interfaces with mouse interaction, they are greatly dependent on textual commands. Following the recent developments in pen and sketch-based interfaces, we propose an alternative to standard mouse-based tools to create agent-based worlds.

The interaction with editION is usually made through pen-based hardware such as a TabletPC or a digitizing tablet. After capturing the scribbles drawn by the user, these must be recognized, interpreted and

⁴ An example application of the ION framework can be found in [Pra05].

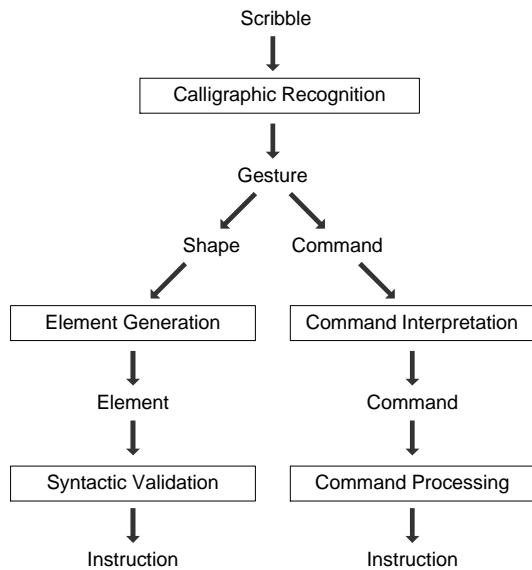


Figure 4: Recognition strategy

validated to become useful. Thus, the whole process of handling the user input and producing corresponding output both to the user and to the framework, plays a major role in our calligraphic tool.

4.1 Symbol Recognition

In order to provide on-line recognition of sketches, our approach processes the scribbles individually and not the entire sketch on demand, as performed by SILK and SIM-U-SKETCH. These scribbles are clusters of strokes drawn by the user which are submitted to a recognition process when the user's pauses are longer than a given time between strokes.

We use a multi-level recognition and parsing strategy, outlined in Figure 4, in order to convert scribbles drawn by users into instructions for ION World Generator. This strategy is divided in several steps, detailed below.

We start by performing the calligraphic recognition of submitted scribbles. To that end, we use CALI, a fast, simple and compact scribble recognizer used in JavaSketchIt. CALI identifies shapes of different sizes and rotated at arbitrary angles, drawn with dashed, continuous strokes or overlapping lines. It detects not only the most common shapes in drawing such as triangles, lines, rectangles, circles, diamonds and ellipses, using multiple strokes, but also other useful shapes such as arrows, crossing lines or wavy lines, as depicted in figure 5.

For this work we only need to detect a subset of the CALI gestures to specify elements and commands. These gestures are depicted in Figure 6. The scribbles to represent the elements were selected based on its visual similarity with the hand-drawn elements, usually sketched by developers when schematically representing their agent worlds.

However, since CALI is size and rotation independent, we need to carry out additional computation to determine scribble orientation and size. This calligraphic recognition step yields two categories of gestures: shape gestures and command gestures. Therefore, after identifying the category to which the detected gesture belongs, we apply distinct parsing paths for shapes and commands.

To perform gesture identification we apply the grammar presented in Figure 7. This set of simple rules provides an efficient manner not only to determine if the scribble is a command or a shape, but also to identify the command or element corresponding to a given scribble.

We consider the application of the grammar mentioned above to be the gesture identification process. This process implements the first levels of the recognition strategy, which is the transformation of scribbles into elements or commands. In this process, rectangles are transformed into entities or actions, depending on their geometric properties, triangles into relationships, circles into properties and lines into connectors. On the other hand, pre-specified gestures are transformed into "delete", "select" and "copy" commands.

When a scribble is identified as an element, it goes through validation. To that end, context information is used to verify if such element makes sense in the current scheme. In the case of a connector, such information is also valuable to determine if it is a simple connector or a role. If the generated element passes the syntactic validation, the corresponding instruction is created and sent to the World Generator. Semantic validation is

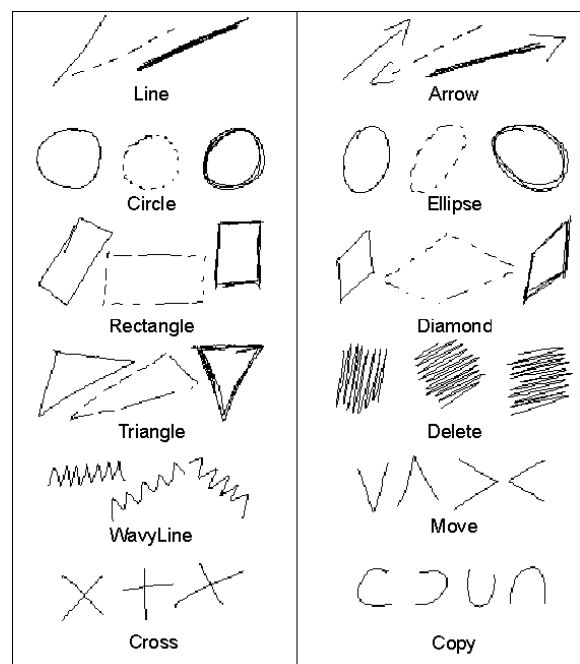


Figure 5: Gestures detected by CALI

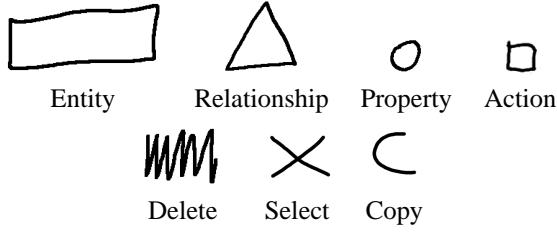


Figure 6: Gestures for elements and commands.

performed by the World Generator and, if the instruction is valid, the `ION` framework is updated. In any case, the World Generator gives proper feedback to the Scheme Manager. Finally, this information is used to provide visual feedback to the user, replacing the sketch by the corresponding element on the screen or showing an meaningful error message.

If a scribble is identified as a command, the context is analyzed to verify its validity. It uses information extracted from context to produce an instruction to send to the World Generator. As for the elements, the world generator processes the instruction and provides information that will be used to give visual feedback to the user.

4.2 Interaction with `editION`

Currently, users sketch their agent world in sheets of paper before coding it into the framework. In `editION` we take advantage of the users' ability to draw agent worlds with a pen to automate the boring and time consuming task of writing unnecessary lines of code. Therefore, users can sketch the world in the `editION` calligraphic interface using a pen-based digitizer and it will be automatically created in the agent framework.

Since we perform on-the-fly gesture recognition, the sketch is interpreted and validated as it is being drawn.

GESTURE-IDENTIFICATION-GRAMMAR(S):=
 $\text{valid_gesture} \rightarrow \text{shape} \mid \text{command}$
 $\text{shape} \rightarrow \text{entity} \mid \text{action} \mid \text{relationship} \mid \text{property} \mid \text{connector}$
 $\text{command} \rightarrow \text{delete} \mid \text{select} \mid \text{copy}$
 $\text{entity} \rightarrow \text{Gesture}(S, \text{RECTANGLE}) \ \& \ \text{SizeWithin}(S, \tau_{\text{max}}^E, \tau_{\text{min}}^E) \ \& \ \text{AspectRatio}(S, 4, 3)$
 $\text{action} \rightarrow \text{Gesture}(S, \text{RECTANGLE}) \ \& \ \text{SizeUnder}(S, \tau_A) \ \& \ \text{AspectRatio}(S, 1, 1)$
 $\text{relationship} \rightarrow \text{Gesture}(S, \text{TRIANGLE}) \ \& \ \text{SizeWithin}(S, \tau_{\text{max}}^R, \tau_{\text{min}}^R)$
 $\text{property} \rightarrow \text{Gesture}(S, \text{CIRCLE}) \ \& \ \text{SizeUnder}(S, \tau_P)$
 $\text{connector} \rightarrow \text{Gesture}(S, \text{LINE})$
 $\text{delete} \rightarrow \text{Gesture}(S, \text{DELETE})$
 $\text{select} \rightarrow \text{Gesture}(S, \text{CROSS})$
 $\text{copy} \rightarrow \text{Gesture}(S, \text{COPY})$
 $\text{Gesture}(sc, t) \rightarrow \text{Scribble } sc \text{ recognized by CALI as } t$
 $\text{SizeWithin}(sc, t_u, t_l) \rightarrow \text{Size of } sc \text{ is within } t_u \text{ and } t_l$
 $\text{SizeUnder}(sc, t) \rightarrow \text{Size of } sc \text{ is below } t$
 $\text{AspectRatio}(sc, w, h) \rightarrow \text{Aspect ratio of } sc \simeq w:h$

Figure 7: Grammar for gesture identification.

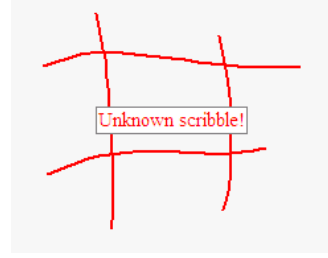


Figure 8: Example of an unrecognized scribble.

Moreover, the on-line connection with the framework allows `editION` to provide immediate feedback to the user from the agent framework. To that end, syntactic and semantic verifications of the sketches are performed while the world is being constructed. Thus, it is no longer necessary to design the complete world to check if any errors exist, as it usually happens in other tools.

Thus, to create an agent world with `editION` the user sketches each element at a time using single or multi-stroke scribbles. The scribble is immediately interpreted by the sketch recognizer. When it is interpreted as a valid element, the corresponding formal symbol replaces the sketch.

Besides elements, the user can also sketch commands. These are also interpreted by the recognizer and, if they are valid, the corresponding action immediately takes place and the drawing area is updated accordingly.

If the scribble is not recognized, it is marked in a different color and a text message informs the user of such situation. Figure 8 depicts an example of an unrecognized scribble, showing the feedback given to the user. This information remains on the screen for a couple of minutes or until the user restarts sketching. Then, both the message and the unknown scribble are deleted.

Similarly, if the user sketches a line, recognized as a connector, but one or both of its endpoints are not over an element, it is drawn in a distinct color with the corresponding message. The same happens if the sketched connection is invalid. Invalid connections occur when the connector extremities are over entities that cannot be connected, for instance if the user is trying to connect two entities or an action to a property, as illustrated in Figure 9.

Besides the unrecognized and invalid scribbles, some elements have no meaning unless they are connected with others. The property and action elements depend

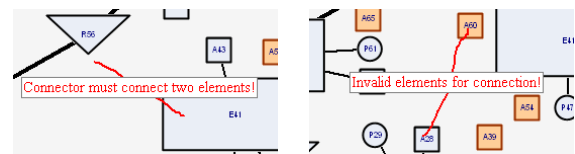


Figure 9: Two examples of invalid strokes.

on an entity and the relationship must be connected to, at least, two entities. In these cases, we consider that the recognized element is incomplete. The *editION* keeps incomplete elements in the drawing area, but they are represented in a different color.

Figure 10 depicts an agent world being created using the *editION* while a new entity is being sketched and its recognition is underway. In this example, some elements have already been recognized and validated. However, the relationship is incomplete, since it needs to be connected to, at least, two entities. Likewise, one action and one property remain unconnected, thus incomplete.

Incomplete elements are displayed in a distinct color until the user corrects this situation. While these elements are incomplete they are not considered besides Scheme Manager. This means that no information about them is sent to the World Manager. Thus, their existence is ignored by the framework.

To create agent-based worlds much more information is needed in order to make it fully functional. Specification of the behavior and the state of each element is an important part of the definition of agents. Thus, *editION* provides an efficient way to edit all the details of each element of the world. A simple click over a recognized entity allows the user to access such details through a pop-up window.

An example of a detail pop-up window is depicted in Figure 11. In this case, the user is changing the details of an action, more specifically, changing the code associated with an event of that action. This kind of changing is submitted to the World Generator, which performs syntactic and, when possible, semantic validation and provides proper feedback to the user.

Many other details can be changed in all elements using the pop-up window, but any change made here is

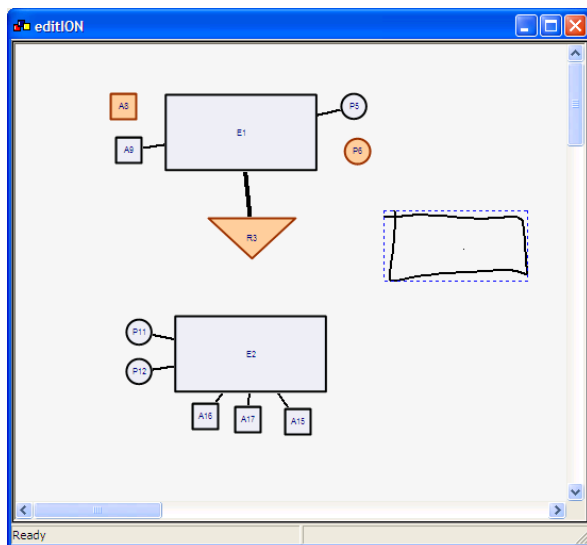


Figure 10: Creating an agent world with *editION*

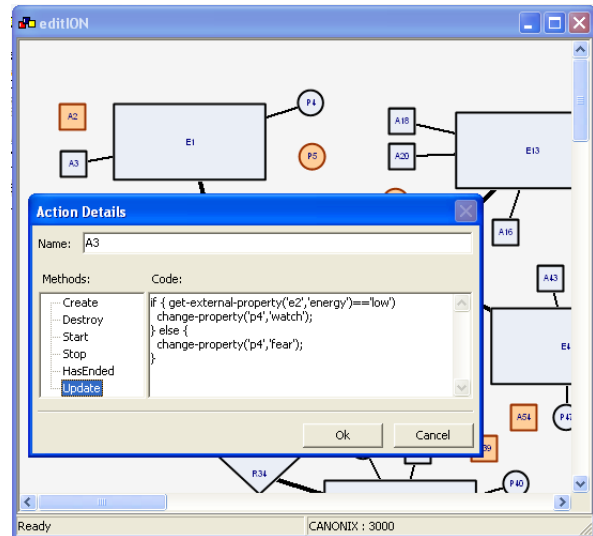


Figure 11: Changing details of an action

always submitted to validation by the Scheme Manager and the World Manager, depending on the type of modification. For instance, when changing an element's name, the name is validated by the Scheme Manager to make sure it does not conflict with other names and then it is validated by the World Manager to check if it is acceptable in the current world.

5 CONCLUSIONS AND FUTURE WORK

The proposed calligraphic interface represents an alternative to current agent management tools. Instead of writing numerous lines of code or dragging and dropping elements from toolbars and menus, with *editION* we bring agent developers closer to traditional paper-and-pencil methods when creating agent worlds.

Since the sketch is interpreted and validated as it is being drawn, the user receives immediate feedback from the agent framework. Therefore, it is no longer necessary to create the complete world to check for errors or incoherence. Most importantly, with this tool, the user avoids writing sometimes long and complex scripts to describe the agent world. It can be simply done by sketching it.

Despite the fact that the proposed tool was devised for management of the *ION* framework, we intend to make it as general as possible in order to allow it to work with other agent frameworks in the future without major changes. To that end, we do not embed the manager in the framework. Instead, we consider *editION* as an independent module that communicates with the framework using a small set of pre-defined instructions. The adopted visual language and identification grammar are, however, best suited for *ION* framework.

The presented version of the *editION* is still under development and offers limited functionality. Basically

it allows the user to create, change and delete world elements. But we feel it has potential to grow into a complete and powerful management tool for agent frameworks, while retaining most of its simplicity. To that end, in a near future we plan to add, among other functionalities, debugging capabilities to `editION`, offering total control over the agent world and providing continuous visual feedback on the world status.

When both `editION` and `ION` framework are fully functional we intend to perform users' evaluation involving developers and researchers from the agents area. In these tests we expect not only to validate the proposed methodology for agent world design, but also to collect information in order to refine our approach, according to the users' needs.

ACKNOWLEDGEMENTS

Alfredo Ferreira was supported in part by the Portuguese Foundation for Science and Technology, grant reference SFRH/BD/17705/2004.

REFERENCES

- [AD01] Christine Alvarado and Randall Davis. Resolving ambiguities to create a natural sketch based interface. In *Proceedings of IJCAI-2001*, August 2001.
- [AV05] Ruth Ayllet and Marco Vala. *Victec deliverable 3.5.1: Toolkit final version*, 2005.
- [BPB⁺05] Lars Braubach, Alexander Pokahr, Dirk Bade, Karl-Heinz Krempels, and Winfried Lamersdorf. Deployment of distributed multi-agent systems. In Franco Zambonelli Marie-Pierre Gleizes, Andrea Omicini, editor, *5th International Workshop on Engineering Societies in the Agents World*, pages 261–276. Springer-Verlag, Berlin Heidelberg, 8 2005.
- [CGFJ02] Anabela Caetano, Neri Goulart, Manuel Fonseca, and Joaquim Jorge. Javasketchit: Issues in sketching the look of user interfaces. In *Proceedings of the 2002 AAAI Spring Symposium - Sketch Understanding*, pages 9–14, Palo Alto, USA, March 2002.
- [Col01] R. W. Collier. *"Agent Factory: A Framework for the Engineering of Agent-Oriented Applications"*. PhD thesis, "University College Dublin", 2001.
- [FPJ02] Manuel J. Fonseca, César Pimentel, and Joaquim A. Jorge. CALI: An Online Scribble Recognizer for Calligraphic Interfaces. In *Proceedings of the 2002 AAAI Spring Symposium - Sketch Understanding*, pages 51–58, Palo Alto, USA, March 2002.
- [HL00] Jason I. Hong and James A. Landay. Satin: a toolkit for informal ink-based applications. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 63–72, New York, NY, USA, 2000. ACM Press.
- [KS04] Levent Burak Kara and Thomas F. Stahovich. Sim-usketch: a sketch-based interface for simulink. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 354–357, New York, NY, USA, 2004. ACM Press.
- [LM01] James A. Landay and Brad A. Myers. Sketching interfaces: Toward more human interface design. *IEEE Computer*, 34(2):56–64, 2001.
- [LNHL00] James Lin, Mark W. Newman, Jason I. Hong, and James A. Landay. DENIM: finding a tighter fit between tools and practice for web site design. In *CHI*, pages 510–517, 2000.
- [Mae95] Pattie Maes. Artificial life meets entertainment: lifelike autonomous agents. *Commun. ACM*, 38(11):108–114, 1995.
- [MKSA03] P. A. Mitkas, D. Kehagias, A. L. Symeonidis, and I. N. Athanasiadis. "a framework for constructing multi-agent applications and training intelligent agents". In *Proceedings of the 4th Int. Workshop on Agent-Oriented Software Engineering (AOSE-2003)*, pages 96–109, 2003.
- [NNLC99] H. Nwana, D. Ndumu, L. Lee, and J. Collis. Zeus: a toolkit and approach for building distributed multi-agent systems. In *Proceedings of the 3rd conference on Autonomous Agents*, pages 360–361. ACM Press, 1999.
- [Pra05] Rui Prada. *Teaming Up Human and Synthetic Characters*. PhD thesis, Instituto Superior Técnico, Technical University of Lisbon, 2005.
- [RN03] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.
- [Sut64] Ivan E. Sutherland. Sketch pad a man-machine graphical communication system. In *DAC '64: Proceedings of the SHARE design automation workshop*, pages 6.329–6.346, New York, NY, USA, 1964. ACM Press.
- [Wil99] U. Wilensky. Netlogo. Center for Connected Learning and Computer-Based Modeling. Northwestern University, Evanston, IL., <<http://ccl.northwestern.edu/netlogo>>, 1999.

Soft Edges and Burning Things: Enhanced Real-Time Rendering of Particle Systems

Tommi Ilmonen
Helsinki Univ. of Technology
Telecommunications Software
and Multimedia Laboratory
Tommi.Ilmonen@hut.fi

Tapio Takala
Helsinki Univ. of Technology
Telecommunications Software
and Multimedia Laboratory
tta@cs.hut.fi

Juha Laitinen
Helsinki Univ. of Technology
Telecommunications Software
and Multimedia Laboratory
Juha.Laitinen@tml.hut.fi

ABSTRACT

This paper describes two methods that can be used to enhance the looks of particle systems. These methods fit applications that use modern graphics hardware for rendering. The first method removes clipping artifacts. These artifacts often appear when a fuzzy particle texture intersects solid geometry, resulting in a visible, undesirable edge in the rendered graphics. This problem can be overcome by softening the edges with proper shading algorithms.

The second method of this paper presents the use of a five-component color model. The parameters of the model are: red, green, blue, alpha and burn. The first four color components have their usual meaning while the "burn" parameter is used to control the additivity of the color. This color model allows particles' alpha blending to range from pure additive (useful for rendering flames) to normal smoke-style rendering. This method can be implemented on most graphics hardware, even without shader support.

Keywords

Particle systems, OpenGL, Shaders

1 INTRODUCTION

Particle systems are typically used to render gaseous phenomena. In real-time applications the particles are rendered as polygons that have a fuzzy texture map. This paper presents two methods that can be used to make such rendering more attractive. Part of these effects can be implemented with any graphics processing unit (GPU) while some methods need shader support in the GPU.

According to our industry contacts these methods are known in the industry and among particle system developers, but there are few if any publications about them. This article is intended to give detailed information on how these techniques can be integrated into real-world applications.

This paper is organized as follows. First we cover available publications on particle systems, then how to render fuzzy particles that intersect solid objects and finally how to achieve a useful blending mode for explosions and fire.

2 BACKGROUND

Particle systems were first published by Reeves [Ree83]. Sims has later described in more detail how to implement the dynamics of particle systems [Sim90]. The author introduced the second order particle system that included moving force fields [Ilm03]. Few publications address the rendering of particle systems. Reeves has published methods to optimize off-line rendering of complex particle systems [Ree85]. Some particle system papers also discuss the rendering of particles, e.g. Burg's introductory article on particle systems gives an excellent overview on how to render particles on graphics hardware [Bur00] (see also McAllister's article on the same topic [McA00]). Burg covers basic topics, such as texturing, alpha blending and texture animation that are widely used throughout the industry.

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.

SHORT COMMUNICATION proceedings
ISBN 80-86943-05-4
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press



Figure 1: A fire effect. Individual fire and smoke particles are visible.

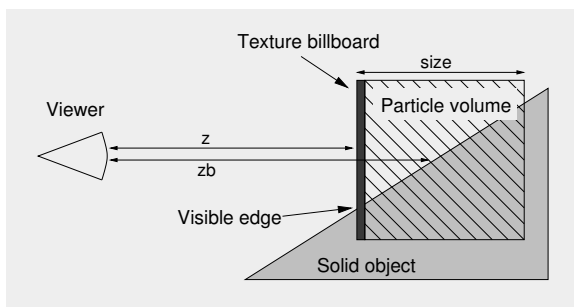


Figure 2: A particle is clipped against a solid object. The volumetric particle is rendered as a single textured billboard.

3 SOFT EDGES

In this section we deal with particles that represent fuzzy objects. Typically such particles are rendered as texture-mapped polygons. In real-time applications the particles are usually so large that individual ones can be identified. This is the case in for example figure 1. In these cases one can often see a sharp edge where the particle texture is clipped against the rigid object.

Figure 2 shows an overview of the situation. The particle is a volumetric, fuzzy object that is shown as a billboard to the user. Where the billboard intersects a solid object a visible edge can be seen. This rendering artifact usually makes a particle system look much less volumetric than intended. It also reveals the volumetric particles are in fact 2D textures. One can counter this problem by using a greater number of smaller particles. Unfortunately this approach is computationally heavy and seldom fit for real-time applications.

A more robust method to make clipping unobtrusive is to hide the edges with progressive alpha-blending. For example in figure 2 the lower part of the particle

billboard should have an alpha value that goes linearly from one to zero. This can be done with the following pixel shader pseudo-code. In this example *pcol* is the user-defined color of the particle, *tcol* the color of the billboard texture at the pixel location, *z* the depth value of the billboard at the pixel locations, *zb* the depth in frame buffer, *size* represents the diameter of the particle volume, *ascale* is the scaling coefficient used to make pixels more transparent when *z* is close to *zb* and *final_color* is the color of the billboard at the pixel location (see figure 2). All colors have three color components and an alpha value that ranges from zero (fully transparent) to one (fully opaque).

```
vector4 pcol = particle_color();
vector4 tcol = texture_color();
float z = pixel_depth();
float zb = frame_buf_depth();
float size = particle_size();
float ascale = (zb - z) / size;
if(ascale > 1)
    ascale = 1;
else if(ascale < 0)
    ascale = 0;
vector4 final_color =
    pcol * tcol;
final_color.alpha *= ascale;
```

The above shader cannot be implemented directly in current PC hardware since in the modern GPU the pixel shaders cannot read depth values from the frame buffer. To counter this problem we use multi-pass rendering:

1. Render all solid objects in the scene.
2. Copy the depth values from the frame buffer to a depth texture.
3. Render particles back to front with a pixel shader that reads the depth values from the depth texture. The shader lowers the fragment's alpha as necessary value to achieve soft clipping.

We have done this with a pixel shader, using the OpenGL shading language (GLSL) [Ros04]. Since soft clipping requires more computation it is inevitably slower than normal rendering. The performance impact depends on multiple factors — the display resolution, the number of particles and their size. Thus any benchmark results are application-specific. In our tests the frame rates dropped typically to one third when soft edges were used with a heavy particle system (as in figures 3 and 5). More optimized shader implementation might improve this situation. The test computer had a 1,5 GHz AMD Athlon processor and an NVidia 5700 graphics card.



(a) Particles rendered with the default OpenGL pipeline. The intersection edges between particles and the brick wall are clearly visible (80 fps).



(b) Particles rendered with custom shading for soft clipping (28 fps).

Figure 3: Hard and soft clipping. The scene and particle locations are identical in both pictures. The frame rate of the soft clipping rendering is significantly lower.

4 ENHANCED BLENDING

The semi-transparent particle billboards need to be blended into the background. In modern GPUs the blending is controlled by blending functions that are a non-programmable part of the graphics pipeline. The two most common blend functions are additive blending and transparency blending. These functions are defined by the following formulas, where the bg is the RGB (red/green/blue) color of the frame buffer, fg is the RGB color of the particle, $alpha$ represents the transparency of the particle (set by the user) and c is the final color of the frame buffer after updating its color value.

Additive blending adds luminance to the frame buffer:

$$c = fg * alpha + bg \quad (1)$$

Transparency blending changes the colors towards the particle color:

$$c = fg * alpha + bg * (1 - alpha) \quad (2)$$

The effect of these blending modes can be seen in figure 4. Both blend functions are useful in particle systems — the additive blend mode is widely used in fire effects and the transparency blending is useful for rendering smoke or fog.

We have developed a new blending function that can be used to create both additive and transparency blending. This is called "controlled additive" blending and it can be used to interpolate smoothly between the additive and the transparency functions.

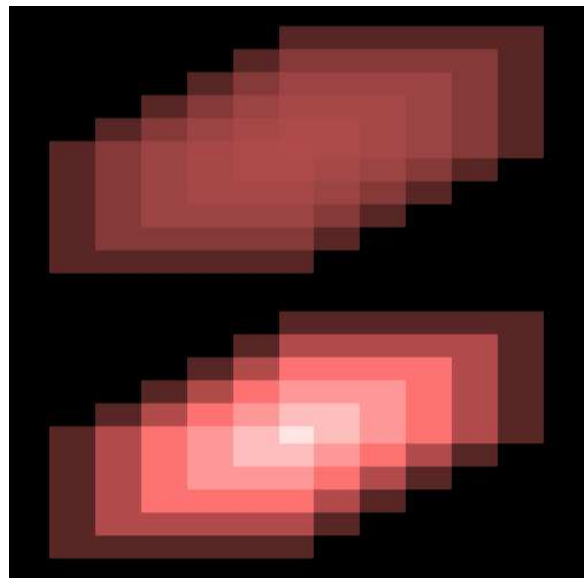


Figure 4: Different blending modes, above transparency blending of several red squares and below additive blending with same colors and geometry.

Interpolation of these two blending modes offers interesting new possibilities. The first is that one can make smooth transitions between additive and transparency blending. For example in the case of fire we can create particles that begin as additive fire particles, but later become transparency particles.

The second use of the controlled additive function is to create particle colors that are slightly additive. That is, large collection of slightly additive particles looks



Figure 5: An explosion rendered with additive blending (top), normal transparency blending (middle) and controlled additive blending (bottom).

brighter than any single particle, but they will not result in white areas like with plain additive blending.

The controlled additivity is defined by the following blend function

$$c = fg * alpha + bg * (1 - alpha * (1 - burn)) \quad (3)$$

Here the *burn* variable is used to control the additivity of the function. The variable ranges from zero (transparency blending) to one (additive blending). This gives us a five-component color model composed of red, green, blue, alpha and burn values. Since graphics hardware only deals with four-component colors and fixed blending functions we need special methods to make the hardware render the particles with this model.

First of all we set the GPU's blending unit to use the following blend equation:

$$c = fgc + bg * (1 - alphac) \quad (4)$$

This equation gives the correct color values when

$$fgc = fg * alpha \quad (5)$$

and

$$alphac = alpha * (1 - burn) \quad (6)$$

Equations 5 and 6 need to be evaluated outside the blend unit due to hardware limitations. They can be easily calculated either in a vertex shader or in the application code. The first alternative may be slightly faster, but it also requires hardware with shader support.

The textures that are used in the rendering need special processing. In general the particle billboard texture is represented as an RGBA texture. Opening equation 6 we see that the *alpha* value of the texture has an important role in the calculations:

$$fg = ucol * tcol * talpha \quad (7)$$

thus, combining equations 5 and 7:

$$fgc = ucol * tcol * talpha * alpha \quad (8)$$

In these equations *ucol* is the RGB color set by the user, *tcol* is the RGB color in the texture and *talpha*

is the alpha value of the texture. The default OpenGL graphics pipeline cannot calculate equation 8 directly. The evaluation can be done either in the pixel shader or by pre-processing the texture. We have used the latter approach since it does not require programmable pixel shaders and the work-load of the pixel units is lower, resulting in potentially better performance. To do so we pre-multiply the alpha values in the particle texture to the RGB values. Thus

$$fgc = ucol * tcola * alpha \quad (9)$$

where

$$tcola = tcol * talpha \quad (10)$$

To clarify the use of the above equations we present a pseudo-code that calculates the colors that are transmitted to the graphics hardware. In the following code *rgb* is the RGB color of particle, *alpha* represents opacity, *burn* controls additivity, *fgc* is the RGB color sent to the GPU and *alphac* is alpha value sent to the GPU. All parameters are assumed be in the range [0-1]. If the particle has a texture map, then the alpha value of the texture should be pre-multiplied to its RGB values.

```
vector3 rgb = get_rgb();
float alpha = get_alpha();
float burn = get_burn();
vector3 fgc = rgb * alpha;
float alphac = alpha * (1-burn);
```

If one wants to use these parameters with OpenGL, then the blending mode needs to be set to match equation 7 with the following function calls:

```
glBlendEquation(GL_FUNC_ADD);
glBlendFunc(GL_ONE,
            GL_ONE_MINUS_SRC_ALPHA);
```

Once this is done the particle color can be sent to the graphics card as a normal RGBA color:

```
glColor3f(fgc[0], fgc[1],
          fgc[2], alphac);
```

So far we have assumed that particle bitmap is a four-component RGBA texture. In this case the burn parameter is assumed to be constant for the whole billboard. With this parameterization the addition of the burn parameter causes minimal performance impact. The number of texture lookups is the same as for the more traditional blending modes. In fact the pixel units of the GPU are not aware of the blending, since all of the work is done in the blend unit. The only extra calculations are the calculation of proper color and alpha

parameters that need to be performed once for each particle. We have been unable to measure any performance loss due to these calculations — compared to all other work required for particle system dynamics and redering these color calculations are insignificant. We have done this calculation in the application code, but it could be moved to the vertex shader for potentially better performance.

It is also possible to add the burn parameter to the texture, creating a five-component texture. This might be useful for situations where one does not want the same burn value to apply to the whole particle. This would probably call for a separate texture with only the burn values in it as graphics hardware cannot deal with more than four color components per texture. A pixel shader would be needed to carry out the calculations. The additional texture lookups would also lower the rendering speed.

5 CONCLUSIONS

This paper has presented two techniques for rendering particle systems on modern graphics hardware. The soft-edge rendering method removes artifacts that appear when particles are rendered close to solid objects. This results in greater flexibility in application programming as developers do not need to hide or minimize the artifacts.

The controlled additive blending adds a new parameter to color definitions. The new burn parameter is useful when one needs to adjust the additivity of a particle. This parameter can be used on all GPUs that support the selection of blend equations with minimal performance impact.

Both presented methods are easy to implement and they can be used together. While these methods have been presented in the domain of rendering particle systems they may be useful in other kinds of applications where a particular special effect is desired.

6 ACKNOWLEDGEMENTS

This work has been funded by the Academy of Finland.

References

- [Bur00] van der Burg, J. Building an Advanced Particle System. In Game Developer, March 2000

- [Ilm03] Ilmonen, T. and Kontkanen, J. The Second Order Particle System, In The Journal of WSCG, volume 11(2), pages 240-247, 2003
- [McA00] McAllister, D. The Design of an API for Particle Systems. Technical report, University of North Carolina, January 2000
- [Ree83] Reeves, W. Particle Systems — a Technique for Modeling a Class of Fuzzy Objects. In Proceedings of the 10th annual conference on Computer graphics and interactive techniques, pages 359–375, 1983
- [Ree85] Reeves, W. and Blau, R. Approximate and probabilistic algorithms for shading and rendering structured particle systems. In Proceedings of the 12th annual conference on Computer graphics and interactive techniques, pages 313–322, 1985
- [Ros04] Rost, R. OpenGL(R) Shading Language, Addison-Wesley, 2004
- [Sim90] Sims, K. 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, 1990

An iterative method for rational pole curve fitting

J. C. Chambelland
jcchambe@esil.univ-mrs.fr

M. Daniel
mdaniel@esil.univ-mrs.fr

J. M. Brun
jmbrun@esil.univ-mrs.fr

LSIS (UMR CNRS 6168). ESIL-Case 925, 163 Avenue de Luminy
13288 Marseille cedex 09 - (France)

ABSTRACT

This paper addresses the problem of least-square fitting with rational pole curves. The issue is to minimize a sum of squared Euclidean norms with respect to three types of unknowns: the control points, the node values, and the weights. A new iterative algorithm is proposed to solve this problem. The method alternates between three steps to converge towards a solution. One step uses the projection of the data points on the approximant to improve the node values, the two others use a gradient based technique to update the control point positions and the weight values. Experimental results are proposed with rational Bézier and NURBS curves.

Keywords

Least-square fitting, rational pole curves, optimization, iterative methods.

1. INTRODUCTION

Pole curve fitting techniques are often used in CAD softwares to smooth a set of data points. Most of these are least-square based methods and aim at minimizing, with respect to control points and node values, a sum of squared Euclidean norms measuring the distance between the set of data points and the curve to be fitted. Among this kind of method, one can emphasize the linear least square method [LS86, Dan96] with fixed parametrizations [Lee89, PT96, ZCM98, Far01], iterative "Hoschek like" methods [Hos88, SD03, CDB05], and global approaches [SKH98, AB01]. These methods are efficient for polynomial and piecewise polynomial pole curves but not for rational curves because the weights linked to the poles are not considered to improve the accuracy of the fitting. Based on the idea initially proposed in [CDB05] for non-rational pole curve fitting, this paper presents a new iterative method allowing to handle the influence of weights in this problem. The issue being to minimize a sum of squared Euclidean norms with respect to three types of unknowns (the control points, the node values, and the weights), the proposed method alternates between three steps to approach a solution. One step uses the projection of the data points on the approximation curve to improve the node values linked to data points. The two others use a robust gradient based technique to update the control point positions and the weight values.

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.

Conference proceedings ISBN 80-86943-05-4
WSCG'2006, January 30 – February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

The problem formulation is given in section 2. Our algorithm is detailed in section 3. Experimental results are proposed with rational Bézier and NURBS curves in section 4.

2. PROBLEM FORMULATION

We want to fit a k -dimensional ($k > 1$) rational pole curve $\mathbf{C}(t)$ ($t \in [a, b]$) with $(n + 1)$ control points to a given set of $(m + 1)$ ordered k dimensional data points \mathbf{D} in the least-square sense. This problem is to minimize a sum of squared Euclidean norms with respect to three kinds of unknowns: the set of control points \mathbf{P} , the set of node values \mathbf{t} linked to data points, and the set of weight values \mathbf{w} linked to control points. Let be:

- $\mathbf{D} = (\mathbf{d}_0, \dots, \mathbf{d}_m)$ the set of ordered points in \mathbb{R}^k
- $\mathbf{P} = (\mathbf{P}_0, \dots, \mathbf{P}_n)$ the set of control points in \mathbb{R}^k
- $\mathbf{w} = (w_0, \dots, w_n)$ the set of weights in \mathbb{R}^{+*}
- $\mathbf{t} = (t_0, \dots, t_m)$ the set of parametric nodes in $[a, b]$

The definition of the rational approximant is:

$$\mathbf{C}(t) = \frac{\sum_{i=0}^n w_i H_i(t) \mathbf{P}_i}{\sum_{i=0}^n w_i H_i(t)} \quad n \leq m, \quad t \in [a, b] \quad (1)$$

And the problem is to minimize the function:

$$d(\mathbf{P}, \mathbf{t}, \mathbf{w}) = \sum_{j=0}^m \|\mathbf{d}_j - \mathbf{C}(t_j)\|_2^2 = \sum_{j=0}^m \|\mathbf{E}_j\|_2^2 \quad (2)$$

fixing $t_0 = a$, $t_m = b$, $\mathbf{C}(t_0) = \mathbf{d}_0$, $\mathbf{C}(t_m) = \mathbf{d}_m$ such that the first and the last data points respectively match with the first and the last points of the curve.

3. PROPOSED ALGORITHM

Outline

Our algorithm allows the minimization of function $d(\mathbf{P}, \mathbf{t}, \mathbf{w})$ while improving the fitting of an initial approximation curve which provides guess values for \mathbf{P} , \mathbf{t} , and \mathbf{w} . This algorithm uses three steps to locate an optimal solution. As the problem depends on three kinds of unknowns, we opted for a relaxation approach which consists of alternately decreasing objective function $d(\mathbf{P}, \mathbf{t}, \mathbf{w})$ with respect to control points, node values and weight values. **The first step** minimizes $d(\mathbf{P}, \mathbf{t}, \mathbf{w})$ with respect to the node values, considering the control point position and the weight values fixed. This corresponds to minimize each term $\|\mathbf{d}_j - \mathbf{C}(t_j)\|^2$ with respect to the node t_j and leads to the solution of the non-linear problem (\bullet is the inner product of two vectors):

$$(\mathbf{d}_j - \mathbf{C}(t_j)) \bullet \frac{d\mathbf{C}}{dt}(t_j) = 0 \quad \forall j \in [0..m]$$

Geometrically, this problem corresponds to find parameter t_j such that $\mathbf{C}(t_j)$ is the orthogonal projection of data point \mathbf{d}_j (see figure 1) on the approximant.

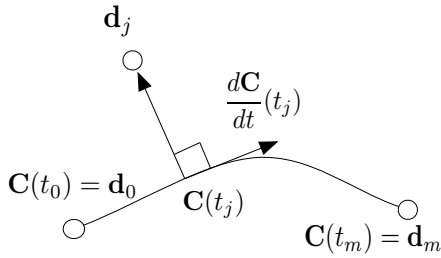


Figure 1: Data point projection

In order to find the closest point and its parametric value, simple "zero order" algorithms can be used. They consist in comparing Euclidean distances between the point to project and a dense set of sample points belonging to the parametric curve. This type of algorithm is inefficient for accurate projections, but useful to provide initial values to "Newton-like" algorithms which use first [Mor97, MH03] or/and second parametric derivatives of the curve [HW05]. According to their rate of convergence, second order algorithms converge theoretically faster than first order algorithms but require more computing time for each iteration and more memory to handle curvature information. In our implementation, we opted for a customized first order projection algorithm. We particularly enforced its robustness, limiting at each iteration the parametric displacement to the value $(b-a)/(m+1)$. This significantly reduces the risk of overshoot and in most cases, very accurate projections are obtained in less than 5 iterations.

The second step consists in reducing $d(\mathbf{P}, \mathbf{t}, \mathbf{w})$ with respect to the control points, considering the node values and the weight values fixed. Since $d(\mathbf{P}, \mathbf{t}, \mathbf{w})$ is infinitely differentiable with respect to \mathbf{P} , we can compute its gradient vector with respect to control points $\nabla^{\mathbf{P}} = (\nabla_0^{\mathbf{P}}, \dots, \nabla_n^{\mathbf{P}})^{\top}$ and affirm that its negative direction is a direction of maximum rate of decrease ([PFTV02, AW95]). Thus, we can "locally" minimize $d(\mathbf{P}, \mathbf{t}, \mathbf{w})$, updating control points by the vector $-\alpha^{\mathbf{P}} \nabla^{\mathbf{P}}$ assuming that $\alpha^{\mathbf{P}}$ is a suitably computed positive scalar value. One way to compute the best scalar value $\alpha_{max}^{\mathbf{P}}$, which minimizes $d(\mathbf{P} - \alpha^{\mathbf{P}} \nabla^{\mathbf{P}}, \mathbf{t}, \mathbf{w})$ with respect to $\alpha^{\mathbf{P}}$, is to use the following equation:

$$\alpha_{max}^{\mathbf{P}} = \frac{\|\nabla^{\mathbf{P}}\|^2}{(H^{\mathbf{P}} \nabla^{\mathbf{P}})^{\top} \bullet \nabla^{\mathbf{P}}} = \frac{\sum_{i=0}^n \|\nabla_i^{\mathbf{P}}\|^2}{(H^{\mathbf{P}} \nabla^{\mathbf{P}})^{\top} \bullet \nabla^{\mathbf{P}}}$$

where $H^{\mathbf{P}}$ is the Hessian matrix of $d(\mathbf{P}, \mathbf{t}, \mathbf{w})$ with respect to \mathbf{P} . This result is used in the robust "steepest descent method" [PFTV02, AW95], but as a matter of fact, requires an explicit handling of the Hessian matrix of the objective function. This task can be sometimes difficult and computationally intensive, but in our case, the Hessian matrix with respect to control points allows us to simplify the denominator $(H^{\mathbf{P}} \nabla^{\mathbf{P}})^{\top} \bullet \nabla^{\mathbf{P}}$. Assuming for convenience that:

$$\phi_i(t_j) = \frac{w_i H_i(t_j)}{\sum_{i=0}^n w_i H_i(t_j)} \quad (3)$$

We can simplify the expression of α_{max} to the reduced expression:

$$\alpha_{max}^{\mathbf{P}} = \frac{\sum_{i=0}^n \|\nabla_i^{\mathbf{P}}\|^2}{2 \sum_{j=0}^m \left\| \sum_{i=0}^n \phi_i(t_j) \nabla_i^{\mathbf{P}} \right\|^2} \quad (4)$$

Proof:

The gradient of $d(\mathbf{P}, \mathbf{t}, \mathbf{w})$ with respect to \mathbf{P} is the $(n+1)$ dimensional column vector:

$$\nabla^{\mathbf{P}} = \begin{pmatrix} \nabla_0^{\mathbf{P}} \\ \vdots \\ \nabla_n^{\mathbf{P}} \end{pmatrix} = \begin{pmatrix} \frac{\partial d(\mathbf{P}, \mathbf{t}, \mathbf{w})}{\partial \mathbf{P}_0} \\ \vdots \\ \frac{\partial d(\mathbf{P}, \mathbf{t}, \mathbf{w})}{\partial \mathbf{P}_n} \end{pmatrix}$$

The Hessian of $d(\mathbf{P}, \mathbf{t}, \mathbf{w})$ with respect to \mathbf{P} is the $(n+1) * (n+1)$ symmetric matrix:

$$H^{\mathbf{P}} = \begin{pmatrix} \frac{\partial^2 d(\mathbf{P}, \mathbf{t}, \mathbf{w})}{\partial \mathbf{P}_0^2} & \cdots & \frac{\partial^2 d(\mathbf{P}, \mathbf{t}, \mathbf{w})}{\partial \mathbf{P}_0 \partial \mathbf{P}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 d(\mathbf{P}, \mathbf{t}, \mathbf{w})}{\partial \mathbf{P}_n \partial \mathbf{P}_0} & \cdots & \frac{\partial^2 d(\mathbf{P}, \mathbf{t}, \mathbf{w})}{\partial \mathbf{P}_n^2} \end{pmatrix}$$

From equ. (2) and (3), we can easily state that:

$$\nabla^{\mathbf{P}} = \begin{pmatrix} -2 \sum_{j=0}^m \phi_0(t_j) \mathbf{E}_j \\ \vdots \\ -2 \sum_{j=0}^m \phi_n(t_j) \mathbf{E}_j \end{pmatrix}$$

and:

$$H^{\mathbf{P}} = \begin{pmatrix} 2 \sum_{j=0}^m \phi_0^2(t_j) & \dots & 2 \sum_{j=0}^m \phi_0(t_j) \phi_n(t_j) \\ \vdots & \ddots & \vdots \\ 2 \sum_{j=0}^m \phi_n(t_j) \phi_0(t_j) & \dots & 2 \sum_{j=0}^m \phi_n^2(t_j) \end{pmatrix}$$

Consequently, the product $H^{\mathbf{P}} \nabla^{\mathbf{P}}$ is the $(n+1)$ dimensional column vector:

$$\begin{pmatrix} 2 \sum_{j=0}^m \phi_0^2(t_j) \nabla_0^{\mathbf{P}} + \dots + 2 \sum_{j=0}^m \phi_0(t_j) \phi_n(t_j) \nabla_n^{\mathbf{P}} \\ \vdots \\ 2 \sum_{j=0}^m \phi_n(t_j) \phi_0(t_j) \nabla_0^{\mathbf{P}} + \dots + 2 \sum_{j=0}^m \phi_n^2(t_j) \nabla_n^{\mathbf{P}} \end{pmatrix}$$

which can be reduced to:

$$H^{\mathbf{P}} \nabla^{\mathbf{P}} = \begin{pmatrix} 2 \sum_{j=0}^m \phi_0(t_j) \sum_{i=0}^n \phi_i(t_j) \nabla_i^{\mathbf{P}} \\ \vdots \\ 2 \sum_{j=0}^m \phi_n(t_j) \sum_{i=0}^n \phi_i(t_j) \nabla_i^{\mathbf{P}} \end{pmatrix}$$

thus:

$$\begin{aligned} (H^{\mathbf{P}} \nabla^{\mathbf{P}})^{\top} \bullet \nabla^{\mathbf{P}} &= 2 \sum_{k=0}^n \sum_{j=0}^m \phi_k(t_j) \sum_{i=0}^n \phi_i(t_j) \nabla_i^{\mathbf{P}} \bullet \nabla_k^{\mathbf{P}} \\ &= 2 \sum_{j=0}^m \sum_{k=0}^n \phi_k(t_j) \nabla_k^{\mathbf{P}} \bullet \sum_{i=0}^n \phi_i(t_j) \nabla_i^{\mathbf{P}} \\ &= 2 \sum_{j=0}^m \left\| \sum_{i=0}^n \phi_i(t_j) \nabla_i^{\mathbf{P}} \right\|^2 \end{aligned}$$

This allows us to state that:

$$\alpha_{\max}^{\mathbf{P}} = \frac{\sum_{i=0}^n \|\nabla_i^{\mathbf{P}}\|^2}{2 \sum_{j=0}^m \left\| \sum_{i=0}^n \phi_i(t_j) \nabla_i^{\mathbf{P}} \right\|^2}$$

q. e. d.

The third step consists in decreasing $d(\mathbf{P}, \mathbf{t}, \mathbf{w})$ with respect to the weight values, considering the control point position and the node values fixed. This step uses the direction of maximum rate of decrease ([PFTV02, AW95]) of the objective function with respect to weights, i.e the negative direction of the vector $\nabla^{\mathbf{w}}$ which is the gradient of $d(\mathbf{P}, \mathbf{t}, \mathbf{w})$ with respect to \mathbf{w} . The gradient of $d(\mathbf{P}, \mathbf{t}, \mathbf{w})$ with respect to \mathbf{w} is the $(n+1)$ dimensional column vector:

$$\nabla^{\mathbf{w}} = \begin{pmatrix} \nabla_0^{\mathbf{w}} \\ \vdots \\ \nabla_n^{\mathbf{w}} \end{pmatrix} = \begin{pmatrix} \frac{\partial d(\mathbf{P}, \mathbf{t}, \mathbf{w})}{\partial w_0} \\ \vdots \\ \frac{\partial d(\mathbf{P}, \mathbf{t}, \mathbf{w})}{\partial w_n} \end{pmatrix}$$

Using equ. (2) and the sum and product derivative rules, we can state that:

$$\nabla^{\mathbf{w}} = \begin{pmatrix} 2 \sum_{j=0}^m \mathbf{E}_j \bullet \frac{\partial \mathbf{E}_j}{\partial w_0} \\ \vdots \\ 2 \sum_{j=0}^m \mathbf{E}_j \bullet \frac{\partial \mathbf{E}_j}{\partial w_n} \end{pmatrix} \quad (5)$$

Using the difference and the quotient derivative rules, we can state that:

$$\begin{aligned} \frac{\partial \mathbf{E}_j}{\partial w_i} &= - \frac{H_i(t_j) \mathbf{P}_i \sum_{i=0}^n w_i H_i(t_j) - H_i(t_j) \sum_{i=0}^n w_i H_i(t_j) \mathbf{P}_i}{\left(\sum_{i=0}^n w_i H_i(t_j) \right)^2} \\ &= - \frac{H_i(t_j) \left(\mathbf{P}_i \sum_{i=0}^n w_i H_i(t_j) - \sum_{i=0}^n w_i H_i(t_j) \mathbf{P}_i \right)}{\left(\sum_{i=0}^n w_i H_i(t_j) \right)^2} \\ &= - \frac{H_i(t_j) \left(\mathbf{P}_i - \frac{\sum_{i=0}^n w_i H_i(t_j) \mathbf{P}_i}{\sum_{i=0}^n w_i H_i(t_j)} \right)}{\sum_{i=0}^n w_i H_i(t_j)} \\ &= - \frac{H_i(t_j) (\mathbf{P}_i - \mathbf{C}(t_j))}{\sum_{i=0}^n w_i H_i(t_j)} \end{aligned} \quad (6)$$

Inserting eq. (6) in eq. (5) finally leads to:

$$\nabla^w = \begin{pmatrix} -2 \sum_{j=0}^m \frac{H_i(t_j) \mathbf{E}_j \bullet (\mathbf{P}_i - \mathbf{C}(t_j))}{\sum_{i=0}^n w_i H_i(t_j)} \\ \vdots \\ -2 \sum_{j=0}^m \frac{H_i(t_j) \mathbf{E}_j \bullet (\mathbf{P}_i - \mathbf{C}(t_j))}{\sum_{i=0}^n w_i H_i(t_j)} \end{pmatrix} \quad (7)$$

While the computation of the suitable positive scalar value α_{max}^P is efficiently handled for step 2, this task is more tedious with weight values. Indeed, the Hessian matrix with respect to \mathbf{w} , H^w , does not allow any significant simplification for the computation of the "optimal" value which is given with respect to \mathbf{w} by:

$$\alpha_{max}^w = \frac{\|\nabla^w\|^2}{(H^w \nabla^w)^\top \bullet \nabla^w} = \frac{\sum_{i=0}^n \|\nabla_i^w\|^2}{(H^w \nabla^w)^\top \bullet \nabla^w}$$

Experimental results have shown that the use of this expression, which requires to compute and store the complex expression of the Hessian matrix H^w , leads to inefficient results. On the other hand, a constant value multiplying the gradient vector often entails the divergence of the objective function near the minimum [PFTV02, AW95]. The solution we opted for is to look for a "satisfactory" scalar value α^w rather than the best one i.e. α_{max}^w . A backtracking approach is applied to achieve this task. The search is initialized with a tiny positive scalar value for α^w (for example 0.1), which is gradually increased while the decrease of the objective function is verified. Note that, we also take care to keep the positivity of the weight values in order to respect the definition of the rational approximant (see eq. 1) verifying $w_i - \alpha^w \nabla_i^w > 0, \forall i \in [0..n]$. The three steps we described allow us to decrease the error function with respect to three kinds of unknowns. The first is an optimization step, while the two others are only reduction steps corresponding respectively to one step of the steepest descent method and one step of the gradient descent method with an estimated "satisfactory" scalar weighting coefficient for the gradient. Even if these steps could be mixed in different ways to minimize the objective function, we opted for a particular blend based on our most convincing experimental results. This aims at reducing a sum of minimum squared Euclidean norms (as proposed by J. Hoschek [Hos88]). Each odd iteration of our iterative algorithm consists in reducing a minimal sum of squared Euclidean norms with respect to control points (**step 1** + **step 2**) while even iterations aim at reducing a minimal sum of squared Euclidean norms with respect to weight values (**step 1** + **step 3**).

Convergence

The convergence is studied from the sequence given by the positive values of the objective function through iterations. Odd iterations achieve the projection of data points (**step 1**) and decrease the sum of squared Euclidean norms with respect to poles (**step 2**). Even iterations also project the data points (**step 1**) and decrease the sum of squared terms with respect to weight values (**step 3**). From guess values $\mathbf{P}, \mathbf{t}, \mathbf{w}$, we can easily state that for an odd iteration i , **step 1** leads to a new set \mathbf{t}' such that $d(\mathbf{P}, \mathbf{t}, \mathbf{w}) \geq d(\mathbf{P}, \mathbf{t}', \mathbf{w})$ and that **step 2** leads to a new set \mathbf{P}' such that $d(\mathbf{P}, \mathbf{t}', \mathbf{w}) > d(\mathbf{P}', \mathbf{t}', \mathbf{w})$.

If we consider the following even iteration $i+1$, this leads from **step 1** to a new set \mathbf{t}'' such that $d(\mathbf{P}', \mathbf{t}', \mathbf{w}) \geq d(\mathbf{P}', \mathbf{t}'', \mathbf{w})$ and **step 3** leads to a new set \mathbf{w}' such that $d(\mathbf{P}', \mathbf{t}'', \mathbf{w}) > d(\mathbf{P}', \mathbf{t}'', \mathbf{w}')$. As illustrated in figure 2, this proves that alternating between odd and even iterations ensures that the positive values of the objective function describe a decreasing lower bounded sequence which proves its convergence.

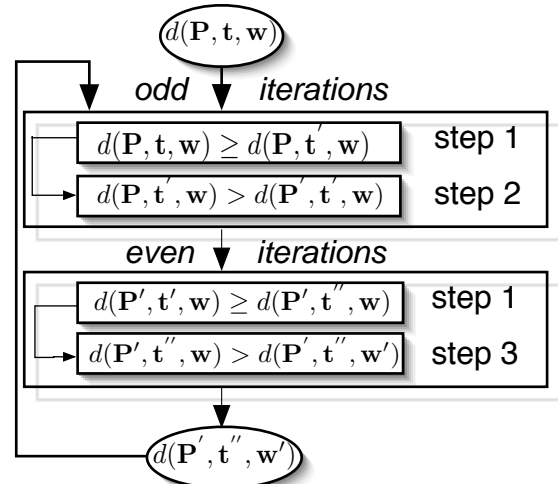


Figure 2: Illustration of the convergence

The convergence is an important robustness criterion but the theoretical efficiency of an iterative optimization method is often given by its rate of convergence. For classical objective functions depending of one type of unknowns, this information corresponds to the speed of convergence of the approximate solution towards the optimal solution. This rate can also be computed from a "residual" function converging towards zero at the solution (for example the squared Euclidean norm of the gradient). Unfortunately, such a study seems unsuitable for our method since three types of unknowns are alternatively modified. The efficiency of this algorithm can however be measured from practical consideration especially from the ratio between the computing time and the approximation accuracy. This analysis is proposed in the last section.

Stopping criteria

As for all iterative methods, a basic stopping criterion is the number of iterations achieved by the process in the case of a wrong expected accuracy or an extraordinary numerical problem. A more meaningful stopping criterion is an expected approximation accuracy for the fitting of a given curve to a given set of data points. When the degrees of freedom of the approximation curve is sufficient, this criterion is particularly suitable for our method, which ensures the strict decrease of the error function through iterations. Another stopping criterion used by classical optimization methods handling one type of unknowns, is the gradient norm of the objective function which converges towards 0 approaching a critical point. In our case, this provides two other stopping criteria, one linked to the norm of $\nabla \mathbf{P}$, the other linked to the norm of $\nabla \mathbf{w}$. Note that an optimum solution is reached when the two norms vanish.

Pseudo-code

```

/**Control values**/
N : positive integer.
 $\epsilon_d$ ,  $\epsilon_{\nabla \mathbf{P}}$ ,  $\epsilon_{\nabla \mathbf{w}}$  : positive real values.

/** Initialization**/
iter=0
initialize  $\mathbf{P}$  and  $\mathbf{w}$ 
compute nodes  $\mathbf{t} = (t_0 \dots t_m)$  such that
 $\|\mathbf{d}_j - \mathbf{C}(t_j)\| = \min_{t \in [a,b]} \|\mathbf{d}_j - \mathbf{C}(t)\|$ 
compute  $d(\mathbf{P}, \mathbf{t}, \mathbf{w})$ 
compute  $\nabla \mathbf{P}$ 
compute  $\nabla \mathbf{w}$ 
/**Main loop**/
while (iter < N) and ( $d(\mathbf{P}, \mathbf{t}) \geq \epsilon_d$ ) and
( $(\|\nabla \mathbf{P}\| \geq \epsilon_{\nabla \mathbf{P}})$  or ( $\|\nabla \mathbf{w}\| \geq \epsilon_{\nabla \mathbf{w}}$ ))
{
    iter=iter+1
    if (iter % 2 == 1) {
        compute  $\alpha_{max}^{\mathbf{P}}$ 
         $\mathbf{P} = \mathbf{P} - \alpha_{max}^{\mathbf{P}} \nabla \mathbf{P}$ 
    }
    else {
        compute a satisfactory positive value  $\alpha^{\mathbf{w}}$ 
         $\mathbf{w} = \mathbf{w} - \alpha^{\mathbf{w}} \nabla \mathbf{w}$ 
    }
    compute nodes  $\mathbf{t} = (t_0 \dots t_m)$  such that
     $\|\mathbf{d}_j - \mathbf{C}(t_j)\| = \min_{t \in [a,b]} \|\mathbf{d}_j - \mathbf{C}(t)\|$ 
    compute  $d(\mathbf{P}, \mathbf{t}, \mathbf{w})$ 
    if (iter % 2 == 1)
        compute  $\nabla \mathbf{w}$ 
    else
        compute  $\nabla \mathbf{P}$ 
}
endwhile

```

4. EXPERIMENTAL RESULTS

In the following, approximation errors are measured from: $E_m = \sqrt{\frac{d(\mathbf{P}, \mathbf{t}, \mathbf{w})}{m+1}}$ and $E_s = \sup_{j=0, \dots, m} \|\mathbf{E}_j\|_2$.

Example 1

We fit a planar rational Bézier curve and a planar quadratic clamped NURBS curve to a set of 10 planar data points given in figure (3). The curves are de-

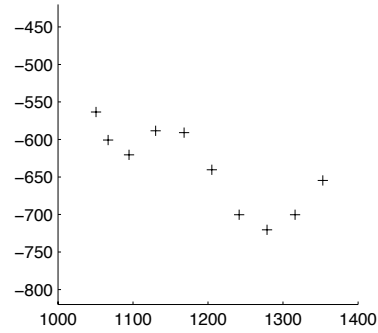


Figure 3: The 10 data points

fined on the range $[a=0, b=1]$ and are controlled by 5 poles. For the NURBS curve, the knot vector is fixed to $(0.0, 0.0, 0.0, 0.33, 0.66, 1.0, 1.0, 1.0)$. The weight values are initialized to 1 so that the two curves are respectively a Bézier curve and a B-Spline curve before starting the process. The initialization of control points \mathbf{P} has been obtained by a linear least-square minimization with respect to control points using four types of parametrization: centripetal, chordal, Foley-Nielson and Zhang [Lee89, Far01, ZCM98]. Initial approximation errors are gathered in tables 1 and 2. Table 3 shows the initial Euclidean norms of both gradients ($\|\nabla \mathbf{P}\|$ and $\|\nabla \mathbf{w}\|$).

Init.	$d(\mathbf{P}, \mathbf{t}, \mathbf{w})$	E_m	E_s
Centripetal	1289.25	11.35	17.62
Chordal	1469.27	12.12	18.65
Foley-N.	1429.62	11.95	19.68
Zhang	729.38	8.54	15.99

Table 1: Initial errors (rational Bézier curve)

Init.	$d(\mathbf{P}, \mathbf{t}, \mathbf{w})$	E_m	E_s
Centripetal	1182.93	10.87	19.00
Chordal	1642.61	12.81	20.54
Foley-N.	1382.21	11.75	20.21
Zhang	530.18	7.28	12.75

Table 2: Initial errors (NURBS curve)

	rational Bézier		NURBS curve	
Init.	$\ \nabla^P\ $	$\ \nabla^w\ $	$\ \nabla^P\ $	$\ \nabla^w\ $
Centripetal	18.81	1416.43	35.25	384.99
Chordal	21.62	1487.47	35.93	357.44
Foley-N.	21.77	1686.07	42.18	501.27
Zhang	14.84	1534.63	28.55	622.46

Table 3: Initial gradient Euclidean norms

For both types of curve, the best results are obtained with the Zhang parametrization. The corresponding approximations are given in figures 4 and 5.

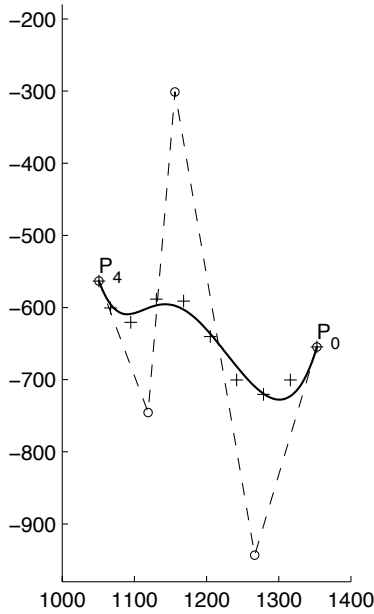


Figure 4: Zhang first approximant (Rational Bézier)

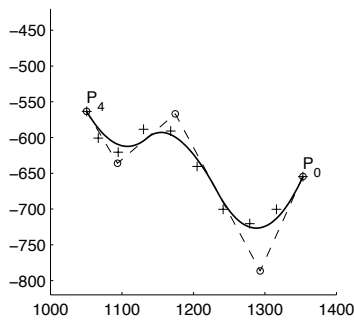


Figure 5: Zhang first approximant (NURBS)

For the rational Bézier curve, we aimed at dividing by at least 1000 the initial norm of both gradients $\|\nabla^P\|$ and $\|\nabla^w\|$ before stopping the process. So according to table 3, ε_{VP} and ε_{Vw} have been respectively set to 0.1 and 1. For the NURBS curve, we expected a very accurate fitting trying to divide by 10000 the norm of

both gradients. In this case this corresponds to fix ε_{VP} and ε_{Vw} to 0.01 and 0.1 respectively. The maximum number of iterations has been set to 10000 and the approximation accuracy ε_d has been set to 10^{-6} . Tables 4 and 5 show the approximation errors and the corresponding iteration number and computing time when the process stops (all tests have been achieved on a 2 Gh PC computer). Tables 6 and 7 give the resulting weights. Figures 6 and 7 show the resulting rational Bézier curve and the resulting NURBS curve first initialized from the Zhang parametrization.

Init.	E_m	E_s	iteration	time (s)
Centripetal	0.286	0.491	9175	< 2
Chordal	0.284	0.487	9123	< 2
Foley-N.	0.301	0.519	8277	< 2
Zhang	0.321	0.558	9145	< 2

Table 4: Resulting errors (rational Bézier curve)

Init.	E_m	E_s	iteration	time (s)
Centripetal	0.006	0.010	2903	< 1
Chordal	0.006	0.010	3187	< 1
Foley-N.	0.006	0.010	2845	< 1
Zhang	0.006	0.010	2751	< 1

Table 5: Resulting errors (NURBS curve)

Init.	w_0	w_1	w_2	w_3	w_4
Centripetal	.53	1.28	1.00	1.38	.38
Chordal	.54	1.28	.99	1.39	.38
Foley-N.	.52	1.31	.98	1.36	.45
Zhang	.55	1.36	.87	1.31	.61

Table 6: Resulting weights (rational Bézier curve)

Init.	w_0	w_1	w_2	w_3	w_4
Centripetal	.76	1.14	1.15	1.04	.84
Chordal	.73	1.14	1.21	1.14	.67
Foley-N.	.77	1.14	1.14	1.02	.88
Zhang	.78	1.15	1.11	0.98	.93

Table 7: Resulting weights (NURBS curve)

One can notice that the initial approximation errors given by the linear least square method are drastically improved. For the rational Bézier curve, E_m is roughly divided by 30 while E_s is roughly divided by 40. For the NURBS curves E_m and E_s are roughly divided by 2000. To measure the influence of handling weights on the resulting approximation errors, we fitted these curves using the iterative method we proposed in [CDB05], which does not handle weight values. Note that according to the weights set to 1, these curves are respectively Bézier and B-Spline curves. For the Bézier curve, the best results we can obtain are $E_m = 1.40$ and $E_s = 2.74$. For the B-Spline curve $E_m = 0.38$ and $E_s = 0.84$. This states that for the rational Bézier curve, handling weights allows to reduce

by roughly 5 the errors obtained without optimizing the objective function with respect to weights. For the NURBS curve, errors are roughly divided by 50.

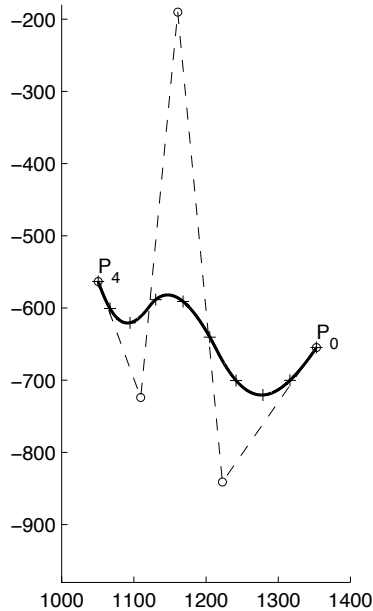


Figure 6: Resulting Rational Bézier

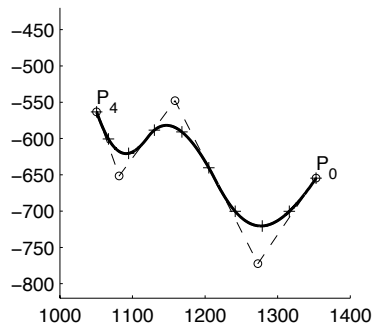


Figure 7: Resulting NURBS

Example 2

In this second example, we fit a clamped cubic NURBS curve to the set of 201 data points given in figure 8. This set is used to emphasize the interest of the improved Hoschek method (IH) proposed in [SD03]. This allows us to make a rapid comparison of our approach with this iterative method providing very convincing results. Note that this method is for B-Spline fitting and does not handle weights. As in [SD03], the curve we fit is defined on the parametric range $[a=0, b=1]$ and is controlled by 19 poles. Intermediate knots are uniformly spaced.

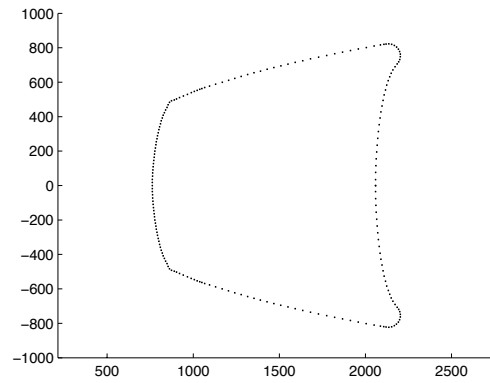


Figure 8: The 201 data points

To start with the same configuration for both methods, weights are initialized to 1 and the first approximation curve (given in figure 9) is obtained by a linear least-square minimization with a centripetal parametrization.

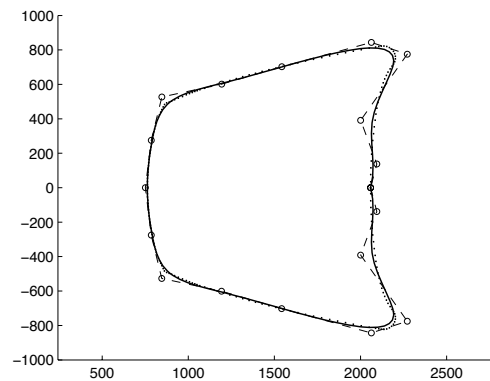


Figure 9: The first approximant

In order to compare both methods we initialized our fitting accuracy with the best mean error which can be obtained with the improved Hoschek method. On a 2Gh PC computer, the best results we can obtain with this method are $E_m = 1.88$ and $E_s = 6.29$ in 4000 iterations (300 s). Setting ϵ_d to 1.88, our process stops after 1953 iterations (18 seconds). The corresponding maximum error is 5.59. This again emphasizes the importance of handling weights in this problem. Moreover, as illustrated in the graph figure 11, where E_m is collected over 5000 iterations, our method leads to a significant improvement of this value. Indeed, after 5000 iterations approximation errors are $E_m = 1.29$ and $E_s = 4.43$. The corresponding approximation curve is given in figure 10. One can also emphasize the smooth and strict convergence of E_m to

wards the minimum, and its very fast decrease during the first iterations.

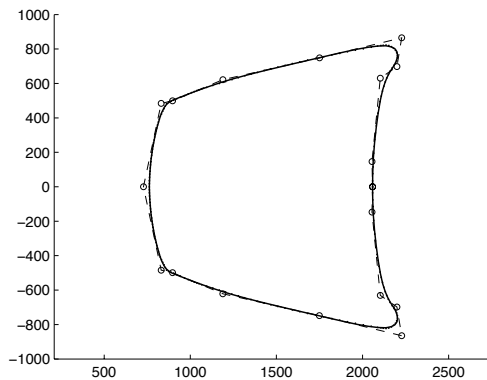


Figure 10: The approximant after 5000 iterations

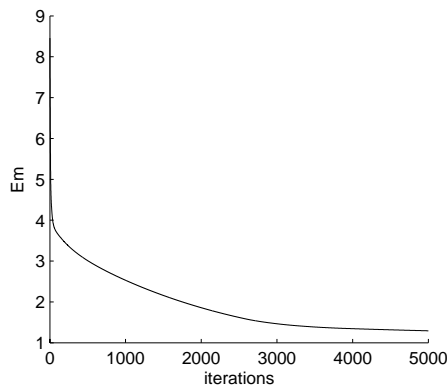


Figure 11: E_m over 5000 iterations

5. CONCLUSION

We have proposed an iterative method to fit a rational pole curve on a set of data points in the least-square sense. Based on a relaxation approach, our algorithm aims at alternatively minimizing a sum of squared Euclidean norms with respect to three types of unknowns: the control points, the node values, and the weights. The method uses a projection step to optimize the objective function with respect to node values and two robust gradient based techniques to optimize the objective function with respect to control points and weight values. The positive error function $d(\mathbf{P}, \mathbf{t}, \mathbf{w})$ monotonically reduces through iterations ensuring the convergence of the process. This algorithm, suitable for all types of rational pole curve, is robust and efficient. We are currently extending it to surfaces. Experimental results emphasize the drastic influence of handling weights in this problem.

6. REFERENCES

- [AB01] M. Alhanaty and M. Bercovier. Curve and surface fitting and design by optimal methods. *Computer Aided Design*, 33(2):167–182, 2001.
- [AW95] G. Arfken and H. J. Weber. *Mathematical Methods for Physicists*, 4th ed. Orlando Academic Press, 1995.
- [CDB05] J. C. Chambelland, M. Daniel, and J. M. Brun. A robust iterative method devoted to pole curve fitting. In *CAD/GRAPHICS 2005 conference (Hong-Kong, Chine, December 7-10, 2005) proceedings*, ISBN 0-7695-2473-7, pages 22–27. IEEE Computer Society, 2005.
- [Dan96] M. Daniel. *Data Fitting with B-splines Curves*. In *Modelling and Graphics in Science and Technology*, J. Teixeira et J. Rix Eds., Springer Verlag, pp 91-104, 1996.
- [Far01] G. Farin. *Curves and Surfaces for CAGD, a Pratical Guide*, 5th ed. Morgan Kaufmann, 2001.
- [Hos88] J. Hoschek. Intrinsic parametrization for approximation. *Comput. Aided Geom. Design*, 5(1):27–31, 1988.
- [HW05] S. M. Hu and J. Wallner. A second order algorithm for orthogonal projection onto curves and surfaces. *Comput. Aided Geom. Design*, 22(3):251–260, 2005.
- [Lee89] E. T. Y Lee. Choosing nodes in parametric curve interpolation. *Computer Aided Design*, 21(6):363–370, 1989.
- [LS86] P. Lancaster and K. Salkauskas. *Curve and Surface Fitting: An Introduction*. Academic Press, 1986.
- [MH03] Y. L. Ma and W. T. Hewitt. Point inversion and projection for NURBS curve and surface: Control polygon approach. *Comput. Aided Geom. Design*, 20(2):79–99, 2003.
- [Mor97] M. E. Mortenson. *Geometric Modeling*, 2nd edition. John Wiley and Sons, 1997.
- [PFTV02] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C++*, 2nd ed. Cambridge University Press, 2002.
- [PT96] L. A. Piegl and W. Tiller. *The NURBS Book*, 2nd ed. Springer, 1996.
- [SD03] E. Saux and M. Daniel. An improved Hoschek intrinsic parameterization. *Comput. Aided Geom. Design*, 20(8-9):513–521, 2003.
- [SKH98] T. Speer, M. Kuppe, and J. Hoschek. Global reparametrization for curve approximation. *Comput. Aided Geom. Design*, 15(9):869–877, 1998.
- [ZCM98] G. Zhang, F. Cheng, and K. T. Miura. A method for determining knots in parametric curve interpolation. *Comput. Aided Geom. Design*, 15(4):399–416, 1998.

3D NOffset mixed-element mesh generator approach

Claudio Lobos and Nancy Hitschfeld-Kahler

Depto Ciencias de la Computación,
FCFM, Universidad de Chile,
Blanco Encalada 2120,
Santiago - Chile, Zip code: 837-0459
clobos@dcc.uchile.cl, nancy@dcc.uchile.cl

ABSTRACT

In this paper we present a new approach to generate a mixed mesh with elements aligned to boundary/interfaces wherever is required. A valid element is: (a) any convex co-spherical element that fulfills the requirements of the underlying numerical method and (b) any element that satisfies domain specific geometric features of the model. The algorithm is based on the normal offsetting approach to generate coarse elements aligned to the boundary/interfaces. Those elements are later refined to accomplish layer density requirements. The main steps of the algorithm are described in detail and examples are given to illustrate the already implemented parts. As far as possible, we contrast this algorithm with previous approaches.

Keywords: mixed-element meshes, normal offsetting approach, advancing front.

1 INTRODUCTION

CVM-conforming Delaunay meshes are Delaunay meshes where the circumcenter (center of the circumcircle of an element) of each boundary/interface element is inside the element itself or inside a neighboring element through internal edges/faces. The previous restriction guarantees that when Voronoi regions are used as control volumes in the control volume method (CVM), a mesh satisfies the conditions for the numerical integration around each boundary or internal point.

CVM-conforming Delaunay meshes are used in several applications, but in particular in the simulation of semiconductor devices. In this application, the meshes must also fulfill two additional requirements: due to the geometry of the devices, they should properly model very thin layers, and due to physical properties, edges parallel to the current flow are desirable.

Octree-based and mixed-element tree-based (MET) mesh generators have been developed for the generation of 3D mixed element CVM-conforming Delaunay meshes [2, 4]. The MET approach generalizes the modified octree approach [8, 6] in several aspects: (1) it considers the whole device no longer encapsulated in a single octree, but partitioned in a set of basic elements: cuboids, rectangular prisms and pyramids; each basic

element becomes the root of a tree, (2) elements are either bisected or refined by introducing appropriate edge points in order to get the density requirements and (3) a Delaunay tessellation that includes tetrahedra, prisms, pyramids and other basic elements [3] is generated. The main advantage of this approach is the use of several element types that naturally fit the requirements of the CVM. The main drawback is that it can not efficiently generate element faces aligned to the boundary/interfaces and it generates too many points if the device geometry is not well-oriented.

The normal-offsetting approach has been used to generate 3D tetrahedral meshes for the CVM with element edges aligned to the boundary [5]. In this approach, the surface and interfaces of the device are triangulated first and then, the front is moved, step by step, to generate almost prismatic elements. These elements are later divided into tetrahedra. In order to generate a CVM-conforming Delaunay tetrahedral mesh, bad-shaped tetrahedra are divided by orthogonal refinement or by applying a more complex strategy in case the orthogonal refinement does not work [7]. The main advantages of this approach is that it generates edges aligned to the boundary, wherever required. The main drawback is that the improvement of bad-shaped tetrahedra along the boundary/interfaces might require the insertion of a high number of points [7, 4]. In addition, the generation of each prismatic element is slow because it requires a front expansion computation.

In this paper we present an algorithm that takes the advantages of the mixed element tree and of the normal offsetting approaches and improves their drawbacks. First, it includes as valid mesh elements all the convex co-spherical elements that fulfill the requirements of the CVM. Cuboids, in particular, are very useful to repre-

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.

Short Communications proceedings ISBN 80-86943-05-4
WSCG'2006, January 30 – February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

sent very thin layers. Tetrahedra are only used where other element types cannot be used. Second, it uses the normal offsetting approach to generate coarse elements edges aligned to the boundary. Coarse elements are later refined to generate the required layer density specified together with the front information. We conjecture that the refinement of an element involves less, more robust and simpler operations than the ones involved in the computation of a front expansion. The main steps of the algorithm are described in detail and several examples are shown.

2 BASIC CONCEPTS

2.1 NOffset

Normal Offsetting (NOffset) is a particular case of the advancing front technique [1]. NOffset adds the constraint that the expansions must be parallel to the fronts.

In order to compute the expansion of a front face in a distance h , the new position of each face point is calculated. The new position of a point P depends on: (a) the normal vector n of the face, (b) a distance h and (c) the fact that other geometry faces that share P can also be front faces or not.

Let k be the number of front faces that share the point P , n_i be the normal vector of each front face and h_i be the distance of expansion associated to each front face. The new point P' is calculated following one of the next rules [5]:

1. Only one front face contains P ($k=1$). Then $P' = P + n_1 h_1$
2. Two front faces share P ($k=2$). Then $(P' - P) \cdot n_i = h_i \mid i = \{1, 2\}$ and $(n_1 \times n_2) \cdot (P' - P) = 0$.
3. Three front faces share P ($k=3$). Then $(P' - P) \cdot n_i = h_i \mid i = \{1, 2, 3\}$
4. Four or more faces share P ($k > 3$). Then, the nearest point to all intersection planes is computed. Our work does not consider this case yet.

Figure 1 shows an example where the new positions of P and Q (P' and Q' , respectively) are obtained by applying rule 2.

3 ALGORITHM DESCRIPTION

In order to generate an adequate mixed element mesh, we have divided this process in the following steps:

- Generation of a coarse anisotropic mixed element discretization
- Generation of a primitive mesh.
- Fulfilling the required layer density
- Fulfilling the required maximum edge length

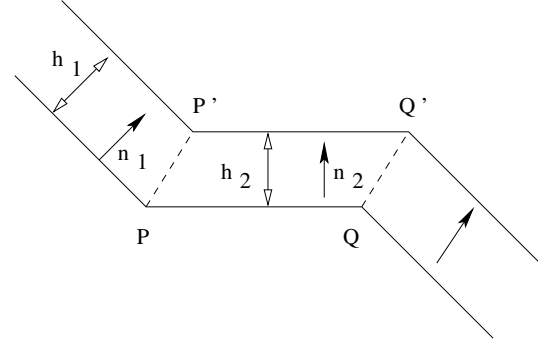


Figure 1: The initial edge PQ is expanded in a distance h_2 .

- Generation of a CVM-conforming mixed element mesh

The next subsections describe each of these steps.

3.1 Generating a coarse mixed element discretization

This first step generates coarse anisotropic elements aligned to the boundary or interfaces according to the front information specified by the user. Two inputs are required: the initial geometry to be meshed and a file where the user specifies the fronts with the following information [5]:

- A list of original geometry faces that conform the front.
- Thickness of first layer (h_{loc}).
- Coarsening factor of the next layers ($factor$).
- Number of layers ($endline$).
- Maximum edge length of the generated elements (mel).

For each front face only one coarse element is generated. The thickness (h_{final}) of each one of these elements is obtained from the number of layers and coarsening factor as follows:

$$h_{final} = h_{loc} \cdot \sum_{i=0}^{t-1} factor^i \quad | \quad t = endline$$

The geometry is specified by a set of polyhedral elements and a subset of the faces that define those elements conform the fronts. Each face is part of just one front.

Let us use Figure 2 to illustrate the algorithm to generate one coarse anisotropic element. Figure 2(a) shows a truncated prism with two front faces: the top one and the right one. Let F_n be the new face obtained by the expansion of the top front face. Figure 2(b) shows the

points of F_n . It can be observed that the position of the points to the right was obtained applying the rule 2 and the position of the points to the left was obtained applying rule 1. If each front face point is shared by only front faces, F_n is already in the right position. But if this fact does not occur, some points of F_n may need to be recalculated. Then, the next step is to find the intersection of F_n with adjacent faces that are not front faces. This is the case of the left face of the truncated prism. The new positions of the left points of F_n are shown in Figure 2(c). Once F_n is well defined (Figure 2(d)), the truncated prism is divided into two elements: the first one formed by the top front face, the face F_n , and the lateral faces that join both faces, and the second one formed by the rest of the original truncated prism. This can be observed in Figures 2(e) and 2(f). The process illustrated in Figure 2 is repeated for each polyhedral element whose boundary faces must be expanded.

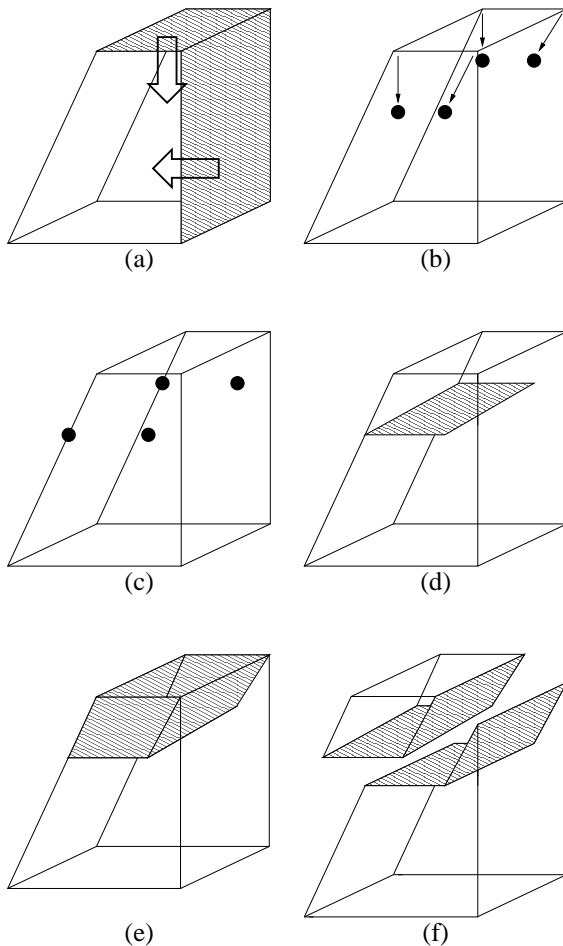


Figure 2: Building a macro element.

The algorithm 1 implements the designed strategy for this step. We assume that each front can be expanded to the maximum distance specified by the user.

Note that when adding a new face it may overlap a previous one. In that case the overlapped face is split

Input: front specification file and the geometry

for each face F of each front **do**

 let E be the element that contains F

for each face point P (not expanded) **do**

 find which rule applies to P

$P' = \text{NOffset}$ of P using the previous rule

afne = adjacent faces to F that do not expand

 move each point P' to the intersection with afne

 join the points P' to form the new face F_n

 find the type of F_n

F_j = lateral faces between F and F_n

E_n = new element defined by F , F_n and F_j

 find the type of E_n

 update element E by $E - E_n$

Output: coarse discretization of the original geometry

Algorithm 1: Generating a coarse mesh

into two new ones. If other element shares the same overlapped face, this element is also updated with the two new faces.

The previous algorithm shows that the calculation of the NOffset for each point P is a very expensive computational task. That is why we do it only once at the generation process of the aligned elements. This strategy avoids applying the previous algorithm each time the user wants to generate a new parallel layer.

We have left out of the scope the detection and handling of collisions among the fronts that require a global approach. Figure 3 shows three cases of problematic collisions and three possible solutions. Figure 3(a) shows the case where an expanded edge becomes an inverted edge. This problem can be solved by computing this edge expansion until the edge length is zero, and then, as shown in the right picture, the expansion of the neighboring edges to the target one should continue. Figure 3(b) shows an edge expansion that goes out of the original geometry because the current implementation of the algorithm only cut it by the neighboring face. Figure 3(c) shows a case where two different fronts collide. This problem can be solved by testing if the current expansion crashes a previous one. Although these problems have been studied, they are not implemented yet.

3.2 Generating a primitive mesh

At this point we have an initial mesh, that might contain general polyhedra as coarse elements. Since it is difficult to develop a method that is capable of refining any type of element, we would like to have a subset of different kind of elements known as primitives elements: tetrahedra, prism, pyramid, bricks and some truncated variants of them. For this type of elements we are capable of defining a refinement strategy.

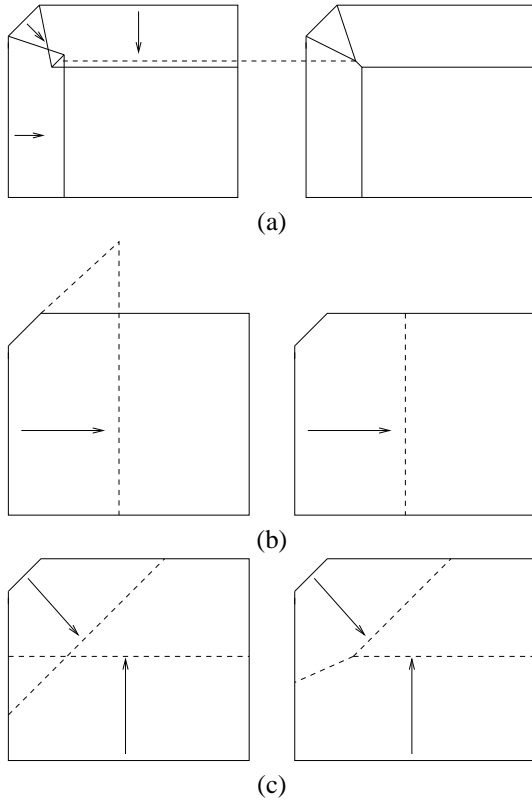


Figure 3: Collisions between expansions and possible solutions

The main idea is to split each coarse element in several primitive elements and refine those elements whenever the density constraints (layer density and maximum edge length) are not accomplished.

Although this step is not very hard to implement it is not done yet, because we have several examples where this is not required. This occurs, for example, when the coarse elements generated after the first step are directly primitive elements. The priority was given to other steps of the algorithm and this was left as part of the ongoing work.

3.3 Fulfilling the required layer density

The input of this step is the initial discretization composed of coarse anisotropic mixed elements aligned to boundary/interfaces and of polyhedral elements that model the part of the device geometry (cavities) where elements aligned to boundary/interfaces are not required. This step can be done independently of the previous one, i.e., the input could be a primitive coarse mesh or just a coarse mesh as coming from the first step.

Each coarse element contains the information of the front by which was generated. Our algorithm splits each coarse element by planes parallel to the front face at a distance defined by the expression $hloc * factor^i$ | $i = 0, \dots, endl ine - 2$ from the front face. The first layer is then located at a distance of $hloc$ from the front

Input: Coarse discretization of the geometry
for each element E **do**
 get front data
 for ($i = 0$; $i < endl ine - 1$; $i++$) **do**
 for each F_j **do**
 compute the intersection points produced by $expansion_i$
 generate $face_i$ defined by the points of $expansion_i$
 generate $element_i$

Output: Discretization with the required layer density

Algorithm 2: Fulfilling layer density requirements

face, the second layer is located at a distance of $hloc + hloc * factor$ from the front face, and so on. Note that the layer obtained with $i = endl ine - 1$ was already calculated in subsection 3.1 to build the coarse element.

Figure 4 illustrates one refinement of a coarse element. Figure 4(a) shows the discretization of the truncated prism shown in Figure 2. The top coarse element is refined once by intersecting its edges with a plane located at the distance $hloc$ from the front face as shown in Figure 4(b). The coarse element is divided into two new elements as shown in Figure 4(c) and (d).

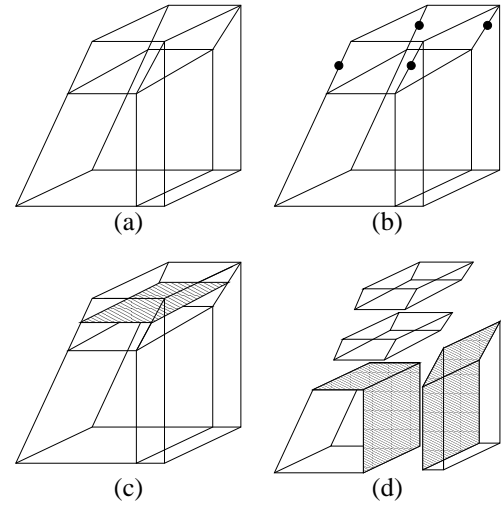


Figure 4: Refining a front element.

Let E be a coarse element, F be the front face of E , F_n be the face of E obtained by the maximum expansion of F , and F_j , the faces of E generated by joining F and F_n . The algorithm splits first all the F_j of the element E at the locations defined by $hloc * factor^i$ | $i = 0, \dots, endl ine - 2$, and then builds the new elements. The coarse element is divided in a number of new elements equal to the value of $endl ine$ hence each F_i is also split in a number of faces equal to this quantity. The pseudo-code of the refinement process is shown in algorithm 2.

3.4 Fulfilling the required maximum edge length

As we mentioned in subsection 3.1 a front specification has several fields. Two of them are the maximum edge length (*mel*) constraint and the list of faces to expand. The *mel* field affects all the elements generated due to an expansion of the faces defined on this front.

We re-use the previous work (subsection 3.3) to accomplish the *mel* constraint on brick type of elements. The layering process of a brick generates only new bricks hence the same algorithm can be applied in other directions until the *mel* is satisfied. Figure 5 shows how this process is done.

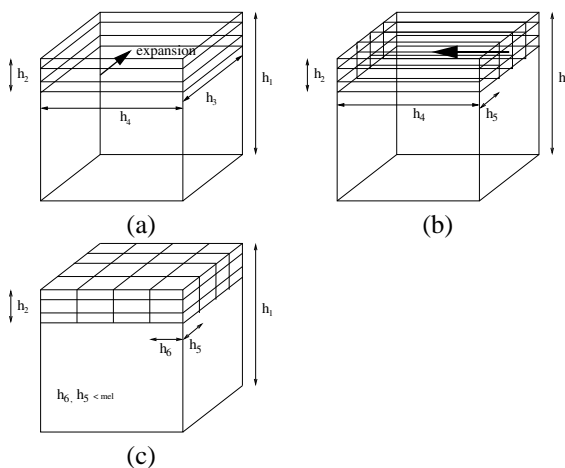


Figure 5: (a) Layer density constraint satisfied (b) *mel* satisfied in one direction (b) *mel* satisfied in the other direction

The same strategy cannot be applied over other types of element, at least not in every direction. It is necessary to specify the way to split the edges and the way to form the new elements for each primitive type. This task is part of the ongoing work.

3.5 Generating a cavity mesh

When all the expansions are done, there might remain a portion of the initial body unmeshed like the example shown in figure 6(b). When this happens it means that the user do not need a specific element density in that zone. Then the final step is to build a conforming mesh.

We have a library capable of building a tetrahedralization of a body. When all the steps are done, the remaining parts of the original body are tetrahedralized and the overall process is finished. Figure 6(f) shows an example of it.

A better strategy is to implement a mix element cavity generator, to produce a real mixed mesh. We have already a mesh generator to accomplish this task [4], however it must be adapted to generate a cavity mesh.

Example 1 (Figure 6) consists of: (a) the initial body, (b) the generation of the coarse mesh in relation to the

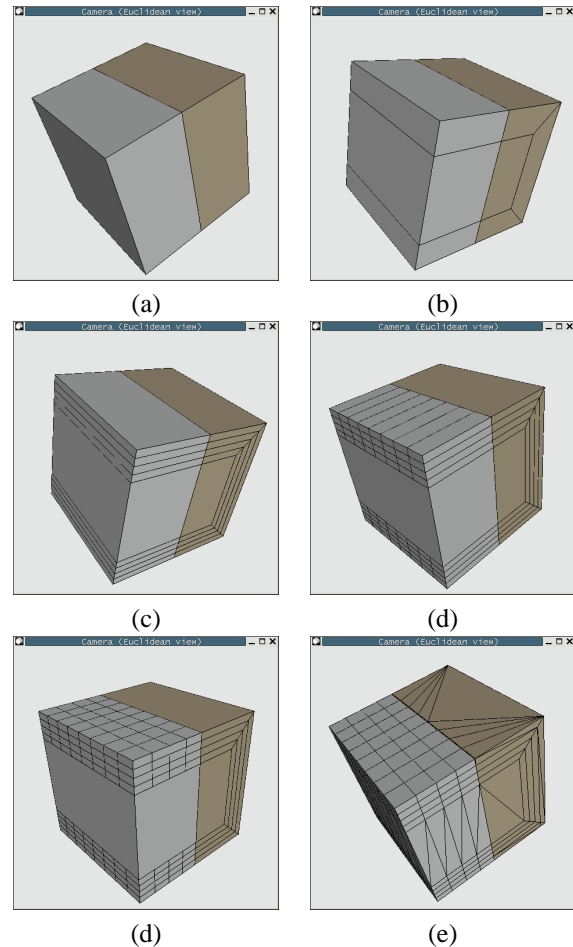


Figure 6: Example 1: mesh generation process for a simple geometry

given front input, (c) satisfying layer density, (d) satisfying the *mel* attribute in one direction (only for brick type of elements), (e) satisfying the *mel* attribute in the other direction and (f) the final mesh including the tetrahedralization of the cavity.

4 COMPARISON BETWEEN NOFF-SET STRATEGIES

There is an implementation of a NOffset strategy with only the tetrahedral type of elements specified in [5]. The differences between that work and ours are: (a) we use mixed elements, (b) we apply NOffset only one time and not every time a new layer is needed and (c) we refine the coarse elements in order to fulfill the layer density.

The more important difference between both implementations is produced in the layering step. The next table shows the number of operations to calculate each new point by each strategy in relation to the number of layers n required by the user.

current implementation	$15 + 6 * n$
old implementation	$15 * n$

The most important result is that we obtain a 60% reduction in the number of operations in relation to the old strategy. This is because the refinement process is much easier than to apply NOffset at each time a new layer is needed.

Another result that we should obtain in the future is that the final mesh should need less elements to accomplish the same required density. This is because we use mixed elements. The worst case is to mesh a body with just tetrahedra so the final number of elements would be the same in both strategies.

5 EXAMPLE AND ONGOING WORK

The second example is a body used for real semiconductor devices analysis called a bipolar transistor. Figure 7 shows the entire process: (a) the original body, (b) the coarse mesh, (c) accomplishment of layer density constraint, (d) generation of a cavity tetrahedralization and (e) and (f) the final mesh.

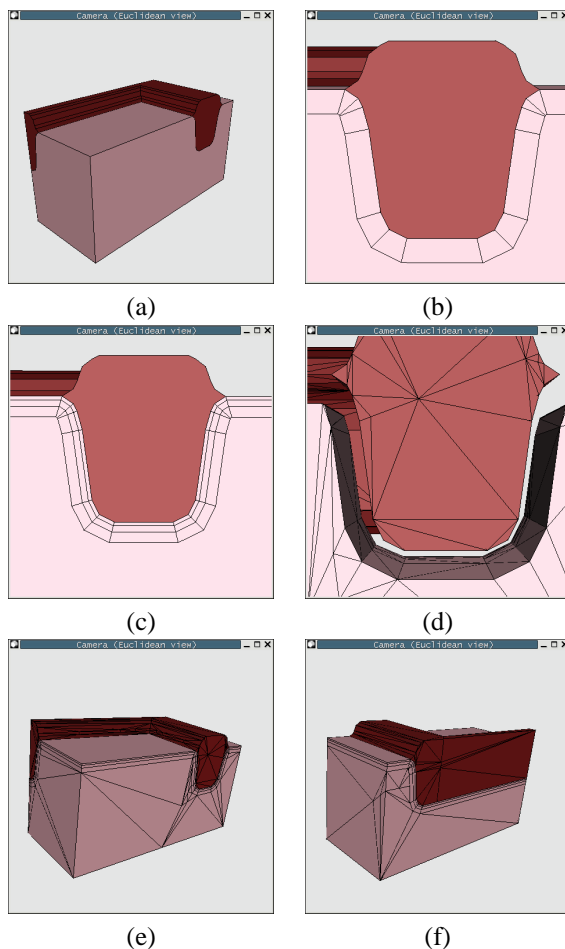


Figure 7: Example 2: mesh generation for a bipolar transistor

Currently, we are working on: (1) the generation of a non-conforming tessellation composed of coarse co-spherical elements such as cuboids, some kinds of

prism and pyramid, and tetrahedra inside the coarse anisotropic elements before the layer density is generated, (2) the generation of a final mixed element mesh, (3) making the mesh Delaunay and (4) improving the quality of the elements inside the cavity.

6 ACKNOWLEDGMENTS

This work has been supported by Fondecyt Project N° 1030672.

REFERENCES

- [1] Pascal J. Frey, Houman Borouchaki, and Paul L. George. Delaunay tetrahedralization using an advancing front approach. In *5th International Meshing Roundtable*, pages 31–46, 1996.
- [2] N. Hitschfeld, P. Conti, and W. Fichtner. Mixed Elements Trees: A Generalization of Modified Octrees for the Generation of Meshes for the Simulation of Complex 3-D Semiconductor Devices. *IEEE Trans. on CAD/ICAS*, 12:1714–1725, November 1993.
- [3] N. Hitschfeld and R. Farías. 1-irregular element tessellation in mixed element meshes for the control volume discretization method. In *Proceedings of the 5th International Meshing Roundtable*, pages 195–204. Pittsburgh, Pennsylvania, U.S.A., 1996.
- [4] N. Hitschfeld-Kahler. Generation of 3d mixed element meshes using a flexible refinement approach. *Engineering with Computers*, November 2004. Accepted for publication.
- [5] Jens Krause. *On boundary conforming anisotropic Delaunay meshes*. PhD thesis, ETH Zürich. Series in Microelectronics, Vol. 115, 2001.
- [6] Mark S. Shephard and Marcel K. Georges. Automatic Three Dimensional Generation by the Finite Octree Technique. In *International Journal for Numerical Methods in Engineering*, volume 32, pages 709–749, 1991.
- [7] L. Villablanca. *Mesh Generation Algorithms for Three-Dimensional Semiconductor Process Simulation*. PhD thesis, ETH Zürich. Series in Microelectronics, Vol. 97, 2000. Hartung-Gorre Verlag, Konstanz, Germany.
- [8] M.A. Yerry and M.S. Shephard. Automatic Three-dimensional Mesh Generation by the Modified-Octree Technique. *International Journal of Numerical Methods in Engineering*, 20:1965–1990, 1984.

A background-priority discrete boundary triangulation method

Robert Edward Loke
IIT-CNR,
Via Giuseppe Moruzzi 1,
56124 Pisa, Italy
robert.loke@iit.cnr.it

Frederik W. Jansen
Dept. of Mediamatics, EWI,
Delft University of Technology,
2600 GA, The Netherlands
f.w.jansen@ewi.tudelft.nl

Hans du Buf
Vision Laboratory, FCT,
University of Algarve,
8000 Faro, Portugal
dubuf@ualg.pt

ABSTRACT

Discrete boundary triangulation methods generate triangular meshes through the centers of the boundary voxels of a volumetric object. At some voxel configurations it may be arbitrary whether a part of the volume should be included in the object or could be classified as background. Consequently, important details such as concave and convex edges and corners are not consistently preserved in the describing geometry. We present a “background priority” version of an existing “object priority” algorithm [6]. We show that the *ad hoc* configurations of the well-known Discretized Marching Cubes algorithm [13] can be derived from our method and that a combined triangulation with “object priority” and “background priority” better would preserve object details.

Keywords: Computer Graphics: Curve, surface, solid, and object representations; Computer aided design (modeling of curves and surfaces); Computational geometry; Image processing.

1 INTRODUCTION

Volume models consist of three-dimensional arrays of measured or computed values. Isosurfacing and other segmentation techniques are often applied to select object boundaries within a volume, which may then be converted into triangular meshes. The Marching Cubes algorithm [11], for instance, builds a triangle mesh through interpolation points between the centers of the discrete voxels. This gives an accurate isosurface but also leads to a large number of triangles. The Discretized Marching Cube (DMC) method [13] constrains the interpolated positions to midpoints between voxel centers. This reduces the number of triangles and their possible orientations, which makes it easier to construct larger surface elements out of neighboring triangles with equal orientation. An alternative approach is to first segment the data, by thresholding or edge detection, into binary components and then to construct a surface through the centers of the boundary voxels of the discrete objects [6]. This method is fast and therefore often used for previewing, without computing a more accurate surface construction. It has similar advantages as the DMC method in that the number of triangles is strongly reduced compared to the exact interpolation surface method. In fact, as we will see below, DMC can be seen as a special case of the “discrete”

surface method. In the following we equate the boundary consisting of only voxels with the surface model through the boundary voxel centers. We will call them both “discrete surface” in contrast to the “continuous” or “exact” surface that is interpolated between the voxel centers.

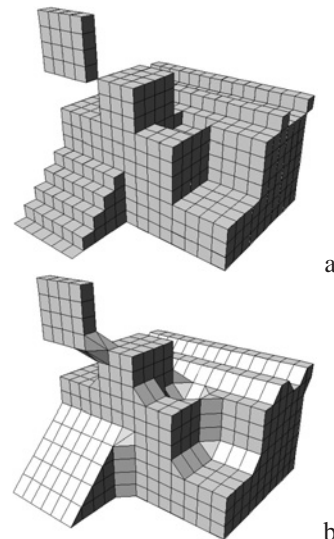


Figure 1: Object with 6-connected surface (a) and 18/26 connected surface (b)

As volume models are likely to become larger in the near future due to increased scanning and simulation resolutions (1024^3 is a regular size nowadays), the differences in accuracy will diminish and a limited triangle count of the discrete models will become a major advantage.

Since the early seventies, quite some mathematical ingenuity has been invested in the study of boundary voxel configurations to find out under which condi-

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.

Short communication proceedings, ISBN 80-86943-05-4
WSCG'2006, January 30 – February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

tions a discrete surface (consisting of only neighboring boundary voxels) has similar topological properties as a regular manifold surface model, i.e. correctly separates the interior from the exterior. This is relevant for thinning and skeletonization [4, 1, 15], ray casting volumetric objects [2, 5, 7], and surface construction [17, 16]. The study of these properties is known as digital topology [9, 8] and has resulted in several boundary definitions by Morgenthaler and Rosenfeld [14], Malgouyres [12], Kovalevsky [10] and Couprie and Bertrand [3]. The latter introduced the notion of simplicity surface, which constitutes a boundary consisting of boundary points that are adjacent and are not simple points. Simple points are points that can be removed from the boundary without altering the topology. They give several operational definitions for simple points and they prove that a simplicity surface can be built out of only 8 different $2 \times 2 \times 2$ voxel configurations [3]. On the basis of these configurations it could be easy to define a boundary triangulation method.

However, the definition of simplicity surface is not of much value from a practical point of view, as we will see in Section 2. A more useful method was proposed by Kenmochi et al. [6] (below we refer to this method as the Kenmochi method). They define the boundary of a discrete solid as the boundary of the set of connected tetrahedra that constitute the volumetric object. To construct the boundary, the object is first decomposed into a set of tetrahedra, and after removing the “double” surfaces shared by neighboring tetrahedra, the “single” outside faces constitute the overall boundary. They also presented a construction method that directly generates the composite boundary and deals with degenerated cases as dangling edges and folded surfaces.

The Kenmochi method is a sound and useful method. However, we can make an interesting observation: the method generates a surface with a maximum envelope, which is not in all cases desired. We will illustrate this issue by showing how the configurations of the Discretized Marching Cubes (DMC) method [13] can be derived from the Kenmochi method. It will make clear that the Kenmochi method has some implicit assumptions that do not in all cases give the expected and desired result. In Section 4, we propose an alternative triangulation scheme which is closer to the DMC configurations. But first, we start with an introduction on discrete surface representations.

2 DISCRETE SURFACE REPRESENTATIONS

A volumetric representation consists of a three-dimensional grid of voxel cubes, where each voxel cube stores one or multiple values. We limit the discussion here to regular grids that constitute a discrete space with voxels that are either black (the object) or white (the background). Each voxel has

a $3 \times 3 \times 3$ neighborhood with other voxels which are 26-adjacent to the central voxel from which 6 voxel centers have a Manhattan distance to the neighborhood center of one, 12 centers a distance of 2, and 8 centers a distance of 3 steps in orthogonal directions. This voxel neighborhood can be decomposed into 8 cuberille cells, see Figure 2; the figure shows the cuberilles between the voxel centers.

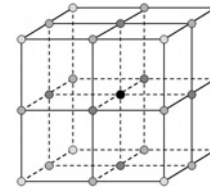


Figure 2: Adjacency relations

In the 3D example shown in Figure 3, the black points constitute a discrete object with only one interior point (marked gray) and 26 boundary points. If we define the boundary in terms of a set of 6-connected points, then we have 26 boundary points. If we also include 18- and 26-connectivity then the edge and corner points become “simple,” i.e. they do not longer border directly to the interior. The 26-surface would be only the octahedron around the interior point (Figure 3b), whereas the 6-surface would be the full cube (Figure 3a), what in most practical cases would be the desired result. Hence, if we want to maintain straight corners then we should eliminate the “short cuts”. However, if we have an oblique or curved surface, then we would like to use the diagonal short cuts of the 26-adjacency in order to avoid the stair-casing of the 6-surface representation (see Figure 1).

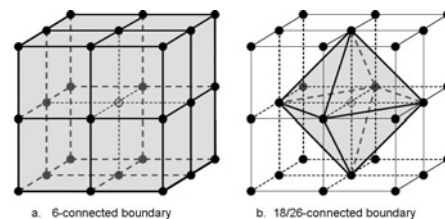


Figure 3: 3D example: 6-connected and 26-connected boundaries

Kenmochi circumvents this dilemma in her method by defining the boundary of the discrete object as the boundary of the set of discrete simplexes that constitute the object. In 2D the simplex is a triangle and in 3D a tetrahedron. If the 3D discrete object can be decomposed into a set of connected (non-degenerated) polyhedra, then its boundary is a manifold. This condition excludes degenerated cases as dangling faces and edges, and other parts that are not properly connected to the main volume. The result of this definition is a surface that encloses the whole object volume and that is locally 6-connected and 18 or 26-connected in case of an oblique surface or a concave corner (see Figure 1b).

Kenmochi gives a slice-by-slice and cell-by-cell construction method that directly generates a correct surface using 14 triangulation patterns for $2 \times 2 \times 2$ voxel configurations (Figure 5). The names of the configurations (P3a, etc.) are taken from [6].

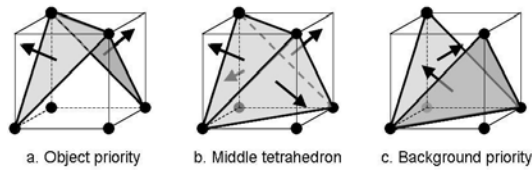


Figure 4: Two different triangulation patterns for Kenmochi configuration P6b, one with the middle tetrahedron (a) and one without (c)

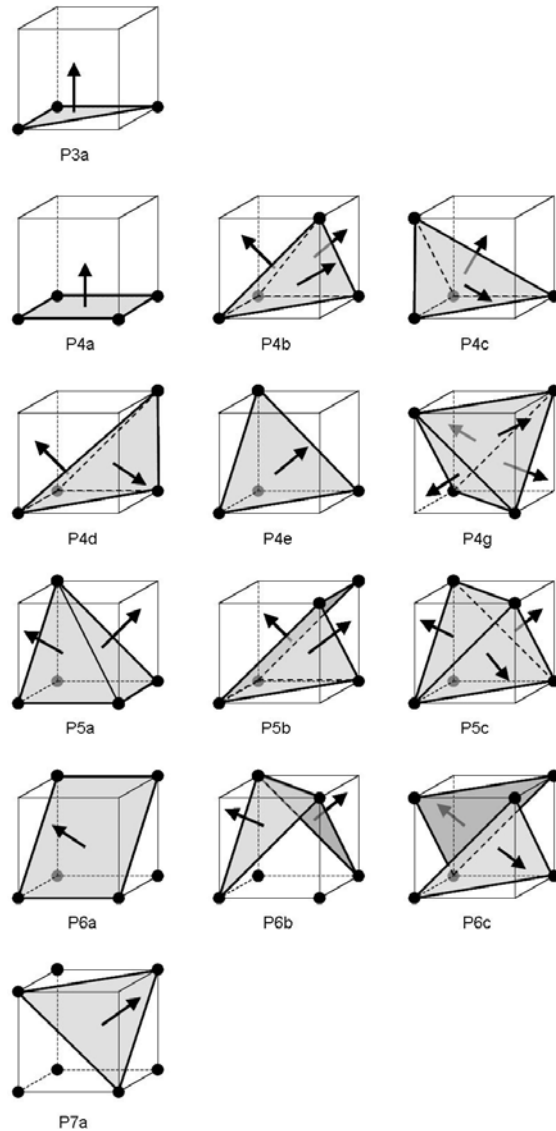


Figure 5: Kenmochi configurations

Although the Kenmochi method maintains sharp corners in case of convex configurations, it still generates oblique faces in concave situations due to the fact

that the method generates tetrahedra in corners (see Figure 1b). If we want to avoid this we should give background points a higher priority than object points, and avoid the “short cuts” associated with 18 and 26-connectivity. However, then we also turn correctly interpolated oblique faces into staircases (Figure 1a), and the result would be equal to 6-connectivity only.

We may notice that the Kenmochi method in general gives priority to the object over the background, because it uses all possible tetrahedra between object points for its boundary definition. For instance in cell-configuration P6b, the middle tetrahedron (Figure 4b) can be arbitrarily assigned to the object or to the background. If the central tetrahedron is removed we get another triangulation (compare Figure 4c with Figure 4a). This alternative triangulation may be useful in certain situations, as illustrated in Figures 14a and 15a.

The main problem is that we do not know *a priori* the shape of the real object before discretization, but it is clear that we may arbitrarily choose the one or the other configuration: “object over background” or “background over object” priority. To further exemplify these issues we will take a closer look at the DMC method.

3 DISCRETIZED MARCHING CUBES

Montani et al. [13] presented their discretized version of the Marching Cubes algorithm as a method to reduce the number of “pathological” cases produced by the standard Marching Cubes algorithm. Instead of taking the exact intersection of the isosurface with the edge between neighboring voxels, they take the midpoint. This increases the number of configurations from 14 in MC to 16 in DMC. We are in particular interested in their configurations “e,” “k,” “l,” “n,” “o” and “p” (see Figure 6).

It is not clear from [13] how these configurations were derived, but we may assume that they started by defining patterns for covering the faces of the cell (see Figure 7a, b, c and d), to guarantee continuity between cells. In addition, they selected the midpoint of the cell to be either in the object or not, and in case the midpoint is part of the object boundary, the midpoint is connected to the midpoints at the edges.

We may conclude that the authors of the DMC method gave priority to the background instead of the object, because otherwise the configurations would have been based on the face-configuration “7e” instead of “7c” (Figure 7). Of course, it is arbitrary whether we choose the one over the other and there is also no need to be consistent. The choice of whether the midpoint (in the DMC configurations) is part of the object is also arbitrary. For instance, the “DMC-e” configuration has five white corner points against three black, so it would be more natural to have the center point white,

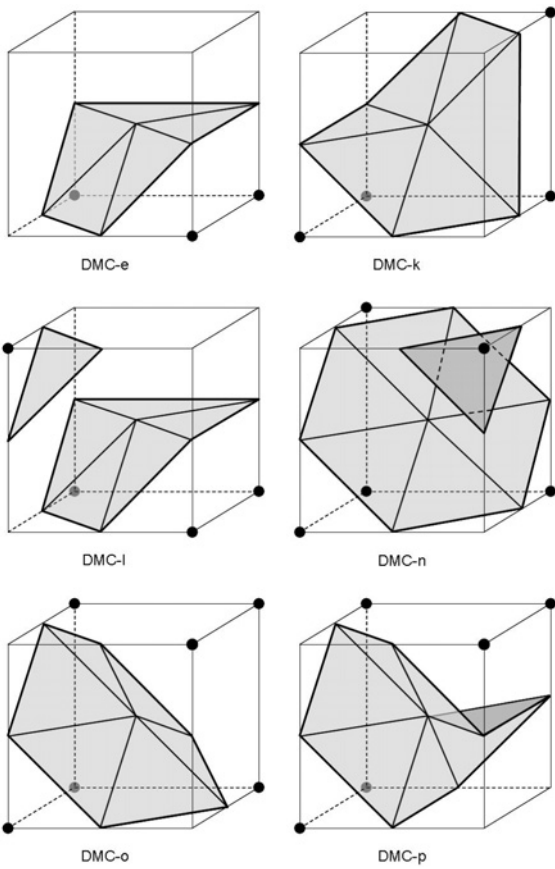


Figure 6: Six of the 16 DMC configurations (cf. Figure 3 in Montani94)

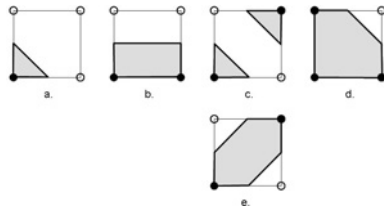


Figure 7: DMC configurations can be derived from the above 2D patterns. It is arbitrary whether configuration c (background priority) is chosen or e (object priority)

but a black center point generates a nice triangulation comparable to configuration “DMC-I”.

An alternative method to construct the DMC configurations is to build them by using the Kenmochi algorithm. We do this by mapping the $2 \times 2 \times 2$ DMC voxel configuration to a $3 \times 3 \times 3$ voxel block. In the $3 \times 3 \times 3$ block the DMC midpoints are now black and the other midpoints have the color of their neighboring corner points. We apply the configurations of Kenmochi (Figure 5) to the eight individual $2 \times 2 \times 2$ cells, and the results are shown in Figure 8. We will notice some differences with Figure 6.

In fact, as mentioned earlier, in the Kenmochi scheme the object has priority over the background, because it uses all possible tetrahedra between black points for its boundary definition. If we reverse the priority in the

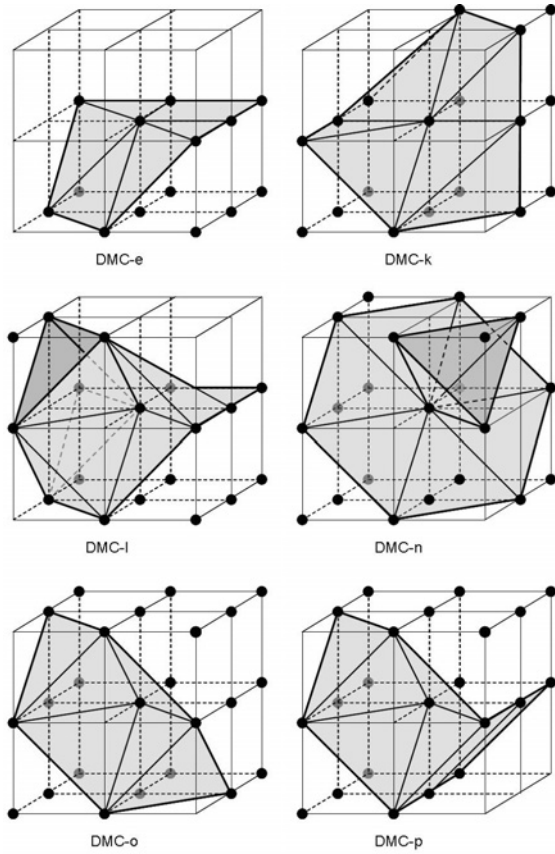


Figure 8: The six DMC configurations derived with the Kenmochi method

DMC method, then the Kenmochi method will generate a better approximation. However, there will still remain small local differences due to the DMC construction method, which connects the central point directly to the edge points.

In conclusion, we saw that DMC is a simplified version of a $3 \times 3 \times 3$ surface construction with the restriction that the cell edges are only black-black, white-white, black-white or white-black voxel combinations (which is due to the $2 \times 2 \times 2$ starting situation) and not white-black-white or black-white-black (which is only possible in the $3 \times 3 \times 3$ configuration).

4 TRIANGULATION WITH BACKGROUND PRIORITY

Looking at the above examples we may wonder whether there exists a triangulation that directly generates the standard DMC configurations. This is indeed so (see Figure 9). Instead of using object priority (cf. Figure 7e) we follow the DMC convention (cf. Figure 7c) and give priority to the white diagonals. This means that object voxels are not 26-connected, but only 18-connected when they are 6-adjacent. Some of the Kenmochi configurations remain (P3a, P4a, P4e, P6a and P7a), whereas other configurations reduce to triangles or to combinations with dangling voxels

(P4b and P5c). P4g has no triangle anymore, only isolated points. The isolated points will have to be removed with a postprocessing similar to the removal of non-manifold situations in the Kenmochi method.

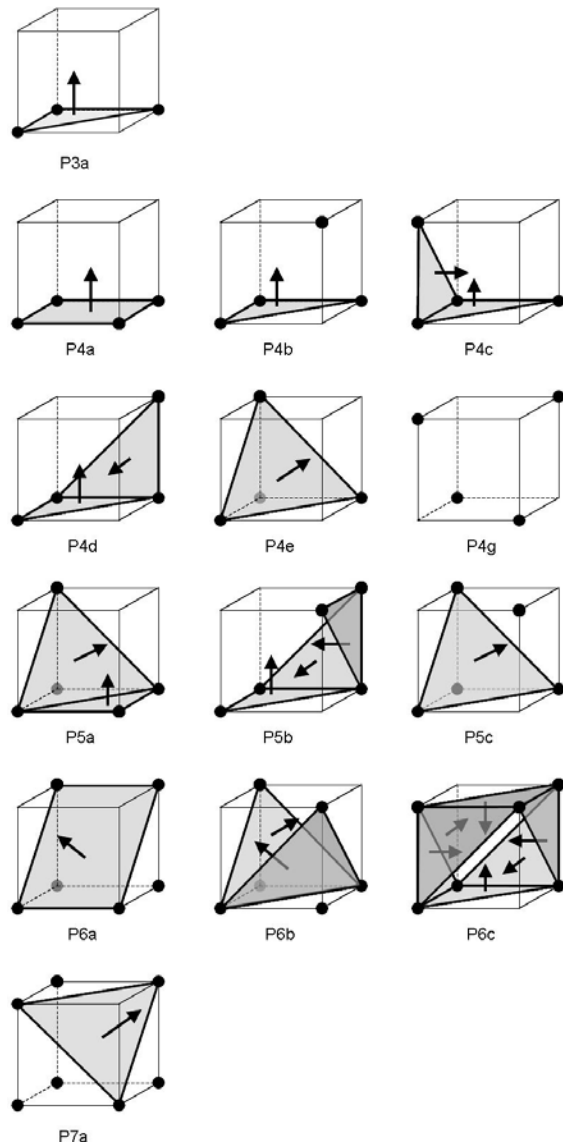


Figure 9: Background Priority scheme

Now we can build the standard DMC configurations, see Figures 10 and 6, although there still remain some differences. This is because the triangulation using DMC configurations is done starting from the central point (if there is one). We can copy this by generating an edge-list from the eight cells and then connect the edge corner points with the center point. This reduces the number of triangles and makes the configuration smoother; compare “DMC-p” in Figures 6 and 10. In addition, if we introduce “don’t care” voxels then we can reduce the number of triangulation patterns to seven for the Kenmochi method and six for the Background Priority method (see Appendix).

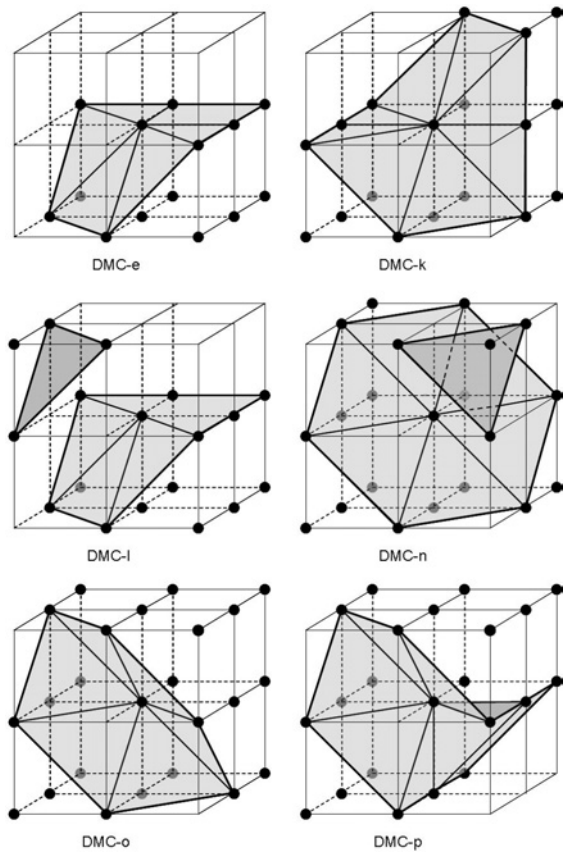


Figure 10: The six DMC configurations with Background Priority algorithm

Figures 14 and 15 show results obtained with two test objects. Note that the sharp edges and the sharp corner of the cut-out part are correctly triangulated by the use of Kenmochi configurations P5c and P6b. However, in the cube where the cut-out part is taken from this is not the case. There, configuration P6b leads to aliasing at the sharp edges and configuration P7a to a cut-off of the sharp corner inside the cube. In the case of triangulations obtained with Background Priority the opposite effects are obtained, except for the sharp corner inside the cube, see below. The upper part of the Hoppe object obtained with the Background Priority method is better, whereas the bottom part is worse.

5 FUTURE RESEARCH

Obviously, the Kenmochi method should be used in order to triangulate convex boundaries and the Background Priority to triangulate concave boundaries. In practice, we can check whether a voxel is at a convex or concave boundary by inspecting its $3 \times 3 \times 3$ voxel neighborhood. Under normal conditions, when the number of voxels which belongs to the object in the neighborhood is larger than 18 the voxel in the center of the neighborhood is at a concave boundary. When the number is smaller than 18 the voxel in the center of the neighborhood is at a concave boundary. Figure 16

shows the results when we apply this triangulation strategy. As can be seen both convex and concave edges are better preserved. Since these results are promising, in the future we want to substitute the concave-convex heuristic by a robust algorithm which determines convexity or concavity in the local neighborhood and to topologically validate the combined triangulation approach.

Figure 11 shows an alternative triangulation of pattern P7a which for oblique surfaces is not desired, but which in some cases, such as the sharp corner inside the cube in Figure 15a, better preserves the object detail. In order to decide which of the two triangulations must be applied it seems necessary to determine which object voxels belong to the boundary and which not. Also for some other configurations (e.g. P6a) other (extreme background priority) triangulations may be asked for in certain situations. These are important issues for future research.

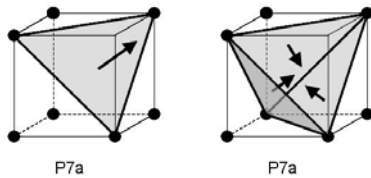


Figure 11: Two possible triangulation patterns for Background Priority configuration P7a

ACKNOWLEDGEMENTS

The first author acknowledges the financial support provided through the European Community's Human Potential Programme under the contract HPRN-CT-2002-00278, Combinatorial Structure of Intractable Problems (COMBSTRU). This research is partly funded by the Portuguese FCT program POSI, framework QCA III.

APPENDIX

The triangulation patterns for the voxel configurations of Figures 5 and 9 can be built from a reduced set of basic triangulation patterns that use "don't care" voxels for positions that can either be black or white. Figure 12 shows the basic patterns for the Kenmochi method and Figure 13 for the Background Priority method. We note that in each configuration of Figure 12 with "don't cares" (K4-K7), at least one of the "don't cares" must be black. For clarity, these patterns can be specified in (((bottom-behind-left)-right)-fore)-top order as:

```

K1:  1 1 1 0 0 0 0 0
K2:  1 1 1 1 0 0 0 0
K3:  1 1 1 0 1 0 0 0
K4:  x x 1 1 1 0 0 0
K5:  x x 1 1 0 1 0 0
K6:  x x 1 1 1 1 0 0
K7:  x x x 1 x 1 1 0

B1:  1 1 1 0 0 x x x

```

```

B2:  1 1 1 1 0 0 0 1
B3:  1 1 1 1 0 0 0 0
B4:  1 1 1 x 1 0 0 x
B5:  1 1 1 1 1 1 0 0
B6:  1 1 1 1 1 1 1 0,

```

where 0 denotes a white point, 1 a black one and x a "don't care" case. Table 1 specifies the conversion.

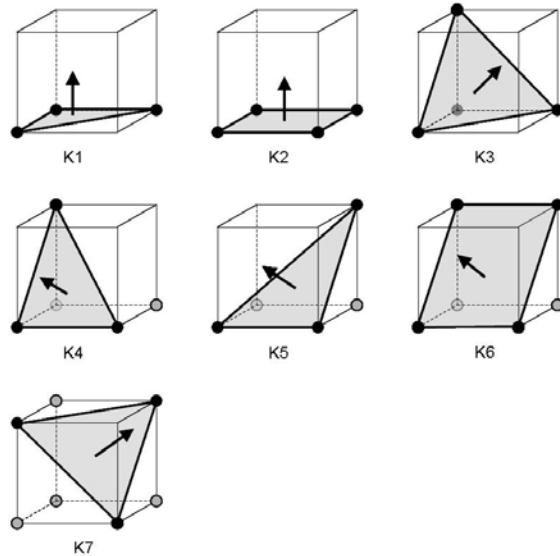


Figure 12: Kenmochi method with "don't care" voxels denoted in gray

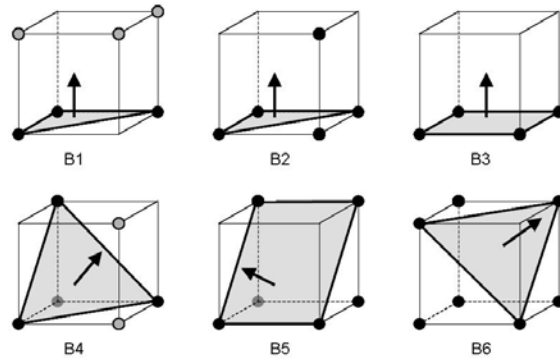
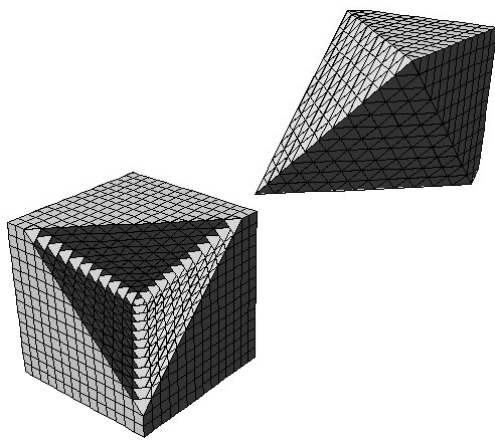


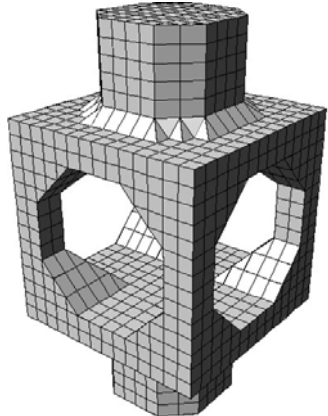
Figure 13: Background Priority method with "don't care" voxels denoted in gray

REFERENCES

- [1] C. Arcelli and G. Sanniti di Baja. Skeletons of planar patterns. In T. Y. Kong and A. Rosenfeld, editors, *Topological Algorithms for Digital Image Processing*, pages 99–143. Elsevier, 1996.
- [2] D. Cohen-Or, A. E. Kaufman, and T. Y. Kong. On the soundness of surface voxelizations. In T. Y. Kong and A. Rosenfeld, editors, *Topological Algorithms for Digital Image Processing*, pages 181–204. Elsevier, 1996.
- [3] M. Couprie and G. Bertrand. Simplicity surfaces: a new definition of surfaces in Z^3 . In *Proc. SPIE Vision Geometry VII*, volume 3454, pages 40–51, 1998.
- [4] R. Hall, T. Y. Kong, and A. Rosenfeld. Shrinking binary images. In T. Y. Kong and A. Rosenfeld, editors, *Topological Algorithms for Digital Image Processing*, pages 31–98. Elsevier, 1996.

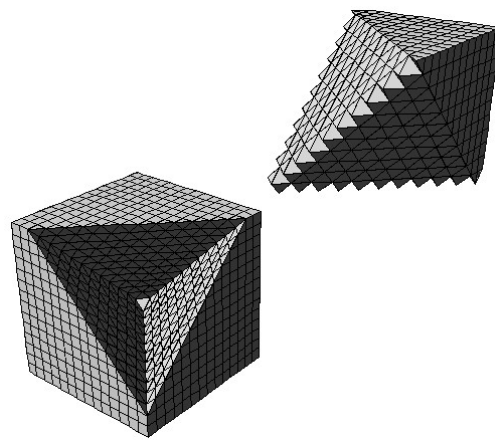


a

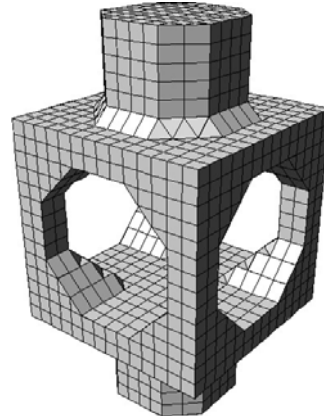


b

Figure 14: Triangulation with the Kenmochi method of a cube with cut-out part (a) and a Hoppe object (b)



a



b

Figure 15: Triangulation with the Background Priority method

- [5] A. Kaufman. *Volume Visualization*. IEEE Computer Society Press Tutorial, Los Alamitos (CA), USA, 1991.
- [6] Y. Kenmochi, A. Imiya, and A. Ichikawa. Boundary extraction of discrete objects. *Computer Vision and Image Understanding*, 71(3):281–293, 1998.
- [7] A. Kodosh, D. Cohen-Or, and R. Yagel. Tricubic interpolation of discrete surfaces for binary volumes. *IEEE Trans. Visual Comput. Graphics*, 9(4):580–586, 2003.
- [8] T. Y. Kong and A. Rosenfeld. Digital topology: introduction and survey. *Comp. Vision, Graphics and Image Proc.*, 48:357–393, 1989.
- [9] T. Y. Kong and A. Rosenfeld. *Topological Algorithms for Digital Image Processing*. Elsevier, 1996.
- [10] V. A. Kovalevsky. Finite topology as applied to image analysis. *Comp. Vision, Graphics and Image Proc.*, 46:141–161, 1989.
- [11] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.
- [12] R. Malgouyres and G. Bertrand. Complete local characterization of strong 26-surfaces: continuous analog for strong 26-surfaces. *Int. Journal on Pattern Recognition and Artificial Intelligence*, 13(4):465–484, 1999.
- [13] C. Montani, R. Scateni, and R. Scopigno. Discretized marching cubes. In R. D. Bergeron and A. E. Kaufman, editors, *Proc. Visualization '94*, pages 281–287, Washington D.C., USA, 1994.
- [14] D. G. Morgenthaler and A. Rosenfeld. Surfaces in three-dimensional digital images. *Information and Control*, 51:227–247, 1981.
- [15] K. Palágyi and A. Kuba. A 3D 6-subiteration thinning algorithm for extracting medial lines. *Pattern Recognition Letters*, 19:613–627, 1998.
- [16] J. K. Udupa. Multidimensional digital boundaries. *CVGIP: Graphical Models and Image Processing*, 56(4):311–323, 1994.
- [17] J. K. Udupa. Connected, oriented, closed boundaries in digital spaces: Theory and algorithms. In T. Y. Kong and A. Rosenfeld, editors, *Topological Algorithms for Digital Image Processing*, pages 205–231. Elsevier, 1996.

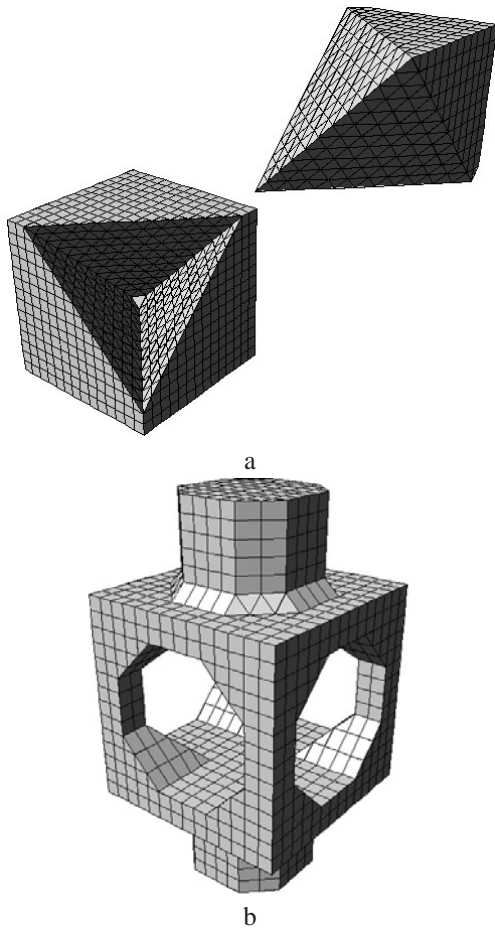


Figure 16: Triangulation with the Kenmochi method at convex boundaries and the Background Priority method at concave boundaries

Table 1: Triangle lookup table for the Kenmochi algorithm with “don’t cares” and for the Background Priority algorithm

Kenmochi	with don’t cares	background priority
P3a	K1	B1
P4a	K2	B3
P4b	K4+K5+K7	B1 + one point
P4c	K5+K5	2*B1
P4d	K4+K4	2*B1
P4e	K3	B4
P4g	4*K7	four points
P5a	K4+K5	B2+B4
P5b	K6+K7	3*B1
P5c	3*K7	B4 + one point
P6a	K6	B5
P6b	2*K7	2*B4
P6c	2*K7	6*B1
P7a	K7	B6

Dynamic Progressive Triangle-Quadrilateral Meshes

Stefan Wundrak
Fraunhofer IGD
Fraunhoferstr. 5
64283 Darmstadt, Germany
stefan.wundrak@igd.
fraunhofer.de

Thomas Henn
Technical University Darmstadt
Fraunhoferstr. 5
64283 Darmstadt, Germany
thomas.henn@gris.
informatik.tu-darmstadt.de

André Stork
Fraunhofer IGD
Fraunhoferstr. 5
64283 Darmstadt, Germany
andre.stork@igd.
fraunhofer.de

ABSTRACT

We present an extension of the original progressive mesh algorithm for large dynamic meshes that contain a mix of triangle and quadrilateral elements. The demand for this extension comes from the visualisation of dynamic finite element simulations, such as car crashes or metal sheet punch operations. These methods use meshes, which consist mainly of quadrilaterals, due to their increased numerical stability during the simulation. Furthermore, these meshes have a dynamic geometry with about 25 to 100 animation steps. Accordingly, we extend the original progressive mesh algorithm in two aspects: First, the edge collapse operation is extended for meshes with a mixture of triangle and quadrilateral elements. Second, we present an algorithm on how to extend quadric error metrics for the simplification of large dynamic meshes with many animation steps. The results are dynamic progressive triangle-quadrilateral meshes – a progressive multi-resolution mesh structure that has two interactive degrees of freedom: simulation time and mesh resolution. We show that our method works on meshes with up to one million vertices and 25 animation steps. We measure the approximation error and compare the results to other algorithms.

Keywords

Progressive Meshes, Level of Detail, Crash Simulation, Animation, Quadrilaterals, Dynamic Meshes.

1. INTRODUCTION

For dynamic finite element simulations in structural mechanics, such as car crashes or sheet metal forming, analysts prefer quadrilateral meshes over triangular meshes (Figure 1), since three-noded constant strain triangles behave poorly in bending [MG97]. The results of these simulations are meshes with a constant mesh topology, which contain between 1 and 5 million vertices, and a dynamic geometry with about 25 to 100 animation steps. In addition, during the optimization process hundreds of variants are simulated and stored, leading to huge amounts of data. Tools to visualise the three-dimensional simulation results help the engineer to interpret the crash behaviour and to optimise the car

body. These tools have to efficiently deal with large time dependent data sets [Som03]. Even though for finite element simulations the model has to be highly tessellated over the complete mesh, during post-processing analysis, data base browsing, or interpolation of simulation results, it is often sufficient to display the animated mesh at a reduced granularity first and refine the animation on demand only. Furthermore, only the mesh sections with a high deformation need to be displayed at full resolution. This approach reduces the amount of data that needs to be transmitted and displayed. The base for these methods will be a progressive data structure that supports dynamic triangle-quadrilateral meshes.

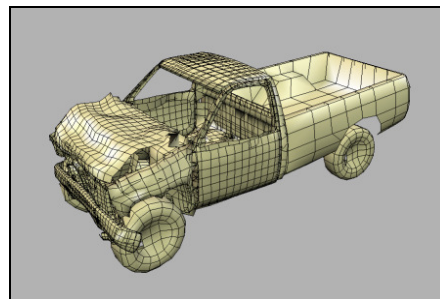


Figure 1. Example of a small triangle-quadrilateral mesh used for crash simulations.

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.

*Short Communications proceedings, ISBN 80-86943-05-4
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press*

Goals and Contribution

We will show how Dynamic Progressive Triangle-Quadrilateral Meshes can easily be generated out of common dynamic meshes as a pre-processing step to visualisation.

Accordingly, we extend the original progressive mesh algorithm of Hoppe [Hop96] in two aspects: First, the edge collapse and vertex split operations are extended for meshes with a mixture of triangle and quadrilateral elements. Therefore, a new constraint is added to avoid the creation of degenerated meshes. Second, we present an algorithm that extends quadric error metrics [GH97] for the simplification of large dynamic meshes with many animation steps. The result is a progressive multi-resolution mesh structure that has two interactive degrees of freedom: simulation time and mesh resolution. We show that our method works on meshes with up to one million vertices and 25 animation steps, and compare the approximation error to other algorithms.

Related Work

Progressive Meshes, first introduced by Hoppe [Hop96], are a progressive data structure for triangular meshes based upon the *edge collapse* operation. By means of this operation it is possible to reduce the complexity of a given triangle mesh by iteratively removing edges and thus deleting the adjacent triangles resulting in the base mesh M^0 :

$$M^n \rightarrow (\text{ecol}_{n-1}) \rightarrow \dots \rightarrow (\text{ecol}_1) \rightarrow M^l \rightarrow (\text{ecol}_0) \rightarrow M^0$$

The inverse *vertex split* operation allows undoing these changes and restoring the removed mesh items. This allows for storage and transmission of the original mesh M^n as multi-resolution representation consisting of M^0 and a sequence of n vertex split operations:

$$M^0 \rightarrow (\text{vsp}_0) \rightarrow M^l \rightarrow (\text{vsp}_1) \rightarrow \dots \rightarrow (\text{vsp}_{n-1}) \rightarrow M^n$$

Ramsey et al. [RBH03] describe an extension to the edge collapse operation for meshes composed of non-planar multi-sided polygons. However, extending the necessary preconditions for legal collapse operation was not covered.

Gumhold [Gum04] introduced the *remove edge* and *join edges* operations to simplify arbitrary polygonal meshes (Face Clustering). The remove edge operation joins two faces by removing its shared edge and thus creates polygons of higher complexity.

To create the sequence of operations from the original mesh one has to define an error metric. Selecting the error measurement is a trade-off between the quality of the simplified mesh and the performance of the simplification process. For a discussion of the different methods see [PS97].

Garland and Heckbert [GH97] introduced an error metric based on *quadrics* that accumulates the error during simplification. It approximates the maximum error of the geometric distance and the deviation of surface normals. A quadric is assigned to each vertex of the mesh, which represents a weighted combination of all faces adjacent to the vertex. To estimate the approximation error of an edge collapse operation the quadrics of the two adjacent vertices are added and evaluated. This leads to a fast algorithm with low memory needs and a good approximation quality. In [GH98] a generalization of the quadric error metric is presented that also considers surface properties, such as colours, texture coordinates, or surface normals. We will extend this approach in this work to support dynamic meshes with many animation steps.

In [KEH97] Kuschfeldt has described how to convert quadrilateral meshes into triangles and how to display the original element boundaries using texture mapping. In combination with progressive meshes this method has several disadvantages. First, inherent information about the element boundaries is lost and has to be stored additionally, which increases the amount of data and complicates the method. Second, storing twice as many triangles increases the connectivity data in many representations. Thus, we decided to include quadrilaterals as basic elements into the progressive mesh algorithm which leads to a simple and elegant solution.

2. PROGRESSIVE TRIANGLE-QUADRILATERAL MESHES

Our aim is to extend progressive meshes to support a mix of triangle and quadrilateral elements. We start with some definitions related to meshes that will be used throughout the paper.

A polygonal mesh M can be denoted by a tuple (V, F) , where $V = \{v_0, \dots, v_n\}$ is a set of vertex positions defining the shape of the mesh in R^3 , and $F = \{f_0, \dots, f_i\}$ is a set of closed faces. The set of edges is denoted by $E = \{e_0, \dots, e_k\}$. An animation with a constant topology is defined by m sets of vertex positions $A = \{V_0, \dots, V_{m-1}\}$.

An edge is a *boundary edge* if it is part of only one face. A vertex is a *boundary vertex* if it is part of a boundary edge. A vertex is an *inner vertex* if it is not a boundary vertex.

For the representation of a legal surface, polygonal meshes have to fulfil the following conditions (cp. [Gum04] [FDF+90] [BSBK02]):

- (M1) *The mesh is manifold with boundary.*
- (M2) *The minimum valence of an inner vertex is three.*
- (M3) *The minimum degree of a face is three.*

Our work is based on the OpenMesh library [BSBK02] that internally works with a half-edge data structure. Non-manifold meshes are initially converted to manifold meshes.

Extending the Collapse and Split Operation for Mixed Meshes

Figure 2 shows the half-edge collapse operation used for triangle meshes collapsing the edge $\{v_0; v_l\}$ into the vertex v_l . The adjacent faces f_l and f_r degenerate and vanish in this process. Also the vertex v_0 is removed from the mesh.

An extension to the half-edge collapse is the edge collapse operation, which allows optimal placement of the remaining vertex v_l , but will not be used in this work. Although the usage of the edge collapse operation usually increases the quality of the decimated mesh it also increases the amount of data that has to be stored in a progressive mesh as we will describe later.

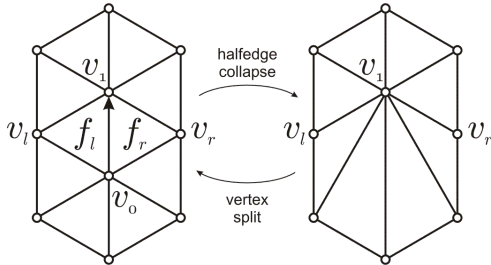


Figure 2. Half-edge collapse and vertex split operation in triangle mesh.

Figure 3 shows the new extended half-edge collapse operation for quadrilateral meshes. In this case, collapsing the edge $\{v_0; v_l\}$ into the vertex v_l does not lead to degenerated faces. Instead, the quadrilaterals are transformed into triangles. Only the vertex v_0 is removed.

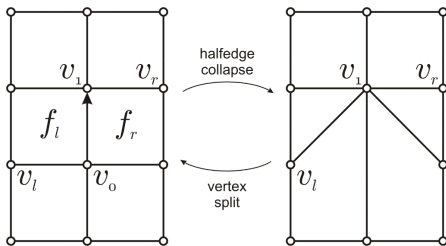


Figure 3. Half-edge collapse in quad mesh.

How these half-edge collapse and vertex split operations are stored in a data structure is described in section 4.

Legal Collapses for Triangle Edges

Depending on the topology of the mesh an edge collapse operation may produce a degenerated mesh.

To avoid this one has to validate legal operations using a topology test. Hoppe et al. introduced three preconditions in order to collapse an edge $\{v_0; v_l\} \in E$ [HDD+93].

Definition: Two vertices v_i and v_j are neighbours, if an edge $\{v_0; v_l\} \in E$ exists.

- (P1) For each vertex $v_i \in V$, that is a neighbour of v_0 and v_l , v_i shares a face with v_0 and v_l .
- (P2) If v_0 and v_l are both boundary vertices, $\{v_0; v_l\}$ is a boundary edge.
- (P3) M has more than 4 vertices if neither v_0 nor v_l are boundary vertices, or M has more than 3 vertices if either v_0 or v_l are boundary vertices.

While (P2) avoids the separation of M into two meshes that are only connected by a single vertex, (P3) terminates the simplification. Precondition (P1) avoids the creation of vertices with a valence of two, which are only allowed at the boundary of the mesh. Figure 4 shows a half-edge collapse operation that violates the first precondition.

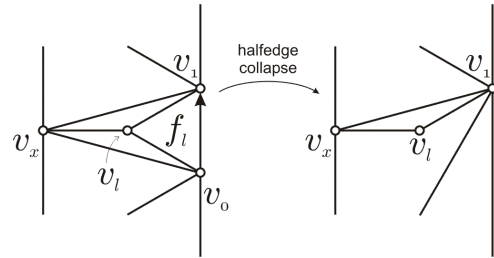


Figure 4. Illegal half-edge collapse operation that creates a vertex with a valence of two. This case is avoided, because (P1) is violated by v_x .

Legal Collapses for Quadrilateral Edges

For mixed meshes containing triangles and quadrilaterals the precondition (P1) is not sufficient and has to be extended. It is necessary to differ between edges that are adjacent to a triangle or to quadrilaterals only. We define that a *triangle edge* is adjacent to at least one triangle, whereas a *quadrilateral edge* is adjacent to quadrilaterals only.

The collapse of a quadrilateral edge needs no further tests. As seen in Figure 3, the valence of each involved vertex is unchanged or increased. The valence of the adjacent faces is decreased by one, since the quadrilaterals are turned into triangles. Only precondition (P2) is necessary in this case.

On the other hand, the collapse of a triangle edge within a quadrilateral mesh needs further attention. Similar to an edge collapse in a triangle mesh a vertex with a valence of two might be created.

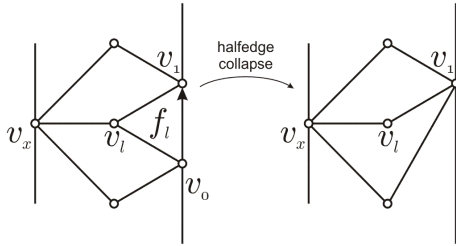


Figure 5. Creation of a vertex with a valence of two in a quadrilateral mesh. This case is not avoided, because (P1) is not violated by v_x .

In Figure 5 the shown half-edge collapse operation leads to a vertex with a valence of two, even though precondition (P1) is not violated. Thus, precondition (P1) has to be extended in order to avoid this case.

Definition: Two vertices v_i and v_j are neighbours of second order, if a face $f \in F$ exist for that $v_i \in f$ and $v_j \in f$ holds true.

Corollary: Two vertices v_i and v_j that are neighbours are also neighbours of second order.

Precondition (P1) can be reformulated as follows:

(P1*) For each vertex $v_i \in V$, that is a neighbour of second order of v_0 and v_1 , v_i shares one face with v_0 and v_1 .

Precondition (P3) for termination is not changed, since successive collapses of quadrilaterals always end up in triangles. The more costly precondition (P1*) is only needed for quadrilateral meshes. A proof for the extended preconditions is given in the Section 7.

Extending the Quadric Error Metric for Mixed Meshes

As described in [GH97] fundamental quadrics represent the triangles of the original mesh. Each fundamental quadric is defined by a plane, which in turn is defined by a triangle of the mesh. Each quadric is based on a set of these fundamental quadrics. However, in a quadrilateral mesh the fundamental quadrics are only well-defined for planar quadrilaterals. For skew quadrilaterals with non-coplanar vertices a plane has to be defined that represents the quadrilateral best. We propose to compute the plane as follows:

To define the plane a normal vector n and a distance d are needed. At first, a normal vector is computed for each vertex of the quadrilateral (cp. [RBH03]). Then n is set to the mean of these normals, and d is defined as the centre of gravity of the quadrilateral multiplied by n . This leads to a fundamental plane that has the same distance to all vertices of the quadrilateral.

3. LARGE DYNAMIC PROGRESSIVE MESHES WITH MULTI QUADRICS

In [GH98] a generalization of the quadric error metric was presented that considers surface properties, such as colours, texture coordinates or surface normals. This approach can be extended to support animated meshes with fixed topology, since the characteristic property of a dynamic model is the changed geometry during each animation step.

There are two principle approaches for the extension of quadrics to consider multiple properties (e.g. A and B) during simplification [GH98]:

- (1) The generation of multi quadrics Q_A and Q_B for each vertex and the definition of the error as $Q_A(v_A) + Q_B(v_B)$.
- (2) The generation of one higher dimensional quadric Q_{AB} for each vertex and the definition of the error as $Q_{AB}(v_{AB})$.

While in method (1) the memory and computation costs rise linearly with the number of attributes, method (2) shows a quadratic behaviour.

In the case of dynamic progressive meshes the number of attributes corresponds to the number of animation steps. A mesh with 25 animation steps would thus lead to a factor of $25^2 = 625$ in computation time and memory consumption compared to the static mesh. This disqualifies method (2) for our work. Using multi quadrics instead of higher dimensional quadrics means however, that optimal vertex placement is not easily possible.

In contrast to the simplification of polygonal meshes with colour or texture attributes a dynamic mesh with m animation steps has additional geometry data in form of m positions for each vertex. The summation of the error values in method (1) however doesn't fit the needs of dynamic meshes very well. Thus, we choose a modified method (1*) that better preserves the geometric variation over time:

- (1*) The generation of multi quadrics Q_A and Q_B for each vertex and the definition of the error as $\max(Q_A(v_A), Q_B(v_B))$.

The m animation steps can be interpreted as m distinct meshes $\{(V_0, F), \dots, (V_{m-1}, F)\}$. For each vertex v_i , m quadrics $Q_{t,i}$ will be generated and initialised using the geometry of the associated animation step (V_t, F) . During simplification the meshes of the animation steps can be processed in a parallel manner. Thus, an edge collapse will be rated considering the $2 \cdot m$ associated quadrics.

The collapse of an edge (v_i, v_j) into vertex v_k needs m quadric additions $Q_{t,k} = Q_{t,i} + Q_{t,j}$, generating m

quadrics, one per animation step. The overall approximation error for this operation is defined by the maximum of these m quadrics:

$$\max_{t=0..m} (Q_{t,k}(v_k))$$

For large models with many animation steps it is possible to further optimise the error metric. In the case of crash results, one can notice that the geometric difference between two successive animation steps is relatively small. Therefore, it is possible to take into account only selected animation steps. For crash simulations we achieved good results by selecting the first, the last, and one animation step in between, preferably the one with the maximum vertex displacements. A generalization to arbitrary animations would be to automatically detect the animation steps with the strongest deformation. Restriction to a small number of animation steps allows for reduced computation effort. Measurements are discussed in section 5.

4. DATA STRUCTURE

As described in [Hop96] progressive meshes are stored in two parts. First the small base mesh, second a stream of n vertex split operations, which are the inverse of the n edge collapse operations generated during the simplification process. This structure may then be used for Level-of-Detail or progressive streaming of the mesh.

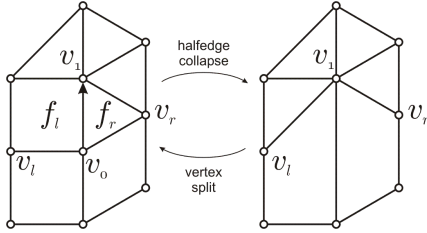


Figure 6. The vertex split operation as inverse operation of edge collapse allows for the refinement of the base mesh.

As shown in Figure 2 for a pure triangle mesh it is sufficient to know the references of the vertices v_l ; v_r ; v_0 as well as the position of the vertex v_0 . If optimal vertex placement was used during simplification the new position of v_l has also to be stored. If the vertex has additional properties, such as colour or displacement vectors, one additionally has to store the properties of the vertex v_0 . The references to v_l and v_r indicate the triangles f_l and f_r .

To define a vertex split operation within a mixed mesh one also has to save whether the half-edge collapse operation did remove any triangles. For

example Figure 6 shows the half-edge collapse removing triangle f_r . Thus, the inverse operation of the half-edge collapse operation $hec(v_0; v_l)$ is well-defined by the vertex split operation $vsp(p_0; v_l; v_r; t_l; t_r)$ with the following parameters:

$v_l; v_r; v_0$	vertex reference
p_0	position of removed vertex v_0
$t_l; t_r$	bool, true if f_l / f_r triangle

5. RESULTS

Evaluation

To evaluate the approximation error of a simplified mesh we compare it to the original mesh. We used Metro [CRS96] to measure the two-sided Hausdorff distance. Tools such as Metro do not offer the comparison of dynamic simplified meshes. As a workaround we compared the single animation steps manually and calculated an overall approximation error as average of the all animation steps.

In addition, Metro is only able to compare triangle meshes, thus, before measuring, the mixed meshes had to be triangulated. To avoid approximation errors due to non well-defined triangulation we used the star-shaped triangulation as described in [Gum04] that uses the centre of gravity as a new vertex position.

In the following we express the two-sided Hausdorff distance in percentage of the diagonal of the meshes bounding box. Each simplification was computed using a PC with a Pentium 4 processor with 2.0 GHz and 1GB of RAM running Windows 2000.

Results Static Mixed Meshes

First, we test our algorithm against QSlim, an implementation of the quadric error metric using edge collapse simplification with optimal vertex placement as described in [GH97].

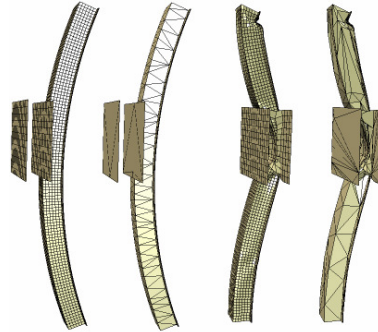


Figure 7. The pillar mesh at animation step t0 and t15 and in resolutions of 100% and 5% of original vertices.

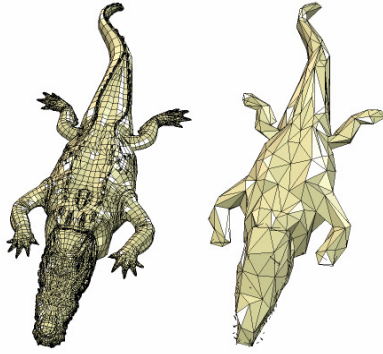


Figure 8. The croco mesh in resolutions of 100% and 5% of original vertices.

We use two different animation steps (t0, t15) of a technical model (Figure 7) and a non-technical model (Figure 8) to evaluate the quality of the simplification of our algorithm compared to QSlim. The results are shown in Table 1.

Mesh	Tris	Quads	5%	5% QSlim	Time [sec]
pillar.t0	0	6190	0,08	0,03	0,7
pillar.t15	0	6190	0,68	0,68	0,7
croco	9358	12523	1,78	1,75	2,1

Table 1. Results static meshes.

The results are generally comparable to QSlim, even though no optimal vertex placement algorithm was used. One thing to notice is that for the non-deformed mesh (pillar.t0) the approximation error is significantly lower than to the deformed mesh (pillar.t15). Meshes prepared for finite element analysis are generally tessellated in a high resolution, even for plane faces. For an undeformed mesh these triangles can be eliminated easily without increasing the approximation error. For deformed meshes however, these triangles are needed to represent the deformed geometry with the needed accuracy. This explains the difference between the two pillar meshes.

Results Dynamic Mixed Meshes

In a second experiment we measure the quality of our algorithm for dynamic meshes. We used the front part of the large neon mesh (Figure 9) as a subset, since this part is deformed most. The Metro tool does not support dynamic meshes, thus we compared the single animation steps for the neon.front mesh as if they were static meshes and calculated an overall approximation error afterwards. The results are shown in Table 2. Above 4% of the original vertices, the maximum overall approximation error falls below 1%.

Percent of vertices	t0 [%]	t12 [%]	t24 [%]	Max. t0...t24
1%	2,06	1,63	1,52	2,06
2%	1,21	1,35	1,30	1,35
4%	0,89	0,78	0,80	0,89
11%	0,26	0,27	0,27	0,27

Table 2. Approximation error for dynamic mesh neon.front, at various resolutions and animation steps.

Figure 9 shows the resulting meshes subject to the two degrees of freedom in mesh resolution and simulation time.

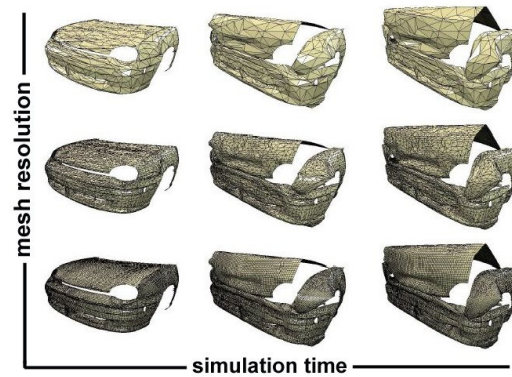


Figure 9. This picture shows the two degrees of freedom, with resolution 4%, 18%, and 100% and animation steps t0, t6, and t12.

One thing to notice is that for dynamic meshes the use of quadric error metrics leads to a higher mesh resolution for the areas with a high deformation.

Our last experiment analyses if it is sufficient for the error metric to consider only a subset of animation steps for meshes that are typically produced by crash simulations, in order to save computation time during the simplification process.

Considered Animation Steps	1%	4%	6%	Time [sec]
All	1,60	0,73	0,38	36:52
t0,t6,t12	1,67	0,68	0,34	3:33
t0	3,92	1,97	1,24	2:16
t12	1,80	0,93	0,57	2:16

Table 3. Mean approximation errors for dynamic mesh neon.front.

Table 3 shows a summary of the results. Including only a subset of three animation steps leads to nearly

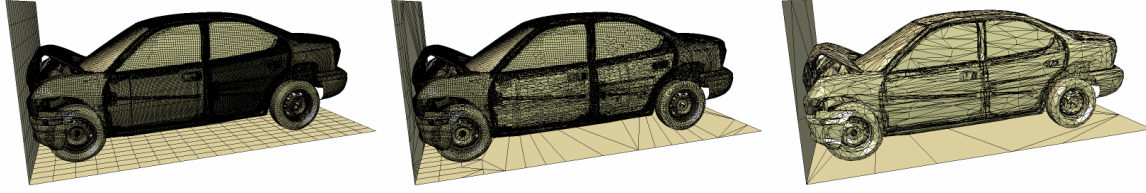


Figure 10. The complete neon mesh at resolutions 100%, 50%, and 8% of the original vertices.

no increase of the approximation error but significantly reduces the computation time for the simplification. Using only one animation step is only acceptable if the animation step with the maximum deformation (t12) is chosen.

Finally, Figure 10 shows the complete finite element mesh of a neon car body in various resolutions. The mesh contains more than 1 million vertices and 25 animation steps. The second mesh shows the mix of triangular and quadrilateral elements.

6. SUMMARY AND CONCLUSION

We have extended the original progressive mesh algorithm in two aspects: First, to support meshes with a mixture of triangle and quadrilateral elements. Second, we defined an error metric for the simplification of large dynamic meshes with many animation steps.

Together we have created a multi-resolution mesh structure that has two interactive degrees of freedom: simulation time and mesh resolution. In future work extensions towards selective refinement and optimal vertex placement shall be researched.

7. PROOF OF PRECONDITIONS

Let M be a *valid* mixed mesh which means the rules (M1) to (M3) are true. Let col be an edge collapse that fulfils the preconditions (P1*), (P2) and (P3) on M .

We want to proof that $col(M)$ also is a valid mesh.

We use a proof by contradiction and presume the opposite: “ $col(M)$ is an invalid mesh”. Thus, the mesh violates one of the rules (M1) to (M3).

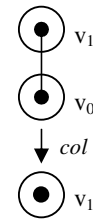
We will show that this presumption leads to a contradiction.

There are three reasons why the produced mesh $col(M)$ may be invalid, namely the violation of (M1), (M2), or (M3).

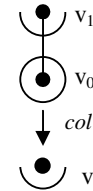
Case 1: $col(M)$ violates (M1)

Since M is manifold with boundary there are three cases:

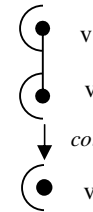
1) The vertices v_0 and v_1 are inner vertices. Since M is manifold the topology around v_0 and v_1 are homeomorphous to discs. After edge-collapse $col(M)$, v_1 still has a disc topology, all other vertices have an unchanged topology. Thus, $col(M)$ is manifold and does not violate (M1). Contradiction!



2) The vertex v_0 is an inner vertex and v_1 is a boundary vertex. Since M is manifold the topology around v_0 is homeomorphous to a disc and the topology around v_1 to a half-disc. After col , the topology around v_1 is homeomorphous to a half-disc, all other vertices have an unchanged topology. Thus, $col(M)$ is manifold and does not violate (M1). Contradiction!

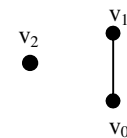


3) The vertices v_0 and v_1 are boundary vertices. Due to (P2) this means $\{v_0, v_1\}$ is a boundary edge. After col , the topology around v_1 is homeomorphous to a half-disc. All other vertices have an unchanged topology. Thus, $col(M)$ is manifold and does not violate (M1). Contradiction!



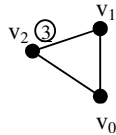
Case 2: $col(M)$ violates (M2)

Let the vertices v_0 and v_1 be the inner vertices involved in col . Let v_2 be the inner vertex that violates (M2), such that $val(v_2) < 3$ in $col(M)$.

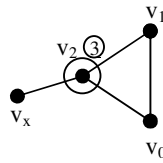


The degree of an inner vertex can only be decreased by col if it is a neighbour of v_0 and v_1 (proof left to the reader). The valence of v_2

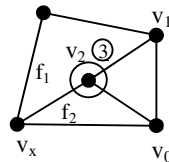
must be 3. Since $val(v_2) < 3$ would violate already (M2) and $val(v_2) > 3$ would not lead to $val(v_2) < 3$ in $col(M)$.



Since $val(v_2) = 3$, v_2 must have exactly one additional neighbour v_x . In addition, v_2 is an inner vertex and M is manifold, thus the topology around v_2 must be homeomorphous to a disc.



Since the topology around v_2 is homeomorphous to a disc and $val(v_2) = 3$, there must be f_1 and f_2 with $\{v_x, v_2, v_1\} \in f_1$ and $\{v_x, v_0, v_2\} \in f_2$ to close the circle around v_2 .



Due to f_1 and f_2 , v_x is a neighbour of 2nd order to v_0 and v_1 but does not share one common face with v_0 and v_1 . This means that col already violates (P1^{*}).

Contradiction!

Case 3: $col(M)$ violates (M3)

This case is trivial since all faces with degree < 3 are removed after an edge collapse col . Thus, $col(M)$ can never violate (M3), which also leads to a contradiction in this case.

Since all three cases lead to a contradiction, this means the presumption $col(M)$ is *invalid* must be wrong. Thus, $col(M)$ must be valid. *Q.E.D.*

8. ACKNOWLEDGMENTS

This work has been partially supported by the European Network of Excellence AIM@SHAPE, IST project 506766.

9. REFERENCES

- [BSBK02] M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt. OpenMesh. A generic and efficient polygon mesh data structure. OpenSG Symposium, 2002.
- [CRS96] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. Metro: measuring error on simplified surfaces. Technical report, Paris, France, 1996.
- [FDF+90] James Foley, Andries van Dam, Steven Feiner, and John Hughes. Computer Graphics: Principle and Practice, Second Edition. Page 473.

Addison-Wesley Publishing Company, Reading, Massachusetts, 1990.

- [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, ACM Press, New York, NY, USA, 1997.
- [GH98] Michael Garland and Paul S. Heckbert. Simplifying surfaces with colour and texture using quadric error metrics. In VIS '98: Proceedings of the conference on Visualization '98, pages 63-269, IEEE Computer Society Press, Los Alamitos, CA, USA, 1998.
- [Gum04] Stefan Gumhold. Progressive polygonal meshes. Technical Report WSI. 2004.4, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, July 2004.
- [HDD+93] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, pages 19-26, ACM Press, New York, NY, USA, 1993.
- [Hop96] Hugues Hoppe. Progressive meshes. In SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pp. 99-108, ACM Press, New York, NY, USA, 1996.
- [KEH97] S. Kuschfeldt, T. Ertl, and M. Holzner. Efficient visualization of physical and structural properties in crash-worthiness simulations. In IEEE Visualization '97, IEEE Computer Society Press, pp.487-490,583, 1997.
- [MG97] Anish Malanlara and Walter Gerstle. Comparative study of unstructured meshes made of triangles and quadrilaterals. Proc. of 6th Intl. Meshing Roundtable, pp. 437-447, 1997.
- [PS97] Paolo cignoni, Claudio Montani, Enrico Puppo, and Roberto Scopigno. Multiresolution representation and visualization of volume data. IEEE Transactions on Visualization and Computer Graphics, pp. 352-369, October-December 1997.
- [RBH03] Shaun D. Ramsey, Martin Bertram, and Charles Hansen. Simplification of arbitrary polyhedral meshes. In IASTED Computer Graphics and Imaging 2003, pp. 117-222, 2003.
- [Som03] Ove Sommer. Interaktive Visualisierung von Strukturmechaniksimulationen. PhD-Thesis, Universität Stuttgart, 2003. URN: urn:nbn:de:bsz:93-opus-15600.

Multiresolution representation and deformation of wavelet-based 3D objects

Heurtebise Xavier

Thon Sébastien

Gesquière Gilles

LSIS: Laboratoire des Sciences de l'Information et des Systèmes
IUT de Provence, Rue Raoul Follereau, Route de Crau
13200 Arles – France

{xavier.heurtebise,sebastien.thon,gilles.gesquiere}@up.univ-mrs.fr

ABSTRACT

In a virtual sculpture project, we would like to sculpt in real-time 3D objects sampled in volume elements (voxels). The drawback of this kind of representation is that a very important number of voxels is required to represent large and detailed objects. As a consequence, the memory cost will be very important and the user/object interaction will be slowed down. In order to allow real-time performance, we propose in this paper a multiresolution model that represents the object with more or less detailed levels thanks to a 3D wavelet transform. We use the marching cubes algorithm to display a triangular surface of the 3D object in various resolutions. To update quickly this surface during sculpture process, we propose a storage method for all the triangles that allows to rebuild only the modified parts of the 3D object. Moreover, to speed up processing and user/object interaction, we also propose a cache system to store in memory the most frequently used levels of details of the 3D object.

Keywords

Computer graphics, virtual sculpture, voxels, levels of details, 3D wavelets.

1. INTRODUCTION

In this paper, we tackle the problem of virtual sculpture of 3D objects with tools, both represented with spatial enumerations. Such a spatial enumeration is a set of volume elements called voxels, obtained by sampling the volume of a 3D object. It can be seen as a 3D image composed of voxels, where a 2D image is an array composed of pixels. To make a spatial enumeration from a 3D object, several methods have already been suggested. The simplest way is a uniform spatial enumeration, by regularly sampling the 3D object into voxels with the same size. However, a major drawback of this representation is the large number of voxels needed to represent large objects with detailed features. This entails three main problems. The first one is the important memory cost to store this uniform spatial

enumeration. The second one is that the display of these objects become slower. Finally, operations on these objects such as sculpture actions or displacements become less and less interactive.

To prevent these inconveniences, adaptive sampling methods have been developed. Libes [Lib91] uses an octree to gather groups of adjacent voxels having same values to reduce the number of elements stored in memory. It's very simple to use and to implement this method. Ferley [Fer02] also works on a n-tree where each cell can be divided in 27 ones. This method looks like an octree and allows to reach a high level of details. However, for an object with small details, the subdivision level of an octree or n-tree will be very high. So the processes (construction and use) will be slow.

Several other sampling methods used in collision detection propose to modify properties of the voxels, such as the size, the orientation or even the shape.

With the AABB method (Axis Aligned Bounding Boxes), Bergen [Ber97] suggests to use voxels with different sizes. Gottschalk [Got96] proposes to modify not only the size of the voxels but also their orientation, with the OBB method (Oriented Bounding Boxes). Thanks to these two methods, 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.

*Uj qtv'Ego o wplec vkpu'r t qeggf kpi u'KDP': 2/: 8; 65/27/6
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press*

object rendering is optimized because the original object shape can be approached with less voxels than with a simple uniform spatial enumeration. The modeling is finer with OBB tree than AABB tree for a same number of bounding volumes. However, AABB tree uses less memory storage than OBB tree for a same number of bounding volumes. Indeed, an OBB is represented by using 15 scalars (9 scalars for the orientation, 6 scalars for position and extent), whereas an AABB only requires 6 scalars (for position and extent). Moreover, to optimize the modeling of the 3D object, these two methods suggest to reduce overlaps between bounding boxes and to increase their filling by the object, with the less possible boxes. This optimization is expensive in processing time, so we prefer using a uniform spatial enumeration or an octree, because they are faster than AABB and OBB methods.

Liu [Liu88] and Hubbard [Hub95] propose to replace cubes by spheres in an octree to form a spheres tree, because spheres accelerate collision detection between objects. Later, Hubbard [Hub95] [Hub96] and Bradshaw [Bra04] suggest a finer modeling using spheres tree thanks to an approximated medial-axis of the 3D object, but this method is slower and more complicated than an octree.

To further improve the use of spatial enumeration, several methods of multiresolution representations have been proposed. So, processing and display times are adapted with the desired level of details. Among these methods, there are octree and wavelet decomposition.

An octree can also be seen as a hierarchical representation of 3D object. The maximum level of subdivision of the octree defines the maximum level of details of a multiresolution representation. Boada [Boa01] defines a section in an octree that determines the displayed nodes for a defined level of details. This method is extended to a n-tree by Ferley [Fer02].

The second multiresolution method uses wavelet decomposition. Wavelets are a mathematical tool for representing functions hierarchically. In our case, these functions are discrete 3D functions that define a set of voxels. More information about wavelets will be given in the section 2.1. Muraki [Mur92] [Mur93] shows the use of 3D Haar wavelets to represent a 3D object. Pinnamaneni [Pin02] builds a 3D Haar wavelet decomposition from a sequence of 1D Haar wavelet decomposition in each direction of the 3D voxels grid. Wavelet decomposition allows to display a 3D object faster according to the level of details. It also permits to drastically cut down the memory cost,

because high compression ratio can be achieved on wavelets coefficients, especially if lossy compression schemes are used.

The previously cited methods are about discrete representation of 3D objects. Different methods have already been proposed to sculpt these kinds of objects.

In the Kizamu project, Frisken [Fri01] uses ADFs (Adaptively sampled Distance Fields) to model and to sculpt the matter. A 3D object is sampled adaptively with a 3D grid according to the details of the object. Each grid cell contains a scalar specifying the minimum distance to the object shape. This distance is signed to distinguish between the inside and outside of the shape. Ferley [Fer02] also uses distance fields, stored in a “n-tree” hierarchical representation where the sampling rate depends on object’s details. Bærentzen [Bae02] proposes Level-Set method to deform the matter. This method stores distance fields around the exterior of a 3D object.

Raffin [Raf04] proposes a model of virtual sculpture based on a multiresolution representation: the octree. The artist can create his own tools. Each tool modify the data in uniform spatial enumeration, so it modifies the nodes of the octree. This method is more useful and easier than previous method, because it operates directly on spatial enumeration.

Finally, Ayasse [Aya01] proposes to perform sculpture operations by the use of CSG (Constructive Solid Geometry). Complex objects are created by successive modifications of the matter with a tool according to simple operations such as difference, union or intersection. The object and the tool are represented by uniform spatial enumerations. Ayasse proposes to reduce the computation time for each sculpture operation by using only the effective voxels of a movement. This method can be useful because the computation times are reduced. However, it doesn’t use a multiresolution representation that could improve the display performance.

As we can see, many models have been proposed in order to represent 3D objects as discrete sets of voxels. However, there remain open issues. The three main problems are the computation time during sculpture operations and the display that must remain real-time, as well as the important memory cost. In this paper, we propose a model based on 3D wavelets that tackles these issues. Although wavelets have already been used to represent discrete 3D objects, it has never been used in a virtual sculpture context

where the object must be dynamically updated. Thus, problems linked to real-time modifications of 3D wavelets have never been studied.

The use of 3D wavelets allows to solve the three main problems mentioned above.

- First, we take advantage of the multiresolution nature of wavelets to manage the interaction time between the user and the object to obtain real-time interaction.
- Second, the display of the discrete object is done thanks to the marching cubes algorithm [Lor87] that smoothes the rugged aspect of voxels, thus improving the representation of 3D objects. To avoid rebuilding the whole triangular mesh after each sculpture operation, we propose a storage method for the triangles of the mesh, which permits to modify locally and rapidly the mesh of a 3D object. Moreover, the level of details for the display is automatically selected to satisfy the conditions about user/object interaction.
- Third, our method tackles memory issues. As the memory space is limited, we propose a cache system that offers a compromise between the processing, preprocessing and display times and the memory cost. Last but not least, the wavelets representation of the object allows important compression ratio to reduce the memory cost, especially if lossy compression schemes are used.

The remainder of the paper is organized as follows: in section 2, we present our method in five parts: the multiresolution model based on 3D Haar wavelets, a display method, an automatic management of levels of details to accelerate the display, a cache system and finally some virtual sculpture operations. We conclude in section 3. Finally we expose some future works in section 4.

2. OUR MULTIREOLUTION MODEL OF SPATIAL ENUMERATION

2.1 The Model

As the uniform spatial enumeration of a 3D object is expensive in processing and display times, we propose a multiresolution model based on 3D Haar wavelets.

On the following example, we explain Haar wavelet decomposition on a 1D case. First, consider a sequence of p values, where p is a power of two (here, $p = 4 = 2^2$):

$$X^0 = [9, 7, 3, 5]$$

Then, by applying Haar wavelet transform, we can represent this sequence in terms of a low-resolution sequence X^1 and a set of detail coefficients Y^1 :

$$X^1 = \left[\frac{9+7}{2}, \frac{3+5}{2} \right] = [8, 4]$$

$$Y^1 = \left[\frac{9-7}{2}, \frac{3-5}{2} \right] = [1, -1]$$

So, by repeating these operations, we obtain several sets of coefficients corresponding to different levels of details, as shown on the following decomposition table:

level of details #	low-resolution coefficients	detail coefficients
0	[9 7 3 5]	
1	[8 4]	[1 -1]
2	[6]	[2]

So, the higher the number of the level of details, the less detailed the sequence. The sequence obtained by Haar wavelet decomposition has the same size than the original sequence. Its coefficients are the low-resolution coefficients of the last level of details and the different detail coefficients:

Original sequence: [9 7 3 5]

Final sequence: [6 2 1 -1]

The extraction of the original sequence from the final sequence uses the inverse wavelet transform:

$$X^1 = [6 + 2, 6 - 2] = [8, 4]$$

$$X^0 = [8 + 1, 8 - 1, 4 + (-1), 4 - (-1)] = [9, 7, 3, 5]$$

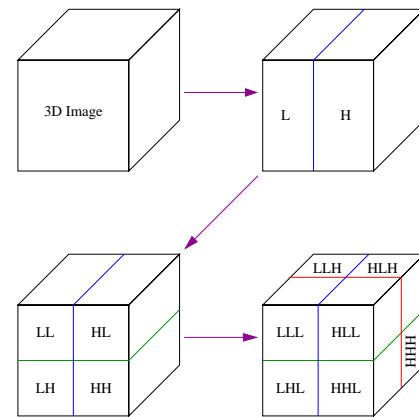


Figure 1. 3D Haar wavelet decomposition.

Similarly, we can use this wavelet transformation in a 3D case. First, the 3D discrete object is defined by a uniform spatial enumeration. Then, by using the wavelet transform we build a hierarchical structure that stores the coefficients of each level of details of this 3D object.

We use the hierarchical structure proposed by Pinnamaneni [Pin02]. For each level of details, the 1D Haar Wavelet transformation is applied in x-, y- and z-direction successively (figure 1).

For each transformation step, we obtain a bloc 'L' with low-resolution coefficients obtained by a low-pass filter, and a bloc 'H' with detail coefficients obtained by a high-pass filter.

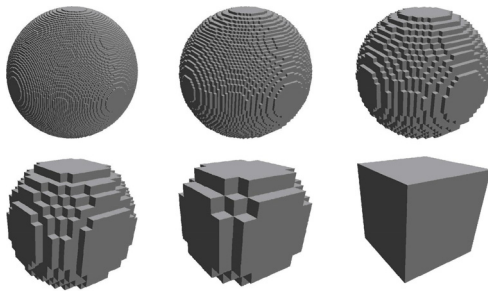


Figure 2. 3D wavelet enumeration for a sphere in 128x128x128 with 6 levels of details (from 0 to 5, from left to right, and from top to bottom).

The figure 2 shows a 3D Haar wavelet decomposition for a sphere with 6 levels of details. Each voxel contains a density value coded in a byte (from 0 for an empty voxel to 255 for a full one).

Several objects permit us to measure the influence of the size of a 3D image on the building time (for wavelet decomposition of the 3D image) and the extracting time (for extraction of a level of details from wavelet enumeration). The results given in this paper have been obtained on a PC with an AMD 3GHz, 1GB of RAM and a NVIDIA Geforce FX 5200 with 128MB video memory.

64x64x64	Objects	Building (4 levels)	Extracting	
			Level 0	Level 1
	cube	0.04599 s	0.06178 s	0.00181 s
	sphere	0.04647 s	0.06228 s	0.00184 s
	torus	0.04611 s	0.06268 s	0.00183 s

128x128x128	Objects	Building (5 levels)	Extracting	
			Level 0	Level 1
	cube	0.69834 s	0.75055 s	0.06286 s
	sphere	0.69753 s	0.75292 s	0.06272 s
	torus	0.70104 s	0.75064 s	0.06252 s

Table 1. Building and extracting times for 3D Haar wavelet enumerations.

As reported on table 1, the building and extracting times do not depend on the kind of 3D object. The building time only depends on the number N of voxels of the initial uniform spatial enumeration and on the maximum level n of details of wavelet enumeration (a is a constant depending on computer power):

$$t_{building} = a \cdot N \cdot [1 - 0.125^n]$$

The extracting time depends on the number N of voxels of the initial uniform spatial enumeration, the maximum level n of details of wavelet enumeration and the extracted level p of details (b is a constant depending on computer power):

$$t_{extracting} = b \cdot N \cdot [0.125^p - 0.125^n]$$

These properties will be useful to estimate the building and extracting times in section 2.4.

2.2 The Display

To display a 3D discrete object, there are several methods such as a display with cubes, spheres or surfels [Sze92]. Nevertheless, the rendering of these methods has a very rough aspect.

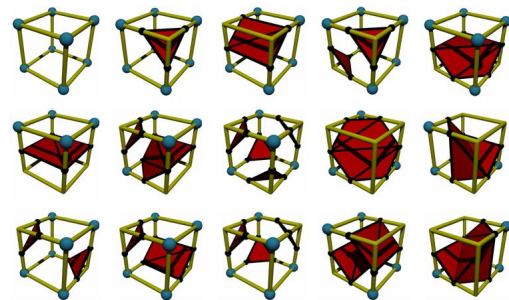


Figure 3. The 15 configurations proposed by Lorensen and Cline in 1987 [Lor87].

To obtain a smooth surface instead of a set of boxes, we use the marching cubes algorithm [Lor87] to build a triangulated surface, by using the different configurations shown on figure 3 (The voxels are the eight vertices of the cubes). For each configuration of full and empty voxels, from 0 to 5 triangles are generated.

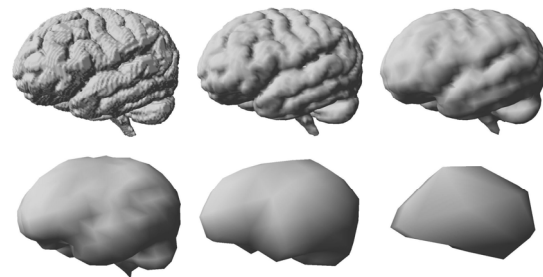


Figure 4. Marching cubes method for an object in 128x128x128 with 6 levels of details (from 0 to 5, from left to right, and from top to bottom).

In the case of a binary coding of the object, the marching cubes algorithm would lead to a rather angular rendering: the surface would always run exactly in the middle of two inside and outside voxels, thus generating angles multiple of 45° . However, as we have values between 0 and 255, we obtain a smoother set of triangles by an interpolation of voxels values.

The marching cubes algorithm is applied on each level of details of 3D wavelet enumeration, so a triangulated surface is obtained for each level of details (figure 4).

Preprocessing times			
	Level 0	Level 1	Level 2
sphere	0.43098 s	0.06498 s	0.01113 s
torus	0.38808 s	0.05524 s	0.01092 s
brain	0.49629 s	0.08615 s	0.01310 s

Display times			
	Level 0	Level 1	Level 2
sphere	0.03294 s	0.00824 s	0.00221 s
torus	0.01843 s	0.00473 s	0.00113 s
brain	0.05552 s	0.01268 s	0.00276 s

Number of triangles			
	Level 0	Level 1	Level 2
sphere	147 968	36 672	9 160
torus	72 368	17 800	4 584
brain	296 416	68 124	13 968

Table 2. Preprocessing and display times and number of triangles of the triangulated surface for 3D Haar wavelet enumerations of 3D objects in 128x128x128.

However, the marching cubes algorithm is a rather slow process. Using this algorithm to compute the triangulated surface of a whole 3D object each time it needs to be displayed would be too time consuming to achieve real-time display (table 2). The more detailed the 3D object, the higher the computation time, as a lot of triangles are computed. In order to improve computation time during sculpting process and to have real-time interactions between a user and the 3D object, we propose the following strategy: we compute the marching cubes on the whole object only once, as a preprocessing step before sculpting. Then, during the sculpting process, the marching cubes algorithm is only applied to the voxels modified by the sculpting tools. Thus, we only compute the new triangles of the surface locally altered by the tool.

In order to modify locally and rapidly the triangulated surface during sculpting process, we propose a method to store the triangles of the surface, which permits to easily find and modify the triangles for the modified parts of the 3D object.

This method uses a data structure (figure 5), composed by:

- A linked list for the storage of triangles of the surface for each block of 8 voxels.
- A 3D matrix of pointers to improve the access to the linked list. When the triangulated surface is

created, the 3D image is processed by block of 8 voxels. For each block, a piece of surface is created with triangles (from 0 to 5) thanks to the marching cubes algorithm. For 0 triangle, the pointer of the 3D matrix is *NULL*. Else, it point to a item in the linked list containing the triangles data.

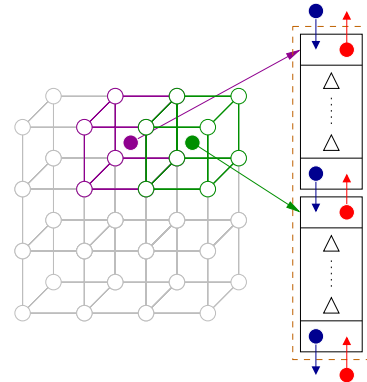


Figure 5. Display structure with a 3D matrix of pointers (on the left) to a linked list (on the right) for the storage of triangles of the surface.

2.3 Automatic Management of the Levels of Details

In order to accelerate the display of a 3D object defined with our model, we take advantage of its multiresolution nature given by the wavelets. We propose an automatic management of the levels of details that selects the more adequate level of the 3D object to display. For that, we use the three following criteria:

- If the user interacts with the 3D object, we define a frame rate N_1 , according to the power of the computer, to display the 3D object in real-time. Consequently, the greater the frame rate N_1 , the faster the display.
- If the user doesn't interact with the 3D object, we define a frame rate N_2 , according to the power of the computer, to display the 3D object. In fact, it's unnecessary to display object in real-time, but the user can interact rapidly with the object. So, the frame rate N_2 is smaller than N_1 .
- Finally, we rectify the level of details obtained by respecting the previous conditions, according to the proximity between the user and the object. So, the greater the distance between the user and the object, the more rough the level of details, because the details become invisible for the user. For a very great distance (right of figure 6), the coarser level of details is displayed. On the contrary, when the distance is small (left of figure 6), the most detailed level of details is displayed.

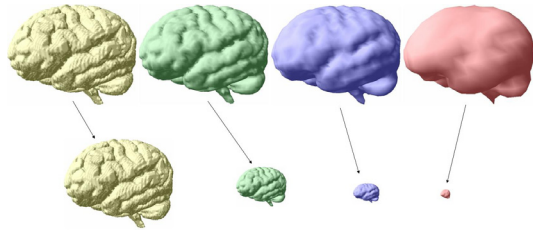


Figure 6. Effect of the proximity between the viewpoint and a 3D object in 128x128x128 with 4 levels of details (from 0 to 3, from left to right).

2.4 The Cache System

Thanks to the automatic management of the levels of details, the display time is improved. But, as the memory space is limited, we can't store in memory all the levels of details of the 3D wavelet enumeration. However, in order to preserve access performance to a given level of details of the 3D object (to modify it, to display it, etc.), it is crucial to avoid to extract this level each time we need to use it. Consequently, we propose a cache system method that offers a compromise between memory usage and the performances of processing, preprocessing and displaying.

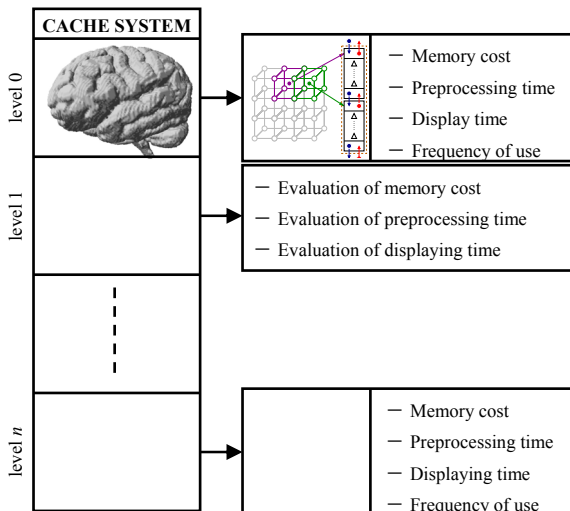


Figure 7. Data structure of the cache system.

The principle of our cache system is to store the most frequently used levels of details of the 3D object. So, we propose the storage of levels of details in memory space, according to several criteria:

- maximum memory size allowed;
- frequency of use of each level;
- processing, preprocessing and display times.

The cache system (figure 7) uses a data structure to store the levels of details of the 3D object. The size of this structure depends on the maximum level of details of the 3D wavelet enumeration. Each level of this structure corresponds to one level of details of

the multiresolution representation. For each level of details, this structure is composed of:

- if the level of details exists:
 - a 3D image;
 - a data structure for displaying the object ;
 - the memory cost;
 - the preprocessing and display times;
 - the frequency of use of the level of details;
- else:
 - the evaluation of the memory cost;
 - the evaluation of the preprocessing and display times;

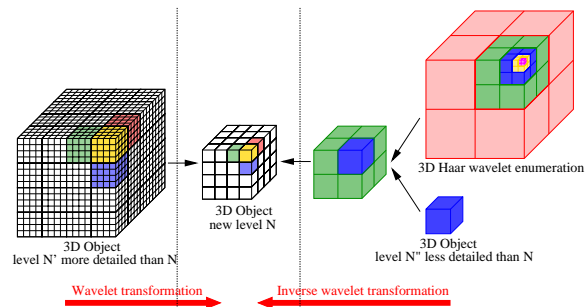


Figure 8. Building the new level from a more detailed level (on the left) or from a less detailed level (on the right).

When a new level has to be added to the cache structure, it is crucial not to extract it from the wavelet enumeration, but to extract it from a level already present in the cache, as less reconstruction steps are needed. For that, the new level is built from:

- the first existing more detailed level by using the wavelet transform only for the low-resolution coefficients;
- the first existing less detailed level by using the inverse wavelet transform.

We automatically choose between these two methods the fastest one to rebuild the new level according to the different existing levels in the cache system (Figure 8).

2.5 The Virtual Sculpture

In this paper, we propose a simple case of sculpture of the matter: adding and subtracting voxels to the 3D wavelet enumeration. A major advantage of our method is that the tool used for virtual sculpture has the same representation than the matter. So, the user can create his or her own tools to sculpt another 3D object. Other tools and the interaction with other representation (like implicit functions, etc.) will be studied in future works.

In this paper, the tool can only be positioned in translation relatively to the matter, without rotation. Rotation issues will be studied in future works. In the case of a tool in translation, a collision test is made between the bounding boxes of the tool and the matter to speed up the sculpture time. If there is a collision, the following operations are performed:

- find the voxels of the matter and the tool which are in the collision zone;
- find which voxel of the tool intersects which voxel of the matter in this zone.

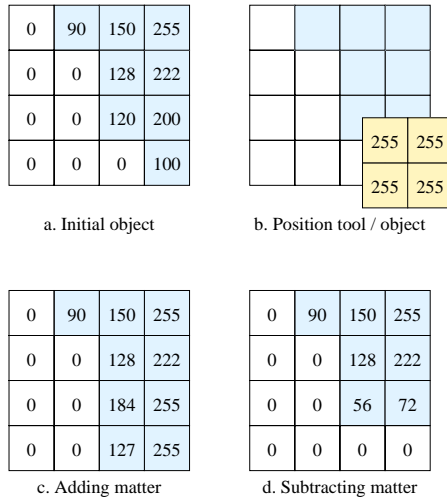


Figure 9. Adding or subtracting matter to an object with a tool.

If there is intersection between voxels of the matter and the tool in the collision zone, the filling percentage of the voxel of the matter by the one of the tool is computed. It's the ratio between the volume of the intersection between the two voxels and the volume of the one of the matter. The two following modes are illustrated in 2D on figure 9:

- in the “Adding matter” mode, if the voxel of the tool isn't null, the filling percentage is added to the value of the voxel of the matter. If this value becomes greater than 255, it is put to 255;
- in the “Subtracting matter” mode, if the voxel of the tool isn't null, the filling percentage is subtracted to the value of the voxel of the matter. If this value becomes negative, it is put to 0.

When the computation is made on all the voxels of the matter in the collision zone, only the images for the levels of details existing in the cache system and the 3D wavelet enumeration are updated.

Moreover, for each level of details existing in the cache system, the triangulated surface is rebuilt only for the modified parts of the 3D object, to improve the computation time. First, the coordinates of the modified voxel are used to access, thanks to the 3D

matrix presented on figure 5, to the items of the linked list that are deleted. Then, the new triangles are computed and the linked list as well as the 3D matrix are updated.

Several examples of sculpture, in 128x128x128, are illustrated on figures 10, 11 and 12 :

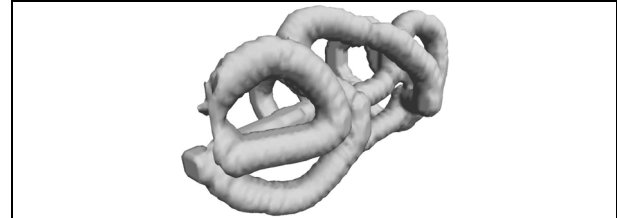


Figure 10. A random sculpture built with a spherical tool in less than 1 minute.

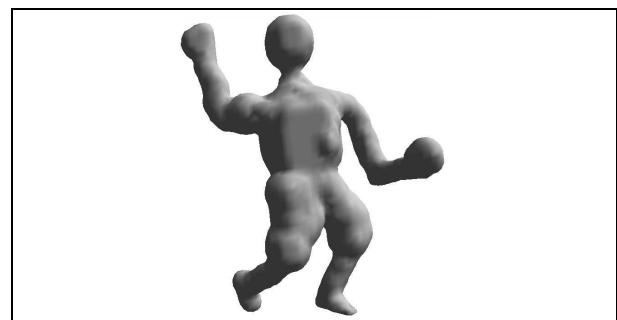


Figure 11. A sportsman sculpted with 3 spherical tools at different sizes in less than 5 minutes.



Figure 12. A relic sculpted with a spherical tool and two ring tools in less than 1 minute.

3. CONCLUSION

We have presented in this paper a model of 3D object that allows real-time virtual sculpture on an average PC. This object is represented as a set of voxels, but we avoid speed and memory issues inherent to this kind of representation thanks to a multiresolution approach based on 3D Haar wavelets. In order to enhance real-time performance, we also have developed several structures, as follows:

- A management of the interaction time between the user and the object to obtain real-time interaction.
- A storage method for the triangles of the mesh, generated by the marching cubes algorithm, to

modify locally and rapidly the mesh of a 3D object after each sculpture operation.

- An automatic management of the level of details for the display to satisfy the conditions about user/object interaction.
- Finally, a cache system to offer a compromise between the memory cost and the processing, preprocessing and display times.

To verify the applicability of our sculpting system, using simple sculpture tools, we have conducted many sculpting sessions which have resulted in numerous interesting sculptures. Some sculptures examples are shown on figures 10, 11 and 12, and several other examples can be seen on <http://www.iut-arles.univ-mrs.fr/thon/>.

4. FUTURE WORKS

Many improvements of our sculpture system are possible, as interaction, speed and memory cost will always be challenging issues.

Concerning interaction, we plan to manage the rotation of tools relatively to the matter. Moreover, more sculpture operations will be added. We will also improve the realism of sculpture actions, by adding parameters to the voxels to imitate physical behavior.

In order to accelerate the display when the user interacts with the matter, different display methods will be investigated.

Finally, we plan to reduce the memory cost of the sculpted object by taking advantage of the important compression ratio allowed by the wavelet representation.

5. REFERENCES

- [Aya01] Ayasse, J., and Müller, H. Interactive Manipulation of Voxel Volumes with Free-formed Voxel Tools. In Proceedings of the Vision Modeling and Visualization Conference 2001, pp.359-366, 2001.
- [Bae02] Bærentzen, J.A., and Christensen, N.J. Volume sculpting using the Level-Set method. Shape Modelling International 2002, IEEE Computer Society, pp.175-182, 2002.
- [Ber97] Bergen, G.V.D. Efficient collision detection of complex deformable models using AABB trees. Journal of Graphic Tools, 2(4), pp.1-13, 1997.
- [Boa01] Boada, I., Navazo, I., and Scopigno, R. Multiresolution volume visualization with a texture-based octree. Visual Computer, 17, pp.185-197, 2001.
- [Bra04] Bradshaw, G., and O'Sullivan, C. Adaptive medial-axis approximation for sphere-tree construction. ACM Transactions on Graphics, 23(1), pp.1-26, 2004.
- [Fer02] Ferley, E. Sculpture virtuelle. Ph.D. thesis, Institut National Polytechnique de Grenoble, 2002.
- [Fri01] Frisken, S.F., and Perry, R.N. Kizamu: a system for sculpting digital characters. Proceedings of ACM SIGGRAPH 2001, pp.47-56, 2001.
- [Got96] Gottschalk, S., Lin, M.C., and Manocha D. OBB-Tree: A hierarchical structure for rapid interference detection. In Proceedings of ACM SIGGRAPH'96, pp.171-180, 1996.
- [Hub95] Hubbard, P. Collision detection for interactive graphics applications. Ph.D. thesis, Dept. of Computer Science, Brown University, 1995.
- [Hub96] Hubbard, P. Approximating polyhedra with spheres for time-critical collision detection. ACM Transactions on Graphics, 15(3), pp.179-210, 1996.
- [Lib91] Libes, D. Modeling dynamic surfaces with octrees. Computer & Graphics, 15(3), 1991.
- [Liu88] Liu, Y., Noborio, J., and Arimoto, S. Hierarchical sphere model (HSM) and its application for checking an interference between moving robots. In Proceedings of the IEEE International Workshop on Intelligent Robots and Systems, pp.801-806, 1988.
- [Lor87] Lorensen, W.E., and Cline, H.E. Marching Cubes: a high-resolution 3D surface construction algorithm. Computer Graphics, 21(4), pp.163-169, 1987.
- [Miz98] Mizuno, S., Okada, M. and Toriwaki, J. Virtual sculpting and virtual woodcut printing. The Visual Computer, 14(2), pp.39-51, 1998.
- [Mur92] Muraki, S. Approximation and rendering of volume data using wavelet transforms. In Proceedings of Visualization '92, Boston, pp.21-28, 1992.
- [Mur93] Muraki, S. Volume data and wavelet transforms. IEEE Computer Graphics and Applications, 13(4), pp.50-56, 1993.
- [Pin02] Pinnamaneni, P., Meyer, J., and Saladi, S. Remote transformation and local 3-D reconstruction and visualization of biomedical data sets in Java3D. In Proceedings of Electronic Imaging Science & Technology Visualization and Data Analysis Conference, San Jose, CA, pp.44-54, 2002.
- [Raf04] Raffin, R., Gesquière, G., Remy, E., and Thon, S. VirSculpt: a virtual sculpting environment. GraphiCon '04 Proceedings, pp.184-187, 2004.
- [Sze92] Szeliski, R., and Tonnesen, D. Surface modeling with oriented particle systems. Computer Graphics, 26(2), pp.185-194, 1992.

mv number: Effective Key to Represent Images

Pranam Janney⁺
Pranam_Janney@satyam.com

Hrishikesh Amur⁺⁺

Sridhar G⁺
Sridhar_Gangadharpalli@satyam.com

⁺Applied Research Group,
Satyam Computers Ltd,
Indian Institute of Science (IISc)
Bangalore – 560041, India

Sridhar V⁺
Sridhar@satyam.com

⁺⁺Intern at Applied Research
Group, Third year (B.Eng(CS))
student from National Institute of
Technology, Surathkal, India.

ABSTRACT

Recent research has given more importance on the optimization of the indexing structure; however there is a great need for research in the area of image representation for efficient retrieval. In this paper we present a method for reducing dimensions in multi-dimensional multimedia data while preserving similarities between different images. We propose to reduce the multi-dimensionality of the feature vectors to a single unique key called the *mv* number. Using this *mv* number as an effective key to represent image, we could achieve better efficiency in image matching and retrieval.

Keywords

Image representation, Dimensionality Reduction, Multi Dimensional Feature moments, Image retrieval.

1. INTRODUCTION

Content - based image retrieval helps users to retrieve relevant images from image databases based on their contents. There are numerous image retrieval algorithms already developed [Con92a, Jou95a, Con96a, Con04a, Con05a]. Most of these techniques attempt to characterize the image into a set of features or feature moments. These feature moments would act as matching factors or criteria for retrieval purposes. Traditionally these feature moments were extracted for low-level characteristics. However these low level feature moments fail to correlate high level characteristics. We try to capture the characteristics using feature moments derived through wavelet analysis techniques, which correlate to high and low level characteristics of the image. The feature moments thus derived are multi-dimensional, which creates complexities in indexing of images in databases for efficient matching and retrieval. In this paper, we propose a method to reduce the dimensionality of the feature characteristics which represent the image for matching purposes. We

represent an image using a single unique key called the *mv* number in order to minimize the effort and computation required for forming clusters. An important aspect of this method is automatic rank ordering of the members inside the clusters, which obviates the need for computationally expensive distance calculations in retrieval system. For robust retrieval, using the proposed technique would suffice the need for distance calculation since the retrieved results are rank ordered according to their closeness to the query image.

The paper is organized as follows: Section 2 surveys related work, Section 3 introduces the sequence numbering technique; Section 4 consists of test results. Finally, we conclude in section 5.

2. RELATED WORK

Images are now being represented in a more concise and precise manner for efficient searching and browsing. Color indexing is insensitive to scale, rotation and translation and it is also robust to occlusion and changes in camera view angle. However, one of the main disadvantages of color indexing is that only color information is stored and the spatial relationship between pixels is lost. Moreover color indexing scheme has proven to be a disadvantage for large image databases [Jou00a]. In [Con95b], researches have divided the image into several sub images and represented these sub images with their color histogram for indexing. This method is computationally expensive and storage overheads are very high. Researchers have used image segmentation to single out important regions for indexing [Con99a]. This however is a very

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.

Short Communications proceedings ISBN 80-86943-05-4
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

complicated process in terms of feature extraction and matching.

Some researchers have incorporated the local spatial relations. In [Wor96a], each pixel is classified as coherent or non coherent, based on whether the pixel and its neighbors have similar colors. Such approach can distinguish widely scattered pixels from closely clustered pixels. A similar method, known as the color structure descriptor is defined in the MPEG-7 standard [MPG701]. An 8 X 8 mask is used as the structuring element to collect color statistics such that spatially highly scattered and closely clustered color pixels can be distinguished. In [Jou02a], connected pixels with similar colors are grouped together to form color blobs, and statistics of the color blobs' geometry properties are used to retrieve image. All the above methodologies deal with color statistics and spatial relationship between pixels to derive effective representation of images for efficient retrieval.

In [Jou98a], the researchers have used Hilbert space filling curves to map the multi-dimensional feature vectors of image onto a Hilbert Grid through which they build an indexing scheme called HG tree for searching. The multidimensional feature vectors are reduced to smaller dimensions using Hilbert space filling curves however the complexity involved in mapping these multidimensional feature vectors onto a Hilbert grid are high. In our recent paper [Con05a], we have used complex wavelet decomposition to decompose images and represent images using feature vectors for effective retrieval. Linear indexing was used on these feature vectors. We propose to reduce the multi-dimensionality of feature vectors to a single unique key with an objective to represent images.

3. KEY: *mv* NUMBER

In our previous work [Con05a], we have used sets of μ and σ values as effective representations of an image. These values are derived from the detailed coefficients of the decomposition technique discussed in [Con05a]. Complex wavelets with five scales of decomposition are used to derive 30 detailed coefficient matrices from which mean and variance pairs can be found for each of red, green and blue components of a colored image. Each coefficient matrix represents the behavior of that image at particular decomposition level for a particular orientation. Thus we try to map the variation in the image from every orientation in five decomposition scales. Thus, we can satisfactorily represent an image using set of μ and σ values.

We introduce a set of three parameters which could

be used to reconstruct the feature characteristics (μ and σ). The three parameters are:

$$\alpha_n = \frac{\mu_n}{\sigma_n} \quad (3.1)$$

$$\beta_n = \frac{\mu_n}{\mu_{n+1}} \quad (3.2)$$

$$\gamma_n = (\mu_n - \mu_{n+1}) \quad (3.3)$$

Where

$n = 1, 2, \dots, T-1, T$ = total number of feature vectors.

Fig 1(b) plots the α values for RGB with thirty features for each color of the image 1 shown in fig 1 (a). It is noticeable that for a given color, the α value remains almost consistent with minor fluctuations. With negligible loss in accuracy we can represent each color by a single value of α . Calculating the weighted average of these values:

$$\alpha = \frac{\sum_{i=1}^n \alpha_i \mu_i}{\sum_{i=1}^n \mu_i} \quad (3.4)$$

Where, n = number of color components.

α plot of three images are shown in Fig 1(b), Fig 2 (a) and (b). The α curve of the two similar images follows a similar pattern; whereas the dissimilar image does not follow suit. This dissimilarity in curve characteristics would characterize the difference between two dissimilar images, hence we use α as a criterion for generating our sequence.

The β values measure the trend of the μ graph. They neither indicate the mean about which the curve fluctuates nor do they quantify these fluctuations. But every value indicates the change from the previous value. These values serve to differentiate between, say two curves of similar trend which fluctuate about different means.

The γ value quantifies the change in the first two μ values for each color. Therefore using the γ value and the full set of β values the entire curve can be reconstructed. τ is a combination of β and γ :

$$\tau = (\beta, \gamma) \quad (3.5)$$

β and γ values were calculated for the three images in Fig 1(a). Fig 3 shows the plot of β versus γ . From fig 3, we can see that out of three images two were similar and third was a dissimilar image. The two similar images are clustered and the third is astray. It is evident from Fig 3 that images with similar β and γ values would be clustered

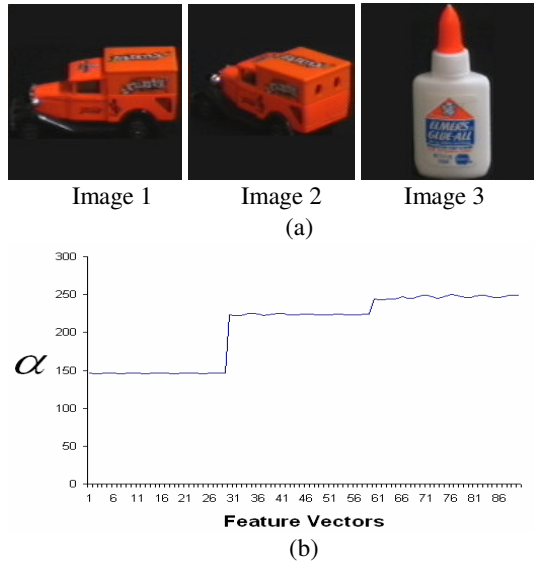


Fig 1. (a) Image 1, Image 2 and Image 3 (b) α characteristics of image 1

thus substantiating our claim that these are valid criteria for image characterization.

Thus, we assume that if a set of parameters can be used to reconstruct a particular distribution, then the set of parameters are unique to that distribution.

Thus multidimensional feature characteristics could be reduced to a set of few parameters which could be used for indexing and similarity calculations. Combining τ and α criteria would result in efficient mv numbering of the images thus resulting in better clustering. α and τ are combined to generate a 13 digit mv number. This scalar key thus approximates the reduction of the multidimensional feature characteristics to a one dimensional quantity. The mv number is a simple key representation of the image using the three proposed parameters i.e. Eqn 3.1, 3.2, 3.3.

4. EXPERIMENTAL SETUP AND TEST RESULTS

The experimental setup for testing the mv numbering technique was implemented in Linux on a Pentium PC. Coil 100 [10] database consists of color images of 100 objects taken in 72 different angles, thus resulting in a database of 7200 images in total. For consolidating our technique we have extracted 30 images from the COIL 100 database. We manually classified the extracted images into two different groups. Fig 4 shows β vs. γ plot for these 30 images. We can see from the figure that the images are automatically clustered into two groups depending on their mv number. The area of these

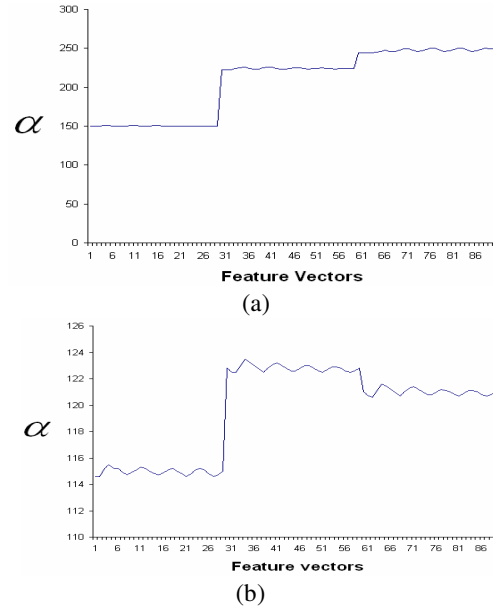


Fig 2. (a) α characteristics of Image 2, (b) α characteristics of Image 3

clusters could be bound by a threshold value. Any subtle changes in the image would reflect on its feature characteristics which in turn would also reflect on its mv number. However the change in the mv number would be proportional to the changes in the image characteristics itself. Changes in light, scale translation etc would result in very small changes in mv number there lending robustness in matching two similar images.

We have used a simple binary tree for indexing the database with mv number as the key for each image in the database. The root of the binary tree is a simple average of the mv numbers of all the images in the tree below. We have compared our test results with the HG tree [Jou98a].

The effectiveness of the indexing schemas with regard to searching was judged by measuring the number of memory accesses (read + write). Since the tree is stored in the memory, every calculation involving a value from the tree is counted as a memory access.

Even though HG tree has tertiary tree structure, it has more disk accesses for a given set of nearest neighbor search when compared to a simple binary tree with our proposed sequence number as a key for each image. Especially for low nearest neighbor queries, index structure with mv number as key has very low disk accesses. Effective and efficient clustering due to mv numbering technique is a major factor for the low disk accesses of the simple binary

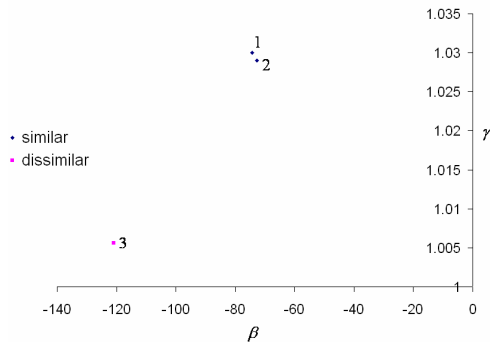


Fig 3. β vs. γ plot for the three images

index structure.

5. CONCLUSIONS

We have proposed *mv* numbering as a key for effective representation of images. Test results support our claim that this *mv* numbering technique helps in forming efficient and effective clusters. This cluster formation would minimize the effort and computation required for creation of clusters. For robust retrieval, the proposed technique would suffice the need for distance calculation since the retrieved results are ranked ordered according to their closeness to the query image. Thus, *mv* numbering could be used as an effective and efficient image representation in matching and retrieval systems. In our future work, we would research on feasibility of the key *mv* in different types of indexing structures for effective and efficient retrieval and also research on better indexing structures in which this key technique would yield optimum performance.

6. REFERENCES

- [Con92a] Kato, T., "Database architecture for content based image retrieval in Image storage and Retrieval Systems" (Jambardino A and Niblack W eds), *Proc SPIE* 2185, pp 112-123, 1992.
- [Jou95a] Flickner, M. et al., "Query by Image and Video content: the QBIC system", *IEEE Computer*, vol 28, pp 23-32, 1995.
- [Con96a] Bach, J.R., et al., "The virage image search engine: an open framework for image management", *Proc. SPIE: Storage and retrieval for Still Image and Video Databases IV*, vol 2670, pp 76-87, 1996.
- [Con04a] Avinash, T. et al., "PATSEEK: Content Based Image Retrieval System for Patent Database", *International Conference on Electronic Business*, 2004.
- [Con05a] Pranam Janney et al., "Enhancing capabilities of Texture Extraction for Color Image retrieval", *Proc. of World Enformatika Conference*, Turkey April 2005.

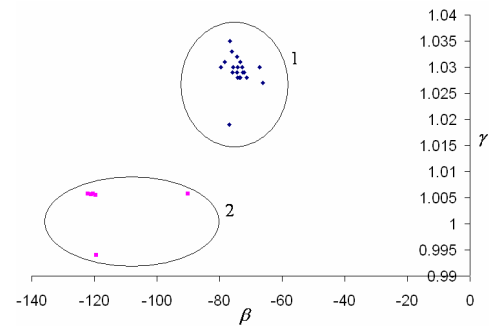


Fig 4. β vs. γ plot for 30 images.

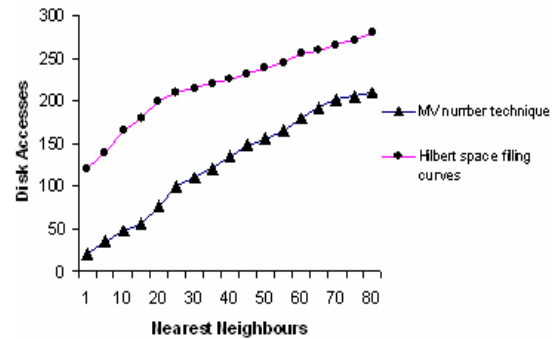


Fig 5. Disk accesses

- [Jou00a] A.W.M.Smeulders et al., "Content-based image retrieval at the end of the early years," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, pp 1349-1380, Dec. 2000.
- [Con95b] W.Hsu, et al., "An integrated color-spatial approach to content-based image retrieval," in *Proc. 1995 ACM Multimedia Conf.*, pp 305-313.
- [Con99a] C.Carson et al., "Blobworld: A system for region-based image indexing and retrieval," in *Proc. Int. Conf. Visual Information Systems*, 1999.
- [Wor96a] G.pass and R. Zabih, "Histogram refinement for content-based image retrieval," in *Proc. IEEE Workshop on Applications of Computer Vision*, 1996, pp. 96-102.
- [MPG701] "MPEG7 FCD," Singapore, ISO/IEC JTC1/SC29/WG11, 2001.
- [Jou02a] G.Qui et al., "A binary color vision framework for content-based image indexing," *Recent Advances in Visual Information Systems*, S-K Chang et al., Eds. New York: Springer, 2002, vol. LNCS 2314, pp. 50-60.
- [Jou98a] Guang-Ho et al., "A New Indexing Scheme for Content Based Image Retrieval", *Multimedia Tools and Applications* 6, 263-288 (1998), Netherlands.
- [Dat98a] "Columbia object image library (COIL-100)", Department of Computer Science, Columbia University, www.cs.columbia.edu/CAVE/research/oftlib/coil-100.html

Automatic Human Body Modeling for Vision-Based Motion Capture

Antoni Jaume i Capó, Javier Varona, Manuel González-Hidalgo,
Ramon Mas, Franciso J. Perales

Unitat de Gràfics i Visió, Dept. Matemàtiques i Informàtica,
Ed. A. Turmeda, Universitat de les Illes Balears (UIB),
07122 Palma de Mallorca, Spain

antoni.jaume@uib.es
<http://dmi.uib.es/~ajaume/>

ABSTRACT

In this paper we present a computer vision algorithm for building a human body model skeleton in an automatic way. The algorithm is based on analyzing the human shape. We decompose the body into its main parts by computing the curvature of a B-Spline parameterization of the human contour. This algorithm has been applied in a context where the user is standing in front of a camera stereo pair. The process is completed after the user performs a predefined initial posture for identifying her principal joints, and building the human model. Using this model, we solve the initialization problem of a vision-based markerless motion capture system of the human body.

Keywords

Motion capture, human body modeling, shape analysis.

1. INTRODUCTION

Nowadays, human motion capture has a wide variety of applications such as 3D interaction in virtual worlds, performance animation in computer graphics, or motion analysis in clinical studies. This problem has been solved by commercially available motion capture equipment, but this solution is far too

expensive for common use and it requires special clothing with retro-reflective markers [Vic05a]. Markerless motion capture systems are cheaper and they do not require any special clothing. They are based on computer vision techniques [Moe01a, Wan03a] and therefore they are named as vision-based motion capture systems. However, their results are less accurate but enough for applications such as 3D interaction in virtual worlds.

In vision-based motion capture a main issue is the human body model. This model must be accurate for representing motions by means of body postures but also simple to make the problem treatable and to obtain a real-time feedback of the application. Usually, this model is built previously and can be modeled from user's images [Rem04a]. Common based techniques for modeling are visual-hulls [Car03a, Che03a]. All these previous models have a realistic appearance but they are too accurate for their use in real-time applications. We are most interested in build less accurate models that

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

SHORT COMMUNICATION proceedings
ISBN 80-86943-05-4
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

represents well enough users' motions. Then, we try to model the user's kinematical structure [Hil05a].

In addition, vision-based approaches are based on the temporal coherence of user's motions. This fact implies to know the user previous postures and his initial posture. That is, the body model has to be initialized at the first frame. We define this initialization as finding the joints 3D position of the user in the first frame. Currently, a common practice of vision-based works is overcome this problem by manual annotation [Bre04a, Deu05a].

In this paper we present an automatic initialization for a vision-based motion capture system. Our algorithm is based on analyzing user's body shape projected into the images, that is, his image silhouettes. The key idea is to cut each silhouette in different body parts assuming that the user stands in a predefined posture. Subsequently, from these cuts we can estimate the user joints 3D position. Therefore, we can also build the kinematical human model of the user.

The remainder of this paper is organized as follows. Section 2 gives our approach to obtain user's joint positions. In this section, we explain how to parameterize the user's silhouette and to automatically find the cuts. Next, section 3 overviews the application where we use the described algorithm. Finally, the conclusion and future work are described in section 4.

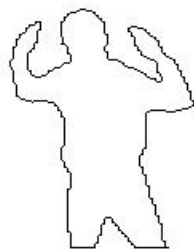


Figure 1: The human body shape.

2. FINDING USER'S BODY JOINTS

As started before, this work is based on the analysis of the human body shape. In this work, when we talk about *shape* we refer to the human 2D silhouette projected on the image, see Fig. 1. Our approximation is to decompose the human shape into different parts by means of *cuts*. Therefore, if the user stands in a predefined posture it is possible to assume that the joints are placed following a known order, then we can correctly label these joints to build user's body model.

First, we describe the body model used in the process and the required initial posture. Next, the human body shape decomposition is described. Finally, we explain how to find and label the joints to build the user's body model.

2.1 The simplified articulated body model

The selection of the human model has to be done according to the kind of information that we want to extract from the image data, i.e human body models used for synthesis applications may need to be more accurate and realistic than human body models used in motion capture applications. Our model is used for motion capture purposes, specifically for 3D interaction in virtual worlds. Therefore, the human model has to be accurate enough for representing motions by means of body postures but simple enough to make the problem treatable and to obtain a real-time feedback of the application.

As result, the body model used in this work is an articulated model with 9 joints (Fig. 2), where every joint have 3 degrees of freedom and the links between the segments are rigid:

1. Virtual foot: roots the body to the floor.
2. Back: corresponds to the beginning of the spine.
3. Neck.
4. 2 x Shoulders.
5. 2 x Elbows.
6. 2 x Wrists.

Using this model, our goal is to describe the configuration of the human body while user is performing a predefined posture in the initialization stage.

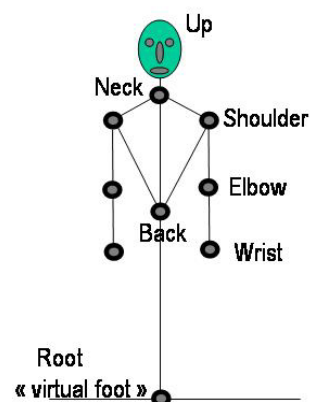


Figure 2: Articulated body model.

2.2 Human body decomposition

Before finding body parts it is necessary to segment the human body shape from the scene background. We have used two methods to do this task: a chroma-key approach [Smi96a] and by means of background subtraction [Hor00a]. Currently, we use the chroma-key approach due to the real-time nature of the final application of our system. However, the algorithm for human body modeling that we present performs equally well without chroma-key. The shapes found using both methods can be seen in Fig. 3.

Once the human shape has been obtained we need to cut the silhouette into different parts to find the joints. According to human intuition about parts, segmentation into parts occurs at negative minimum curvature (NMC) of the silhouette, leading to Hoffman and Richards's minima rule: "For any silhouette, all negative minima curvature are boundaries between parts" [Hof84a].

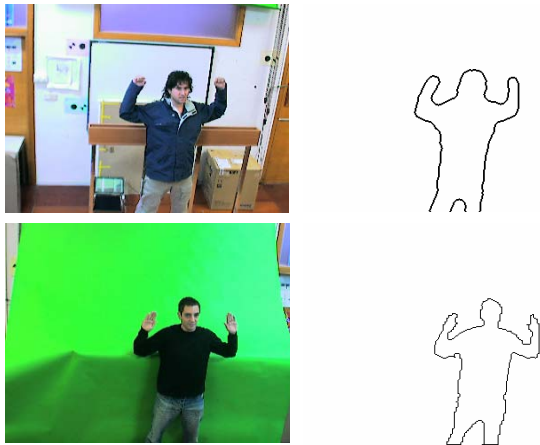


Figure 3: Human body segmentation.

Therefore, it is necessary to find the NMC points of the shape. It is possible to compute the shape curvature directly by means of finite differences over the discrete shape. However, this local computation of curvature is not robust in images. Hence, we parameterize the human body shape using a B-Spline interpolation.

A p th degree B-Spline is a parametric curve, where each component is a linear combination of p th degree basis curves. From the image we obtain a set of contour points $\{Q_k\}$, $k=0, \dots, n$ and we interpolate these points with a cubic B-Spline [Pie97a]. It can be seen in Fig 4 how the B-spline interpolation smoothes the human silhouette.



Figure 4: B-Spline interpolation of human silhouette.

Using the B-Spline shape parameterization it is possible to analytically compute the partial derivatives up to order 2 to obtain the curvature values along the shape. In Fig. 5 the maximum and minimum of the curvature values are shown.



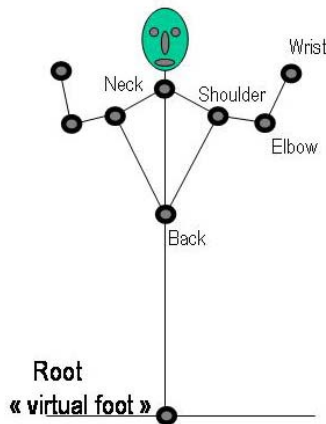
Figure 5: Minimum (in white) and maximum (in green) of the curvature.

However, the minima rule only constrains *cuts* to pass through the NMC points, but does not guide the selection of *cuts* themselves. On one hand, Singh et al. noted that when the boundary points can be joined in more than one way to decompose a silhouette, human vision prefers the partitioning scheme which uses the shortest *cuts* [Sin99a]. This leads to the short-cut rule which requires a cut:

1. Be straight line.
2. Cross an axis of local symmetry.
3. Join two points on the outline of a silhouette, such that at least one of the two points has negative curvature.

4. Be the shortest one if there are several possible competing cuts.

On the other hand, if we know the user's posture it is possible to predict where the *cuts* are. In order to easily obtain the main *cuts*, it is required that the user stands in a predefined posture adequate for finding all the joints of our body model, see Fig. 6.



Studying the initial posture of Fig. 6, we find that negative and positive minimum curvature points lay near the joints that we aim to find. This fact is clearly visible in Fig. 5. Then, according to the short-cut rule and taking into account the user's initial posture we propose the next rules to decompose the human shape:

1. The Back is found at the negative minimum curvature point with the lowest y component.
2. Neck is placed at the middle point of the *cut* between the two negative minimum curvature points with the highest y component.
3. Build the body principal axis with the Neck and Back points. This axis divides the human body shape in two parts.
4. Shoulders are located at the middle point of the *cut* between the negative minimum curvature point with the highest y of the left/right side, and the negative minimum curvature point with lowest y component of the left/right side, excluding the Back point.
5. Elbows are placed at the middle point of the *cut* between the negative minimum curvature point with highest/lowest x component and positive maximum curvature with highest/lowest x component.

Where x and y refers to the horizontal and vertical image coordinates respectively. Applying these rules we obtain the user's body shape decomposition as it is showed in Fig. 7. Fig. 8 shows the human body model from the obtained *cuts*.

Subsequently, we estimate the positions of the shoulders and elbows joints as the *cut* middle point. Once we have the joints 2D positions in each image of the stereo pair, we can estimate their 3D position using the mid-point triangulation method. With this method, the 3D position is computed projecting each joint 2D position on each image to infinity and computing their 3D coordinates as the nearest point to these two lines [Tru98a].



Figure 7. Obtained cuts.

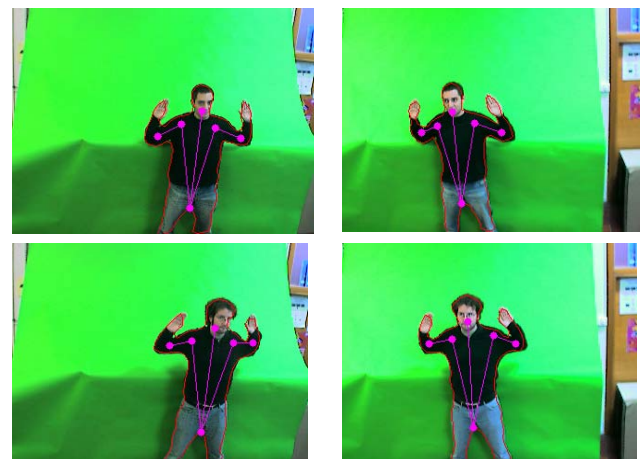


Figure 8. Human Body Models of different users.

Finally, wrist joints need to be estimated to complete our body model. However, they can not detect using the proposed shape analysis method. In this case we use the color cue to find in the images the hands [Var05a]. To locate wrists we approximate the hands by means of ellipses. Using the 3D positions of elbows previously located, we search in the image for the intersection between a 2D line, defined by the elbow and center of the hand positions, and the 2D ellipse hand approximation.

3. VISION-BASED MOTION CAPTURE

The described method to build a human body model has been used at the initialization stage of a motion capture system [Bou05a]. The main advantage of the proposed system is to avoid invasive devices on the user. Besides, the whole process must be done in real-time because the system is used for interacting with virtual environments, where the interaction must be done under very strict deadline times to achieve good feedback rates.

This approach combines video sequence analysis, visual 3D tracking and geometric reasoning; to deliver the user's motions in real-time. This brings to the end user to make large upper body movements in a natural way in a 3D scene. For example, user can explore and manipulate complex 3D shapes by interactively selecting the desired entities and intuitively modifying their location or any other attributes in real time. This technology could be used to implement a wide spectrum of applications where an individual user could share, evaluate key features, and edit virtual scenes between several distributed users. One key novelty of the present work is the possibility to interact in real-time, in 3D, through the current body posture of the user in the client application.

Presently, the system is able to process, not only 3D position of the end effectors, but also a set of human gesture signs, hence offering a richer perceptual user interface. Results obtained using this system is shown in Fig. 9.

4. CONCLUSIONS

In this paper we have presented an algorithm for initializing a human body model in an automatic way. This model is adequate for motion capture purposes. The algorithm is mainly based on shape analysis and human body silhouette decomposition. In order to automatically model the user's body we

have defined a set of rules that performs well if the user stands in a predefined posture.

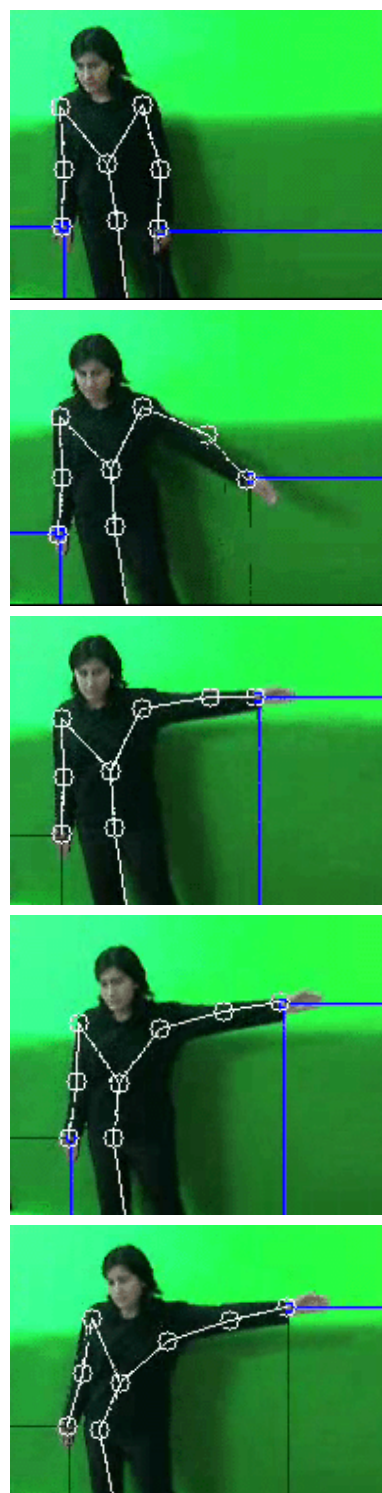


Figure 9: Using the body model in vision-based motion capture.

5. ACKNOWLEDGMENTS

This work was supported by the project TIN2004-07926 of Spanish Government. Besides, Javier Varona acknowledges the support of a Ramon y Cajal fellowship from the Spanish MEC.

6. REFERENCES

- [Bou05a] R. Boulic, J. Varona, L. Unzueta, M. Peinado, A. Suescun, F. Perales, "Real-Time IK Body Movement Recovery from Partial Vision Input", Proceedings of the 2nd International ENACTIVE Conference, Genoa, 2005.
- [Bre04a] C. Bregler, J. Malik, K. Pullen, K., "Twist based acquisition and tracking of animal and human kinematics", International Journal of Computer Vision, 56(3):179-194.
- [Car03a] J. Carranza, C. Theobalt, M. Magnor, H. Seidel, "Free-Viewpoint video of Human Actors", Proceedings of ADM SIGGRAPH 2003:569-577.
- [Cha05a] J. Chai, J.K. Hodgins, "Performance animation from low-dimensional control signals", Proceedings of SIGGRAPH 2005, ACM Trans. Graph. 24(3):686-696, 2005.
- [Che03a] G. Cheung, S. Baker, T. Kanade, "Shape-From-Silhouette of articulated objects and its use for human body kinematics estimation and motion capture", Proceedings of IEEE CVPR 2003.
- [Deu05a] J. Deutscher, I. Reid, "Articulated Body Motion Capture by Stochastic Search", International Journal of Computer Vision, 61(2):185-205, 2005.
- [Hor00a] T. Horprasert, D. Harwood, L. S. Davis, "A robust background subtraction and shadow detection", Proc. ACCV'2000, Taipei, Taiwan, 2000.
- [Hil05a] A. Hilton, M. Kalkavouras, G. Collins, "3D studio production of animated actor model", Vision, Image and Signal Processing, IEEE Proceedings-Volume 152, Issue 4, 5 Aug. 2005 Page(s):481 - 490.
- [Hof84a] D. Hoffman, W. Richards, "Parts of recognition", Cognition 18: 65-96, 1984.
- [Lee04a] M.W. Lee, I. Cohen, "Human Upper Body Pose Estimation in Static Images", Proc. ECCV'2004, LNCS 3022: 126--138, 2004.
- [Moe01a] T.B. Moeslund, E. Granum, "A Survey of Computer Vision-Based Human Motion Capture", Computer Vision and Image Understanding 2001, 81(3):231-268
- [Pie97a] L. Piegl, W. Tiller, "The NURBS Book ", Springer, ISBN 3-540-61545-8, 1997.
- [Rem04a] F. Remondino, "3-D reconstruction of static human body shape from image sequence", Computer Vision and Image Understanding, 93(1):65--85, 2004.
- [Sin99a] M. Singh, G. D. Seyranian, D. D. Hoffman, "Parsing silhouettes: The short-cut rule", Perception and Psychophysics, 61:636-660, 1999.
- [Smi96a] A.R. Smith and J.F. Blinn, "Blue screen matting," Proc. SIGGRAPH 96, pp: 259-268, ACM SIGGRAPH, Addison-Wesley, 1996.
- [Tru98a] E. Trucco, A. Verri, "Introductory Techniques for 3D Computer Vision", Prentice-Hall, 1998.
- [Var05a] J. Varona, J.M. Buades, F.J. Perales, "Hands and face tracking for VR applications", Computers & Graphics, 29(2):179-187, 2005.
- [Vic05a] Vicon Systems, 2005. <http://www.vicon.com>.
- [Wan03a] L. Wang, W. Hu, T. Tan, "Recent developments in human motion analysis", Pattern Recognition 2003, 36(3):585-601.
- [Zha01a] L. Zhao, "Dressed human modeling, detection and parts localization", PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2001.

Virtual Grasping of Deformable Objects with Exact Contact Friction in Real Time

J  r  mie Le Garrec Claude Andriot Xavier Merlhiot

CEA LIST SCRI

Fontenay-aux-Roses, France

(jeremie.legarrec , claud  .andriot , xavier.merlhiot)@cea.fr

Philippe Bidaud

Laboratoire de Robotique de Paris LRP

Fontenay-aux-Roses, France

bidaud@robot.jussieu.fr

ABSTRACT

This paper describes a physically-based simulation for grasping tasks in an interactive environment. Fingertips and interacting objects are based on quasi-rigid models. The quasi-rigid model combines a rigid model for dynamic simulation and a deformable model for resolving local contact with friction and surface deformation. We simulate deformation by adding compliance on control points in the contact area based on point primitives. We use a Coulomb's law description to add friction phenomenon in the virtual environment. A linearized formulation of this model in the contact space and an iterative Gauss-Seidel like algorithm are able to solve complex multi-contacts problem between deformable objects in real time. Our method computes consistent and realistic contact surface in a stable way. This allows us to couple this system with an optical motion capture system for Virtual Reality applications.

Keywords

grasping, real-time simulation, deformable model, motion capture

1 INTRODUCTION

Deformable objects play an important role in many interactive virtual environments and this role has become crucial in grasping tasks. Indeed deformation of human fingertips is a key factor in stable manipulation, as they conform to the object's shape. Compliant contact with friction on a larger area than one point are able to resist more efficiently to perturbations in the contact space and prevent an object from slipping. Many approaches used to simulate real time deformations in interactive grasping task are based on mass-spring model or on finite element analysis. In this paper we introduce a compliant contact model adapted to a point-based representation to model the surface of the objects for this kind of tasks. Point primitives has been investigated in computer graphics in recent years. Using this representation seems to be the most

adapted geometric modelling for fingertips to get objects exactly conform in the contact area. Indeed triangle meshes have irregular connectivity and objects have commonly incompatible connectivity graphs on the two sides of the contact area, which requires refinement with high computational cost.

Soft grasped objects and fingertips deform due to contact forces. So the collision detection algorithm as well as the collision response model play a major role in this kind of real time simulation. In this paper we present a robust collision detection for point-based simulation and a deformation model based on compliant contacts with friction. We use a constant time step integration called time-stepping method. The motion dynamic formulation solves for the contacts appearing between steps. This simulation can be coupled with an optical motion capture system for Virtual Reality applications.

2 RELATED WORK

Various theoretical and experimental results have been presented by robotics communities for modelling soft finger contact or controlling robotic hands [1], [2]. In the context of real-time interaction, [3] uses a Finite Element approach and examines a method to compute a grasp quality metric which represents the ability of a grasp to maintain relative object position in the face of disturbances. To resolve contact forces for haptic feed-

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.

*Short Communications proceedings ISBN 80-86943-05-4
WSCG'2006, January 30-February 3, 2006,
Plzen, Czech Republic.
Copyright UNION Agency-Science Press*

```

foreach time step  $t$  do
  Free Motion()
  Collision Detection()
  LCP Construction()
  Gauss Seidel()
  Constrained Motion()

```

Figure 1: Overview of our framework

back, [4] and [5] use Finite Element Methods. [6] propose a method based on a mass-spring model to compute repulsive forces for each finger of a user wearing a Cybergrasp force-feedback glove.

The surface deformation can be simulated with a fully deformable model, but it can be prohibitive since the size of the system to solve for contact resolution is large, even for small deformations. This method is not well adapted for modelling fingertips, because the surface around the deforming contact area stay intact. Instead, we use quasi-rigid object introduced by [9]. This model combines a rigid model for dynamic simulation and a deformable model for local contact resolution with friction and surface deformations. Resolving the contact problem exactly is a difficult problem because we do not know if a possible contact point on the surface, will stay in contact or not, depending on the boundary conditions and on the other points in the neighbourhood. We use a Linear Constraint Problem form (LCP) to solve the contact space [9], [4] and [10]. Point-primitives, which has become popular in recent years [7], [9], are adapted to conforming contact, as both interacting objects share contact points that have zero separation. In this context, we must also adapt the collision detection algorithm. [8] presents an approach with bounding volume hierarchy, but they do not adress the problem of contact response. [11] presents an efficient collision detection and a response based on penalty force computation.

3 OUR METHOD

3.1 Overview of Our Approach

Figure 1 gives an overview of our collision detection and response framework. We use a constant time step integration called time-stepping method. The *Free Motion* will integrate the forces applied on fingertips by the user through the motion capture device, or forces from control laws for manipulation tasks. Other forces like gravity, but not contact forces, are also integrated. Then the *collision detection* algorithm gives contact normals and geometric configurations of the contact space (section 3.3). We use a *LCP formulation* and an algorithm close to the *Gauss Seidel* method [4] to solve the contact forces with friction such that the

displacement of control points will deform the surfaces not to interpenetrate (sections 3.4 and 3.5). This is called the Signorini problem. The Gauss Seidel algorithm is easy to implement and robust to converge toward a solution even if several solutions are possible. Finally contact forces are integrated to find the *Constrained Motion*.

3.2 Surface Definition

Fingertips and grasped objects are modelled by an unstructured set of points. Given a point cloud with N primitives p , the surface is defined as the zero set S of an implicit function f using the Weighted Least Square (WLS) method which we breafly recap [7], [8]

$$S = \{p | f(p) = 0\} \quad (1)$$

$$f(p) = n(p) \cdot (a(p) - p)$$

where $n(p)$ is the normal vector and $a(p)$ is the weighted average of all points at p :

$$a(p) = \frac{\sum_{k=0}^N \theta(\|p_k - p\|) p_k}{\sum_{k=0}^N \theta(\|p_k - p\|)} \quad (2)$$

We choose a truncated Gaussian as weighted function $\theta(x) = e^{-x^2/h^2}$ and h is the bandwidth of the kernel which tune the decay of the influence of points. The normal $n(p)$ is defined by moving last square and is exactly the smallest eigenvector of matrix B with:

$$b_{ij} = \sum_{k=0}^N \theta(\|p_k - p\|) (p_{k_i} - a(p)_i) (p_{k_j} - a(p)_j) \quad (3)$$

3.3 Collision Detection

Given two point clouds A and B , we need an approximation of the interpenetration distance for collision response. We use the same approach as in [11]. Given a point p_A on the surface S_A of the object A , we project this point onto S_B yielding the projected point $p_{A,B}^{proj}$. The projection operator is described in [7]. We then define p_A as intersecting with S_B if:

$$(p_A - p_{A,B}^{proj}) \cdot n_{p_{A,B}^{proj}} < 0 \quad (4)$$

To reduce the number of primitives to be tested, we use a bounding volume (BV) hierarchy for collision detection, which is a very efficient data structure for time critical application [8], [12]. We choose to implement a binary tree with a sphere as BV. Each node stores a sample of points underneath, which yields different levels of detail of the surface, and the leaf nodes must achieve a hole-free coverage of the surface, to avoid missed collisions. The traversal of the hierarchies is inspired by [8]. After this step, we obtain a set of coupled nodes which overlap. To accelerate

calculation versus accuracy, we project the center of each leaf node, not all the primitives contained in this node, on the surface in the node coupled. If the number of primitives in the coupling node is too small for correctly computing $a(p)$ and $n(p)$, we take into account points in a given border of this node (called r_ϵ -border in [8])(figure 2).

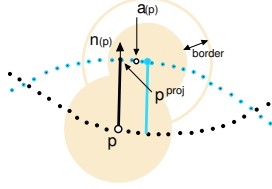


Figure 2: Estimation of the coupling interpenetration distances.

The bandwidth of the projection operator is chosen as the distance between the center of the two nodes. To avoid artefacts that can appear in high curvature region, we don't take into account cases when the projection point is not situated inside the border of the coupling node.

At each time step, the overlapping surfaces are computed after the Free Motion step, so in an undeformed state. However, because we deal with deformable models, collisions could be potentially missed even for small deformations. So we must update the BV hierarchy, whereas this operation is often very costly. We notice that the updated hierarchy is used to obtain a set of overlapping nodes, not to compute the interpenetration distance for collision response. This update is done by Bottom-Up strategy, but to accelerate this operation, we stop the process when we reach a given level in the hierarchy.

3.4 Linear Complementarity Formulation

Given the set of active points $\{(p, q) \mid p \in S_A, q \in S_B\}$ on the surface of two objects, we must find the contact forces acting on this points such that the displacement will deform the surfaces not to interpenetrate (figure 3).

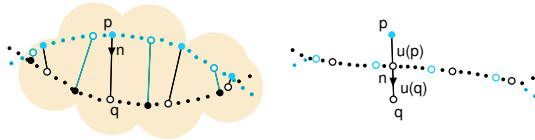


Figure 3: On the left, active points (p, q) obtained with collision detection algorithm, on the right, displacement (u_p, u_q) to obtain zero gap.

This problem is solved using the method introduced by [4]. We choose an arbitrary direction for the con-

tact normal n and in this direction the force f_p^n is positive : $f_p = -f_q = f^n n$ and we define the gap : $\delta_p^n = (p - q) \cdot n$. Points in the contact space are chosen in the set of points determined by the collision detection with $\delta_p^n < \delta_\epsilon$ and $\delta_\epsilon > 0$. A positive threshold δ_ϵ permits to introduce in the contact space the neighborhood area of contact, which could penetrate during the step of contact resolution. To solve the contact forces, we use a linear complementarity problem LCP formulation [4], [9]. For each potential contact point, we can write :

$$0 \leq f^n \perp \delta^n \geq 0 \quad (5)$$

To linearize Signorini's problem, the contact space is frozen during the current time step, called the free motion. With the collision detection detailed above, we can compute the value of each gap, denoted $\delta^{n, free}$, and the contact normal without contact forces. Deformation displacements u_p and u_q are defined between the free motion and the constrained motion, after solving Signorini's problem and the integration of the contact forces :

$$\delta_p^n = (u_p - u_q) \cdot n + \delta_p^{n, free} \quad (6)$$

Displacements are then first calculated at a coarser level (set of points (p, q)) and propagated to the surface using interpolation, allowing us to accelerate calculation versus accuracy. For visual accuracy, the normal vector of each contact point is recomputed using the method introduced in section 3.1.

3.5 Contact Resolution

3.5.1 Quasi-Rigid Model

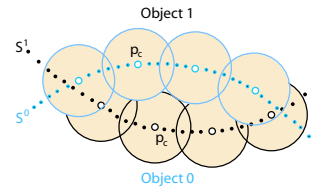


Figure 4: Control points p_c on the interpenetrating surfaces S^0 and S^1 . They are chosen as the centers of the leaf nodes in the BV hierarchy.

We use the approach of quasi-rigid model introduced by [9], which splits the global motion, driven by a rigid model, from a local relative displacement driven by a linear deformable model. The deformation of the surface points can be linearized to $\tilde{K}u = f$ where \tilde{K} is analogous to a stiffness and depends on the intrinsic properties of the object. To obtain deformation compliance $C = (\tilde{K})^{-1}$, we can use a model based on continuum mechanics for mesh free [13], or a local model [9] [14]. We choose the second type

to speed up the calculation of compliance, but this contact resolution method is still valid for more complex deformation models. The deformation is simulated by adding compliance on N control points p_c on the surface. They are chosen as the centers of the leaf nodes in the BV hierarchy (figure 4). A linear relation links the constrained control node displacements $u_{p_c} = [u_{p_{c_0}} \cdots u_{p_{c_N}}]^T$ and the contact forces $f_{p_c} = [f_{p_{c_0}} \cdots f_{p_{c_N}}]^T$:

$$u_{p_c} = [C_{p_c}] f_{p_c} + u_{p_c}^{free} \quad (7)$$

The displacement of a contact point p is interpolated from the controlled points by :

$$u_p = \frac{\sum_{i=0}^N \theta(\|p - p_{c_i}\|) u_{p_{c_i}}}{\sum_{i=0}^N \theta(\|p - p_{c_i}\|)} \quad (8)$$

where $\theta(x)$ is a gaussian weighted function. We can equivalently interpolate the unknown contact force at p_{c_i} , with the force f_p^n obtained in the contact space. By stacking relations (6), (7) and (8) for each contact on the surface S_i for a couple of objects, we can write matrices $[H^{S_i}]$ such that :

$$\delta_p^n = \underbrace{\left(\sum_i [H^{S_i}] [C^{S_i}] [H^{S_i}]^T \right)}_W f_p^n + \delta_p^{n,free} \quad (9)$$

W is named Delassus operator [15]. See [4] for more information. To simulate friction, Coulomb's law describes the behaviour in tangent contact space :

$$\begin{aligned} \delta_p^t = 0 &\Rightarrow \|f^t\| < \mu \|f^n\| \\ \delta_p^t \neq 0 &\Rightarrow \|f^t\| = -\mu \|f^n\| \frac{\delta_p^t}{\|\delta_p^t\|} \end{aligned} \quad (10)$$

Then we linearize the system along two tangential directions, f^{t_1} and f^{t_2} :

$$\begin{bmatrix} \delta^n \\ \delta^t \end{bmatrix} = \begin{bmatrix} W_{nn} & (W_{nt})_{(1 \times 2)} \\ (W_{tn})_{(2 \times 1)} & [W_{tt}]_{(2 \times 2)} \end{bmatrix} \begin{bmatrix} f^n \\ f^t \end{bmatrix} + \begin{bmatrix} \delta^{n,free} \\ \delta^{t,free} \end{bmatrix} \quad (11)$$

We use a Gauss-Seidel like algorithm to solve Signorini's and Coulomb's laws with a guaranteed convergence [4], [16].

As the quasi-rigid approach [9] splits the global motion from a local linear relative displacement, we can sum up these two models in compliance within the contact space. The equation (9) can be rewritten as

$$\delta_p = [W_{rigid} + W_{deform}] f_p + \delta_p^{free} \quad (12)$$

When the stiffness increases, the behavior tends to the rigid motion.

3.5.2 Deformable Model

A displacement $u_{p_{c_i}}$ of a control point on the surface at p_{c_i} induced by a force $f_{p_{c_j}}$ acting at p_{c_j} is given by

$$u_{p_{c_i}} = \phi_j(\|p_{c_i} - p_{c_j}\|) f_{p_{c_j}} \quad (13)$$

where ϕ a function of influence, which can be physically motivated as the Boussinesq approximation [9]. In our simulation, we choose to introduce a local coupling between points ensuring that the Gauss-Seidel algorithm yields coherent converging solutions in time,

$$\phi_j(x) = gauss(x) C_{p_{c_j}} \quad (14)$$

where $C_{p_{c_j}}$ is the compliance at point p_{c_j} and $gauss(x)$ a gaussian. Since we assume linear elasticity, the total displacement at p_{c_i} is the superposition of contribution of all neighboring points. Using an implicit euler scheme, the compliance include stiffness (K) and damping effects (D): $C_{p_{c_i}} = (K + \frac{D}{\Delta T})^{-1}$. Then we can compute W_{deform} using the equation (9).

3.5.3 Rigid Model

We use the quasi-static generalized form for the rigid model [17] :

$$b(q, \dot{q}) \dot{q} = \Gamma^{ext} + \Gamma^{contact} \quad (15)$$

where q is the vector of generalized degrees of freedom, b the viscosity, Γ^{ext} the resultant of external forces and $\Gamma^{contact}$ the contact forces. [17] introduced a methodology to map the rigid motion space into the contact space, using a jacobian J_c . We can now express the motion of the gaps in the contact space due to a rigid motion :

$$\dot{\delta}_p = [J_c B^{-1} J_c^T] f_p + \dot{\delta}_p^{free} \quad (16)$$

We can obtain the rigid form of the Delassus operator integrated with an implicit euler method :

$$W_{rigid} = \left[J_c \left(\frac{B}{\Delta t} \right)^{-1} J_c^T \right] \quad (17)$$

This model allows for a stable real time simulation with an arbitrary choice of time step.

4 DISCUSSION

We implement this method in C++. As of yet, the implementation is not fully optimized. In the following, all results have been obtained on a 2 Ghz Pentium-IV with 1 Go RAM.

An example for grasping task is shown in Figure 5. A box is maintained due to friction between three fingertips while the gravity force is pulling it down. The

virtual hand is composed of 3 fingers with 4 degrees of freedom for each finger. Each finger is controlled in position with generalized motor torques. The box is composed by 36K sampled points and 4800 control points and each fingertip is composed by 2400 sampled points and 320 control points. Figure 6 and 7 give performance measurements for the detection collision algorithm and the contact resolution method with friction.

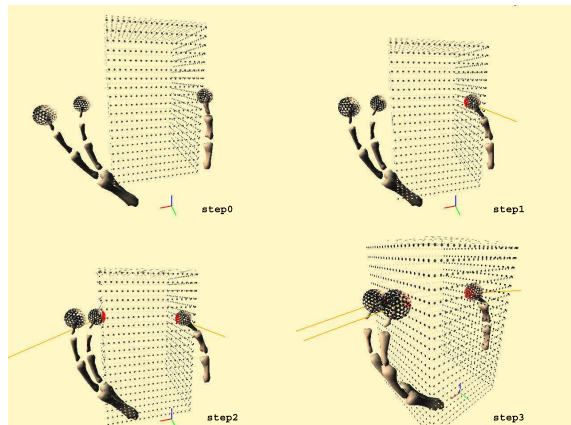


Figure 5: Grasping simulation of a box with gravity. The three fingers are controlled in position. Contact forces are drawn for each finger.

	step1	step2	step3
Traverse BV Hierarchy (ms)	1.170	1.904	2.973
Nb Contacts	60	108	180
Interpenetration Computing (ms)	1.394	2.376	4.100
Hierarchy Update (ms)	2.468	2.717	2.549
Total Time (ms)	5.279	7.382	10.200

Figure 6: Collision Detection computing time for each step of the simulation.

	step1	step2	step3
Nb Contacts	56	105	164
Fill W_{Rigid} (ms)	8.838	16.560	25.030
Fill W_{Deform} (ms)	1.159	3.140	7.350
Gauss Seidel Iterations	4	4	5
Gauss Seidel Resolution (ms)	0.409	2.061	5.410
Total Time(ms)	14.150	28.374	43.060

Figure 7: Contact Resolution computing time for each step of the simulation.

There are many factors influencing the performance. For large number of colliding points, the estimation of the different coupling distance has the more computational cost. Significant gain could be achieved exploiting the temporal coherence during collision detection.

Using this particularity, different temporal resolution could be used for the different part of this method, which can be easily parallelized.

Like many discrete collision detection algorithm, our approach is susceptible to fail for deep penetrations, where bounding volumes of the hierarchy don't overlap. Furthermore the computation of the contact forces depends on the accurate estimation of the contact normal direction, especially for surface with high curvature like edges. Some artefacts can appear because our intersection resolution algorithm in the leaf nodes and the local deformation model are not well adapted in that case.

The Gauss Seidel algorithm is well suited for interactive application and robust to converge. Indeed convergence is reached for a hundred contacts in less than 2ms. To another part, the building of the Delassus operator dominates the computational overhead. We expect substantial speedups by reusing parts of the computation over multiple time steps.

There are many avenues for further work. First, we can use our contact resolution as a framework to test more advanced friction models. In fact, the geometry of the contacting surface and the friction model have an important influence on how the grasping simulation evolves. That's why more complex control laws are useful for computing non-sliding contact forces to move the object on a given consign position.

We expect that this system can be coupled with an optical motion capture system (ART [18] Advance Realtime Tracking). The fingertips are linked to trackers with damped springs, following the orientation of the hand and the position of the three fingers.

5 CONCLUSION

In this paper, we present a method to use point-primitives for modelling fingertips in interactive grasping tasks. We also present an exact contact resolution with friction between quasi-rigid objects. Thanks to Gauss Seidel algorithm and the use of Delassus operator, this approach is suitable for real time application. More advanced physical models could be integrated in our system such as advanced friction models. This model can be efficiently combined with an optical motion capture system for real time grasping task. The improvement of this kind of application is still in progress.

6 ACKNOWLEDGMENTS

We would thanks Christian Duriez for his help with the Delassus operator and the Gauss Seidel algorithm formulation.

References

- [1] N. Xydias and I. Kao, Modeling of contact mechanics with experimental result for soft fingers. In *IEEE Intl. Conference on Intelligent Robots and Systems*, pp. 488-493, 1998.
- [2] C. H. Xiong, M. Y. Wang, Y. Tang, and Y. L. Xiong, Compliant grasping with passive forces. In *Journal of Robotic Systems*, 22(5), pp. 271-285, May 2005.
- [3] M. Ciocarlie, A. Miller and P. Allen, Grasp analysis using deformable fingers. In *IEEE Intl. Conference on Intelligent Robots and Systems*, Aug. 2005.
- [4] C. Duriez and C. Andriot, A multi-threaded approach for deformable/rigid contacts with haptic feedback. *Intl. Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems*, 2004.
- [5] G. Picinbono, J. Lombardo, H. Delingette and N. Ayache, Improving realism of a surgery simulator : Linear anisotropic elasticity, complex interactions and force extrapolation. In *J. Visualization and Computer Animation*, 2002.
- [6] A. Maciel, S. Sarni, O. Buchwalder, R. Boulic and D. Thalmann, Multi-finger haptic rendering of deformable objects. In *Proc. of Eurographics Symposium on Virtual Environments*, 2005.
- [7] A. Adamson and M. Alexa, Approximating and intersecting surfaces from points. In *Proc. of Eurographics Symposium on Geometry Processing SPG 03*, pp. 230-239, June 2003.
- [8] J. Klein and G. Zachmann, Point Cloud Collision Detection. In *Computer Graphics Forum (Proc. of Eurographics 2004)*, 23, 3, Grenoble, France, pp. 567-576, Sep. 2004.
- [9] M. Pauly, D. Pai and L. Guibas, Quasi-Rigid Objects in Contact. In *ACM Siggraph/Eurographics Symposium on Computer Animation 2004*.
- [10] D. Baraff, Fast contact force computation for non penetrating rigid bodies. In *Proc. of Siggraph*, pp 23-34, 1994.
- [11] R. Keiser, M. Müller, B. Heidelberger, M. Teschner and M. Gross, Contact Handling for Deformable Point-Based Objects. *Proc. of Vision, Modeling, Visualization VMV'04*, Stanford, CA, Nov. 2004.
- [12] G. Van Den Bergen, Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools* 2, pp. 1-14, 1997.
- [13] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross and M. Alexa, Point-Based Animation of Elastic, Plastic, and Melting Objects. *ACM Siggraph/Eurographics Symposium on Computer Animation 2004*.
- [14] P. Song, J.S. Pang and V. Kumar, A semi-implicit time-stepping model for frictional compliant contact problems. *Intl. Journal of Robotics Research*, 60, pp. 2231-2261, 2004.
- [15] J-J. Moreau and M. Jean, Numerical Treatment of contact and friction : the contact dynamics method. *Engineering Systems Design and Analysis*, 4:201-208, 1996.
- [16] F. Jourdan, P. Alart and M. Jean, A gauss-seidel like algorithm to solve frictional contact problems. *Computer Methods in Applied Mechanics and Engineering*, pp. 33-47, 1998.
- [17] D. Ruspini and O. Khatib, Collision/Contact Models for the Dynamic Simulation of Complex Environments *IEEE/RSJ IROS'97*, Sept. 1997.
- [18] www.ar-tracking.de

Face-Based Perceptual Interface for Computer-Human interaction

Cristina Manresa-Yee, Javier Varona and Francisco J. Perales

Universitat de les Illes Balears

Departament de Matemàtiques i Informàtica

Ed. Anselm Turmeda. Crta. Valldemossa km. 7.5

07122 Palma

{cristina.manresa, javier.varona, paco.perales}@uib.es

ABSTRACT

Nowadays accessibility to new technologies for everyone is a task to accomplish. A way to contribute to this aim is creating new interfaces based on computer vision using low cost devices such as webcams. In this paper a face-based perceptual user interface is presented. Our approach is divided in four steps: automatic face detection, best face features detection, feature tracking and face gesture recognition. Using facial feature tracking and face gesture recognition it is possible to replace the mouse's motion and its events. This goal implies the restriction of real-time response and the use of unconstrained environments. Finally, this application has been tested successfully on disabled people with problems in hands or arms that can not use the traditional interfaces such as a mouse or a keyboard.

Keywords

Visual Tracking, Gesture Recognition, Perceptual User Interfaces, Assistive Technologies, e-Inclusion, Information and Communication Technologies, Human-Computer Interfaces

1. INTRODUCTION

Nowadays accessibility to new technologies for everyone is a must-do task to accomplish. One way of achieving this aim is to provide new natural user interfaces at a low cost. Perceptual User Interfaces (PUIs) offer a more natural human-computer interaction (HCI) by means of trying to "perceive" what the user is doing using computer vision, speech recognition and/or sketch recognition [Lan02a].

The use of computer vision techniques for creating

perceptual interfaces is a growing field in research because they offer great advantages over other systems: non intrusive devices on the user and hand-free control.

These kind of systems are oriented to take into account the needs of all users, including those with disabilities. Assistive technologies (AT) and e-Inclusion have become a major driving force for an open information society. Therefore, these interfaces can be very useful for disabled people with problems in hands or arms or it can be used for leisure purposes.

Taking advantage of the benefits that vision-based interfaces yield and using a low cost standard webcam, a system for replacing the traditional devices for computer interaction such as a mouse or a keyboard can be implemented.

After an accurate bibliographic revision, some systems are being already commercialized and other more sophisticated solutions are in research phases. These approaches differ in the selection of the facial feature to track. Moreover, they don't include gesture recognition in their solutions. Toyama's work

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.

SHORT COMMUNICATION proceedings
ISBN 80-86943-05-4
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

[Toy98a] uses the whole face, Crea Ratón Facial [Cre05a], Tash CameraMouse™ [Bet02a][Tas05a] and Nouse™ [Gor02a], use the nose, and EyeTech Digital System Quick Glance [Eye05a] uses the gaze and infrared illumination.

The system presented here works with a standard webcam and non attached devices on the user. It is based on face features tracking and face gesture recognition. The system aims to act as a human-computer interface replacing the mouse's actions and with a complementary graphical keyboard, the user can interact with the computer completely hands-free. By means of the nose tracking the mouse's position is carried out and the mouse's events can be achieved by gesture recognition.

Important requirements to take into account are that the system has to work in real-time, under unconstrained backgrounds and with conventional light conditions. Furthermore, the quality of the images is poor due to the use of webcams.

The paper is organized as follows. Section 2 describes the system's architecture, the face detection, the face features selection, the features tracking and smoothing and the gestures recognition. Section 3 presents the performance evaluation and the last section concludes this paper.

2. FACE-BASED PERCEPTUAL USER INTERFACE SYSTEM

System Architecture

The application presented in this paper uses the nose region as facial feature for tracking. The nose region is the selected feature because it is visible for everyone in all face positions facing the computer screen, fact that does not occur always with other features that can be occluded by beards, moustaches or glasses.

In figure 2.1, a graph of the system process can be seen.

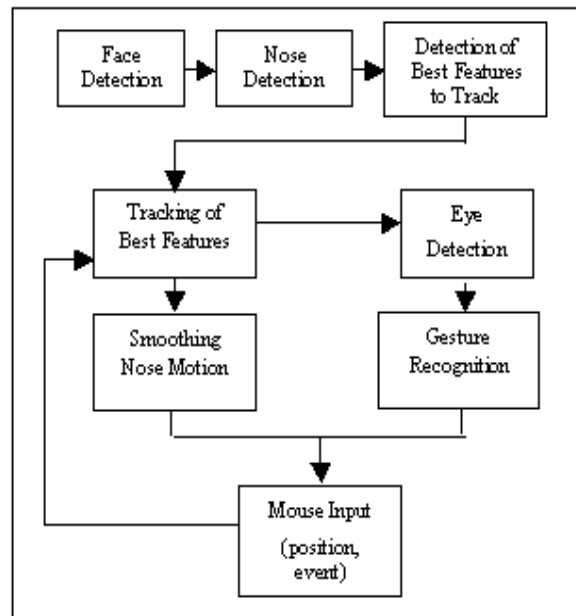


Figure 2.1 System Process

Face Detection

The system automatically detects the face by means of Viola and Jones algorithm [Jou04a], which does not require any previous calibration process. This approach slides a window across the image and applies a binary classifier that discriminates between a face or the background. This classifier is trained using a boosting machine learning meta-algorithm [Vio01a].

When a face is detected, for it to be accepted as a face and for continuing with the image process, the user must keep it steady for a number of frames. When the user executes the system, he has to place himself correctly and steady in front of the screen and the webcam. In figure 2.2, correct and incorrect face detection examples are shown.

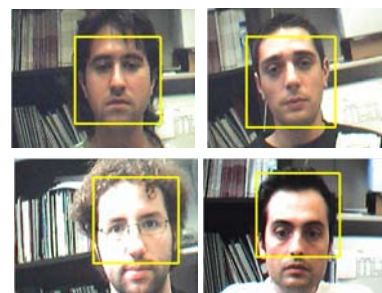


Fig. 2.2 Face detection examples: upper row correct; down row incorrect.

The face bounding box is defined as $F = \{\vec{X}_F, \vec{S}_F\}$, where \vec{X}_F is the rectangle's origin point and \vec{S}_F is the rectangle's size.

Best Face Features Detection & Tracking

Once the face is detected, the system looks for the nose section. Based on anthropometrical face measurements, the nose is found in approximately the second third of the face. The nose rectangle is defined as $N = \{\vec{X}_N, \vec{S}_N\}$ where the nose rectangle origin is $\vec{X}_N = (\vec{X}_F + \frac{\vec{S}_F}{3})$ and the nose rectangle size is $\vec{S}_N = (\frac{\vec{S}_F}{3})$.

Over the nose rectangle, the best features to track are selected controlling those points for which the derivative energy perpendicular to the prominent direction is above a threshold. This typically selects corners in images [Shi94a].

When applying the algorithm, due to the illumination, some points can appear and not be in the nose corners. The real position point of the mouse to track will be a mean of all the features found, due to this, the selected features to track should be in both sides of the nose and symmetrical, because this point should be as centred as possible.

With the points found with the algorithm, the final list of points $\vec{P} = (\vec{p}_1, \vec{p}_2 \dots \vec{p}_n)$ to track, where n is the number of features and $\vec{p}_i = (px_i, py_i)$, is created considering symmetry constraints. In figure 2.3, the features found applying directly the Shi & Tomasi algorithm and the final list of features that will be considered due to their symmetry respect to the vertical axis are shown .

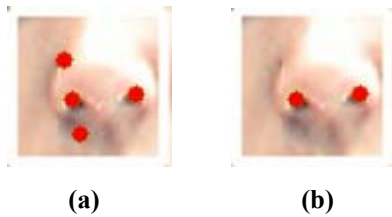


Fig. 2.3 Best Features to Track (a) Shi and Tomasi best features to track (b) Enhancement of the features for our purposes

For tracking the selected feature points, the spatial intensity gradient information of the images is used for finding the best image registration [Jou04b]. The Lucas-Kanade algorithm is robust for handling rotation, scaling and shearing, so the user can move

in a more flexible way. However, due to illumination changes or a fast movement, the length between a particular feature to track and the mean of the features can be greater than the length of the nose rectangle diagonal. If this occurs, this feature is not tracked anymore, as only the points beneath the nose region are in the region of interest.

Smoothing the Nose Motion

When the user wants to move the mouse position to a particular place, there is a tendency in the direction of the movement. For avoiding jitter and for smoothing the movement, it is applied a linear regression method to a particular number of tracked nose positions through consecutive frames. The computed nose points of several consecutive frames, s , are adjusted to a line, and therefore the mouse's motion can be carried out over that line direction.

Additionally, for avoiding discontinuity problems between the estimations of the nose positions at each time step, previous estimated points are considered for computing the current frame nose position.

When the new smoothed point for the frame t , n_t is calculated, the system calculates the horizontal distance and the vertical distance in pixels between n_t and the previous nose point estimation.

$$Dx(n_t) = (nx_t - nx_{t-1}),$$

$$Dy(n_t) = (ny_t - ny_{t-1}),$$

Finally, these distances will be used in the computation of the final screen coordinates S for the new point n_t

$$S_x = V_x + Dx(n_t) \cdot Kx$$

$$S_y = V_y + Dy(n_t) \cdot Ky$$

where $\vec{V} = (V_x, V_y)$ is the previous mouse position point in the computer screen and Kx and Ky are predefined constants

Finally, the computed mouse screen coordinates \vec{S} are sent as real mouse inputs for placing the cursor in the desired region.

Gesture Recognition

The PUI can be enhanced including gesture recognition for replacing the different clicks of the mouse, for example, by means of eye blinking detection. This is a very difficult task due to the poor quality of the images and no use of infrared lights.

The eye blink detection is based on finding the iris. That is, if the iris is detected in the image the eye will be considered as open, if not, the eye will be considered closed. For achieving this aim, a region of interest, ROI, is calculated for including the eyes' region. This eyes' ROI is proportional to the face dimensions and it avoids including hair or the face frontiers. Then the eyes' ROI is segmented by a threshold for finding the dark zones. After that, the Sobel operator is used for detecting the vertical edges of the eyes' ROI (searching only for vertical contours eliminates the eyebrows if they were included in the image). The two sides of the eye are the ones with the two longest vertical edges. The left eye or right eye is distinguished controlling the centre of the image. In figure 2.4 the eye blinking detection process is shown.

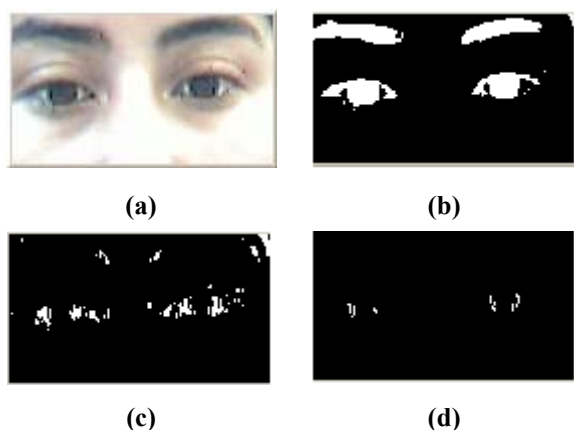


Fig 2.4 (a) Original image, (b) Thresholding results, (c) Sobel contours, (d) Eye vertical edges

If the two vertical edges of an eye don't appear in the image for a number of frames (for gesture consistency) the system will assume that the eye has been closed and will send the mouse event associated.

Some examples of the blink recognition are shown in figure 2.5.

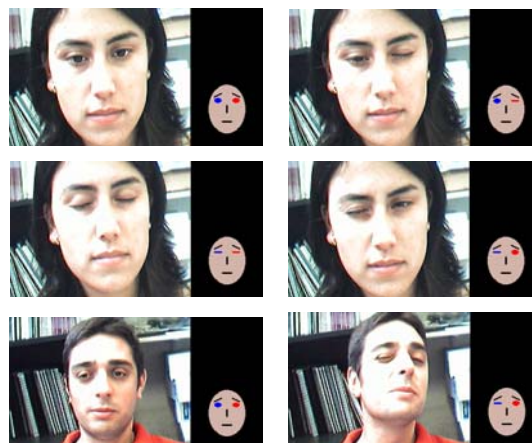


Figure 2.5 Examples of eye blinking detection

3. SYSTEM'S PERFORMANCE EVALUATION

In this section we show the accuracy and robustness of the nose tracking system. The application is implemented in Visual C++ using OpenCV libraries. The images were captured with two webcams: a Genius VideoCAM Express USB Internet Video Camera and with a Logitech QuickCam Messenger. The cameras are not assumed to be calibrated and they provide 320x240 images at a rate of 25 frames per second. The computer configuration was a Pentium IV, 3.2 GHz, 1GB RAM. Although the system has been tested on machines with less resources and it worked fine.

The user placement is very important too. The user was seating in a comfortable position without stretching his neck or forcing a strange pose. The webcam was placed on the computer screen at a forehead height. See figure 3.1



Figure 3.1 Correct user placement

In order to prove the robustness, the user was asked to rotate and move his head always facing the computer screen and therefore the webcam too. The results were that the user could move in a quite wide

range without losing the features to track. In figure 3.2 images of the range of motion can be seen. Although the system is quite robust, fast movements or illumination changes can cause the loss of the good features to track, in this case, the system is re-initialized for detecting the face and the best features to track.



Figure 3.2 Face Motion Range

Two datasets were implemented for evaluating the accuracy of the application and users with different skills tested the system. The dataset A, a 4 x 4 point grid was presented in the computer screen. Each point has a radius of 15 pixels. Fig. 3.3 illustrates the pattern. This test was performed on a set of 13 persons without any training. The user had to try clicking on every point and distance data between the mouse's position click and the nearest point in the grid was stored.

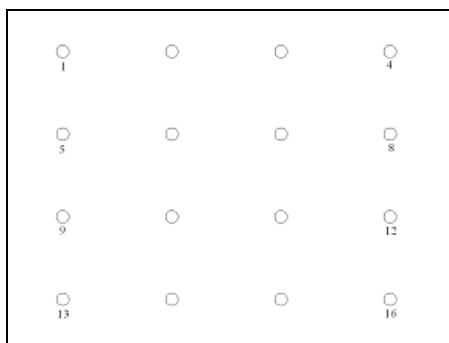


Figure 3.3 Dataset A: a point grid of 4x4, where the circle radius is 15 pixels

In figure 3.4 it is shown the percentage of error caused by clicks on incorrect positions for the first set of users without training. The graph shown in figure 3.5 represents the mean distance in pixels of the errors originated trying to click on a point for the users without experience.

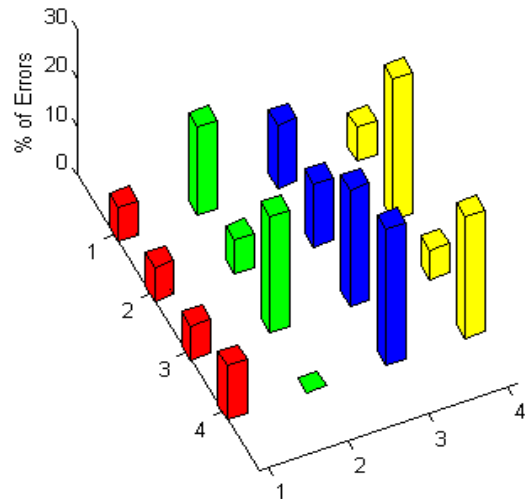


Figure 3.4 % of errors of Test 1: non-trained users

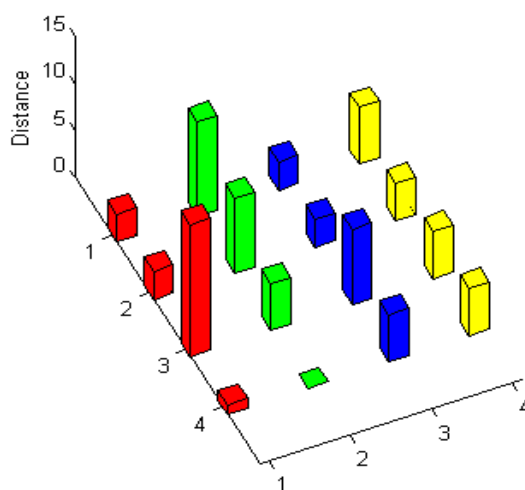


Figure 3.5 Distance errors (in pixels) of Test 1: non-trained users

In figures 3.4 and 3.5 it is shown that a person without previous training can place the screen cursor at the desired position in the main number of cases. Besides, the distance error is related with the circle position, that is, it augments when the user tries to click near the screen boundaries.

Next, the following experiments will show how after a short training period, the user improves fast his skills. Therefore the results improve. To show this fact a dataset B, a 5 x 5 point grid was presented in the computer screen. Each point has a radius of 15 pixels. Figure 3.6 illustrates the pattern.



Figure 3.6 Dataset B. A point grid of 5x5. The circle radius is 15 pixels

This second test was performed on a set of 9 persons that already had used the system several times. The graphs shown in figures 3.7 and 3.8 illustrate how the number and distance of errors decrease dramatically (users only failed once). Besides, in this case (with trained users), there is not a relation between the screen positions and the accuracy of the system. However, in order to state this assumption this test should be completed with more persons.

Finally, to point out, the continuous use of this system produced some neck fatigue over several users. Therefore, some errors clicking the point grid could be caused due to this reason.

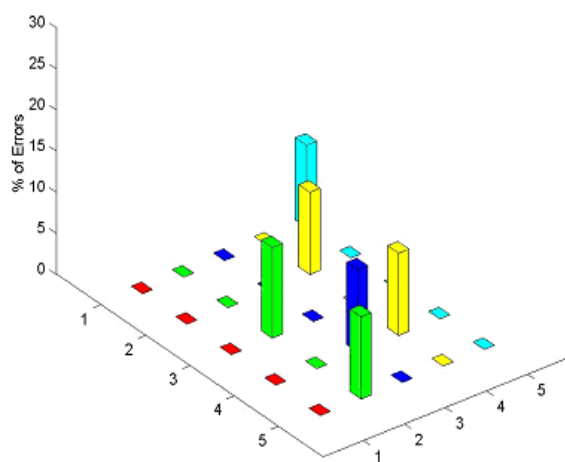


Figure 3.7 Errors of Test 2: trained users.

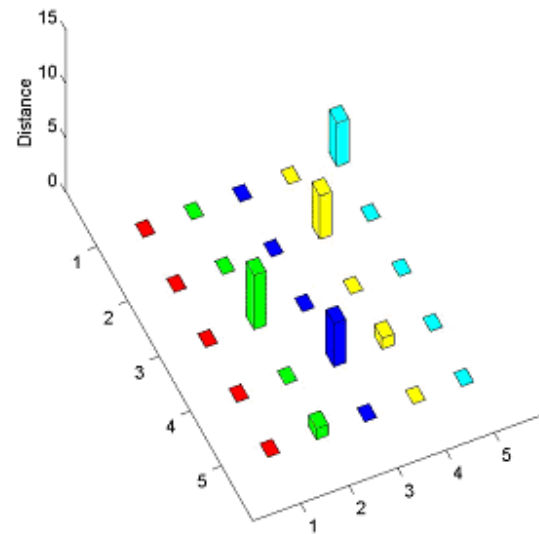


Figure 3.8: Distance errors (in pixels) of Test 2: trained users.

4. CONCLUSIONS AND FUTURE WORK

In this paper a real-time vision-based perceptual user interface has been presented for interacting with the computer. A system based on computer vision algorithms has been proposed. To build this system algorithms for face detection, best feature selection and tracking and face gesture recognition have been used.

The system's performance evaluation results have shown that the system can replace a traditional mouse with a low-cost interface such as a webcam with efficiency and robustness. Besides, the experiments have confirmed that continuous training of the users results in higher skills and, thus, better performances and accuracy for controlling the mouse position.

The system's disadvantages are that the environment has to be controlled regarding to illumination conditions and the user must be capable of moving the neck in quite a wide range.

As future work a more exhaustive evaluation of this system remains. Besides, it is planned to extend the gesture recognition to other face gestures for a more rich interaction with the computer. Therefore, it will be possible to obtain a MPEG4 compliant face description to build and control a user avatar for enhanced communication between users.

5. ACKNOWLEDGMENTS

This work has been subsidized by the national project TIN2004-07926 from the MCYT Spanish Government and TAGrv S.L., Fundación Vodafone, Asociación de Integración de Discapacitados en Red. Javier Varona acknowledges the support of a Ramon y Cajal grant from the Spanish MEC.

6. REFERENCES

- [Bet02a] Betke, M., Gips J., Fleming, P. The Camera Mouse: Visual Tracking of Body Features to Provide Computer Access for People with Severe Disabilities, IEEE Transactions on neural systems and Rehabilitation Engineering, vol 10, n°1, 2002
- [Cre05a] <http://www.crea-si.com/esp/rfacial.html>
- [Eye05a] <http://www.eyetechds.com/qglance2.htm>
- [Gor02a] Gorodnichy, D.O., Malik, S. , Roth, G. Nouse 'Use Your Nose as a Mouse' – a New Technology for Hands-free Games and Interfaces.
- [Jou04a] Viola, P., and Jones, M. Robust Real-Time Face Detection. International Journal of Computer Vision 57(2), pp.137-154, 2004
- [Jou04b] Baker, S. and Matthews, I. Lucas-Kanade 20 Years On: A Unifying Framework. International Journal of Computer Vision 56(3), pp.221-225,2004
- [Lan02a] Landay, J.A., Hong, J.I., Klemmer, S.R., Lin, J., Newman, M. W. Informal PUIs: No Recognition Required. AAAI 2002 Spring Symposium: Sketch Understanding Workshop. pp. 86–90. 2002
- [Shi94a] Shi, J., and Tomasi, C. Good Features to Track. IEEE Conference on Computer Vision and Pattern Recognition. Pp.593-600, 1994
- [Tas05a] http://www.tashinc.com/catalog/ca_camera_mouse.html
- [Toy98a] Toyama, K. Look, Ma – No Hands!”Hands-Free Cursor Control with Real-Time 3D Face Tracking. Proc. Workshop on Perceptual User Interfaces. pp. 49–54, 1998.
- [Vio01a] Viola, P. and Jones, M. Rapid Object Detection Using a Boosted Cascade of Simple Features. IEEE Conference on Computer Vision and Pattern Recognition, volume 1, pp. 511–518, 2001.

Attention-Based Target Tracking for an Augmented Reality Application

Morgan Veyret
European Center for Virtual Reality,
France
veyret@enib.fr

Eric Maisel
European Center for Virtual
Reality, France
maisel@enib.fr

ABSTRACT

When visiting an aquarium, people may be disturbed or, at least, disappointed by the amount and diversity of available information. Moreover, one can find it very difficult to match the information of notices on the wall to the reality of the fishes. Therefore, we propose a virtual guide, an autonomous teaching assistant embodied in the real world using augmented reality techniques, for helping people in their visit of aquariums. This virtual guide will interact with the real world using multiple modalities (e.g. speech, facial expression, ...). Thus, it should be aware of the aquarium's state and content, and use perceived information and prior knowledge to inform the visitor in a structured fashion. Due to the high mobility and unpredictable behaviour of the fishes, our guide requires an adequate perception system. This camera-based system has to keep track of the fishes and their behavior. It is based on the focalisation of visual attention that allows to select interesting information in the field of view. This is achieved by extracting a number of focuses of attention (FOA) using a saliency map and a multi-level memory system, which is filled (or updated) with the extracted information. It allows our system to detect and track targets in the aquarium. This article describes how we use the saliency map and memory system, along with their interactions, to set up the first part of our perception system.

Keywords: Augmented Reality, Virtual Guide, Autonomous, Perception, Computer Vision, Tracking, Saliency Map, Memory

1 INTRODUCTION

When visiting an aquarium, matching notices on the wall with the content of the aquarium can be very difficult. Even if one can find the appropriate notice for a specific fish, when looking back to the aquarium, this fish may have moved or be hidden. The attractiveness and pedagogical goal of the aquarium may suffer from these problems.

To solve such a problem, we propose to use a virtual guide to help people during their visit of an aquarium. This guide will be embodied in the real world using augmented reality techniques¹ and will interact with its environment (see figure 1). It will use several modalities to deliver information concerning the aquarium and its contents to the visitor like speech and facial expressions.

The guide's discourse will have to be structured based on both prior static knowledge concerning the aquarium (and the fishes it contains) and dynamic knowledge of the activity inside the aquarium. To build



Figure 1: The virtual guide embodied in the real world to help people during their visit.

such a discourse, our guide will have to extract information related to its current concerns. As an example, while the guide is talking about a shark, it may be more interesting to find information about other shark or the shark's behavior. On the opposite, if its talk is about to end and another fish, that hasn't been described yet, appears, the guide may use this opportunity to start talking about another topic. The above situations show the need for a specific perception system that can be controlled by our guide while providing information about unexpected events.

We plan to build such a perception system using the focalisation of visual attention in a hybrid model. This model will take into account the two main influences of visual attention (see section 2). Using such an approach, our perception system may show the control and reactivity compromise our guide requires.

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.

SHORT COMMUNICATION proceedings
ISBN 80-86943-05-4
WSCG'2006, January 30 – February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

¹ Projection on a transparent screen

This article describes the first part of the perception system that is able to detect and track targets using the focalisation of attention to select interesting information out of video streams. This is achieved using a simplified classic attention model [30, 18, 14] based on the use of a saliency map in addition to a memory system. The saliency map is used to extract focuses of attention (FOA) according to the encoded importance of visual features. Those FOAs are then used as an input for a multi-level memory system, which is used to store information about targets over time and track them. Here we show how our model works and may be used in the planned guide's perception system based on a hybrid visual attention model.

This document is organized as follow: the first section of this article will describe a few domains our model is connected to, which includes the focalisation of visual attention, as well as target tracking models. The second section describes our work in details while the third shows some results concerning target tracking using the proposed approach on aquarium static ² videos. In the last section, we will consider what remains to be done for the virtual guide application and the perception model in particular.

2 RELATED WORK

A classic model of visual attention is proposed by Koch and Ullman in 1985 [18]. Their architecture describes the attentional process based on two steps: a parallel process followed by a sequential one. The first step is based on features maps computed over the entire field of view and combined in a subsequent saliency map (see figure 2). Then a Winner-Take-All (WTA) algorithm is used to extract the most salient locations in a sequential manner. An *inhibition of return* (IOR) mechanism is set up to avoid multiple selections of the same location. They considered a small number of basic features that have been found to be used in the human visual system like: colors, orientations and contrast. In 1998, Itti *et al.* [14] proposed a similar model of bottom-up (BU) visual attention that showed good results at simulating human focalisation of visual attention. They used center-surround feature computation and a new combination strategy. Later, they proposed a new combination strategy [12] to improve their saliency map model. The proposed strategy is designed to enhance locations that differ from their surrounding. Other BU perception models were proposed: Terzopoulos *et al.* [28] proposed a simulation of virtual fishes and their environment. They used peripheral vision through a number of concentric virtual cameras to model the focus of attention. In [8], Courty and Marchand proposed to use simplified saliency maps

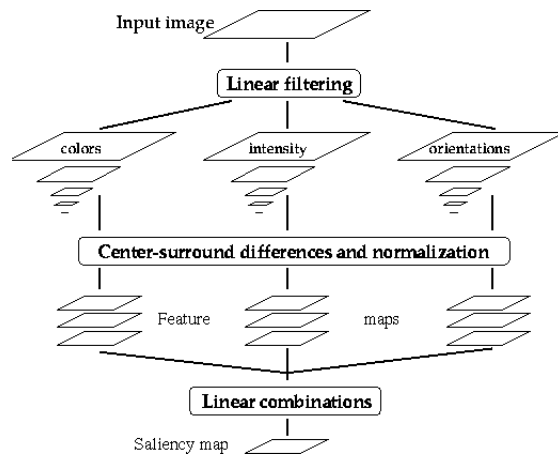


Figure 2: Itti *et al.* saliency map model. Features are extracted and combined across different scales in a center-surround manner. The obtained feature maps are linearly combined to build the saliency map.

to model visual attention and improve the realism of their virtual actor. This was done by making the actor look towards salient location (in 3D space). In such BU models, salient locations are areas that correspond to the most active features considered. This allows a non-negligible reactivity during the perception process. Those classic models describe the focalisation of visual attention in a bottom-up manner and are closely related to Treisman's *Feature Integration Theory* [30] and Wolfe's *Guided Search* [33]. However, Wolfe's model proposed top-down (TD) cueing of the feature maps that used prior knowledge in active search tasks. For example, when looking for a red object, a color feature map can be biased to account only for red color, improving the saliency of red objects in the field of view. In 2002, Itti and Navalpakkam [22] proposed a new visual attention model based on Itti's earlier work [14, 15, 13] integrating TD influence and a memory system. This influence biased the saliency map through the modulation by a so-called "attention map" which was built using prior knowledge contained in memory and concerning the search task as well as recently known information (from a working memory). In 2005 [23], they improved their model with the noticable addition of recognition. Olivia *et al.* [25] also used the notion of saliency map (defined as a probability of feature presence) and its modulation to build an attention model reflecting both BU and TD path. These hybrid approaches take advantages from BU models while enabling active search tasks which requires the control of visual attention.

Hayhoe *et al.* [11] also showed the presence of control in visual attention as well as different memory level ranging from short term (working) memory to an "unlimited" long term one. Several perception systems used memory systems. Kuffner and Latombe [20] used

² Camera's position is fixed

it to store information about perceived objects (position, last time seen, ...) and then used it in an inhibition of return mechanism, which was based on objects rather than classic spatial locations. Noser *et al.* [24] used the notion of visual memory and synthetic vision to model the perception of a digital actor which used this memory to find a path through its environment in an obstacle avoidance problem.

Work have also been done concerning target tracking, mainly for video based surveillance systems [6], robots vision [7] and augmented reality registration [17]. A classic approach is to extract feature points (e.g. markers or corners) and track those points over time [29, 31] based on trajectory and movement assumptions to help reducing matching possibilities. These approaches are known as *Feature Based Tracking*. Burghardt *et al.* used this approach to detect and track animal faces in wildlife footages [3]. Chetverikov and Verestoy [5] used a similar approach to track dense feature point sets. Feature tracking was also used by Coifman *et al.* [6] to detect and track moving vehicles making multiple assumptions like fixed cameras and car's straight trajectory. Other authors used an active contours approach [27] to track 3D objects' pose in 2D space. Another approach is to try to match an area of a frame with a known model [19, 4, 7] (e.g. color distribution [2]). This approach is known as *Model Based Tracking* (or *Region Based Tracking* [21]). Ramanan and Forsyth used such an approach in 2003 [26] to detect and track a human based on a model of its body parts, using bayesian network inference. There was also recent attempts to build a tracking system based on the focalisation of attention [10] to help oceanographic researcher to annotate underwater video sequences.

3 OUR WORK

The proposed model is inspired from biological and psychological studies about visual attention [30, 11], work in computer vision concerning saliency maps [14] and the focalisation of visual attention in general.

3.1 Bottom-up Focalisation of Visual Attention

As we said in the introduction of this document, we want to build a perception system for a virtual guide. This perception system must be able to provide unexpected information as well as requested one. We are looking at creating a tracking system based on visual attention to allow the future guide perception behavior (using an hybrid visual attention model integrating both low-level and high-level influences). Here we only describe a "Bottom-Up" approach that is used to detect and track targets in the aquarium.

Saliency Map Our tracking system is inspired from work by Itti *et al.* [14] on saliency maps (SM). They built a SM model (see figure 2), based on biologically

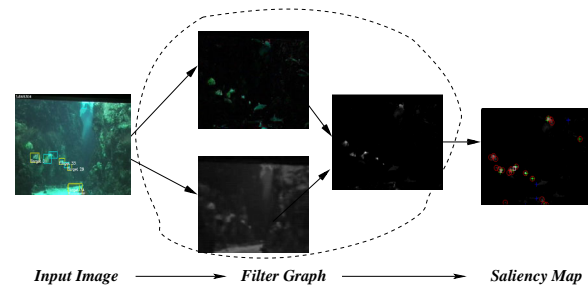


Figure 3: Saliency map creation process: an input image (on the left) is passed through a filter graph (in the middle) representing the creation process and the saliency map is outputted (on the right)

inspired features maps, that showed good results at predicting focus of attention localisation when facing natural and noisy images. Their system was able to predict most of the salient locations a human visual system would have looked at. The saliency map is responsible of encoding the interest of image areas according to considered features (intensity, colors, orientations,...). In [32], we used a simplified saliency map model using biologically inspired features to predict salient locations in a visual attention system designed for understanding road situations. In this model, we proposed a loop where saliency was used to modify entries of a bayesian network and the output of this network was used to modify the saliency map creation process. We wanted the saliency computation system designed for this work to be modular in order to allow modifications of the saliency map creation process (to model the top-down influence through features biasing).

This system is based on the notion of *filter*. A filter is an object that handles image processing operations. Each filter may have a set of parameters that allows us to control its behavior as well as a number of inputs and outputs. Filters may be connected to each other in a directed acyclic graph (DAG) to describe the entire saliency map creation process (see figure 4 and 3). At the top of this graph, there is the input image for which we want to compute saliency and the output node correspond to the computed saliency map.

Different kinds of filters are available like background substraction operator, color filtering (Red, Green, Blue, Yellow) or multiscale pyramid creation. Other filters are combination operators that allow different combination strategies between feature maps like maximum combination (where the maximum value of each input map is kept in the resulting map) or weighted combination.

Using this modular scheme, we are able to build the desired saliency map by describing the filtering sequence as a graph. Resulting map is computed in a backward manner, meaning that each filter requests its

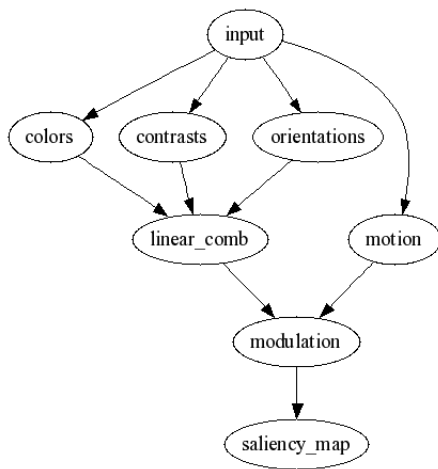


Figure 4: Example of a directed acyclic graph (DAG) that represent one saliency map creation process

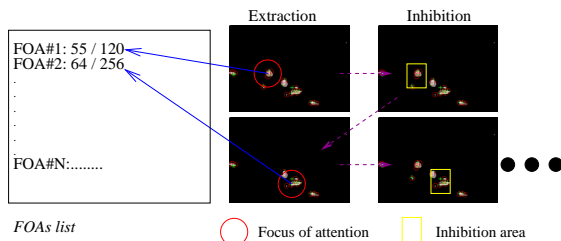


Figure 5: Extraction of focuses of attention with inhibition of return (IOR). The algorithm output a requested number of points by repeating an, extraction and inhibition process.

inputs from its parents in the graph. Then, only required computation is done.

For now, we use a simplified version of saliency maps in order to allow for real-time computations. Hence we only use a limited amount of features (colors, motion, maybe orientations) that are linearly combined into the resulting saliency map. Our modular architecture allows us to experiment multiple saliency map models.

Focalisation Once the saliency map (SM) is computed, it can be used to extract multiple focuses of attention. A focus of attention (FOA) is an area of the field of view corresponding to the most salient location in the saliency map. This focus of attention is defined by a position and a size in the saliency map coordinate system (in pixels). FOAs are extracted from the saliency map by using a Winner-Take-All (WTA) style algorithm. We tried different extraction algorithms. Since we want to extract multiple FOAs, we set up an “inhibition of return” mechanism (like in [14, 9]) to ensure that the same location is not selected multiple times (see figure 5).

3.2 Memory System

The bottom-up attention system described above doesn’t store information over time. Therefore it is not

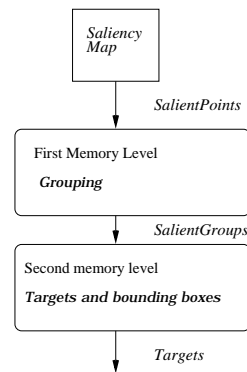


Figure 6: Overview of the different memory levels and the objects they handle.

sufficient to set up the requested tracking system. To allow to store perception information from the saliency map over time, we propose a memory system that will enable tracking.

This memory system is composed of two main levels. The first level, which can be considered as a short term (or working) memory, stores information about focuses of attention, while the second one, which can be considered as a medium term memory, uses information from the short term memory to detect and track targets over time (see figure 6).

First Level The first memory level is based on *salient points* which are built from the extracted focuses of attention. A salient point associates multiple informations to a location on the saliency map (FOA). As an example, it may contain information about intermediate saliency values. These salient points are the base elements of our memory system and are used to fill its first level.

For each of this points presented to the memory input, we try to match an existing one (already stored in this memory level) by using an euclidian distance computed in image space (feature space computation are also planned). If the computed distance is less than a specified threshold (FOA size in the case of image space distance) then points are considered to match each other.

To enhance this simple matching process, we use Kalman’s filtering techniques to predict positions of existing points in memory and distance is computed using the predicted point’s position.

If a point is matched correctly with an existing one, we update existing information with the newly perceived point. This update procedure means correcting the Kalman’s filter parameters to take into account the perceived trajectory modifications as well as updating lifetime properties of the point in memory

Here, each point has a timestamp, indicating its last update time, and other time properties: maximum lifetime, active time and time spent in memory.

Maximum lifetime is used to determine whether a point has to be removed from memory or not. As this level is “short term”, we use a small lifetime value (generally < 3 seconds). The active time parameter serves as an up-to-date marker that allows us to know if the point has been updated recently (this value is smaller than the maximum lifetime of a point since we consider that the point is up-to-date only a few instants after being updated). The last parameter is a focalisation one that enables us to consider only points that have been in memory for a sufficiently long time.

If the point isn’t matched with an existing one, then it is added in memory. There’s a limitation concerning the number of points that can be present simultaneously in the memory. This limitation allows us to add a new level of focalisation by considering only points that are perceived multiple times (this also avoids some possible noise from the saliency map).

Once we have updated our memory with new perceived FOAs, active points (recently perceived) that have been there for sufficient time are grouped to take into account spatial “similarities”. Thus, considering the salient point sizes, we build groups according to their distance in the image³. This allows us to reduce the computation time required to process those points.

Grouping is also inspired by the human visual system where groups are built during perception (e.g. based on features or spatial similarities).

Those groups will serve as an input for the second memory level (medium term memory).

Second Level At this level, we manipulate groups of salient points and *targets*.

A target describes an area of the processed image that may be considered as an object. It stores information about this area as a bounding box defined by its size and position.

For each group, extracted from the previous memory level, we check if an associated target exists. If such a target is found, then it is updated directly from the group without any further test. If there is no associated target, then a new target is created.

Each target uses a Kalman’s filter to predict both its size and position from one frame to the next. When updating a target, lifetime and Kalman’s filter parameters are modified according to the perceived information. To extract the bounding box’s size and position from the salient group, we use a blob extraction algorithm based on thresholding that allows us to extract size limits of a segmented image area. If we can’t find bounding box information using this segmentation algorithm (due to bad segmentation or thresholding problem for example), we use the salient group information to extract an approximation of the expected bounding box.

This is done by computing minimum and maximum coordinates, using the points in the associated group, and using this information as the bounding box. If the target has been updated before, this won’t affect the Kalman’s filter too much thanks to the previously perceived information.

Lifetime management is the same as for the preceding memory level except that the targets have no indication of time spent in memory. Moreover, this level doesn’t have any capacity limitation. This was not necessary since focalisation is done at the preceding memory level.

This memory system and the interactions between the different levels set up a new focalisation mechanism on top of the saliency map we used to extract focus of attention. Our system is then able to focalize on targets that are already known for a long time while allowing unexpected targets perception, due to the use of the saliency map. This is the first part of our perception system where targets will be used by our guide to build a representation of the real world. Then it will use this representation to build an explanation and interact with the visitors.

4 EXPERIMENTS

We implemented our system using the OpenCV [1] computer vision library that provides low-level image processing functions as well as higher level algorithms for computer vision applications.

To test the system, we used a video of a tropical aquarium recorded at a 320x240 pixels resolution, shot with a digital video camera. The video was played in a different thread in our tracking system at 25 FPS, which is the expected frame rate from the camera system we envisage. Our system grabs frames while the video is still being played at the same frame rate. This is why we use the *number of processed frames* as a time measurement in our results.

We used a saliency map based on simple features: motion, intensity and colors. To extract the motion mask, we used the algorithm from KaewTraKulPong and Bowden [16] which is implemented in the OpenCV [1] library. Other features are extracted as follow:

- Intensity is extracted directly out of color corrected images by computing mean of all channels:

$$I(x,y) = \frac{R(x,y)+G(x,y)+B(x,y)}{3}$$

- Colors are extracted using Itti’s [14] formulae:

$$R = r - \frac{g+b}{2} \qquad G = g - \frac{r+b}{2}$$

$$B = b - \frac{r+g}{2} \qquad Y = \frac{r+g}{2} - \frac{|r-g|}{2}$$

³ We also plan to consider feature space to check for similarities between points

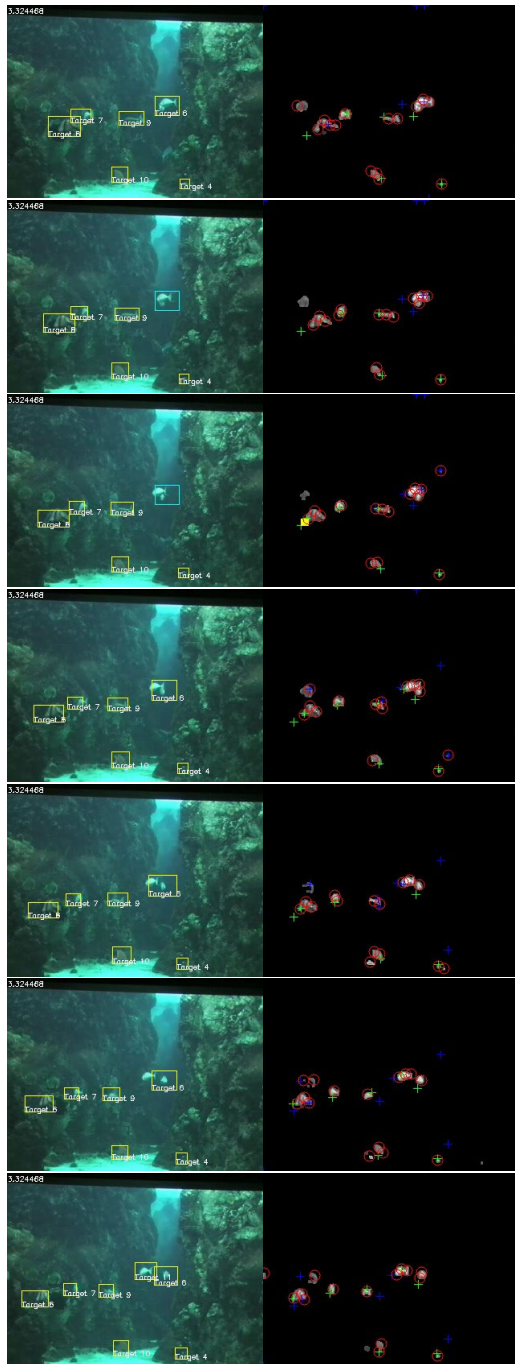


Figure 7: Tracking sequence with corresponding saliency maps (from top to bottom). The left column shows tracking output with yellow rectangle to represent currently tracked targets and light blue ones for lost targets that still being in memory. The right column show the corresponding saliency maps with the current focuses of attention as red circle, salient point in memory as cross (green for active one and blue for those we're forgetting) and salient groups as filled yellow rectangles.

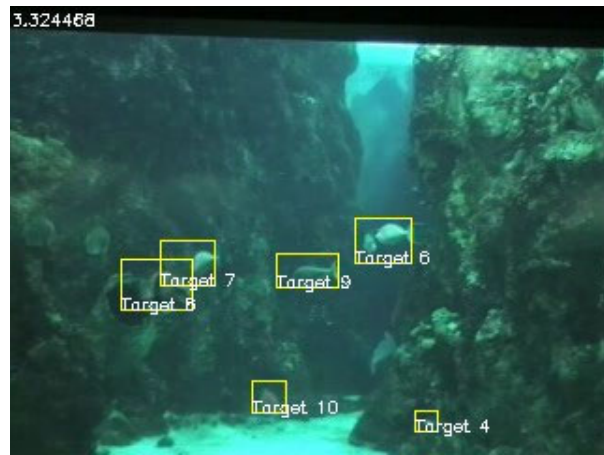


Figure 8: Tracking results. “Target 4” is a false positive due to some illumination artifacts.

Simple feature maps (intensity and colors) are combined into a global map using a maximum combination operator:

$$Global(x, y) = \max_{i \in \mathbb{F}} (F_i(x, y)).$$

Then the motion map is used as a modulation parameter to obtain the final saliency map:

$$SM(x, y) = Global(x, y) \bullet Motion(x, y)$$

There was no need of a training phase and our system was able to detect and track targets at a rate of about 5 FPS, which is acceptable for our real-time purpose since no real optimisation has been done yet.

Figure 7 shows the tracking process over a few number of consecutive processed frames. We can see that our system is able to detect and track multiple targets at the same time (this may be partially controlled by the number and size of extracted focuses of attention). Targets may also be lost for a certain number of frames. When lost, a target can be recovered if updated by a new salient point thanks to our memory system that keeps track of targets during a certain amount of time.

Focuses of attention, salient points and groups can be seen on figure 9. Groups help our system when a previously merged target (two overlapping objects) is split. The apparition of two groups instead of the previous one allows us to create a new target to track the newly discovered object.

We can also notice the presence of a false target detection (see figure 8) that passed successfully through our memory system. This target is detected due to some illumination artifacts that persisted. This accounts for the need of a control system. Recognition, or trajectories information for example, can be used to disable this kind of targets and enhance the perception system.

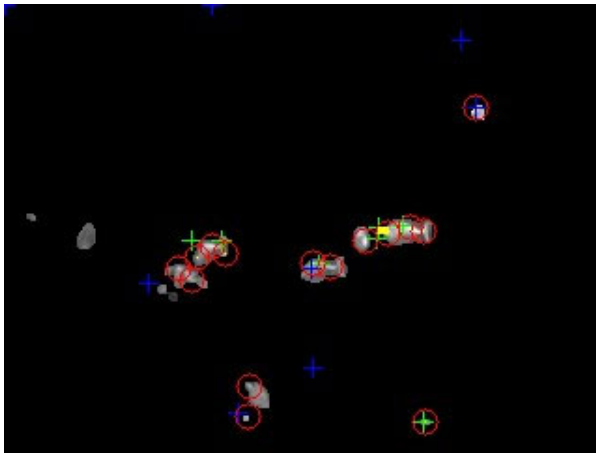


Figure 9: Saliency map. FOAs are the red circles, salient points are represented as crosses (active ones are in green) and salient groups as yellow filled rectangles.

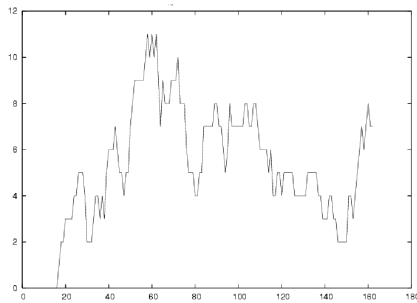


Figure 10: Evolution of the number of targets (vertical axis) over time (horizontal axis), time is defined in number of processed frames

We also measured the number of targets that are tracked at once. As showed in figure 10, this number is limited even though our second memory level has no capacity limitation. This is due to the capacity of the first memory level that keeps our system from accounting for every single salient location detected. We can also see the memory “initialisation process” through the absence of tracked targets at the beginning of the experiment. This is due to the time needed by salient points to be allowed to go to the second memory level.

Figure 11 shows the lifetime of some tracked targets in terms of number of processed frames⁴. This lifetime ranges from about 4 processed frames to 60. The average is about 13, thus showing that our simple tracking system is able to track targets over time without using complex trajectory models or assumptions. For now, new targets aren’t matched with dead ones as they aren’t stored in memory. This would require a long term

⁴ A processed frame is different from consecutive frame since the camera can grab multiple consecutive frames while only one frame is processed

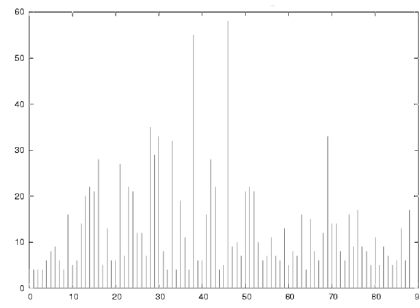


Figure 11: Lifetime of targets in number of processed frames (vertical axis), each impulse correspond to a target

memory storing information on old targets and is more appropriate for attention control. This long term memory could, for example, use trajectories to try to match new targets with dead ones and, thus, improving tracking capabilities.

5 DISCUSSION AND FUTURE WORK

The tracking system presented here is only a part of our proposed guide perception system which needs to include attentional control. This leads to the necessity of a recognition system to identify tracked fishes and the modelling of our guide’s goals. These goals will then be used to control the focalisation of visual attention. We’ll probably need to define visual strategies to abstract this control. We’ll also need to describe our guide’s knowledge in order to use it in the perception process.

The limited size of the tracked targets will also be a serious limitation when trying to recognize the fishes⁵. To overcome this difficulty, we plan to use an extended camera system with two movable cameras in addition to the fixed one used for tracking. Those cameras will have to focus on selected targets using information provided by our tracking system.

The next step will consist in adding recognition capabilities to our system and use it in our guide perception system. The intended recognition approach will be based on previous work [32]. Based on focused targets (from the extended camera system), we plan to extract specific salient information (e.g. a tuple containing a color, its spatial location on the target and a saliency value) and use this information to update a belief network. This network will then be used in a backward manner to select the information to extract.

6 CONCLUSION

We proposed a virtual guide to help people in their visit of aquariums. We described the needs of perception of

⁵ Average target size is about twenty pixels large.

this guide and presented a new tracking system based on the focalisation of visual attention.

This system used a saliency map inspired from a biological model in association with a multi-level memory. It showed some tracking capabilities when tested against static (no camera movement) videos of an aquarium. Fishes were successfully detected and tracked over a few frames while the total number of considered information was still low.

We have also given a brief overview of the intended recognition process and work that still need to be done in order to achieve the virtual guide.

REFERENCES

- [1] OpenCV computer vision library. <http://www.intel.com/research/mrl/research/opencv/>.
- [2] Gary R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, 2(2):1–15, 1998.
- [3] Tilo Burghardt, Janko Calic, and Barry Thomas. Tracking animals in wildlife videos using face detection. In *European Workshop on the Integration of Knowledge, Semantics and Digital Media Technology*, October 2004.
- [4] Andrea Cavallaro, Olivier Steiger, and Touradj Ebrahimi. Tracking video objects in cluttered background. In *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, 2004.
- [5] D. Chetverikov and J. Verestoy. Motion tracking of dense feature point sets. In *Proc. 21th Workshop of the Austrian Pattern Recognition Group*, pages 233–242, Halstatt, Oldenbourg Verlag, 1997.
- [6] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research: Part C*, 6(4):271–288, 1998.
- [7] Andrew I. Comport, Éric Marchand, and François Chaumette. Robust model-based tracking for robot vision. In *IEEE/RSJ Intl. Conf on Intelligent Robots and Systems, IROS'04*, Sendai, Japan, September 2004.
- [8] Nicolas Courty and Eric Marchand. Visual perception based on salient features. In *Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, October 2003.
- [9] Nicolas Courty, Eric Marchand, and Bruno Arnaldi. A new application for saliency maps: Synthetic vision of autonomous actors. In *IEEE Int. Conf. on Image Processing, ICIP'03 Barcelona, Spain*, September 2003.
- [10] Walther Dirk, Duane R. Edgington, and Christof Koch. Detection and tracking of objects in underwater video. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Washington, D.C., 2004.
- [11] Mary M. Hayhoe, Dana H. Ballard, Hiroyuki Shinoda, Jochen J. Triesch, Pilar Aivar, and Brain T. Sullivan. Vision in natural and virtual environments. *Eye Tracking Research and Applications Symposium*, 2002.
- [12] L. Itti and C. Koch. A comparison of feature combination strategies for saliency-based visual attention systems. In *SPIE Human Vision and Electronic Imaging (HVEI'99)*, San José, CA, pages 373–382, January 1999.
- [13] L. Itti and C. Koch. A saliency based search mechanism for overt and covert shifts of visual attention. In *Vision Research* 40, pages 1489–1506, May 2000.
- [14] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20, pages 1254–1259, November 1998.
- [15] Laurent Itti and Christof Koch. Learning to detect salient objects in natural scenes using visual attention. In *Image Understanding Workshop*, 1999. (in press).
- [16] P. KaewTraKulPong and R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *2nd European Workshop on Advanced Video Based Surveillance Systems*, 2001.
- [17] Georg Klein and Tom Drummond. Robust visual tracking for non-instrumented augmented reality. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality*, page 113, 2003.
- [18] C. Koch and S. Ullman. Shifts in selective visual attention: Towards the underlying neural circuitry. *Human Neurobiology*, 4:219–227, January 1985.
- [19] D. Koller, K. Daniilidis, and H. H. Nagel. Model-based object tracking in monocular image sequence of road traffic scenes. *International Journal of Computer Vision*, 10:257–281, 1993.
- [20] J. Kuffner and J. Latombe. Fast synthetic vision, memory, and learning models for virtual humans. *Computer Animation*, pages 118–127, 1999.
- [21] François Meyer and Patrick Bouthemy. Region-based tracking in an image sequence. Technical report, INRIA (Programme 4) : Robotique, Image et Vision, 1992.
- [22] V. Navalpakkam and L. Itti. A goal oriented attention guidance model. In *Proc. 2nd Workshop on Biologically Motivated Computer Vision (BMCV'02)*, pages 453–461, Tuebingen, Germany, November 2002.
- [23] V. Navalpakkam and L. Itti. Modeling the influence of task on attention. *Vision Research*, 45:205–231, 2005.
- [24] H. Noser, O. Renault, and D. Thalman. Navigation for digital actors based on synthetic vision, memory and learning. *Computer & Graphics*, 19(19):7–19, 1995.
- [25] Aude Olivia, Antonio Torralba, Monica S. Castelhana, and John M. Henderson. Top-down control of visual attention in object detection. In *Proceedings of the IEEE International Conference on Image Processing*, volume 1, pages 253–256, Barcelona, Spain, September 2003.
- [26] D. Ramanan and D. A. Forsyth. Finding and tracking people from the bottom up. In *Computer Vision and Pattern Recognition (CVPR)*, Madison, WI, June 2003.
- [27] K. Stark. A method for tracking the pose of known 3-d objects based on an active contour model. In *ICPR96*, pages 905–909, Vienna, August 1996.
- [28] Demetri Terzopoulos, Tamer Rabie, and Radek Grzeszczuk. Perception and learning in artificial animals. In *Artificial Life V : Proc. Fifth Inter. Conf. on the Synthesis and Simulation of Living Systems*, Nara, Japan, May 1996.
- [29] Carlos Tomasi and Takeo Kanade. Detection and tracking of point features. Technical report, Carnegie Mellon University, April 1991. CMU-CS-91-132.
- [30] Anne M. Treisman and Garry Gelade. A feature-integration theory of attention. *Cognitive Psychology*, 12(1):97–136, January 1980.
- [31] J. Verestoy and D. Chetverikov. Comparative performance evaluation of four feature point tracking techniques. In *Proc. 22nd Workshop of the Austrian Pattern Recognition Group*, pages 255–263, Illmitz, Austria, 1998.
- [32] Morgan Veyret and Eric Maisel. Simulation de la focalisation de l'attention visuelle: application à la simulation d'automobilistes virtuels. In *AFIG*, Poitiers, 2004.
- [33] J.M. Wolfe. Guided search 2.0: A revised model of visual search. *Psychonomic Bulletin & Review*, 1(2):202–238, 1994.

Fast and Stable Tracking for AR fusing Video and Inertial Sensor Data

Gabriele Bleser, Cedric Wohlleber, Mario Becker, Didier Stricker
Fraunhofer IGD
Fraunhoferstraße 5
64283 Darmstadt, Germany
{gbleser, cwohlleb, becker, stricker}@igd.fhg.de

ABSTRACT

Accurate acquisition of camera position and orientation is crucial for realistic augmentations of camera images. Computer vision based tracking algorithms, using the camera itself as sensor, are known to be very accurate but also time-consuming. The integration of inertial sensor data provides a camera pose update at 100 Hz and therefore stability and robustness against rapid motion and occlusion. Using inertial measurements we obtain a precise real time augmentation with reduced camera sample rate, which makes it usable for mobile AR and See-Through applications.

This paper presents a flexible run-time system, that benefits from sensor fusion using Kalman filtering for pose estimation. The camera as main sensor is aided by an inertial measurement unit (IMU). The system presented here provides an autonomous initialisation as well as a predictive tracking procedure and switches between both after successful (re)-initialisation and tracking failure respectively. The computer vision part performs 3D model-based tracking of natural features using different approaches for yielding both, high accuracy and robustness. Results on real and synthetic sequences show how inertial measurements improve the tracking.

Keywords: Augmented Reality, Sensor fusion, Tracking.

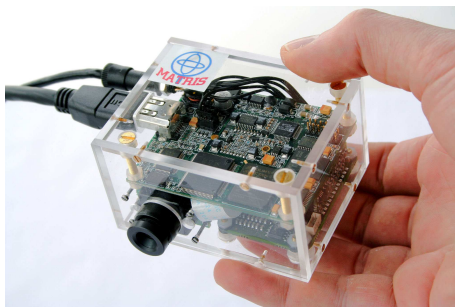


Figure 1: The integrated camera and IMU hardware.

1 INTRODUCTION

To obtain accurate registration of the camera pose, 2D/3D correspondences provided by vision-based tracking algorithms are fused with measurements of a miniature inertial measurement unit (IMU) [7] [1] [16] [6]. The hardware we use is shown in Figure 1. It includes a color firewire camera integrated with a miniature IMU that processes sensor readings at very

high update rates and triggers the image grabber, hence supplying a synchronized stream of video and inertial measurements, namely acceleration and angular velocity, which are integrated to obtain relative pose estimates. However, without a constant correction, noise and computational errors lead to a rapid drift in the camera position and orientation. For this reason, it is essential to track and update image features in a reliable way.

Different types of feature detectors [2] [10], descriptors [12] [9], alignment [18] and update methods [8] can be applied. The detection and description of point features can be performed in a robust, Euclidean or even affine invariant manner [11]. This is mostly very time-consuming and not absolutely required during tracking. In some cases it can even be a disadvantage since information gets lost through the abstracted description of the feature. Another approach is to directly use the pixel intensity information, what is less time-consuming but sensitive to illumination and viewpoint changes. However, the latter can be eventually compensated, if predictions of the camera pose are available. A frequent update of the features' descriptors increases their uncertainties and often introduces drift into the camera registration. Hence, the careful selection and long-term tracking of a small set of landmarks is desirable [13].

2 APPROACH

Our approach differentiates two phases of the vision-based tracking: initialisation and predictive tracking.

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'2006, January 30 – February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

The system (re)-initialisation performs the computation of the initial camera parameters within a global reference coordinate system without any assumption about the approximate pose. It has to work robustly without user interventions and has to be very accurate. Errors within the initial orientation are eminently critical, as the acceleration measurements of the IMU include gravity and therefore are interpreted with respect to the orientation (see 3.4). The predictive tracking processes consecutive frames. It has to be very efficient and can take advantage of temporal coherence and the high-quality prediction with the IMU as additional sensor. The pose computation bases on a Kalman filter, which processes the camera data with the measurements of the IMU and generates the best estimate of the current pose.

In [4] we used Euclidean invariant SIFT features [10] for both, initialisation and tracking, and updated the feature map and descriptors constantly. We keep this approach for the initialisation of the tracker only, since very robust results have been reached.

For the on-line tracking we take advantage of the IMU measurements and designed a vision-based tracking module, that applies simple feature tracking. In the first implementation stage, our features (so-called anchors) are 3D patches described by image textures. They are generated offline using structure from motion techniques and build a map of well-defined landmarks, which are highly stable and precisely defined in 3D. We register those anchors within the current image by pre-warping their descriptors with help of the predicted camera pose of the Kalman filter.

3 FRAMEWORK

The run-time system consists of three interacting components and a database, which contains all relevant offline data. The computer vision component is further divided into an initialisation (see 3.2) and a predictive tracking part (see 3.3) and has access to the database for either querying data for initialisation or for tracking. Initialisation yields a 6 DOF pose, whereat the predictive tracking generates 2D/3D correspondences for single frames.

The IMU does not only provide measurements of its acceleration and angular velocity, but is also able to autonomously determine a very precise estimate of its absolute orientation, which is helpful for initialisation.

The sensor fusion component (see 3.4) receives measurements from both sensors, and updates the state of the Kalman filter using the appropriate model with respect to the given kind of data. As the IMU runs at 100 Hz, the state of the Kalman filter is refreshed at the same rate, making a high-quality pose estimation available to the vision-based predictive tracking. Moreover, this component controls the application flow by solely being authorised to switch between initialisation

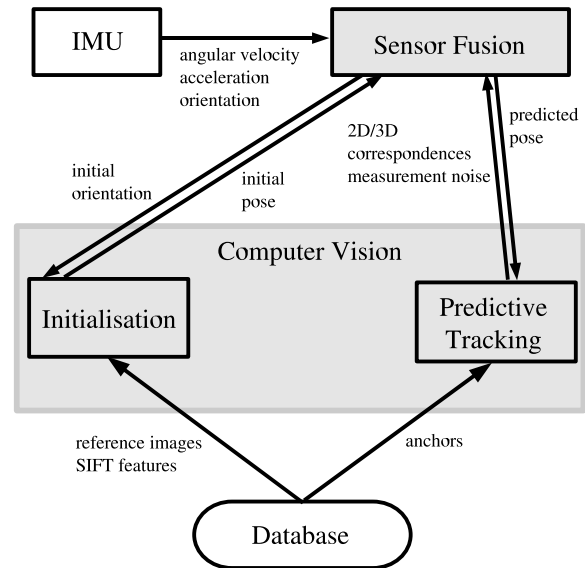


Figure 2: Architecture of the run-time system.

and tracking. If the computer vision component reports failures, the tracking can be holded up with the IMU data over a short period of time before leading to a divergence of the Kalman filter, that leads to reinitialisation. Figure 2 shows the architecture of the system.

3.1 Offline Preparation

The vision-based component requires some pre-processed input data for initialisation and tracking. This data is generated within an offline step for every 3D scene and camera setup and exported to an xml file for permanent storage and re-use. At system startup, the data is imported to the run-time system by the database component. The database contains:

- Reference images - these consist in fully calibrated images that show the 3D scene from a sufficient number of viewpoints. Furthermore a group of 3D SIFT features is associated to every reference image. They are used during initialisation to get a pose estimate for a given camera frame. Reference images can be either exported by an automatic 3D scene reconstruction step [15] or they can be manually calibrated using a CAD model [4].
- Anchors - they represent persistent features defined in 3D. Currently, we process 3D patches, given as four 3D corners and a surface normal, with an associated image texture that has been sampled from the reference image with the most frontal view onto the plane.

3.2 Initialisation

The initialisation is achieved by matching SIFT features between the live video frame and pre-calibrated reference images, for which the features are known in 3D.

To deal with multiple reference images efficiently, this task is divided into two steps:

Firstly, the database is queried with the current camera frame to find the most likely reference image using a content based similarity measure [14]. At the current implementation, the image retrieval is based on color histograms. Additionally, the initial orientation estimate of the IMU is exploited by finding the closest reference pose with respect to the camera view direction. The roll angle is neglected, as the SIFT features used for obtaining 2D/3D correspondences in the second step, are invariant against in-plane rotations. The second step is carried out with the pre-selected reference image and its associated features as detailed in [4].

3.3 Predictive Tracking

The task of this module is the generation of 2D/3D correspondences by locating the 3D anchors provided by the database within the current camera frame using the pose prediction of the sensor fusion module. This is done within the following steps:

1. query anchors from the database, that are visible from the predicted pose
2. pre-warp their descriptors to the appearance expected from that viewpoint
3. register the anchors within small uncertainty regions around the expected positions using the transformed descriptors
4. establish 2D/3D correspondences by associating each registered 3D anchor centroid to its measured image position
5. perform an outlier rejection to verify valid correspondences
6. pass all valid correspondences to the sensor fusion
7. report a tracking failure, if there have been few inliers
8. update the search radius

As mentioned in paragraph 3.1 we define an anchor as a 3D plane that is described by an image texture. The latter is sampled from a reference image with a preferably frontal view onto that plane. As the reference images are fully calibrated and a pose prediction for the current frame is available, the pre-warping of each anchor description (step 2) is described by a homography

$$H = K(\delta R - \delta T \cdot \frac{\vec{n}^T}{d})K^{-1}$$

where $\delta R, \delta T$ is the relative camera motion between the current prediction and the reference image, the anchor description has been sampled from, and \vec{n}, d are the

plane parameters given in the camera coordinate system of the reference image [5]. The camera intrinsics K are assumed to be static. An example anchor description and pre-warping is given in Figure 3 a,b. The closer the prediction comes to the actual pose the more is the current anchor appearance resembled by the pre-warped one and the more confident becomes the registration. But we made the experience that small distortions do not harm the registration.

Referring to step 3 the pre-warped descriptors are located within the current frame by performing a block matching within a small uncertainty region around the expected anchor positions. The latter are simply computed by projecting the 3D centroids with the current pose prediction. Typically we extract a patch (16x16 or 32x32 pixels) from the warped texture around the centroid and use this for the registration. To be independent of intensity changes like $\alpha f(i, j)$, we use the normalized cross correlation as similarity measure [3]. Computing this measure for all pixel positions around the expected image location within a square search region, we find the position that minimizes the criterium and take this as the correct anchor location for establishing a 2D/3D correspondence. Figure 3 points out this procedure.

As the performance of the block matching (assuming a fixed patch size) depends exclusively on the size of the search region, we keep this as small as possible while assuring a stable tracking. We give a fixed upper T_{max} and lower T_{min} threshold for the search radius R in pixel but choose this parameter dynamically during tracking (step 8). After initialising this parameter with the upper value, we update the search range for the next video image R_{t+1} depending on the maximal observed feature displacement D_{max} using the following rules:

$$\begin{aligned} R_{new} &= D_{max} + T_{min} \\ \alpha &= \frac{D_{max}}{R_t} \\ R_{t+1} &= (1 - \alpha) \cdot R_t + \alpha \cdot R_{new} \\ R_{t+1} &= \min(T_{max}, \max(T_{min}, R_{t+1})) \end{aligned}$$

The formulas are interpreted as follows: the weight $\alpha \in [0..1]$ (as $D_{max} \leq R_t$) defines the influence of R_{new} , which merely depends on the measured displacements (and the constant T_{min}), onto the changeover of R_t , the currently used search radius, to R_{t+1} , the radius for the next frame. If the maximal measured displacement is near to the search radius (D_{max} is near to R_t), the weight onto the measured displacement gets near to 1 and as a result the search range increases quickly. This is very important as the block matching can't return the correct registration, if it does not lie in focus. If D_{max} is far away from R_t , the weight becomes small, hence the search range decreases slowly.

The set of correspondences resulting from the anchor

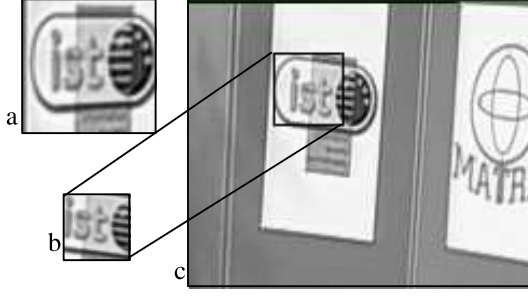


Figure 3: pre-warping and block matching: a: attached image texture as stored in the database, b: pre-warped texture, c: start position for block matching within camera frame

registration are filtered by RANSAC-like outlier rejection [5]. As the RANSAC algorithm also yields a 6 DOF pose, a tracking failure is not only reported, if there have been few inliers, but also if the vision-based pose estimate is completely different from the prediction.

3.4 Sensor Fusion

This component runs a Kalman filter on the measurements from both sensors and makes a pose estimate available to the vision-based tracking. More details can be found in [17] and [7][6].

Camera Motion Model

To fit the Kalman filter framework the motion of the camera is described as a nonlinear system model [17] [7]:

$$x_{k+1} = f(x_k, u_k, v_k), \quad v_k \sim N(0, Q_k) \quad (1)$$

with state vector $x_t = [p \ v \ q]^T$ representing the camera position, velocity and orientation quaternion in a global coordinate system. This equation is used to predict the pose at each time step by integrating the sensor data $u_t = [\omega \ a]^T$ composed of the angular velocity ω and the acceleration a . As the sensor and the camera have different coordinate systems the data has to be transformed using the so-called hand-eye calibration R_X . We assume that this transformation is only rotational i.e. the camera and sensor center points are very close.

$$\begin{aligned} p_{k+1} &= p_k + v \cdot \Delta t + (R(q) \cdot R_X \cdot a + g) \cdot \frac{\Delta t^2}{2} \\ v_{k+1} &= v_k + (R(q) \cdot R_X \cdot a + g) \cdot \Delta t \\ q_{k+1} &= A(\theta) \cdot q_k \end{aligned}$$

with

$$A(\theta) = \cos\left(\frac{\|\theta\|}{2}\right) \cdot I_4 + \sin\left(\frac{\|\theta\|}{2}\right) \cdot \Omega$$

$$\begin{aligned} \theta &= \omega \cdot \Delta t \\ \Omega(\omega) &= \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \end{aligned}$$

This integration of sensor data is also known as dead-reckoning.

Every system update increases the pose uncertainty described by a covariance matrix P

$$P_{k+1} = A_k P_k A_k^T + Q_k, \quad A_k = \frac{\partial f}{\partial x}(\hat{x}_k, u_k) \quad (2)$$

by adding the system noise covariance Q resulting from the sensor data uncertainty.

Using Vision Measurements

While the exclusive integration of the sensor data leads to a rapid drift in the pose prediction, the vision-based tracking provides 2D/3D correspondences that are used as measurements to correct the state.

- **2D/3D Correspondences** 2D feature positions are related to the filter state using their 3D points and the camera projection model [17] [7]. The measurement equation for one feature point $\xi_i = (u, v)$ in the camera frame corresponding to the 3D point s_i in the world coordinate system is expressed as

$$\begin{bmatrix} Zu - fX \\ Zv - fy \end{bmatrix} = 0 \quad (3)$$

with $[X \ Y \ Z]^T$ the coordinate of the feature point in the camera system i.e. $[X \ Y \ Z]^T = R^T(q) \cdot (s_i - p)$

The uncertainties of the 2D and 3D feature points have to be modelled as independent normal distribution to fit in the Kalman filter. The covariances of the 2D feature points are given by the block matching.

- **2D/2D Correspondences** 2D/2D correspondences of points or lines could also be used in the Kalman filter by relating the velocity of the features to that of the camera. The derivation of the measurement equation can be found in [17]. The expression for one feature $[\xi_k \ \psi_k]$ is

$$\begin{aligned} [-f\alpha, f, \xi_k \alpha - \psi_k] v_k &= 0, \\ \alpha &= \frac{\dot{\psi}_{Dis} - \dot{\psi}_{Rot}}{\dot{\xi}_{Dis} - \dot{\xi}_{Rot}} \end{aligned} \quad (4)$$

with $\dot{\xi}_{Dis}, \dot{\psi}_{Dis}$ being the velocity of the feature due to the camera displacement, $\dot{\xi}_{Rot}, \dot{\psi}_{Rot}$ being the velocity due to the rotation and f being the focal length of the camera.

- **Divergence Monitoring** Loss of feature points over a longer period, e.g. because of successive tracking failures, causes the Kalman filter to diverge. As the filter is not able to recover itself, it has to be reinitialized externally (see 3.2). If the variance of the error $h(\hat{x}_k, 0)$ over the last frames exceeds the standard gaussian error, this is taken as a hint for an irreparable drift in the filter state and causes the sensor fusion component to request an initial pose estimate from the computer vision part.

Data Synchronization

The data provided by the inertial and vision sensors has to be synchronized to provide a correct real time pose estimation. Two problems have to be solved to obtain proper synchronization:

Multi-Rate sensor fusion The computer vision and the IMU run at different sampling rate. The Kalman filter is able to perform several system updates before new point correspondences are provided. A similar multi-rate sensor fusion problem can be found in the vehicle navigation literature where GPS position updates are not available as often as inertial data. In our system, the camera shutter is triggered by the IMU so that the camera images are synchronized with the inertial data at hardware level. As the IMU is running at 100 Hz, the camera can be run at 25 or 12.5 Hz.

Lag of vision data The computer vision (CV) takes some time to process a camera frame (Figure 4). So the point correspondences are delayed when they are provided to the sensor fusion. The Kalman filter performs system updates (SU) for each IMU data sample. As long as no vision data is received the system state and IMU data are buffered. When point correspondences are available, the system state is reverted to the vision data timestamp and updated to current time with the buffered IMU data.

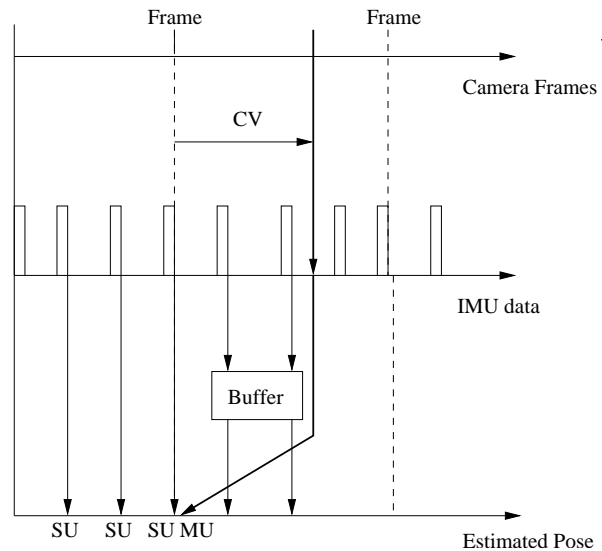


Figure 4: Synchronization of the run-time system.

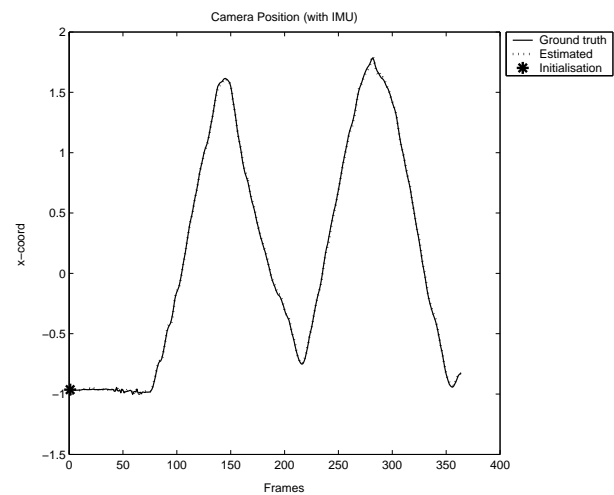


Figure 5: Position (x coordinate) of the camera with use of IMU data, vision data at 5 Hz.

4 RESULTS

We tested our system on live video using the hardware represented in section 1. Figures 9-10 show some augmentations from the live video, where the camera has been triggered with 12.5 Hz. The system has also been tested on synthetic data to provide repeatable results. Some synthetic offline sequences have been generated with resolution 720x576 pixels. The observed 3D scene is a textured 3D model of a room. Four reference images and 40 planar texture patches (anchors) have been pre-processed as input for the vision-based initialisation and the predictive tracking. As the scene has been rendered from a 50 Hz camera track, the ground truth pose data is available at 50 Hz and can be compared to the estimated pose of our system. The inertial sensor data has also been computed from the virtual camera track

and simulates the synchronized 100 Hz samples of an IMU moving on this track.

Data fusion at low vision rates

The system has been tested at different vision data rates from 5 to 50 Hz. The improvement of the tracking using the IMU data is most visible at low vision rates. Figures 5 and 6 show the estimated camera position in comparison with the ground truth data with and without the use of IMU data.

Without the IMU support, the pose computation is less accurate and the Kalman filter even diverges so that several re-initialisations are needed. The IMU data allow to track rapid camera movements which are difficult to follow with a purely vision based system.

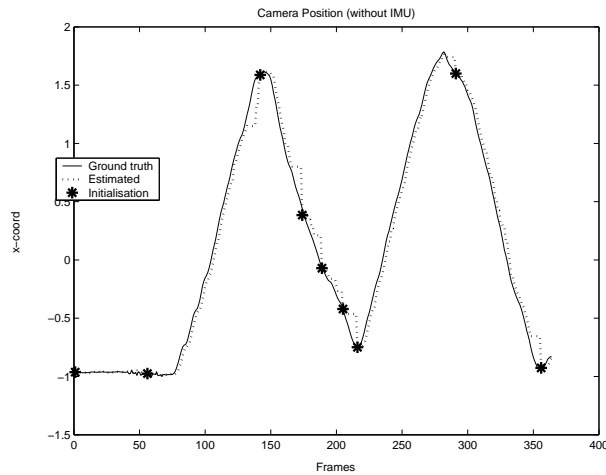


Figure 6: Position (x coordinate) of the camera without use of IMU data, vision data at 5 Hz.

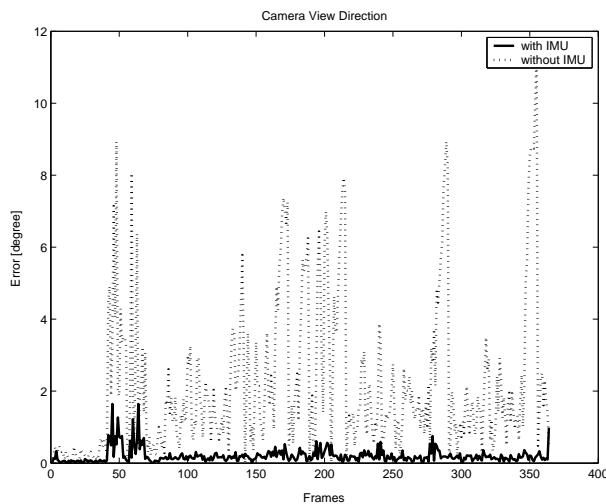


Figure 7: Orientation error of the camera with and without use of IMU data.

Figure 7 plots the orientation error with and without the use of the IMU data. As the IMU gives very accurate gyroscopic measurements, the system is able to compute the rotational part of the camera pose very precisely.

Pose prediction

As the IMU delivers an update rate of 100 Hz, the camera pose is updated at the same rate. Hence the current pose is always available to the system so that it can provide a high-quality prediction of the feature positions to the vision part. The search ranges of the block matching can then be reduced, as shown in figure 8.

5 CONCLUSION

We designed a flexible run-time system for real-time 3D camera tracking fusing vision-based data with

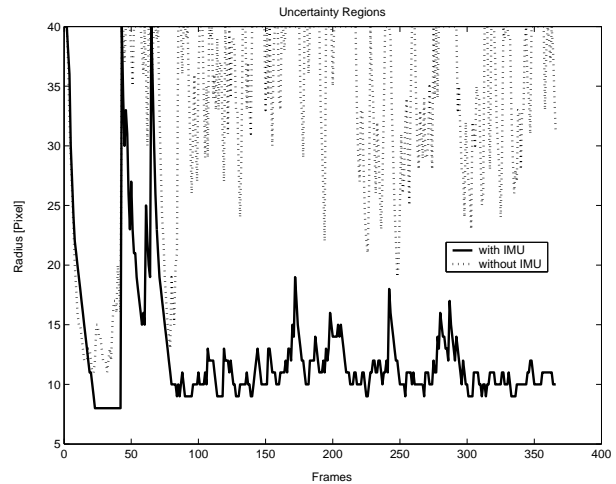


Figure 8: Search ranges needed by the block matching.



Figure 9: Predictive Tracking: predicted and matched patches are overlaid.



Figure 10: Video frame with live augmentation. The texture is correctly overlaid on the video image using the computed camera pose from the system.

inertial sensor data. As the IMU provides angular velocity and acceleration measurements at a refresh rate of 100 Hz, the camera pose is updated at the same rate. Simultaneously, the camera sample rate can be reduced, whereat a high-quality pose prediction is nevertheless passed to the vision-based tracking.

We showed, that the integration of inertial data improves the tracking significantly, while minimizing CPU costs and yielding high precision and rendering frame-rate, which is crucial for mobile See-Through applications. At the current state the vision-based tracking module operates exclusively with 3D planar features that have been generated offline. Therefore the tracking is restricted to parts of the environment that have been reconstructed within the preparation step. Future work includes the online reconstruction of temporal features, which are less confident but hold up the tracking, whenever the camera observes additional territory.

6 ACKNOWLEDGEMENTS

This work has been performed within the MATRIS consortium, a research program within the European Union (IST-002013). The authors would like to thank the EU for the financial support and the partners within the consortium for a fruitful collaboration. The synthetic image sequences have been provided by the university of Kiel, the synthetic sensor measurements have been provided by Xsens Motion Technologies. For more information, please visit its website, www.ist-matris.org.

REFERENCES

- [1] L. Armesto, S. Chroust, M. Vincze, and J. Tornero. Multi-rate fusion with vision and inertial sensors. *Robotics and Automation, 2004. Proceedings. ICRA '04*, pages 193–199 Vol.1, 2004.
- [2] A. Baumberg. Reliable feature matching across widely separated views. In *Computer Vision and Pattern Recognition, Volume 1*, pages 774–781 vol.1, 2000.
- [3] O.D. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, 1993.
- [4] G.Bleser, Y.Pastarmov, and D.Stricker. Real-time 3d camera tracking for industrial augmented reality applications. In *Proc. of WSCG 2005*, pages 47–54, 2005.
- [5] R. Hartley and A. Zissermann. Multiple view geometry in computer vision. 2000.
- [6] J. Hol. Sensor fusion for camera pose estimation. *Master Thesis, University of Twente*, 2005.
- [7] J. Hol, P. Slycke, T. Schoen, and F. Gustafsson. 2d-3d model correspondence for camera pose estimation using sensor fusion. In *Proc. of InerVis workshop at the IEEE International Conference on Robotics and Automation*, 2005. <http://www.xsens.com/>.
- [8] I. Takahiro I. Matthews and S. Baker. The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 810–815, 2004.
- [9] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Proc. Computer Vision and Pattern Recognition*, pages 506–513, 2004.
- [10] D. G. Lowe. Object recognition from local scale invariant features. In *Proc. of the International Conference on computer Vision ICCV*, pages 1150–1157, 1999.
- [11] K. Mikolajczyk and C. Schmid. An affine invariant point detector. In *European Conference on Computer Vision, Volume 1*, pages 128–142, 2002.
- [12] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *Proc. Computer Vision and Pattern Recognition*, pages 257–264, 2003.
- [13] N. Molton, A. Davison, and I. Reid. Locally planar patch features for real-time structure from motion. In *Proc. of British Machine Vision Conference*, 2004.
- [14] S. Philippi and G. Bleser. A framework for the comparison of similarity measures in multimedia databases. *Accepted for publication in Proc. of the International Conference on Imaging Science, Systems, and Technology*, 2005.
- [15] R.Koch, J.-F. Evers-Senne, J.-M. Frahm, and K.Koeser. 3d reconstruction and rendering from image sequences. In *Proc. of WIAMIS*, 2005.
- [16] U. Neumann S. You and R.T. Azuma. Hybrid inertial and vision tracking for augmented reality registration. pages 260–267, 1999.
- [17] T. Schoen and F. Gustafsson. Integrated navigation of cameras for augmented reality. *International Federation of Automatic Control (IFAC) World Congress, Prague*, 2005.
- [18] T. Zinßer, C. Graeßl, and H. Niemann. Efficient feature tracking for long video sequences. In *Proc. DAGM Symposium*, pages 326–333, 2004.

3D Human Animation from 2D Monocular Data Based on Motion Trend Prediction

Li Zhang

Department of Computing
Curtin University of Technology
Perth, WA 6845
GPO Box U1987 Perth
zhangl@cs.curtin.edu.au

Ling Li

Department of Computing
Curtin University of Technology
Perth, WA 6845
GPO Box U1987 Perth
ling@cs.curtin.edu.au

ABSTRACT

A model-based method is proposed in this paper for 3-dimensional human motion recovery, taking un-calibrated monocular data as input. The proposed method is able to generate smooth human motions that resemble the original motion from the same viewpoint the sequence was taken, and look continuous from any other viewpoint. The core of the proposed system is the motion trend prediction for reconstruction. To focus the research effort on motion reconstruction, “synthesized” input is first employed to ensure that the reconstruction algorithm is developed and evaluated accurately. Experiment results on real video data indicate that the proposed method is able to recover human motion from un-calibrated 2D monocular images with very high accuracy.

Keywords: human animation, 3D motion reconstruction, motion trend prediction

1. INTRODUCTION

Animation is the production of consecutive images, which, when displayed, conveys a feeling of motion [HOB99]. In the past decade, with the rapid development of computer technology, computer animation has become very popular in many applications. In computer animation, the representation of human body and its motion receives great attention, since human animation are widely employed in many areas, such as games, movies, surveillance, scientific visualization, etc. As monocular images and video sequences are easily available, many great efforts have been made to reconstruct 3D human motion from monocular images. However, such attempt remains very much under-developed due to many technical difficulties.

[Tay00] and [RR03] suggested adjusting the posture

of a human model according to camera calibration information and biomechanical constraints applied on the model. Orthographic projection is used in their approaches, which is greatly different from the perspective projection used in any real camera. [LZP99] and [PCS02] made use of the motion library. The former took motion attributes achieved through reconstruction as guidance for estimation of unknown human motion, while the latter use motion library to resolve the depth ambiguity in recovering 3D configuration from 2D image features. In both attempts, a large motion library needs to be maintained and upgraded continuously. In [CB04] the concept of prioritized constraints is introduced. Based on it the proposed method can get quite good results, which is only suitable for adding variations to motions known before the reconstruction. [DTJR01] introduced an interactive system which combines biomechanical constraints on 3D motion with user interferences to reconstruct sequences in 3D; similarly three possibilities for solving inverse kinematics problem during human animation are discussed when interactive direct manipulation is applied [Fěd03]. There are also attempts in automatically generating accurate inverse dynamics solutions to simulate and deform human motion [TSF04] and [KM04], however such efforts have been concentrated on hand posture recovery only. [ZL04] proposed a Criterion Function (CF) to represent the residuals between the image feature and the projected features from the reconstructed 3D model in a Global Adjustment (GA) system. In their

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.

SHORT COMMUNICATIONS proceedings,
ISBN 80-86943-05-4
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

method the accuracy and the consistency of the recovered postures are only guaranteed from the same viewing direction as the original.

Most existing methods introduce simplifications on human motion or require assistance such as user interference or motion library. This paper aims to propose a novel model-based human motion reconstruction method from un-calibrated 2D monocular data without user interferences and the human motion is truly unrestricted.

The key component of our proposed system is a Motion Trend Prediction (MTP) technique which aims to achieve the posture reconstruction at every frame based on information from previous frames (except the 1st frame). Unlike the proposed method for estimating 3D motion of an articulated object in [HNHQ94], we do not divide our skeletal model into many small components, recover their motions separately and integrate them together to compose the motion of the skeleton itself. In that way although the estimation of each small component might be simple and feasible, the recovered motion of the entire skeleton cannot be ensured to be favorable. In our algorithm, the whole skeletal model is always treated as a single object to ensure the consistency between motions of different body parts and their smoothness. Some systems utilized silhouettes for estimation of the posture with human body depth and collision constraints [MG01], where only arms were studied. Later such efforts were extended [FRDC04]. To better produce silhouettes from 2D monocular view and to resolve the 3D pose prediction ambiguities Principal Component Analysis (PCA) and Radial Basis Function (RBF) are both employed. Besides them, motion library and key frame technique are also needed for accurately rendering 3D animal gaits. Simplicity and computational efficiency is sacrificed in the method to achieve accurate motion simulation. Compared to them our proposed MTP technique is fairly simpler yet more accurate. As the MTP technique is a per-frame approach, we introduce a filtering process to smooth the results. The filtering algorithm can be repeated if necessary in order to re-enforce important constraints. Therefore the accuracy of the MTP can be strictly guaranteed. Further more, unlike the Kalman or the particle filtering prediction tracking algorithm [WB95], calculations of partial derivatives or an even more complicated infrastructure are not necessary in our algorithm. Through a very small set of simple Deviation Function (DF) equations, 3D human postures in a motion sequence can be tracked and recovered. The settings of the weighting parameter (WP), which plays an important role in determining the MTP's accuracy, can be easily found through a low-pass filtering process as well. The MTP technique proposed here is very simple, but it can

efficiently generate smooth human animation. It demonstrates great advantages over most of the methods proposed before, when only 2D monocular data is available.

The rest of this paper is organized as follows: 3D skeleton model used in motion reconstruction is introduced in Section 2; in Section 3 assumptions on the input data are explained; MTP and the detailed reconstruction procedure are described in Section 4; experimental results are presented and analyzed in Section 5; and a brief conclusion is presented in Section 6.

2. MODEL

To reconstruct 3D motion a camera model and a human model are set up. The camera model is located at a fixed position in virtual space with pre-defined focal length, and it does not require knowledge of the actual cameras from which the video sequence is taken. This project focuses on recovering the whole body motion from monocular data; hence an articulated 3D skeletal model as shown in Fig.1 is sufficient for our purpose. The joint *pelvis* is set as the root in the skeletal tree structure, and the 5 leaf joints are *left wrist*, *right wrist*, *left ankle*, *right ankle* and *head*.

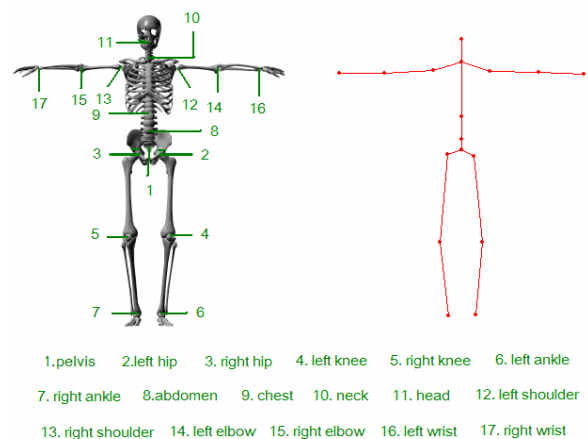


Figure 1: 3D skeletal model, its 2D stick figure and the 17 joints

At the current moment, our attention is focused on the motion reconstruction for segments including pelvis, waist, chest, neck, upper arms, forearms, thighs, and crura. The tips of hands and feet, and the top of the head are not considered as individual joints in our work. Their motions will be determined by the motions of forearm, crus and neck segments respectively. Besides, the leaf joints are assumed to have no degrees of freedom (DOF) since they are located at the end of skeletal branches. Movements of these leaf joints are resulted from the rotations of the connecting parent joints. For reconstruction purpose, we assign each intermediate joint (joints that are not leaf joints mentioned above) 1 to 3 DOF(s) in the

world coordinate system (WCS), and each intermediate joint is associated with a local coordinate system (LCS) as shown in Fig.2. However there are 6 DOFs (3 for translations and the other 3 for rotations) at *pelvis*, since the translation of the whole body is represented by the movement of *pelvis*. Therefore our human model actually has 17 joints, 12 segments and 37 DOFs.

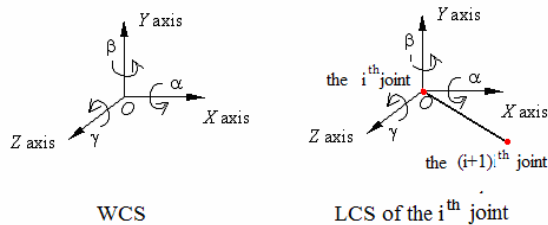


Figure 2: WCS and LCS of the i^{th} joint

The rotational angles of each intermediate joint about the 3 axes in its associated LCS are governed by biomechanical constraints [WBF87, RGB96, PW99]. The joint constraints in [ZL04] are used here with some modifications as listed in Table 1.

	x-axis	y-axis	z-axis
<i>Chest</i>	$-25^{\circ} \sim 30^{\circ}$	$-30^{\circ} \sim 30^{\circ}$	$-15^{\circ} \sim 15^{\circ}$
<i>Left Shoulder</i>	$-90^{\circ} \sim 90^{\circ}$	$-135^{\circ} \sim 45^{\circ}$	$-135^{\circ} \sim 90^{\circ}$
<i>Left Elbow</i>	$0^{\circ} \sim 0^{\circ}$	$-135^{\circ} \sim 0^{\circ}$	$0^{\circ} \sim 0^{\circ}$
<i>Right Shoulder</i>	$-90^{\circ} \sim 90^{\circ}$	$-45^{\circ} \sim 135^{\circ}$	$-90^{\circ} \sim 135^{\circ}$
<i>Right Elbow</i>	$0^{\circ} \sim 0^{\circ}$	$0^{\circ} \sim 135^{\circ}$	$0^{\circ} \sim 0^{\circ}$
<i>Left Knee</i>	$0^{\circ} \sim 135^{\circ}$	$0^{\circ} \sim 0^{\circ}$	$0^{\circ} \sim 0^{\circ}$
<i>Right Knee</i>	$0^{\circ} \sim 135^{\circ}$	$0^{\circ} \sim 0^{\circ}$	$0^{\circ} \sim 0^{\circ}$

Table 1: Modified angular constraints based on biomechanical constraints

Through proper translations of the *pelvis* joint and rotations of the segments about each joint in the skeletal model, any human posture could be obtained from the initial pose in Fig.1.

3. PREPARATION OF DATA

In order to generate reliable motion reconstruction, 2D feature information, such as the joints' positions on the 2D image plane, extracted from the source images or video must be highly accurate. Any error in feature extraction could lead to incorrect 2D configuration of human body geometry and human posture. There have been a large number of approaches to feature extraction in the image processing and computer vision area [AT04, BK03, GODC05, LF04, LZZP04, SJ04, TSF04]. However up to date, no technique is able to produce 2D image feature extraction that is sufficiently accurate. Extraction error remains an unavoidable issue. As a matter of fact, such inaccuracy is one of the main

factors preventing the progress of reconstruction technology. Besides, the lack of depth information in monocular image source makes it extremely difficult to evaluate the performance of any 3D reconstruction algorithm.

The extraction of 2D feature information and the 3D motion reconstruction are actually two independent modules. They should be addressed separately and in parallel for the ultimate development of the complete 3D motion reconstruction system. In this project, the research is focused on the 3D motion reconstruction. To ensure the proper development and evaluation of the reconstruction algorithms, computer synthesized source videos is first used. The popular BVH (Bio-vision hierarchical data) motion files are employed in any computer animation software to generate various animation series on a fixed human model with known geometry. For the time being, the motion is generated for a skeletal model to show the motion more clearly. The 3D animation is projected on the image plane to produce the 2D joint features in every frame, which is the only input to our motion reconstruction system. The 3D motion data and the camera settings are not utilized in any way during the reconstruction process. Such information can later be used for comparison purpose after the motion is reconstructed in 3D to ensure proper evaluation of the reconstruction algorithms developed.

The reconstruction algorithms developed can be used on any kind of actual monocular video to produce reconstructed human motion truthful to the extracted 2D feature information. The "synthesization" part of this project is merely to enable accurate evaluation and assessment of any reconstruction algorithms. The algorithms are later tested on real video to demonstrate their applicability.

Manual adjustment for the 1st frame is required in our system, since information of the 1st frame is the starting point for the MTP. The human posture and body location recovered at the 1st frame have to be very accurate. The skeletal model will first be resized to fit the human geometry shown in the 2D monocular data. Human posture in the 1st frame will then be calculated and recovered using MTP as discussed in the next section. Minor manual rotations about certain joint might be necessary to ensure that the recovered posture resembles the original one in all details. Such manual adjustment is only allowed in the first frame. The reconstruction process of all other frames is completely automatic and user interference is not allowed and totally unnecessary.

4. MOTION TREND PREDICTION

Human motion reconstruction is actually a process to reconstruct human posture at every frame. The goal is to recover human postures which resemble the original postures as much as possible. Our algorithm

is based on two observations about human motion. 1. Despite the complexity of human motion, there are actually only two types of movements involved: the translation of the whole body and the rotations of the body segments about the joints. The former put the human body in a certain location, while the latter generates a particular body posture. Hence, the 3D movement of any joint can be treated as the composition of the following transformations: translation of the whole body, and rotations of all ancestor joints of this particular joint. 2. Most of the human motion is assumed generally smooth (although it's not so true in real human motion), which means that the 3D human postures in neighboring frames are similar.

Based on the above observations, a Motion Trend Predication technique is proposed. As discussed in Section 1, the technique has many advantages over the existing methods. It is developed for two different types of human motion: human motion with or without the body relocation.

4.1. Motion Reconstruction without Body Relocation

In this case, there is no body relocation in the input monocular human motion, which means that the position of the joint *pelvis* is fixed. Therefore the movement of each joint can be considered as a composition of the rotations about all its ancestor joints. According to kinematics principles [IC05], the adjusting order applied in posture reconstruction should be from the root to the leaf joints. The *pelvis* is taken as the origin of the WCS. By minimizing the residuals between the skeleton's projected figure and the 2D image feature, all joints can be rotated to certain 3D positions, which ensure the recovered posture resembles the original one from the same viewpoint. However recovering posture at each frame individually may violate the inter-frame consistency; to ensure natural looking motions could be produced effectively with the MTP, two important 3D predictors are introduced. When a segment connecting the i^{th} joint and one of its direct descendants is being rotated about the i^{th} joint, coordinates of that descendant should be utilized to evaluate if the segment has been rotated to the correct location. Since the 3D coordinates of any joint in neighboring frames should be similar to ensure smooth motion, the 3D coordinates of every joint in the WCS could be used for motion prediction. Hence the first 3D predictor is defined to represent the distance between a certain joint's 3D positions at neighboring frames as follows:

$$predictor_{3D_pos_i} = \sqrt{(x^K - x^{K-1})^2 + (y^K - y^{K-1})^2 + (z^K - z^{K-1})^2} \quad (K \geq 2) \quad (1)$$

where (x^K, y^K, z^K) and $(x^{K-1}, y^{K-1}, z^{K-1})$ stand for the

WCS coordinates of the i^{th} joint's direct descendant(s) at the K^{th} and $(K-1)^{th}$ frame respectively. Another 3D predictor is derived from the Z component of each joint. This predictor is used to stress the movements caused by the absolute depth changes in WCS:

$$predictor_{z_component_i} = abs(z^K - z^{K-1}) \quad (2)$$

During the reconstruction process, once the reconstruction of a frame is finished, the posture configuration of the skeletal model obtained for this frame will be used as the reference for predicting the human posture in the next frame. In other words, the 3D coordinates of every joint and its Z component in WCS at the $(K-1)^{th}$ frame will be utilized when recovering the 3D posture at the K^{th} frame.

Therefore a DF is developed as shown in Eq.(3), which is the parametric representation of the MTP technique for adjusting the skeletal model. It looks similar to the popularly-used energy function defined in [WBF87]. However in our DF, 3D predictors are introduced, which are based on the 3D position and Z component of every joint. The new items greatly enhance the accuracy of the reconstruction in the Z direction, which is the main concern in any 3D reconstruction. Meanwhile the way the DF is formulated in our MTP makes the solution process much simpler than all existing methods.

$$DF_i = weighting_parameter_{orientation_i} \times deviation_{orientation_i} + \\ weighting_parameter_{position_i} \times deviation_{position_i} + \\ weighting_parameter_{length_i} \times deviation_{length_i} + \\ weighting_parameter_{3D_pos_i} \times predictor_{3D_pos_i} + \\ weighting_parameter_{z_component_i} \times predictor_{z_component_i} \quad (3)$$

The above DF is in its basic form. When dealing with limbs, possible ambiguities could be resulted from occlusion due to the high flexibility of limbs. A method is proposed to better handle such situations. Usually postures of limbs differ slightly between neighboring frames, thus it is first assumed that poses of the forearms or shins are exactly the same when rotating the upper arms or thighs. This way the ambiguities could be reduced greatly. Before upper arms or thighs are being rotated at the K^{th} frame, configurations of the forearms or shins obtained at the $(K-1)^{th}$ frame are applied temporarily. The position residuals of the *wrists* or *ankles* between the projection and image features are then included into the DF. Hence the DF for rotating upper arms or thighs is evolved as follows:

$$DF_i^{limb} = DF_i + \\ weighting_parameter_{leaf_joint_i} \times deviation_{leaf_joint_i} \quad (4)$$

$$deviation_{leaf_joint_i} = D_2(P_{j_i}, P_{j_p}) \quad (5)$$

Here the j^{th} joint (leaf joint – wrist or ankle) is an indirect descendant of the i^{th} joint (shoulder or hip); P_{j_i} is the image feature of the j^{th} joint, and P_{j_p} is its corresponding projection feature, as illustrated in Fig.3.

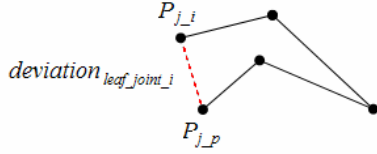


Figure 3: Deviation in the position of indirect child joint (leaf joint)

When the joint *pelvis* is fixed at a certain position in WCS, the 3D coordinates of every joint in WCS are used to define a 3D predictor no matter how the *pelvis* is rotated. Each joint of the skeleton model is rotated one by one from the root to the leaf joints according to the DFs defined above. For instance, when rotating joint *pelvis*, the residuals between the image and projection features of its three direct child joints (*abdomen*, *left* and *right hips*) must be optimized to nearly zero. The 3D positions and Z components of these three joints in the previous frame are taken into consideration as well. This way the 3D position of *pelvis* can be recovered.

Once the posture configuration resulting in the minimum DF values for every joint is obtained, all joint are believed to have been moved to their approximately correct 3D positions. As continuity between recovered postures at each frame is strictly guaranteed by the two 3D predictors in the DF, the whole reconstructed human motion will be smooth, and looks natural from any viewpoint in 3D space.

4.2. General Motion Reconstruction

Next we attempt to reconstruct unrestricted human motions. Two types of movements have to be taken into consideration: the relocation of the human body which is represented by the translations of the *pelvis* joint, and the rotations of every intermediate joint.

Translations parallel to the image plane is easy to implement, since in this case the distance between the human object and the camera is fixed. The DF for recovering such translations is defined as:

$$DF_{translation_1} = deviation_{position_pelvis} \quad (6)$$

$$deviation_{position_pelvis} = D_2(P_{pelvis_i}, P_{pelvis_p}) \quad (7)$$

where P_{pelvis_i} and P_{pelvis_p} are image and projection features of *pelvis* respectively.

However, usually translations perpendicular to the image plane are also present in the movement of the *pelvis*. Such translation is much more difficult to

handle since it is not easy to detect when and how such translation exactly happens. It is a common knowledge that the distance of an object with the projection plane determines its projected sizes on the image plane. Such understanding is used to determine the translation of the *pelvis* perpendicular to the image plane. As 3D human postures in consecutive frames should appear similar, we can first assume the human posture already recovered for the $(K-1)^{th}$ frame and the posture to be recovered in the K^{th} frame to be “the same”. Before translating the *pelvis* in the K^{th} frame, the 3D posture configuration obtained at the $(K-1)^{th}$ frame can be applied to the skeletal model temporarily. If the skeleton model is translated to its correct location at the K^{th} frame, the sum of the projected segment lengths should be equivalent to that of the image features. The following DF is then defined to perform the body translation perpendicular to the image plane:

$$DF_{translation_2} = deviation_{total_length} \quad (8)$$

$$deviation_{total_length} = abs\left(\sum_{i=1}^{16} \|S_{i_i}\| - \sum_{i=1}^{16} \|S_{i_p}\|\right) \quad (9)$$

where S_{i_i} and S_{i_p} represent lengths of the image feature and projection feature of the i^{th} segment respectively.

If more than 3 frames prior to the current frame have been recovered, the average perpendicular translation factor of the *pelvis* at those frames is used as an additional deviation factor for further refinement.

After the translation of the joint *pelvis*, the 3D skeletal model is considered as positioned to the right location in 3D space. Rotations of every joint will then follow using the methods described in Section 4.1. As the *pelvis* position changes in most of the frames, to recover the human posture through the same DFs as discussed in Section 4.1, a new reference coordinate system (RCS) is introduced. It is defined after the translation of *pelvis* in every frame, with its origin located at *pelvis* and its three axes parallel to those of WCS. Such RCS is used as the substitute of the “WCS” for calculating the 3D predictors in MTP. Based on the RCS, rotations of all intermediate joints can be derived.

5. EXPERIMENTS AND STATISTICS

Varies experiments on both the computer synthesized animation and real video are conducted to test the accuracy of the MTP technique. Monocular sequences with resolution of 640x480 and frame rate of 20fps are used.

5.1. Results from “Synthesized” Data

In this section, the MTP technique is evaluated on computer synthesized monocular video data. During

the process a parameter search routine is performed to find the best WP settings to be used in the DFs.

Firstly, two sequences (stretching and sneaking) are reconstructed, both of which mainly concern the motions nearly parallel to image plane. Part of the sneaking action is illustrated in Fig.4.

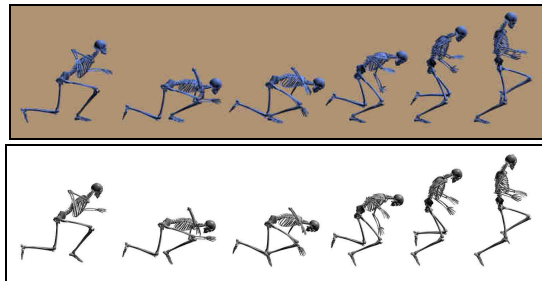


Figure 4: Top: Input motion (Sneaking); Bottom: Reconstruction from the same view direction;

The statistics data obtained during reconstruction from above sequences is shown in Fig.5, where the total 2D residuals of all the 17 joints between image and projection features are presented in form of stacked line. The horizontal axis represents the 17 joints in numerical order as defined in Section 2. It can be seen from Fig.5 that for both sequences, the maximum total residual is below 3 pixels, which is highly satisfactory given the image resolution of 640*480.

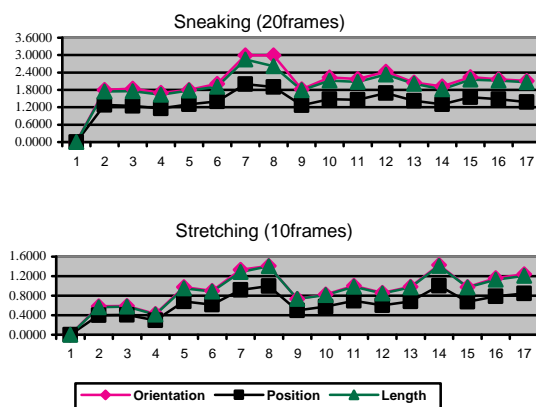


Figure 5: Total 2D deviation value at all frames

Next our efforts are put into animating a human kicking action (22frames), where the joint *pelvis* is moving all the time and the motion is more flexible in 3D space. As shown in Fig.6, the reconstructed results closely resemble the original motion from the same viewing angle.

Fig.7 illustrates the total value of DFs of all joints at each frame for the kicking sequence. Such value reaches its peak at the 11th frame, which is 1.660629 in pixels. Again the result indicates that the projection of the recovered 3D posture at each frame

is very close to the original, given the image resolution of 640*480.

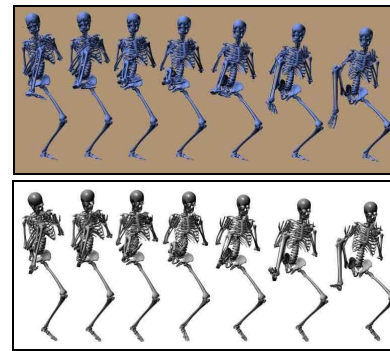


Figure 6: Top: Input motion (Kicking); Bottom: Reconstruction from the same view direction.

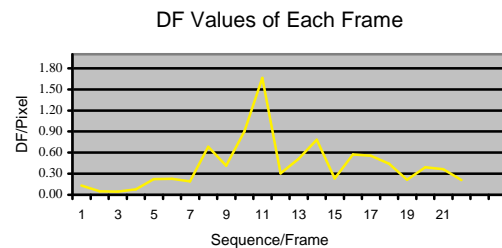


Figure 7: Total DF value at all frames (Kicking)

Since the sequence is computer synthesized, the actual 3D position of every joint is available. Thus the comparison between the original and the reconstructed postures could be done in 3D. In Fig.8 the original and reconstructed motions are still very similar when viewed from another angle.

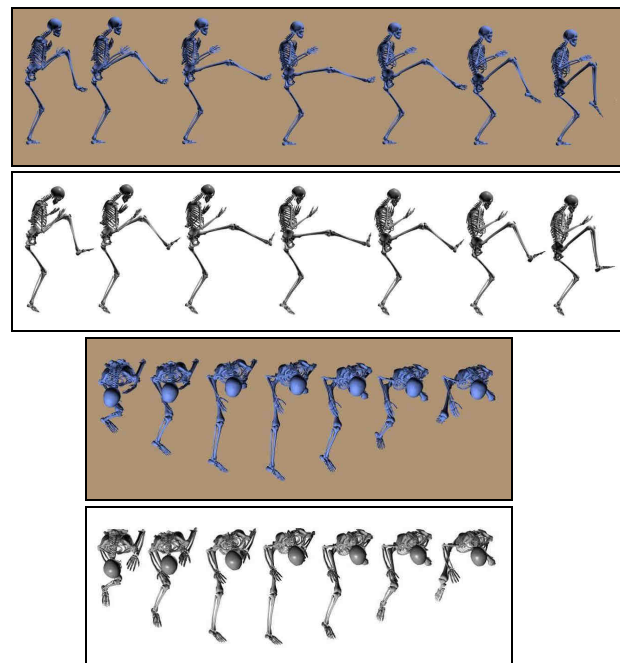
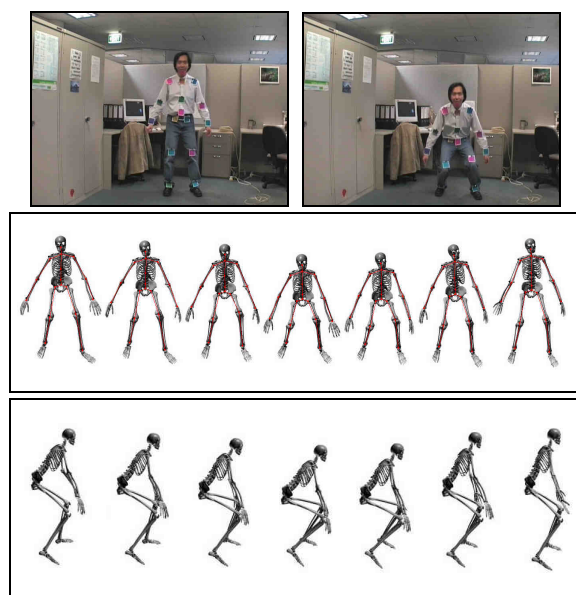


Figure 8: Input and reconstructed motion from other viewing directions

5.2. Results from Real Video Data

Next the technique is tested on real monocular video sequences. One motion sequence composed of walking and squatting (49frames), and the reconstructed human animation based on it are presented in Fig.9. To ensure accurate image feature extraction from real video sequence and to reduce unnecessary noises, color labels are stuck to the human object at joint positions. Image processing techniques such as those mentioned in [BK03, JB04, MRC05] are used to extract the 2D joint features from each frame of the video sequences; however we can only guarantee the noises will be minimized as possible as we can, which will affect the final 3D motion reconstruction.



**Figure 9: Top: Two frames from the input video
Middle: Reconstructed motion - front view
Bottom: Reconstructed motion - side view
(Walking and Squatting)**

Fig.10 shows the total values of DFs of all joints at each frame of this sequence. Compared with Fig.7, the total values of DFs for real video reconstruction is much higher than those from “synthesized” data. However the maximum DF sum value is only 6.28112 at the 18th frame in a resolution of 640*480. The reconstruction from real video still can be considered as highly accurate.

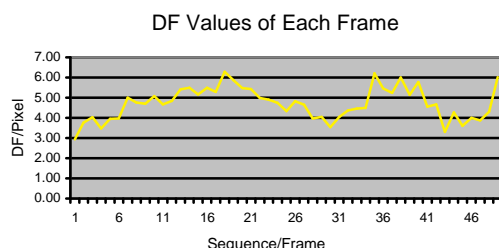


Figure 10: Total DF value at all frames (Walking and squatting)

6. CONCLUSION

A model-based technique is proposed in this paper for motion reconstruction from un-calibrated monocular video sequences containing unrestricted human motion. MTP technique is first developed to reconstruct human motion with no body relocation. The technique is then extended to derive a new RCS at each frame, and hence enable reconstruction of any unrestricted human motion.

The main advantage of our approach is that through it a truly wide range of monocular sequences could be reconstructed, and there is no requirement for camera calibration. From experimental results presented in the paper, the reconstruction results are highly satisfactory as long as the 2D image features are reasonably accurate.

As a future work we are planning to introduce control on the leaf joint into MTP. Plausible motions about these parts are expected to be simulated.

REFERENCES

- [AT04] Agarwal A., Triggs B.: Learning to track 3D human motion from silhouettes. In *Proceedings of the 21st International Conference on Machine Learning* (July 2004), pp. 9-16.
- [BK03] Barrón C., and Kakadiaris I. A.: A convex penalty method for optical human motion tracking. *First ACM SIGMM international workshop on Video surveillance, IWVS'03*, pp. 1-10.
- [CB04] Callenec B. L., Boulic R.: Interactive motion deformation with prioritized constraints. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aug. 2004), pp. 163-171.
- [DTJR01] David E. DiFranco, Tat-Jen Cham, James M. Rehg.: Reconstruction of 3D figure motion from 2D correspondences. In *Proceeding of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, 2001.
- [Fëd03] Fdor M.: Application of inverse kinematics for skeleton manipulation in real-time. In *Proceedings of the 19th spring conference on Computer graphics* (Apr. 2003), pp. 203-212.
- [FRDC04] Favreau L., Reveret L., Depraz C., Cani, M. P.: Animal gaits from video. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aug. 2004), pp. 277-286.
- [GODC05] Gibson D. J., Oziem D. J., Daltion C. J., Campbell N. W.: Capture and synthesis of insect motion. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aug. 2005), pp. 39-48.
- [HNHQ94] Holt R. J., Netravali A. N., Huang T. S., Qian R. J.: Determining Articulated Motion from Perspective Views: A Decomposition Approach. *Pattern Recognition, Vol. 30* (1997), pp. 1435-1449.
- [HOB99] Hodgins J. K., O'Brien J. F., Bodenheimer R.

- E.: *Computer Animation*. In *Wiley Encyclopedia of Electrical and Electronics Engineering*, John G. Webster, ed., v. 3, 686-690, 1999.
- [IC05] Ieronutti L., Chittart L.: A virtual human architecture that integrates kinematic, physical and behavioral aspects to control h-anim characters. In *Proceedings of the tenth international conference on 3D Web technology* (Mar. 2005), pp. 39-48.
- [JB04] Juan C.D., Bodenheimer B.: Cartoon texture. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aug. 2004), pp. 267-276.
- [KM04] Kurihara T., Miyata N.: Modeling deformable human hands from medical images. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aug. 2004), pp. 355-363.
- [LZP99] Liu Xiaoming, Zhuang Yueting, Pan Yunhe.: Video based Human animation technique. In *Proceeding of the 7th ACM International Conference on Multimedia* (Oct. 1999), pp. 353-362.
- [LZZP04] Li Chuanjun, Zhai Peng, Zheng S. Q., Prabhakaran B.: Segmentation and recognition of multi-attribute motion sequences. In *Proceeding of the 12th ACM Annual International Conference on Multimedia* (Oct. 2004), pp. 836-843.
- [MG01] Moeslund T. B., Granum E.: A Survey of Computer Vision-based Human motion capture. *Computer Vision and Image Understanding*, Vol. 81, Issue 3 (Mar. 2001), 231-268.
- [MRC05] Müller M., Röder T., Clausen M.: Efficient content-based retrieval of motion capture data. *ACM Transactions on Graphics* 24, 3 (July 2005), 677-685.
- [PCS02] Park M. J., Choi M. G., Shin S. Y.: Human Motion Reconstruction from inter-frame feature correspondences of a single video stream using a notion library. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (July 2002), pp. 113-120.
- [PW99] Popovic Z., Witkin A.: Physically based motion transformation. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (July 1999), pp. 11-20.
- [RGBC96] Rose C., Guenter B., Bodenheimer B., Cohen M. F.: Efficient generation of motion transitions using spacetime constraints. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (Aug. 1996), pp. 147-154.
- [RR03] Remondina F., Roditakis A.: Human figure reconstruction and modeling from single Image or monocular video sequence. In *Proceeding of the 4th International Conference on 3D Digital Imaging and Modeling* (Oct. 2003), pp. 116-123.
- [SJ04] Sminchisescu C., Jepson A.: Generative modeling for continuous non-linearly embedded visual inference. In *Proceedings of the 21st International Conference on Machine Learning* (July 2004), pp. 9-16.
- [Tay00] Taylor C. J.: Reconstruction of articulated objects from point correspondences in a single image. *Computer Vision and Image Understanding*, Vol. 80, No. 3 (Dec. 2000), 349-363.
- [TSF04] Tsang W., Singh K., Fiume E.: Helping hand: an anatomically accurate inverse dynamics solution for unconstrained hand motion. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aug. 2004), pp. 319-328.
- [WB95] Welch G., Bishop G.: An introduction for kalman filter. Technical report TR 95-041, Department of Computer Science, University of North Carolina at Chapel Hill, 1995.
- [WBF87] WITKIN A., BARR A., FLEISCHER K.: Energy constraints on parameterized models. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (Aug. 1987), pp. 225-232.
- [ZL04] Zhao Jianhui, Li Ling.: Human motion reconstruction from monocular images using genetic algorithms. *Computer Animation and Virtual World*, Vol. 15, No. 3-4 (July 2004), 407-414.

Density estimation optimizations for global illumination

R. Garcia, C. Ureña, J. Revelles, M. Lastra, R. Montes
Dpto Lenguajes y Sistemas Informáticos
University of Granada
{ruben,curena,jrevelle,mلاstral,rosana}@ugr.es

ABSTRACT

Density estimation on the tangent plane (DETP) is a density estimation technique for global illumination. This technique is based on Photon Maps and provides increased accuracy when the surfaces are not locally smooth or continuous. However, the performance of the technique is limited by the large number of ray-disc intersections needed. Some optimizations which increase the performance of DETP have been devised. The first optimization works by creating a set of candidate rays for each radiance calculation. The second optimization uses spatial indexing of the discs around the radiance calculation points. An analytical study of the order of complexity of the algorithms, as well as an heuristic study of the calculation time for the different values of the parameters involved, has been performed. Some rules are given in order to identify the most suitable optimization for a given radiance calculation.

Keywords: Global Illumination, Density Estimation, Range Searching.

1 INTRODUCTION

Density estimation performance depends on the rapid calculation of which rays contribute to a zone in the scene. Current techniques use space partitioning schemes to achieve interactive speeds. Most techniques index the rays in order to calculate density estimation on each of the points in the scene. Our approach, which is based on Density Estimation on the Tangent Plane, uses advanced space partitioning techniques to index the zones which affect each point where density estimation is being calculated and obtains better performance than existing techniques when the zones are relatively small.

In Density Estimation rays are usually included in a spatial indexing in order to provide a fast means to locate rays which affect a given position. Photon Maps [Jen01] indexes the ray impacts using a kd-tree. Sphere cache [LURM02] creates a list of spheres which contain rays. Havran et al. use a lazily constructed kd-tree to index the rays [HBHS05].

In Raytracing (from the eye), most applications create a spatial indexing for the objects in the scene. [AK89] and [Hav01] provide a compendium of techniques.

Our approach consists in applying a spatial indexing of the discs used in Density Estimation on the Tangent Plane.

Some work has been done on applying statistical and analytical methods to calculate the efficiency of spatial indexing used in global illumination algorithms. [HPP00] uses statistical methods to study different spatial indexing techniques for ray tracing and [HP03] presents a framework which eases the comparison among different optimization techniques.

The structure of the article is the following: Section 2 describes the most common density estimation techniques, beginning with simple techniques whose limitations are dealt with by more complex algorithms. Finally our algorithms are described in some detail. An optimization of DETP, disc indexing, is introduced. This technique provides increased performance when the distance between points in which the radiance is being calculated is in the order of magnitude of size of the kernel of the density estimation function. However, the main advantage of this technique is that all existing spatial indexing techniques can be trivially modified to support this approach. Section 3 contains an empirical comparison of time efficiency between different optimizations of Density Estimation on the Tangent Plane, using a test scene. Section 4 has a time efficiency study on the different algorithms. The probabilities of ray object intersection are used to calculate the expected time of the different algorithms, and an order of efficiency using the $O()$ notation is derived for each algorithm. Finally the most important conclusions are summarized and our future work is outlined.

2 DENSITY ESTIMATION METHODS

In order to obtain a radiosity value at a point, an approximation of the integral of the incident radiance for all the directions towards that point must be calculated.

The most basic method is described by Arvo in [Arv86] and Patanaik in [PM92]. It computes 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.

Short communications proceedings, ISBN 80-86943-05-4
WSCG'2006, January 30 – February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

impacts of the photons in the patches and calculates the energy density in those patches. Then vertexes are assigned a radiance which is the average of the patches to which they belong.

One method which has proven useful to obtain an estimate of the integral is the Density Estimation Method, popularized by [WHS97] and [Jen96]. This method consists of three phases. The first phase is based on the particle model of light, and traces a number of photons from the light sources. The second phase (Density Estimation proper) estimates the radiance. The third phase, decimation, simplifies the geometry after illumination has been calculated. This last phase is dropped often because it is considered to be outside the scope of density estimation.

Jensen [Jen96], devised the well known Photon Maps method. It consists in finding the nearest n ray impacts (n is predefined) to the point where radiance is being estimated, adding their energy, and dividing it by the area of the greatest circle of the sphere which contains the n impacts.

The most known limitation of Photon Maps, is that when radiance on a point is calculated, the surface in the neighbourhood should be relatively planar and large. [HP02] presents an algorithm which solves this limitation by using geometry information near the point.

Another less known limitation of Photon Maps and [HP02], mentioned in [LURM02] and [HBHS05], is that if relatively very small surfaces exist in the scene, these zones have a comparatively high variance, and they tend to appear either too bright (in a few cases) or too dark (which is more frequent) if the number of photons is not large enough.

2.1 Density Estimation on the Tangent Plane

A method to avoid the high variance of Photon Maps mentioned in the previous section consists in storing the rays in the scene and using a fixed size disc centered in the point where radiance is being calculated, and contained in the plane tangent to the surface. Rays intersecting a given disc are used to calculate the radiance at the point on which the disc is centered [LURM02]. The algorithm is called Density Estimation on the Tangent Plane (DETP). Note that this algorithm keeps track of the trajectory of the photons (origin, direction and impact point) unlike the original Photon Maps. See Figure 1. To avoid self-shadowing in concave surfaces, the second intersection of the ray and the scene is used instead of the first. This method uses discs of fixed radius [LURM02].

The algorithm has optimum trade-off between accuracy and variance when the disc radius (which is a user defined constant) is in the order of magnitude of half the distance between irradiance calculations. If disc radius were smaller, rays intersecting the tangent plane near

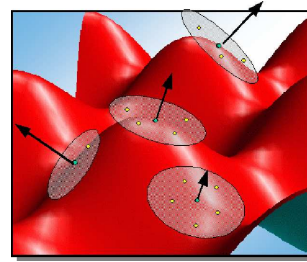


Figure 1: Density Estimation on the Tangent Plane

the middle point of two irradiance calculations would be ignored. If it were larger, intersections would be used for various calculations, creating artificial smoothing.

2.2 Sphere cache

The limitation of DETP is that the number of disc-ray intersections is high, therefore increasing the computation time. To address this, the sphere cache [LURM02] was developed.

The sphere cache consists in creating a hierarchy of spheres of decreasing radius and storing the rays which intersect each sphere in order to decrease the number of ray-disc intersection tests.

Firstly, a sphere tangent (i.e. circumscribed) to the bounding box of the scene is built. This sphere intersects all the rays.

Then, as figure 2 shows, spheres of decreasing radius are built one inside the other (the ratio between two consecutive spheres is a parameter called Q), until the radius is just above the disc radius mentioned in the previous section.

Each sphere has an associated data structure which contains the rays which the aforementioned sphere intersects. These rays are calculated by the intersection of the sphere with the rays in the immediately enclosing sphere.

The first point at which radiance is to be calculated is the center of the spheres of decreasing radius. Therefore, the first disc is contained in the inner sphere. Irradiance can be calculated by checking which rays in the inner sphere intersect the disc as well, and adding their energy. The number of ray-disc intersection tests is clearly reduced.

For the rest of the points, if the disc centered in the point is contained in the inner sphere, the disc is intersected against the rays in this sphere. Otherwise, the sphere is discarded, and the rest of the spheres are tested in order, until one is found to enclose the disc. Then the hierarchy of spheres is recalculated, using this point as center. See Figure 2 right.

Finally, the disc is intersected against the rays in the innermost sphere, in the same way as when no recalculation of spheres is needed.

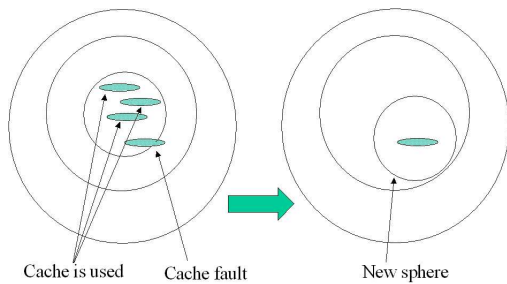


Figure 2: Sphere cache

Lastra et al. [LURM02] demonstrated that the use of space filling curves to reorder the points increments spatial coherence, and therefore reduces computation time. This approach is called point sorting.

2.3 Disc indexing

The disc indexing technique creates a spatial indexing of the discs in the scene. This is accomplished by considering the discs as real geometry, and applying a space partitioning method to them. The discs are initialized with a radiance value of zero. Then the rays traverse the spatial index adding their contribution to the discs they intersect. See Figure 3. The ray need

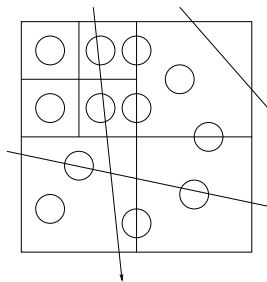


Figure 3: Disc indexing

only be followed until the first intersection with the real scene (or the second if concave surfaces exist). The spatial indexing should be able to store discs and to calculate efficiently all the intersections with a segment (the endpoints of this segment are the origin of the ray and the intersection with the real scene). All the published algorithms meet this criterion.

After radiance has been calculated, each disc contains an estimate of the radiance according to the DETP scheme. The data structure can be considered a sort of irradiance cache [WRC88]; therefore new irradiance values can be estimated using the same interpolation which that paper proposes.

Some work [HP03] has been done on studying characteristics of the scene which make some indexing techniques more efficient than others. Other studies [HPP00][RLGM03] use a fast simulation with few rays to choose the most appropriate indexing method. Since the disc position follows the surface of the objects, this research is applicable for this technique as well.

This method has higher performance than the original sphere cache intersection method when the discs have a radius which is in the order of magnitude of the mean distance among the points in which radiance is being calculated. In other situations, the performance of the sphere cache is higher. Details are provided in Section 3.

3 EMPIRICAL COMPARISON

All the algorithms described in the previous sections have been implemented in our rendering system. The system calculates irradiance samples on the vertexes of the scene even though DETP can calculate irradiance on any surface point.

The first test scene can be seen in Figure 4 (left). It contains 72 500 triangles, and is called the first tree scene.

The second tree scene is a different type of tree and ground, with bigger triangles (Figure 4 (center)).

An axis aligned BSP Tree [SS92] is used in the first examples of this section, and an Octree is used in the latter, more complete comparison. These techniques were chosen because they provided good performance in the photosimulation phase.

The timing results for the BSP Tree are shown in Figure 5. In this article, all times are expressed in seconds.

Triangles in the ground have edges whose length is 2 % of the scene's length and triangles in the tree have edges of 1 %. It can be seen that for disc sizes under 4 % of the scene, disc indexing is faster than sphere cache.

For 20 000 photons, the results are only better for sizes of 1 % and 2 %.

The graph in Figure 5 shows disc indexing's performance decrease as the area of the discs increases. It can be seen that for the second scene the results are better using larger discs than in the first, due to the fact that the triangle size has increased.

In order to make a more complete comparison between the techniques, a series of experiments were conducted on the first scene. Time was measured for the different combinations of radius of the DETP disc (between 1% and 16%), maximum depth of the octree (between 5 and 8 levels) and number of photons (between 100 and 409 600).

The timing results are linear with respect to the number of rays for the executions of disc indexing with octrees of different depths. Sphere cache is also linear, but the time is slightly better than linear when the number of rays is small. This is due to the rays in the inner sphere fitting the processor's cache. According to the experiments performed, the performance difference of sphere cache and disc indexing depends basically on the ratio of the distance between discs ($\overline{\Delta x}$) to the disc radius (d). If we call \bar{r} this number, $\bar{r} = \overline{\Delta x}/d$, and bearing in mind that the performance of the algorithms changes



Figure 4: First and second tree scenes, and large atrium

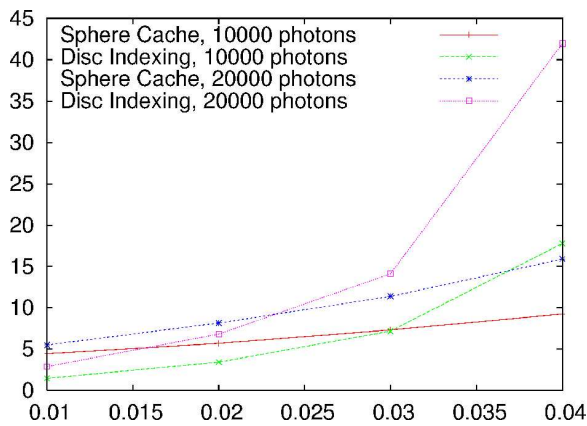


Figure 5: Time in seconds of radiance calculation using sphere cache and disc indexing for the first scene, as a function of disc radius

smoothly with this number and therefore the divisions are not sharp, three regions can be recognized.

- The first corresponds to $\bar{r} \leq 2$, in which disc indexing is better than sphere cache. The difference is more significant as disc radius decreases.
- The second corresponds to $\bar{r} \geq 6$, in which the situation is the opposite.
- The third is the intermediate situation $2 < \bar{r} < 6$. In this case, if the rays in the inner sphere fit the processor's cache, sphere cache is faster due to the coherence of this algorithm. Otherwise, disc indexing is faster, because the performance per ray of sphere cache lowers.

Figure 6 gives the performance of the techniques for each case. In a third, more complex scene, which can be seen in Figure 4 (right), disc indexing obtained reductions in time of up to 50 %. In this scene, the mean distance between points is $0.01\hat{3}$ and the disc radius is 0.01; $\bar{\Delta x}/d=1.1\hat{3}$, therefore it belongs to the first zone.

Summarizing, it is worth noting that the performance of disc indexing decreases faster than that of sphere

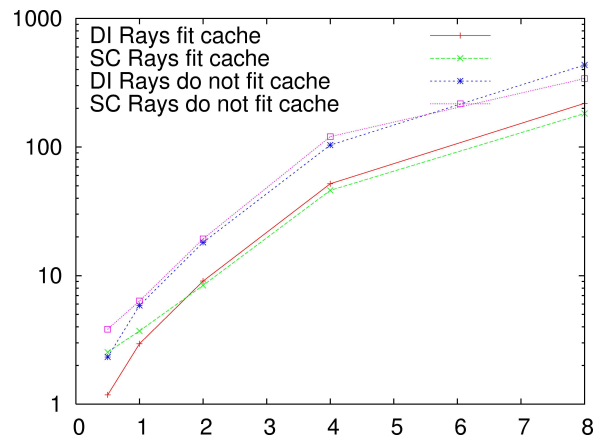


Figure 6: Zones in which each technique is optimal. Time as a function of $\bar{r} = \bar{\Delta x}/d$

cache as disc size grows and that the memory required by disc indexing grows extremely fast as disc size grows, making disc indexing unusable for large discs. For very small radii, disc indexing at maximum depth is always better than sphere cache. It should be noted as well that the spatial partitioning should end when the size of the voxel is similar to that of the disc radius.

4 THEORETICAL STUDY

These algorithms have the same underlying Density Estimation Technique (DETP). Therefore, for any given scene and set of rays, they will calculate the same solution (albeit using different computation time). In this case, the efficiency of the algorithms can be compared using computation time as a function of the size of the scene (that is, the number of discs and their size) and the number of rays.

Of the three algorithms (raw DETP, sphere cache with point sorting and disc indexing), only sphere cache with point sorting and disc indexing are discussed, since they have the highest performance. Notation is sum-

Ray related quantities	
R	Set of Rays
$n_R = \#R$	Number of rays
Sphere related quantities	
$S = \{S_i\}$	Set of Spheres
r_i	Radius of S_i
$r_i = r_{i-1}Q = r_0Q^i$	
$V_i = \frac{4}{3}\pi r_i^3$	Volume of S_i
m_i	Number of recalculations of sphere S_i with point sorting.
k	Number of spheres
Time related quantities	
u	Ray Disc intersection time
t	Ray Sphere intersection time
T_R	Time to recalculate the spheres with point sorting
T_I	Time to intersect the disc against the inner sphere
Other symbols	
$0 < Q < 1$	Ratio of the radii of two spheres
$P = \{P_i\}$	Set of Irradiance samples
$n_P = \#P$	Number of Irradiance samples
d	Disc Radius

Table 1: Symbols used in this article

marized in Table 1. The efficiency of raw DETP is clearly $O(n_R n_P)$. For sphere cache with no point sorting, the points at which the radiance is calculated are assumed to follow a uniform distribution. This prevents sphere cache from obtaining better performance. It can be shown that the algorithm is $O(n_R n_P)$, with a hidden constant slightly over 1. The efficiency of the other two algorithms is discussed in detail in the following sections.

An asymptotical analysis of the performance of the algorithms in the average case is useful to check scalability. Scene complexity (here measured as the number of radiance calculations) and illumination complexity (number of rays) are used as the variables for the study.

To calculate the expected time for the algorithms mentioned in the previous sections we need an estimation of the number of ray-disc intersections, which depend on the distribution of rays. We will suppose that the rays are uniformly distributed in space. This assumption is essential to obtain mathematical formulae for intersection probability. It is a common assumption [ABCC02, ABCC03], also implicitly used in [RKJ96] which works well in practice although it does not correspond exactly to reality.

4.1 Mean time using sphere cache with point sorting

The probability that a ray (with uniform distribution) which intersects a convex body, intersects a second convex body located inside the first can be derived from re-

sults of integral geometry from Santalo [San02], and is the ratio of the areas of the bodies.

In the case of sphere cache, if the distribution of the rays is uniform, the number of rays which intersect the inner sphere is independent of the location of the sphere, and is proportional to the quotient of the square of the radii of the spheres. The number of rays which intersect S_i is then

$$n_i = n_R \frac{r_i^2}{r_0^2} = n_R Q^{2i} \quad (1)$$

The cost of recalculating a sphere S_i once is product of the number of rays in the surrounding sphere multiplied by the ray-sphere intersection test:

$$t_i = t n_{i-1} \quad (2)$$

where t is the ray sphere intersection time.

Now we need to know how many spheres there are. Spheres are created with decreasing radius, until the radius of the sphere is just above the disc radius (i.e. the next sphere would have a smaller radius than the disc).

Let k be the number of nested spheres in the sphere list. To calculate k , we will use d as the disc radius. Recall (Section 2.2) that the quotient between the radii of two adjacent spheres is Q , and that spheres are built until their radius is just above the disc radius d . The value of k should comply with the following equations:

$$r_k = Q^k r_0 \geq d \quad ; \quad r_{k+1} < d \quad (3)$$

Therefore k can be calculated as:

$$k = \left\lceil \log_Q \left(\frac{d}{r_0} \right) \right\rceil \quad (4)$$

The cost of intersecting the disc with the rays in the inner sphere, T_I , is:

$$T_I = u n_k n_P = u n_R Q^{2k} n_P = \quad (5)$$

$$u n_P n_R Q^{2 \lceil \log_Q (\frac{d}{r_0}) \rceil} \lesssim u n_P n_R \frac{d^2}{r_0^2} \quad (6)$$

Since sphere cache is not useful unless the locations of the points in which the radiance is calculated are coherent, a space filling curve is used to sort the points. The z-order or Lebesgue curve is used in this algorithm.

The sorting algorithm divides the cube in 2^{48} cubes. Each small cube has an X, Y and Z coordinate with 16 bits each, ranging from 0 to 16383. These 3 coordinates are concatenated to form a 48-bit number. A function then transforms this number by reordering the bits in the following way: the least significant bits of each coordinate correspond to the three least significant bits of the new one, and this process is repeated until the most significant bit is reached. An example with 3 bits / coordinate would be:

- Original: $(x_2 x_1 x_0 \mid y_2 y_1 y_0 \mid z_2 z_1 z_0)$
- Reordered: $(x_2 y_2 z_2 \mid x_1 y_1 z_1 \mid x_0 y_0 z_0)$

Now the cubes are visited from 0 to $2^{48} - 1$. Changing the least three significant bits means moving the point to a distance whose length does not exceed that of the diagonal of the cubes, $2 * \sqrt{3} * 2^{-16}$, and it is also not less than that of its side, $2 * 2^{-16}$. A sphere with a radius in that order of magnitude would be recalculated at most 2^{48} times. It can be seen that a sphere of radius $2^{-m}r_0$ is recalculated 2^{3m} times. Therefore, since

$$r_i = Q^i r_0 = 2^{-m} r_0 \quad (7)$$

$$m = -\log_2 Q^i \quad (8)$$

sphere S_i is recalculated $m_i = 2^{3*(-\log_2 Q^i)} = Q^{-3i}$ times.

This is the maximum number of times the sphere is rebuilt, and is independent of the number of points at which the density estimation is calculated. There is an independent limit on the number of times which it can be rebuilt, given by the number of irradiance samples. Sphere S_i is recalculated at most n_P times. The total recalculation cost in this case is

$$T_R = \sum_{i=1}^k \min(n_P, m_i) t_i \quad (9)$$

and the total cost is

$$T = T_R + T_I \quad (10)$$

T can be used to provide clues about what value for Q should be used. See Figure 7. The graph in the left shows the cost of recalculating spheres per ray (i.e. 1 would mean n_R intersections; equivalent to no optimization), as a function of the disc radius and Q . A small Q creates few spheres, so the time is small. The disc radius influences the length of the sphere list, but the smallest spheres have very few rays and therefore add little to the time, so the differences in time are small.

Figure 7 (center) provides a graph of the time it takes to intersect the disc to the rays in the inner sphere. This time only depends on the radius of the inner sphere, which depends on Q : the radius is r_k such that $r_k = r_0 Q^k \geq d$ and $r_{k+1} < d$. This graph shows that there is an infinite number of optimum Q ; but if numeric instability makes obtaining the minimum values difficult, Q should be as near as possible to one.

A high Q provides inner spheres which wrap tightly around the discs. On the other hand, the number of recalculations is high. A low Q provides loose fits to the discs; therefore there will be many rays in the inner sphere and the calculation time will be higher. The spikes in the graph correspond to the inner spheres which wrap perfectly around the discs, i.e., $d = r_k$. The local minima of this function have all the same value, which is the time to intersect the rays in a sphere of radius d : $u n_R d^2 / r_0^2$.

Figure 7 (right) provides the sum of the two previous graphs. It forms a U-shaped function, as was expected

of the two composing graphs, and shows how changing the disc radius makes the optimum value of Q change smoothly to create an inner sphere which encases the disc. When Q becomes near to 1, the number of spheres increases exponentially and the time can be seen to diverge in $Q = 1$, which would mean an infinite number of spheres.

The figures show values between 0.6-0.7 as being a good compromise, because it is below 1 on Figure 7 (left) and near the point where the gradient becomes important, and the value in the graph of Figure 7 (center) is quite low also. The result is coherent with practical experiments [Las04].

It was seen above that for a sufficiently large n_P , the number of recalculations is fixed and the time depends only on n_R .

The time to intersect the disc to the rays in the inner sphere, on the other hand, does depend both on n_R and n_P and is therefore $O(n_R n_P)$. The hidden constant, d^2 / r_0^2 , can, in practice, make this algorithm quite efficient.

If the disc radius is approximately equal to distance between irradiance samples, which would be desirable according to Section 2.1, the efficiency of the algorithm can be proven to increase.

The supposition that the disc radius is approximately equal to the distance between samples will therefore be added.

For small n_P (so the limit on recalculations is not reached), since the disc radius is approximately the distance between irradiance samples, then k is approximately:

$$k \approx \log_Q \left(\frac{1}{\sqrt[3]{n_P}} \right) = \log_{Q^{-1}} \sqrt[3]{n_P} \quad (11)$$

Now, to calculate the order of efficiency of the algorithm, the value of some important quantities is shown, and finally the time of the whole algorithm is expanded and the efficiency is calculated. The value of n_i , number of rays in sphere i is

$$n_i = n_R Q^{2i} \quad (12)$$

The time to recalculate sphere i is

$$t_i = t n_R Q^{2i-2} \quad (13)$$

The number of recalculations of sphere i :

$$m_i = Q^{-3i} \quad (14)$$

The time to recalculate sphere i in the whole algorithm is

$$m_i t_i = t n_R Q^{-i-2} \quad (15)$$

Finally, the time of the whole algorithm is then

$$\sum_{i=1}^k m_i t_i = \frac{t n_R}{Q^2} \left(\frac{\sqrt[3]{n_P} - 1}{1 - Q} \right) = O(n_R \sqrt[3]{n_P}) \quad (16)$$

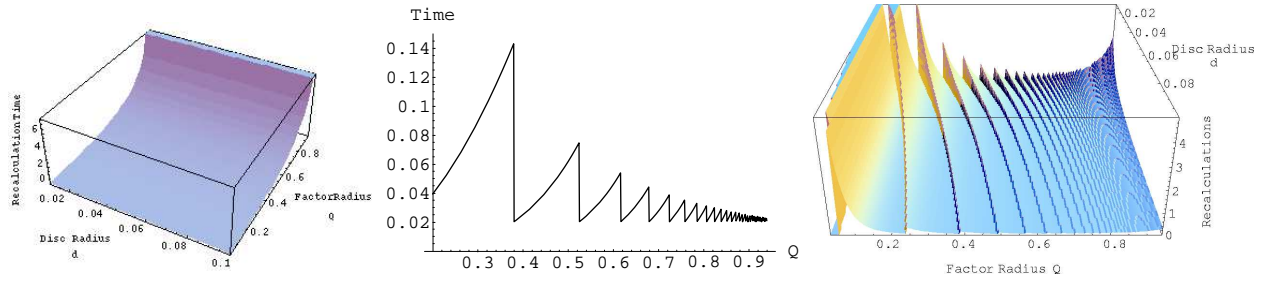


Figure 7: Left: Number of recalculations of sphere cache as a function of Q and Disc Radius. Center: Time to intersect the disc to the rays in the inner sphere, as a function of Q . Right: Recalculation time of sphere cache as a function of Q and Disc Radius.

4.2 Disc indexing

The disc diameter provides a useful minimum size for the voxels in the disc indexing technique. Dividing a voxel of this size creates voxels in which most of the discs belong to all child voxels. This makes the intersection time with the new voxels higher than the original combined voxel.

To illustrate the problem, imagine we were to calculate the radiance on each of the vertexes of the mesh in Figure 8. Remember though that DETP is independent of geometry. There is a disc centered in each of the vertexes of the scene. As the disc size grows, it becomes apparent that space partitioning will not help.

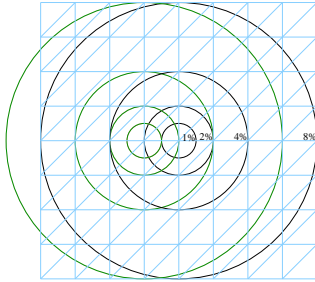


Figure 8: Discs in a typical mesh

Since disc indexing can be used with different indexing techniques, the order of complexity depends on the technique used. However, [RKJ96] proves that the order of complexity of a grid, a binary tree and an octree is the cubic root of the number of cells for the three techniques. [SKM98] studies the average complexity for ray shooting for other techniques. Evidence suggests that most indexing techniques have the same order of efficiency in the average case. We will use an octree as a indexing technique in our analysis due to its performance in generic scenes.

The relative size of the side of a voxel at a depth k with respect to the side of the whole scene is 2^{1-k} . If we set the disc diameter as the side of the smallest voxel, we get

$$2^{1-k} = 2d \quad (17)$$

$$k = \lfloor -\log_2 d \rfloor \quad (18)$$

There are therefore 8^k voxels. A uniform distribution of the irradiance samples mean there are $n_P/8^k$ samples per voxel, and therefore, the intersection time between one ray and one voxel is $n_P u/8^k$. Since each ray traverses a line of these voxels (2^k voxels), and the origin must be found by traversing the tree (k steps) the mean time for this method is

$$T = u k n_R n_P / 4^k \quad (19)$$

If we make the voxel size similar to the disc size, $k = O(\log_2 \sqrt[3]{n_P})$. The algorithm is then $O(n_R \sqrt[3]{n_P} \log n_P)$.

If we use a balanced space partitioning structure, according to [ABCC02], traversing to a neighbor node can be done in $O(1)$. Then the performance of disc indexing becomes $O(n_R \sqrt[3]{n_P})$.

This efficiency is higher than that of the sphere cache, which is $O(n_R n_P)$, for a large n_P . For small n_P , in which the distance between irradiance samples is similar to the disc radius, the efficiency is the same as that of sphere cache.

5 CONCLUSIONS

According to the empirical study of the techniques, the ratio of the mean distance between irradiance samples and the disc radius defines zones in which the algorithms are adequate. Disc indexing is optimal if this ratio is smaller than 2; sphere cache is optimal if this ratio is larger than 6. In the intermediate zone, sphere cache is better if the rays in the inner sphere fit the processor's cache. A universal optimal algorithm would calculate this ratio and the mean number of rays in the inner sphere and would select the best technique with this data.

The results of the theoretical study are the proof that basic DETP and sphere cache are $O(n_R n_P)$, and that sphere cache with point sorting, although $O(n_R n_P)$, has a hidden constant proportional to d^2/r_0^2 , which is the fraction of rays in a sphere whose radius is that of the discs. Since this value is quite small, the algorithm is quite fast in practice. For small scenes, this last algorithm was also proven to be $O(n_R \sqrt[3]{n_P})$. It was also proven that the radius factor should be between 0.6 and 0.7.

Disc indexing is $O(n_R \sqrt[3]{n_P} \log n_P)$ for unbalanced trees and $O(n_R \sqrt[3]{n_P})$ for balanced trees.

6 FUTURE WORK

We plan to use the BART [LAM00] benchmarking in the empirical study to investigate the border between the different useful zones of each algorithm. Some more studies should be done on other algorithms related to Density Estimation on the Tangent Plane, such as Variable Radius [Las04], which allows the size of the discs to change on account of the number of rays in the vicinity (like Photon Maps). In order to extend this approach to be able to compare it to different Density Estimation techniques (such as Photon Maps, or Ray Maps for Global Illumination [HBHS05] by Havran et al.), a study of variance and error should be added, and the algorithms compared by examining the error or variance as a function of computing time.

The framework presented in this article should be used as a basis for the automatic estimation of parameters in DETP. Finally more realistic models of the distribution of the irradiance samples and the rays should be studied.

7 ACKNOWLEDGEMENTS

This work has been supported by the research project coded TIN2004-07672-C03-02 (Spanish Commission for Science and Technology). We would like to thank Miguel Vega for his help, and anonymous reviewers for their comments.

REFERENCES

- [ABCC02] B. Aronov, H. Brönnimann, A. Y. Chang, and Y. Chiang. Cost prediction for ray shooting. In *SCG '02*, pages 293–302. ACM Press, 2002.
- [ABCC03] B. Aronov, H. Brönnimann, A. Y. Chang, and Y. Chiang. Cost-driven octree construction schemes: An experimental study. In *SCG '03*, pages 227–236. ACM Press, 2003.
- [AK89] J. Arvo and D. Kirk. *A Survey of Acceleration Techniques*, chapter 6, pages 201–262. Academic Press, San Diego, 1989.
- [Arv86] James R. Arvo. Backward Ray Tracing. In *ACM SIGGRAPH '86 Course Notes - Developments in Ray Tracing*, volume 12, pages 259–263, 1986.
- [Hav01] V. Havran. *Heuristic Ray Shooting Algorithms*. Ph.d. thesis, Department of Computer Science and Engineering, Czech Technical University in Prague, 2001.
- [HBHS05] V. Havran, J. Bittner, R. Herzog, and H.-P. Seidel. Ray maps for global illumination. *16th Eurographics Symposium on Rendering*, 2005.
- [HP02] Heinrich Hey and Werner Purgathofer. Advanced radiance estimation for photon map global illumination. *Computer Graphics Forum*, 21(3):541–546, 2002.
- [HP03] V. Havran and W. Purgathofer. On comparing ray shooting algorithms. *Computer and Graphics*, 27, Issue 4:593–604, August 2003.
- [HPP00] V. Havran, J. Přikryl, and W. Purgathofer. Statistical comparison of ray-shooting efficiency schemes. Technical Report TR-186-2-00-14, Institute of Computer Graphics and Algorithms, Vienna University of Technology, may 2000.
- [Jen96] H.W. Jensen. Global illumination using photon maps. In *Rendering Techniques '96*, pages 21–30. Springer-Verlag, 1996.
- [Jen01] Henrik Wann Jensen. *Realistic Image Synthesis using Photon Mapping*. AK Peters, 2001.
- [LAM00] J. Lext, U. Assarsson, and T. Moeller. Bart: A benchmark for animated ray tracing. Technical report, Dept. of Computer Engineering, Chalmers University of Technology, Goeteborg, 2000.
- [Las04] Miguel Lastra Leidinger. *Stochastic Rendering Techniques for Complex Environments*. PhD thesis, University of Granada, 2004.
- [LURM02] M. Lastra, C. Ureña, J. Revelles, and R. Montes. A particle-path based method for Monte-Carlo density estimation. *Poster at: 13th EUROGRAPHICS Workshop on Rendering*, 2002.
- [PM92] S.N. Pattanaik and S.P. Mudur. Computation of global illumination by Monte Carlo simulation of the particle model of light. *Proceedings of 3rd Eurographics Rendering Workshop, Bristol*, 1992.
- [RKJ96] Erik Reinhard, Arjan J. F. Kok, and Frederik W. Jansen. Cost prediction in ray tracing. In *Rendering Techniques '96*, pages 41–50. Springer-Verlag, June 1996.
- [RLGM03] J. Revelles, M. Lastra, R.J. García, and R. Montes. A formal framework approach for ray-scene intersection test improvement. In *WSCG'2003*, 2003.
- [San02] L. Santalo. *Integral Geometry and Geometric Probability*. Cambridge University Press, 2 edition, October 2002.
- [SKM98] L. Szirmay-Kalos and G. Márton. Worst-case versus average case complexity of ray-shooting. *Computing*, 61(2):103–131, 1998.
- [SS92] K. Sung and P. Shirley. Ray tracing with the BSP tree. In David Kirk, editor, *Graphics Gems III*, pages 271–274. Academic Press, 1992.
- [WHSG97] B. Walter, P. M. Hubbard, P. Shirley, and D. P. Greenberg. Global illumination using local linear density estimation. *ACM Transactions on Graphics*, 16(3):217–259, July 1997.
- [WRC88] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 85–92, New York, NY, USA, 1988. ACM Press.

A new model for 3D graphical rendering

Alessandro Martinelli
University of Pavia
Faculty of Engineering
Strada Ferrata Pavia PV(27100),Italy
AlexMartinelli@jumpy.it

ABSTRACT

One of the most important tasks of a traditional 3D Rendering engine is the projection on the image plane of geometrical structures (such as triangles or lines). This operation takes place in the middle of the rendering pipeline, between the vertex shader and the fragment shader: its aim is just that of creating fragment data from vertex data. The solution of the projection problem is necessarily bound to the solution of a great number of systems of equations, where the complexity of the equations is in general related to the properties of the geometrical structures. To make this process fast, the most adopted solution is that of using linear models, so that the systems become linear and the module gets the simplest implementation. Unfortunately, linear models have some limitations: the solution is to use approximation, but to get good models they are necessary a lot of linear structures, in particular a lot of triangles; modern 3D Rendering Engines may automate the process of converting non linear models in triangles, but this does not reduce the occupation of memory and doesn't eliminate linear approximation. In this article I consider a non linear model (the Lembo model) for geometrical structures in a 3D rendering engine: firstly I show the properties of the model; then I show an efficient algorithm to solve the projection problem directly on the model equations.

Keywords

Approximation Models, 3D Models, Rendering Algorithms, Non-linear Graphics, Real-Time Graphics

1. INTRODUCTION

I want to show the advantages in using a quadratic triangle model to work with the traditional polygonal model directly in the graphics hardware. Graphics is often concerned with approximation: if you want to represent a function $y = f(t)$, you may consider the linear approximation, or you may consider high order polynomial approximations. In 3D graphics they are used surfaces, and also surfaces may be approximated with many approximation techniques, but there still remains the problem of rendering; rendering is generally concerned with the solutions of systems of equations, and this may take an enormous computational cost. The GPU hardware works with triangles; recently, subdivision meshes have been introduced in the graphical hardware: but they make it possible only the refinement of triangular meshes (see [Bou01a], [Shi00a]).

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.

Short Communications proceedings ISBN 80-86943-05-4 WSCG' 2006, January 30-February 3, 2006 Plzen, Czech Republic. Copyright UNION Agency - Science Press

In general the approximation of a curved surface is concerned with the concept of LOD (Level of Detail), and with some algorithm to decide the triangles to be used to approximate the surface (see for example [Bru00a]).

Only in non-Real Time Graphics, such as for ray-tracing, some techniques are used to render non-linear elements directly without a triangulation process (see [Mar00a]). Our aim is to introduce non linear rendering techniques also in common real-time rendering pipelines.

Scan Line Rendering Techniques have been introduced with this aim. However, they have been thought to work with generical non linear models (for example [Lan00a],[Sch00a],[SAdDC12] e [Sed01a]). This Techniques are known to be slow and less amenable to hardware acceleration. In this article I show that it is possible to construct a fast scan line rendering algorithm for a new model which may be easily introduced in the accelerating parallel hardware. Such a model is a form of Quadratic Bezier Triangle (see [Bru00a]), with some other informations which make it very close to PN Triangles (see [Vla00a] or [Bou00a]) or Steiner Patches (see [Bre00a], [Sed00a]).

2. THE QUADRATIC FIXED-DOMAIN REFERENCED PN TRIANGLE (OR LEMBO) MODEL

The model I have worked on is a quadratic bezier triangle defined on a standard dominion (the triangle with vertices $(0,0)$, $(1,0)$ $(0,1)$) with six reference construction points (the points $(0,0)$, $(1,0)$, $(0,1)$, $(\frac{1}{2},0)$, $(0,\frac{1}{2})$, $(\frac{1}{2},\frac{1}{2})$), and a separate Normal function on the same domain. I use to call it also Lembo, an Italian word which may be translated as 'strip' or 'patch'.

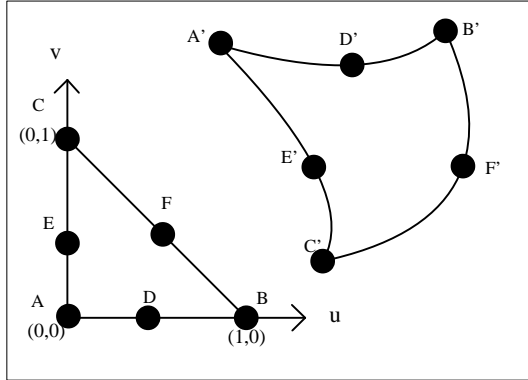


Figure.1 A lembo representation: there are the parametric function (top), the standard dominion with its reference points (bottom), the lembo image and the modeling points.

Such a model is no really new, but it may be considered as the compositions of different ideas from other models. Let us consider Steiner Patches and PN Curved Triangles.

Steiner Patches and Quadratic Bezier Triangles

A Steiner Patch is a Quadratic Bezier Triangles, with a fixed domain expressed in this way

$$\begin{cases} x(u, v) = a_x u^2 + b_x v^2 + c_x uv + d_x u + e_x v + f_x \\ y(u, v) = a_y u^2 + b_y v^2 + c_y uv + d_y u + e_y v + f_y \\ z(u, v) = a_z u^2 + b_z v^2 + c_z uv + d_z u + e_z v + f_z \\ u \geq 0 \\ v \geq 0 \\ u + v \leq 1 \end{cases}$$

The model is similar to Quadratic Bezier Triangles by Bruijns (see [Bru00a]), considering the projection onto the u-v plane of their domain (with the equation $w = 1 - u - v$).

A lot of work have been done to study the properties of similar models, for example in [Bar00a], [Bre00a] and [Sed00a].

Curved PN Triangles

A curved PN triangle (see [Vla00a]) uses two different functions to model the geometry and

normals of a patch. The Geometry of the PN Triangle is defined by a cubic patch b

$$\begin{aligned} b : \mathbb{R}^2 &\rightarrow \mathbb{R}^3 \\ b(u, v) &= \sum_{i+j+k=3} b_{ijk} \frac{3!}{i!j!k!} u^i v^j w^k \\ u, v, w &\geq 0 \\ u + v + w &= 1 \end{aligned}$$

The geometry function does not create C^1 continuity, but it is possible to simulate it with a normals function. The normal component of a curved PN Triangle is a quadratic function of the normal data, defined as

$$\begin{aligned} n : \mathbb{R}^2 &\rightarrow \mathbb{R}^3 \\ n(u, v) &= \sum_{i+j+k=2} n_{ijk} u^i v^j w^k \\ u, v, w &\geq 0 \\ u + v + w &= 1 \end{aligned}$$

The Lembo Model and the Mesh Refinement Techniques

The model I have consider is similar to Curved PN Triangles, but to model geometry I use a quadratic patch instead of a cubic one. The domain of the patch is fixed and it is in the same form of Steiner Patches.

$$\begin{aligned} b : \mathbb{R}^2 &\rightarrow \mathbb{R}^3 \\ n : \mathbb{R}^2 &\rightarrow \mathbb{R}^3 \\ b(u, v) &= \sum_{i \leq 2, j \leq 2} b_{ij} u^i v^j \\ n(u, v) &= \sum_{i \leq 2, j \leq 2} n_{ij} u^i v^j \\ u, v &\geq 0 \\ u + v &\leq 1 \end{aligned}$$

It's possible to construct Lembos meshes, using the six reference points for construction, or it's possible, as for PN Triangles, to use the Lembo

Model to refine a triangular mesh instead of creating a lembo mesh from scratch. For the geometry function it's necessary to find the edges middle points; this is possible for example using the Near Least Square Acceleration proposed in [Bar00a]; such a method produces both the control points and the control normals on the edges. An edge of a quadratic Bezier triangle between two points P_1 and P_2 with normals N_1 and N_2 may be expressed with a parameter t

$$f = at^2 + bt + c$$

The function f has to interpolate the ending points of the edge, and the tangents in the ending points should be tangent to the normal described in the ending points. This conduces to a four equations system. It is not possible to find an exact solution, but it is possible to find f so that the integral

$$\int_0^1 \|f''\|^2 dt$$

has the minimum value. This is the algorithm (see [Bar00a] for details).

$$\begin{aligned} T_1 &= N_2 - N_1(N_1 \cdot N_2) \\ T_2 &= -N_1 - N_2(N_1 \cdot N_2) \\ \alpha &= \frac{T_1 \cdot T_2}{T_1 \cdot T_1} \\ \beta &= \frac{P \cdot T_1}{T_1 \cdot T_2} \\ \alpha' &= \frac{T_1 \cdot T_2}{T_2 \cdot T_2} \\ \beta' &= \frac{P \cdot T_2}{T_1 \cdot T_2} \\ f(t) &= \left(\frac{\alpha' \beta' T_2 - \alpha \beta T_1}{2} \right) t^2 \\ &\quad \left(P + \frac{\alpha \beta T_1 - \alpha' \beta' T_2}{2} \right) t + P_1 \end{aligned}$$

where T_1 and T_2 are the tangent required in P_1 and P_2 and $P = P_2 - P_1$.

The points and the normals are evaluated using only information on the edge ending points, so the algorithm will give the same values for two patches with a common edge.

The construction of the normal function may be done also in the similar way used for PN Triangles; another possibility is to consider the geometry of two lembos with a common edge and to evaluate the average of the two normals in the middle point; this approach has the drawback to construct a lembo using data from other lembos.

Once the six points and six normals for the lembo have been found, it is possible to construct the Lembo function from a simple linear transformation, making the values of the lembo in the reference points $\{A(0,0), B(1,0), C(0,1), D(\frac{1}{2},0), E(0,\frac{1}{2}), F(\frac{1}{2},\frac{1}{2})\}$ be equal to the value of the constructing points $\{A', B', C', D', E', F'\}$ in the Point-Normal Space (for every point it's given the position (x', y', z') and a normal vector (n'_x, n'_y, n'_z)).

$$\begin{cases} A'_x = fx \\ B'_x = ax + dx + fx \\ C'_x = bx + ex + fx \\ D'_x = \frac{1}{4}a_x + \frac{1}{2}d_x + f_x \\ E'_x = \frac{1}{4}b_x + \frac{1}{2}e_x + f_x \\ F'_x = \frac{1}{4}a_x + \frac{1}{4}b_x + \frac{1}{4}c_x + \frac{1}{2}d_x + \frac{1}{2}e_x + f_x \end{cases}$$

$$\begin{cases} f_x = A'_x \\ e_x = 4(E'_x - A'_x) - (C'_x - A'_x) \\ d_x = 4(D'_x - A'_x) - (B'_x - A'_x) \\ b_x = C'_x - A'_x - e_x \\ a_x = B'_x - A'_x - d_x \\ c_x = 4F'_x - a_x - b_x - 2d_x - 2e_x - 4A'_x \end{cases}$$

where I have considered only the x component of the geometry function. It is the same also for the normal function, they are both quadratic.

3. RENDERING WITH TESSELLATION

One possibility for rendering is tessellation. Such technique is fast, in particular because it may be introduced directly in the actual hardware. Recently some GPUs have introduced a module to manage the subdivision meshes, and this allow the programmers to use common meshes in the CPU and get the mesh refinement only on the GPU, and this allow better performance (see [Bou01a],[Shi00a]). The subdivision techniques for a quadratic triangular patch have been discussed for example by Bruijns in [Bru00a], and they divide in fixed step techniques and variable step ones.

4. SCAN LINE RENDERING

Scan Line Rendering is a rendering technique that implies the solution of systems of non linear equations. A non linear model (in general a cubic patch) is intersected with a plane in the R^3 space. Generally they are made some assumptions: the screen plane is the x-y plane, with the z axis orthogonal to the screen; every projection transformation on the model points have already been evaluated. So a parametric model is intersected with the plane $y = y_j$, for a set of y_j which

are in the interval between the minimum and the maximum value for y on the surface. The steps of a scan line algorithm are:

```

; Scan Line Rendering Procedure

proc render(Patch)

   $y_j, y_{min}, y_{max}$ 

  [ $y_{min}, y_{max}$ ]=evalMinMax(Patch)

  for  $y_j = y_{min}; y_j \leq y_{max}; y_j++$ 

    ; Study the equation  $y_j = y(u, v)$ 

    renderEquation( $y_j$ );

  endproc

```

In general a scan line algorithm requires the use of a numerical method, or more then one, and the stability properties of the method depend on the model of the patch rendered. The use of a numerical method and the managing of numerical errors to guarantee stability make this algorithms generally slow.

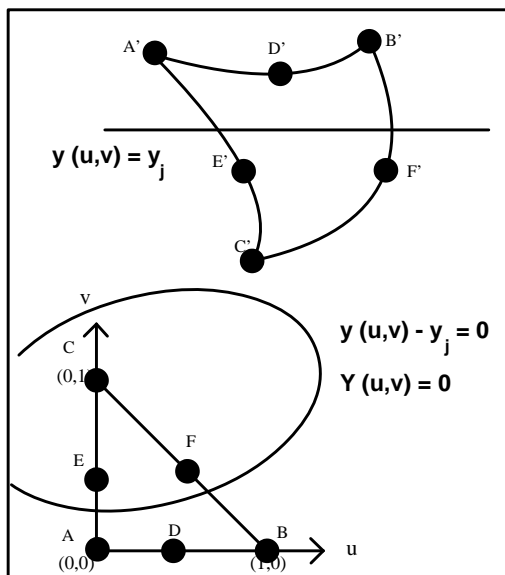


Figure.2 The $Y(u, v) = 0$ curve

5. A VERY EFFICIENT SCAN LINE ALGORITHM FOR LEMBOS

I'm going to show an efficient scan line algorithm for lembos. In particular this algorithm doesn't require numerical techniques, doesn't suffer stabilities problem and all is described in terms of solutions of second degree equations.

The $Y(u, v) = 0$ curve

The $Y(u, v) = 0$ curve has the form

$$Y(u, v) = a_y u^2 + b_y v^2 + c_y uv + d_y u + e_y v + f_y - y_j = 0$$

The equation represent a conic in the (u, v) space. In particular they have to be studied the arcs of this curve which are inside the domain. Along these arcs they are constructed points for the next rendering steps.

Finding maximum and minimum values for y

The maximum and minimum values for y are easy to find. They may be one of the following alternatives:

- (1) One of the Vertices A', B', C' of the Lembo
- (2) One of the maximum or minimum points for the edges, which are arcs of parabola. They may be one of the Vertices (already considered) or one of this values:
 - (a) $u = -\frac{d_y}{2a_y}, v = 0$
 - (b) $u = 0, v = -\frac{e_y}{2b_y}$
 - (c) $u = -\frac{-2b_y + c_y + d_y - e_y}{2(a_y + b_y - c_y)}, v = 1 - u$
- (3) The solution of the linear system of equations with the two partial derivatives functions equal to zero $2a_y u + c_y v + d_y = 0$ and $2b_y v + c_y u + e_y = 0$

The extreme method: linearization of the $Y(u, v) = 0$ curve

The idea is to work with extreme linearization: given o point of the $Y(u, v) = 0$ curve, it has to be found another point so that along the segment between the two points the maximum value of $\|Y\|$ (so the maximum error of the approximation of the curve piece with the segment) is exactly a predefined value k . This is very simple, because the $Y(u, v)$ function along the segment behaves as a parabola. So, the maximum error is easily found in the middle point. All the parts of the equation $Y(u, v) = 0$ inside the lembo domain are approximated with an array of segments, with every segment having its maximum error of approximation in the middle point and that error being exactly the one established. In general I suggest a pixel-oriented error choice, such as $k = \frac{1}{2} \text{pixel}$.

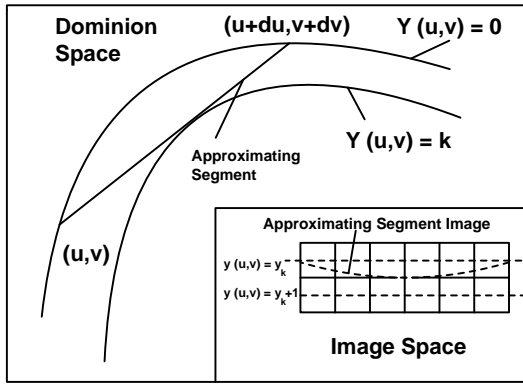


Figure.3 The segmentation of the $Y(u, v) = 0$ equation

Once the segments have been found, it is possible to consider the function $x = x(u, v)$. It is very simple to proceed with the same kind of approximation, trying to create a sub-segmentation, where every subsegment has both the property to have an approximatively constant value for y , with an error not higher than the one established, and to have an x with an approximatively linear trend, with also the error of approximation controlled and not higher than the established k . So the generation of fragments along the subsegment may be done with a simple linear equation for the u and v components, which may be evaluated step by step with simple increments. In the next sub sections I'm going to show the steps of this algorithm: the managing of the intervals, the construction of the segment and the construction and rendering of sub-segments.

Managing Intervals

The first problem is to find the pieces of the $Y(u, v)$ curve which are inside the domain. This may be done constructing intervals inside the domain polygon. The extreme points of this intervals have to be the intersection of the curve with the domain edges, so they may be

- (1) The solutions of the equations $a_y u^2 + d_y u + f_y - y_j = 0, v = 0$
- (2) The solutions of the equations $u = 0, b_y v^2 + e_y v + f_y - y_j = 0$
- (3) The solutions of the equations $(a_y + b_y - c_y)u^2 + (d_y - e_y + c_y - 2b_y)u + b_y + e_y + f_y - y_j = 0, v = 1 - u$

This points may be used as beginning and ending point for the next step, which is the segmentation one.

The segmentation problem

An approximation of the $Y(u, v) = 0$ equation can be evaluated step by step with segmentation. If

$P_0(u_0, v_0)$ is a point of the curve (inside the domain), to construct the segment I found a point $P_1(u_0 + du, v_0 + dv)$, where du and dv satisfy these conditions

$$\|Y(u_0 + \frac{1}{2}du, v_0 + \frac{1}{2}dv)\| = k$$

$$Y(u_0 + du, v_0 + dv) = 0$$

so in the middle-Point of the segment it is required the maximum error k , while the point P_1 has to stay on the curve.

The equations may be written in this way

$$\begin{aligned} & \frac{1}{2} \frac{\partial^2 Y(u_0, v_0)}{\partial u^2} du^2 + \frac{1}{2} \frac{\partial^2 Y(u_0, v_0)}{\partial v^2} dv^2 \\ & + \frac{\partial^2 Y(u_0, v_0)}{\partial u \partial v} du dv \\ & + \frac{\partial Y(u_0, v_0)}{\partial u} du + \frac{\partial Y(u_0, v_0)}{\partial v} dv = 0 \\ & \frac{1}{2} \frac{\partial^2 Y(u_0, v_0)}{\partial u^2} (\frac{du}{2})^2 + \frac{1}{2} \frac{\partial^2 Y(u_0, v_0)}{\partial v^2} (\frac{dv}{2})^2 \\ & + \frac{\partial^2 Y(u_0, v_0)}{\partial u \partial v} \frac{du}{2} \frac{dv}{2} \\ & + \frac{\partial Y(u_0, v_0)}{\partial u} \frac{du}{2} + \frac{\partial Y(u_0, v_0)}{\partial v} \frac{dv}{2} = \pm k \end{aligned}$$

Simplifying these equations I obtain the following

$$\begin{aligned} & \frac{\partial Y(u_1, v_1)}{\partial u} du + \frac{\partial Y(u_1, v_1)}{\partial v} dv = \pm 4k \\ & \frac{\partial^2 Y(u_1, v_1)}{\partial u^2} du^2 + \frac{\partial^2 Y(u_1, v_1)}{\partial v^2} dv^2 \\ & + 2 \frac{\partial^2 Y(u_1, v_1)}{\partial u \partial v} du dv = \mp 8k \end{aligned}$$

which is a second degree system of equations. The choice for the sign of $k \pm$ depends on the curvature of the curve and may be done before the segmentation phase.

The algorithm produces segments step by step. When P_1 has been found, the solution for P_2 (and so for the following points) is very easy to find. In fact, one of the two solutions of the second degree system for (du, dv) is the one which brings back to the point P_0 . So it's possible to find the second solution given the first.

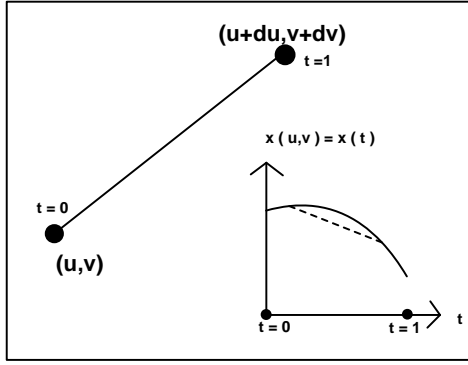


Figure.4 The sub-segmentation with the $x(u, v)$ function along the segment with approximately constant y .

The sub-segmentation

When a segment has been evaluated, it may be expressed in this form

$$\begin{aligned}
 t &\in [0, 1] \\
 u(t) &= bt_u t + ct_u \\
 v(t) &= bt_v t + ct_v \\
 x(t) &= at_x t^2 + bt_x t + ct_x \\
 z(t) &= at_z t^2 + bt_z t + ct_z \\
 &\text{etc. (if there are other functions)} \\
 &\text{as normal functions}
 \end{aligned}$$

The x function may be segmented in a way similar to the one used for y . A step dt is evaluated so that the linear approximation of x between t_i and $t_i + dt$ has k as maximum value; the error function is always a parabola, so the maximum error is in the middle point.

$$\begin{aligned}
 \left\| \frac{x(t_i) + x(t_i + dt)}{2} - x\left(t_i + \frac{1}{2}\right) \right\| &= k \\
 dt &= \pm \sqrt{\frac{\pm 4k}{at_x}}
 \end{aligned}$$

dt is a fixed step. The sub-segmentation process is truly fast. Once a subsegment has been found, it can be considered as linear both for y and x , then rasterization is truly simple.

6. PERFORMANCES

The evaluation of performances is a bit difficult, because it's necessary to consider many factors. Of course the rendering of a lembo is slower than the rendering of a triangle, but if we approximate a lembo with a lot of triangle the rendering

time becomes similar or overcomes the one for the lembo with the rendering. In general the rendering of single lembo is 2 or 3 times more slow than the one of a single triangle, but 2 or 3 are not so much if we consider the possibility to use dedicated hardware.

Another reason which make difficult an evaluation is that actually there is not an hardware for lembos, but a comparison should be done exactly on the hardware level, because there is no reason to perform it via software.

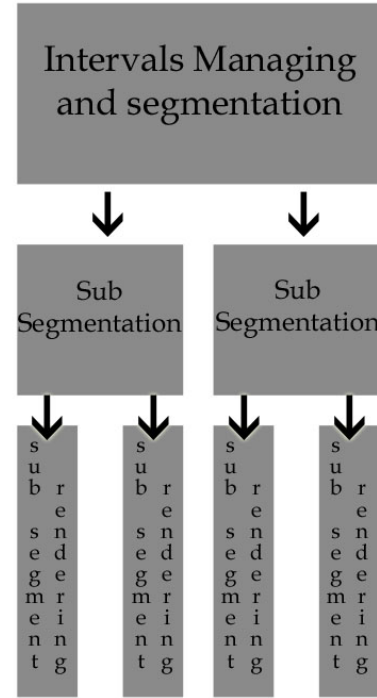


Figure.5 A possible parallel architecture which implements the extreme method for lembos

7. CONCLUSIONS

The Lembo Model makes it possible to get a better approximation reducing the number of elements used (lembos or triangles) and so reducing the number of vertexes processed to obtain high quality renderers; the most evident consequence is the reduction of memory occupied in graphical applications.

The extreme method shown in section 5 is a very fast method for lembos rendering. It's a scan-line method, but it doesn't require numerical approximation if it is possible to have a fast and accurate instrument to solve equations of the second degree. Moreover, the algorithm may be divided in more steps, and these steps are more amenable for hardware acceleration than over methods for scan line rendering, in particular the production

of subsegments from segments and the rasterization of the subsegments; they are also amenable for a parallel architecture, since the production of subsegments and the rendering of different subsegments may be done in parallel.

8. REFERENCES

- [Bar00a] T. Barrera, A. Hast, E. Bengtsson, Surface Construction with Near Least Square Acceleration based on Vertex Normals on Triangular Meshes, SIGRAD (2002)
- [Bou00a] T. Boubekeur, Patrick Reuler, Christophe Schlick, Scalar Tagged PN Triangles, Eurographics 2005
- [Bou01a] T. Boubekeur, Christophe Schlick, Generic Mesh Refinement on GPU, Graphics Hardware 2005, ACM 2005
- [Bre00a] D. E. Breen, Creation and smooth-shading of Steiner Patches Tessellations, proceedings of 1986 ACM Fall Joint Computer Conference, 1986
- [Bru00a] J. Bruijns, Quadratic Bezier Triangles As Drawing Primitives, 1998 Workshop on Graphics hardware Lisbon Portugal, ACM 1998
- [Com00a] V. Comincioli, Analisi Numerica: Metodi, Modelli, Applicazioni, Mc Graw-Hill, 1995
- [Lan00a] J.M. Lane, L.C. Carpenter, T. Whitted, J.F. Blinn, Scanline Methods of Displaying Parametrically Defined Surfaces, J.D. Foley Editor, ACM 1980
- [Mar00a] William Martin - Elaine Cohen - Russell Fish - Peter Shirley, Practical Ray Tracing of Trimmed NURBS Surfaces, <http://www.cs.utah.edu/vissim/papers/raynurbs/raynurbs.html> 2000-08-02
- [Qua00a] A. Quarteroni, F. Saleri, Introduzione al calcolo Scientifico, Springer 2002
- [Sch00a] D. Schweitzer, E.S. Cobb, Scanline Rendering of Parametric Surface, ACM 1982
- [Sed00a] T.W. Sederberg, David C. Anderson, Ray Tracing of Steiner Patches, ACM 1984
- [Sed01a] T.W. Sederberg, A.K. Zundel, Scan Line Display of Algebraic Surfaces, Computer Graphics, Volume 23, Number 3, July 1989
- [Shi00a] L. Shiue, I. Jones, J. Peters, A Real Time GPU Subdivision Kernel, ACM ToG, 2005
- [Vla00a] A. Vlachos, J. Peters, C. Boyd, J. L. Mitchell, Curved PN Triangles, ACM Press, 2001
- [Wil00a] D.F. Wiley, H.R. Childs, B.F. Gregorski, B. Hamann, K.I. Joy, Contouring Curved Quadratic Elements, Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2003)
- [Whi00a] T. Whitted, A ScanLine Algorithm for Computer Display of Curved Surfaces, Proc. 5th Conf Computer Graphics and Interactive Techniques, Atlanta, 1978
- [Wri00a] Richard S. Wright Jr., Benjamin Lipchak, OpenGL SuperBible, Third Edition, SAMS, 2004

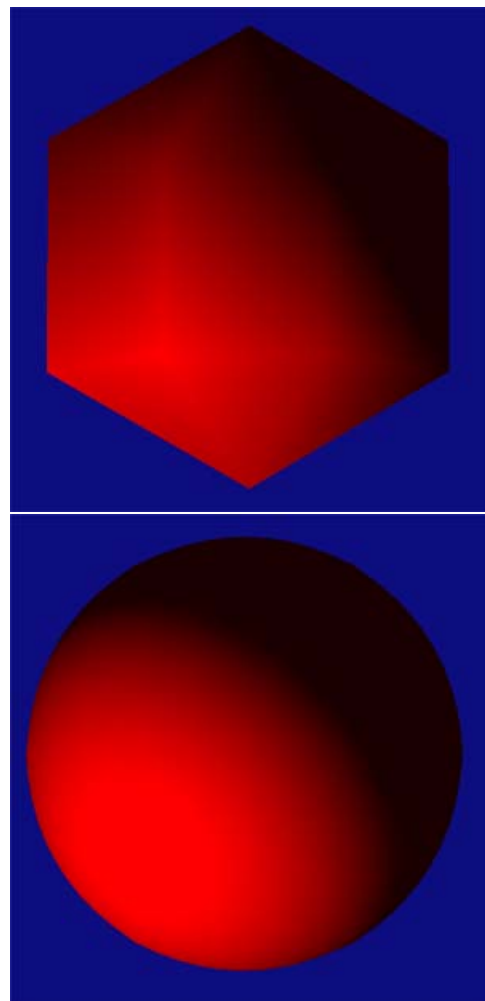


Figure.6 Two spheres, the first constructed with 24 triangles, the second with 24 lembos.

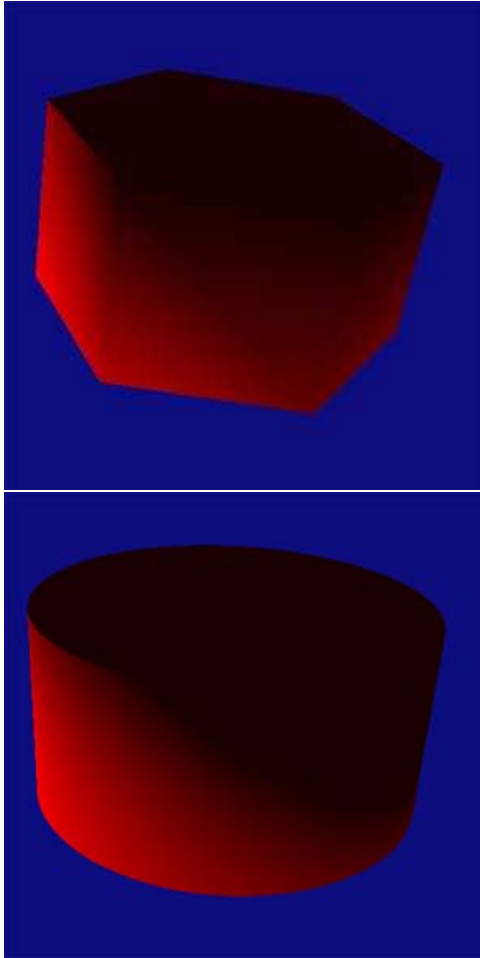


Figure.7 Two cylinders, the first constructed with 24 triangles, the second with 24 lembos.

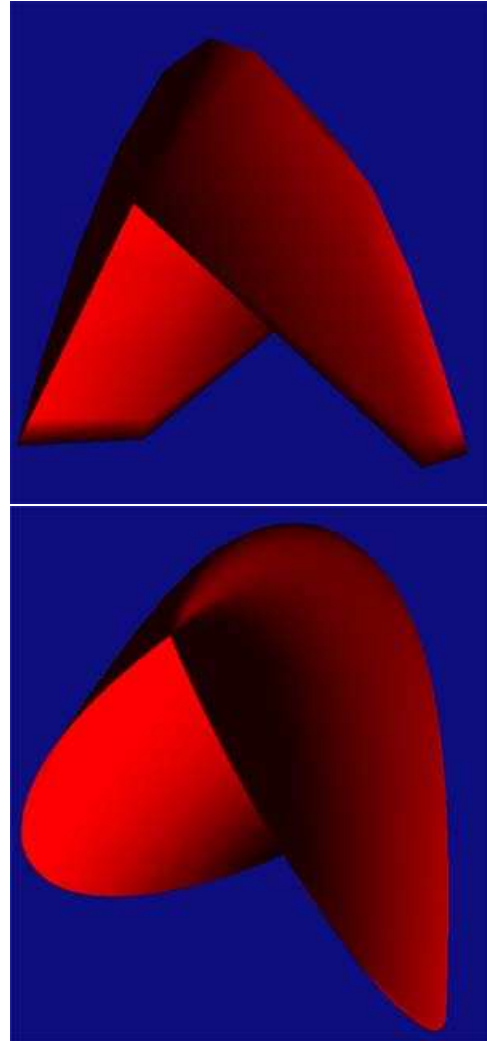


Figure.8 Two images of the same surface, the first constructed with 96 triangles, the second with 96 lembos.

Rational Ruled surfaces construction by interpolating dual unit vectors representing lines

Stavros G. Papageorgiou
Robotics Group,
Department of Mechanical and
Aeronautical Engineering, University of Patras
26500 Patra, Greece
papage@mech.upatras.gr

Nikos A. Aspragathos
Robotics Group,
Department of Mechanical and
Aeronautical Engineering, University of Patras
26500 Patra, Greece
asprag@mech.upatras.gr

ABSTRACT

In this paper, a new representational model is introduced for the rational family of ruled surfaces in Computer Graphics. The surface parameterization is constructed using the NURBS basis functions and line geometry. The ruled surface is defined by interpolating directly dual unit vectors representing lines, which is a single parametric surface and its shape depends on the control lines. All the advantages of the NURBS basis such as shape control and the local modification property are also applicable and bequeathed to the dual NURBS ruled surface.

The problem of drawing the lines defined by dual unit vectors is also resolved. Towards this direction, we propose a simple technique to calculate the surface's striction curve in order to draw the rulings of the surface within the striction curve neighborhood.

The on-screen 3D plot of the surface is realized in a pre-defined specific region close to the striction curve. With the proposed technique a natural representation of the ruled surface is derived. The shape of the surface can be intrinsically manipulated via the control lines that possess one more degree of freedom than the control points. Our method can find application not only in CAD but in the areas of NC milling and EDM.

Keywords

Ruled surface, dual unit vectors, Line Geometry, Plücker coordinates, Striction curve.

1. INTRODUCTION

A 3D surface is called ruled if through any of its points passes at least one line that lies entirely on that surface. The generation of these surfaces is considered simple since they are created by moving a line in 3D space [Far97]. Applications of these surfaces can be found in Computer Aided Design (CAD), CAGD, NC milling, and wire Electric Discharge Machining (EDM) [Yan96]. A ruled surface can be generated by linear interpolation between two given bound curves or using tensor product surfaces [Gra97]. Alternatively, a ruled surface can be defined using a line geometry representation. Ravani and Wang [Rav91] were the

first to show that this approach has the advantage of curve type algorithms and they constructed ruled surfaces of degree $3m$ from $m+1$ control lines. The idea of utilizing dual vector calculus and screw theory in CAGD is not new. Chen and Pottman [Che99], presented a method to derive an approximation of ruled surface in 4-D space by tensor product B-Spline representation. Ding [Din02], used the De Boor algorithm and screw theory to determine ruled surfaces in 6-D space. Xia [Xia00] presented an approach for the problem of motion cutter planning for side milling of ruled surfaces.

In this paper, a method is introduced to define a ruled surface by extending NURBS to dual vector calculus. Most of the published approaches deal with the definition of a ruled surface using approximation algorithms such as the De Boor and De Casteljau. In this paper the ruled surface is represented by a dual NURBS structure. The representation uses the same basis function for curve definition, where the control points are replaced by control lines described by dual unit vectors. Since the rulings of the surface are represented by dual unit vectors, the main representation issue is how to define a specific part of the entire surface. In this paper we are using the

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

SHORT COMMUNICATION proceedings
ISBN 80-86943-05-4
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

locus of the surface's striction points in order to represent the rulings within the neighborhood of this locus. The proposed calculation of the striction curve is focused on the properties of dual line representation and is quite simple. The rulings are projected on the screen ultimately by keeping them under a specific pixel range from the striction curve. In this way we avoid the extra vectors needed for the computer graphics representation of the dually defined rulings.

2. BACKGROUND

Dual unit vector representation of a line in space.

A line in \mathcal{R}^3 can be represented by a dual unit vector, while the elements of this vector are known as Plücker coordinates. Dual vectors are based on the theory of dual numbers invented by Clifford [Cli73]. A dual unit vector \hat{L} is defined as:

$$\hat{L} = \underline{L} + \varepsilon \cdot \underline{L}^0$$

where \underline{L} is the principal vector, \underline{L}^0 the principal moment and ε the dual unit. The reader who is unfamiliar with dual numbers and vectors is referred to the Appendix. The vector $\underline{L} = (L, M, N)$ defines the direction of the given line and the vector $\underline{L}^0 = (L^0, M^0, N^0)$ is the moment of the line about the origin. The dual numbers

$$L + \varepsilon \cdot L^0, M + \varepsilon \cdot M^0, N + \varepsilon \cdot N^0$$

are the dual direction cosines of the line which are the components of the dual unit vector \hat{L} . The definition of an arbitrary line in space is shown in Fig. 1 where \underline{r}_p is the position vector of a point p on the line. Since \underline{L}^0 is the moment of the line around the origin we have $\underline{L}^0 = \underline{r}_p \times \underline{L}$ and obviously $\underline{L} \cdot \underline{L}^0 = \underline{L}^0 \cdot \underline{L} = (\underline{r}_p \times \underline{L}) \cdot \underline{L} = 0$. The symbol \cdot declares the dot product and the symbol \times the cross product. The vectors \underline{L} and \underline{L}^0 are always orthogonal. Additionally, L, M, N are chosen such that $\underline{L} \cdot \underline{L} = L^2 + M^2 + N^2 = 1$. Since $\underline{L} \cdot \underline{L}^0 = 0$ and $\underline{L} \cdot \underline{L} = 1$ these equations are imposed to the six Plücker coordinates resulting to only four independent variables of degrees of freedom for the definition of a line in 3D space.

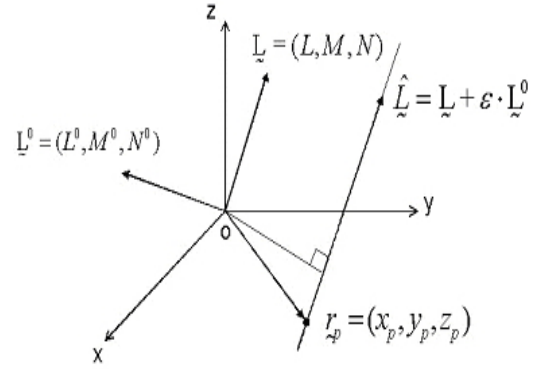


Figure 1: The Plücker coordinates of a line in the space.

Thus, the three dimensional space can be considered to be composed of ∞^4 lines instead of ∞^3 points. A line has four degrees of freedom while a point has only three but in some cases it is convenient to treat a 3D space as a set of lines and to represent its properties in terms of these lines [Roo78].

3. NURBS

NURBS curves

In this part of the paper we briefly outline the properties of NURBS. NURBS curves are rational and present some nice properties such as the ability to represent conic sections and free form surfaces on a rich geometric domain. NURBS also present the properties of the B-Splines, such as the strong convex hull and the local modification property. In fact, a NURBS curve is transformed to a B-Spline curve if all its weights are set equal to 1. The weight which is the additional parameter for the definition of a NURBS curve is quite advantageous. This extension provides one more degree of freedom for shape design and therefore makes NURBS curves more powerful than B-Splines. A NURBS curve is defined by the equation:

$$C(u) = \frac{1}{\sum_{i=0}^n N_{i,p}(u)w_i} \sum_{i=0}^n N_{i,p}(u)w_i P_i = \sum_{i=0}^n R_{i,p}(u)P_i \quad (1)$$

where $N_{i,p}(u)$ is the i -th basis function of degree p

defined by:

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u \in [u_i, u_{i+1}) \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+p}} N_{i+1,p-1}(u)$$

where

$$R_{i,p}(u) = \frac{N_{i,p}(u)w_i}{\sum_{j=0}^n N_{j,p}(u)w_j}, \quad 0 \leq i \leq n \quad (2)$$

are the NURBS basis functions and $w_i \geq 0$ the weight to control point P_i . For the NURBS curve we have: **a)** $n+1$ control points P_i ($0 \leq i \leq n$), **b)** a knot vector U that holds $m+1$ knots and $0 = u_0 \leq u_1 \leq u_2 \leq \dots \leq u_{m-1} \leq u_m = 1$, **c)** a degree p satisfying $m = n + p + 1$. NURBS curves are rational since $R_{i,p}(u)$ are rational functions.

Dual definition of NURBS ruled surface

The dual definition of the NURBS ruled surface is straight-forward. The control points are replaced by dually defined lines using the corresponding dual unit vector $\hat{L}_i = L_i + \varepsilon \cdot L_i^0$. The equation:

$$\hat{M}(u) = \sum_{i=0}^n R_{i,p}(u) \cdot \hat{L}_i \quad (3)$$

defines a NURBS ruled surface where \hat{L}_i are the dual unit vectors representing the control lines and $R_{i,p}(u)$ are the NURBS basis functions given by Eq (2). For the dual definition of the NURBS ruled surface we have the following: **a)** $n+1$ control lines represented by the corresponding dual unit vector \hat{L}_i , ($0 \leq i \leq n$), **b)** a knot vector U that holds $m+1$ knots and $0 = u_0 \leq u_1 \leq u_2 \leq \dots \leq u_{m-1} \leq u_m = 1$, **c)** a degree p satisfying $m = n + p + 1$. In order to fully understand the meaning of the weight, let us consider a control line \hat{L}_i and a weight $w_i \geq 0$. The multiplication of the control line by the weight $w_i \geq 0$ gives: $w_i \cdot \hat{L}_i = w_i \cdot L_i + \varepsilon \cdot (w_i \cdot L_i^0)$ which is the same line as this operation has no affect to its position and to its direction due to the use of unit vectors. However, it affects the shape of the ruled surface since we can assign a specific weight on one or more control lines. In Fig. 2, the same ruled surface is projected but with the use of weights $w_2 = 1.1$ and $w_2 = 1.4$ assigned to the control line represented by \hat{L}_2 . The result is apparent; the ruled surface has been “pulled” from the corresponding

control line, maintaining local control at the same time. For reasons of simplicity, in the rest of this paper, all examples are constructed utilizing weights equal to one. Moreover, for illustration purposes, throughout this paper the control lines are colored red and the striction curve in bold blue.

In Fig. 3 and 4 two dually defined NURBS ruled surfaces with the control hull and their striction curve are shown defined by 5 and 4 control lines respectively. In Fig. 3 and 4, it can be noticed that the first and last control lines, coincide with the rulings of the surface. In both these examples, the boundaries of the control hull are marked with a dashed red line. From these illustrations it is evident that the dually defined ruled surface inherits two of the most important properties that NURBS curves possess: The local effect of weights to the shape of the surface and the “restriction” of the surface by the hull that the control lines form.

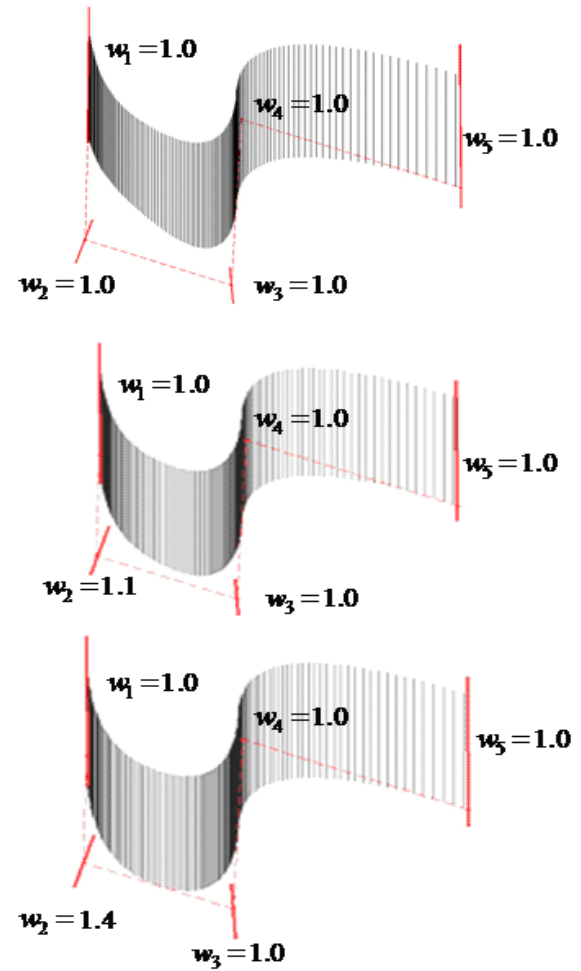


Figure 2 Ruled Surface with alternative weights ($w_2 = 1.1$ in the second case and $w_2 = 1.4$ in the third)

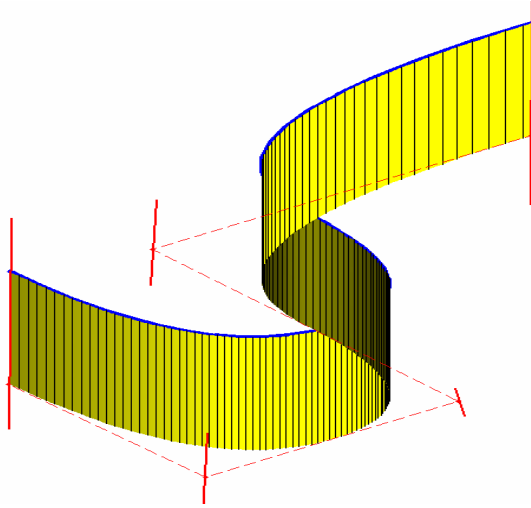


Figure 3 Control hull of 5 dually defined control lines and striction curve.

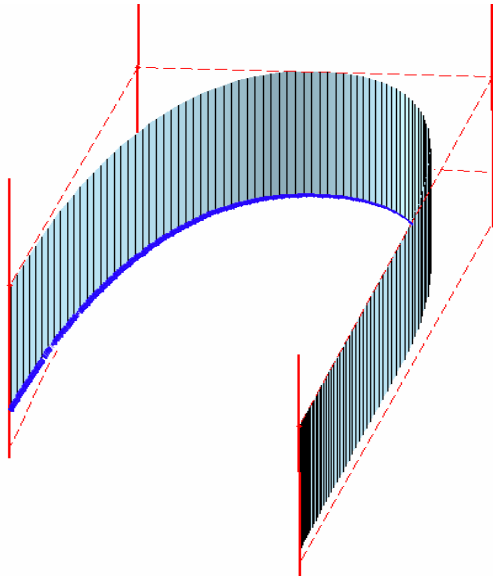


Figure 4 Control hull of 4 dually defined control lines and striction curve.

Since NURBS curves can represent conic sections and a circle, it is clear that using Eq.(3) representations of conical surfaces, generalized cylinders etc can be derived easily. For a specific value of u , Eq.(3) returns a dual line vector:

$$\hat{M}(u) = \underline{M}(u) + \varepsilon \cdot \underline{M}^0(u)$$

We normalize the $\underline{M}(u)$ vector to obtain the unit vector:

$$\underline{\hat{S}}(u) = \frac{\underline{M}(u)}{|\underline{M}(u)|}$$

which is the direction of the line. Then we subtract the product of the pitch $p(u)$ and $\underline{M}(u)$ from $\underline{M}^0(u)$:

$$\underline{S}^0(u) = \underline{M}^0(u) - p(u) \cdot \underline{M}(u)$$

where $p(u)$ is given by:

$$p(u) = \frac{\underline{M}(u) \cdot \underline{M}^0(u)}{\underline{M}(u) \cdot \underline{M}(u)}.$$

The derived dual unit vector:

$$\hat{\underline{S}}(u) = \underline{\hat{S}}(u) + \varepsilon \cdot \underline{S}^0(u)$$

satisfies the Plücker conditions and represents a ruling of the ruled surface for a given value of u . The main problem with this kind of representation is that line segments cannot be easily represented and therefore line to point transformations are needed [Rav91]. In this paper we introduce another approach for the representation of the rulings by utilizing the striction curve of the surface. The striction curve plays an important role in the design and in the differential geometry of ruled surfaces since it possesses some interesting properties. The maximum Gaussian curvature of the generators is located on each striction point of the surface. The point on the ruling which is closest to the successive ruling is called the striction point and is reference system independent. The locus of the striction points, known as striction curve, presents interesting properties and is useful for the evaluation of ruled surfaces in general. A non-cylindrical ruled surface can be re-parameterized using a striction-directrix curve representation taking into account that the striction curve does not depend on the choice of the base curve. Furthermore, the striction curve is always in contact with all the rulings of the surface and all the singularities of the surface are located on this curve.

4. DETERMINATION OF THE STRICTION CURVE

The locus of the striction points is called striction curve and has attracted the interest of researchers due to the role that plays in differential geometry of ruled surfaces. Pottman [Pot96] calculated the striction curve via the De Casteljau algorithm and line geometry. The authors of [Sch98], presented a technique for the calculation of the striction curve of a ruled surface utilizing its classical definition and dual numbers. The definition was derived utilizing the arc length of the indicatrix curve as the parameter and was depended on an arbitrary directrix of the ruled surface. In this paper, the striction curve is calculated using elements that are included in the proposed line representation of the ruled surface. The striction points are determined as follows: Let $\hat{\underline{N}}(u, \Delta u)$ be the common perpendicular between two successive rulings $\hat{\underline{S}}(u)$ and $\hat{\underline{S}}(u + \Delta u)$, $u \in \mathbb{R}$, where $\Delta u \in \mathbb{R}$ is very small as it is shown in Fig. 5.

Let

$$\hat{\tilde{S}}(u) = \tilde{S}(u) + \varepsilon \cdot \tilde{S}^0(u)$$

and

$$\hat{\tilde{S}}(u + \Delta u) = \tilde{S}(u + \Delta u) + \varepsilon \cdot \tilde{S}^0(u + \Delta u).$$

Then the common perpendicular is given by:

$$\begin{aligned} \hat{\tilde{N}}(u, \Delta u) = & \tilde{S}(u) \times \tilde{S}(u + \Delta u) + \\ & \varepsilon \cdot (\tilde{S}(u) \times \tilde{S}^0(u + \Delta u) + \tilde{S}^0(u) \times \tilde{S}(u + \Delta u)) \end{aligned} \quad (4)$$

in dual vector form. The reader who is not familiar with the dual number operations is referred to the appendix. Utilizing the straight forward definition of the dual point [Aza01] by two dual unit vectors we get for the striction point $S_p(u)$:

$$\begin{aligned} S_p(u) = & \tilde{S}(u) \times \tilde{S}(u + \Delta u) + \\ & (N(u, \Delta u) \times \tilde{N}^0(u, \Delta u) \cdot \tilde{S}(u)) \cdot \tilde{S}(u) \end{aligned} \quad (5)$$

where

$$\tilde{N}(u, \Delta u) = \tilde{S}(u) \times \tilde{S}(u + \Delta u)$$

and

$$\tilde{N}^0(u, \Delta u) = (\tilde{S}(u) \times \tilde{S}^0(u + \Delta u) + \tilde{S}^0(u) \times \tilde{S}(u + \Delta u))$$

Two dual unit vectors intersect when the dual component of their dual scalar product vanishes. This property is used for the above definition of the striction point. The resulting construction has enough freedom to hold three point vectors simultaneously. Thereby, extra information can be stored into the expression of the striction point such as the tangent vector, the curvature vector etc. In Fig. 5 the striction point $S_p(u)$ is illustrated for two successive rulings on a ruled surface along with the common perpendicular $\hat{\tilde{N}}(u, \Delta u)$.

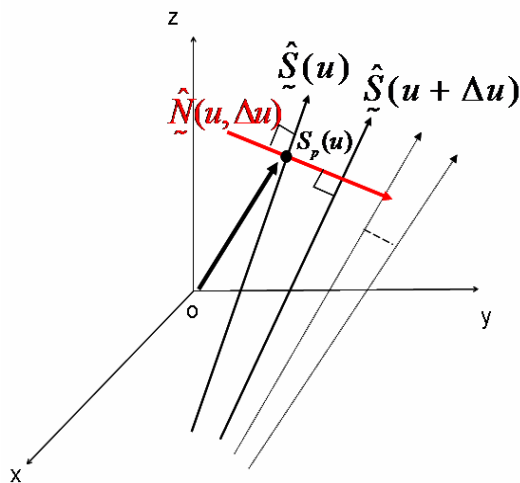


Figure 5 The Striction Point $S_p(u)$.

5. RULED SURFACE DRAWING

Our method is based on the calculation of the striction curve for the rendering of dual NURBS ruled surfaces. In the current literature, the problem

of drawing an infinite line represented by a dual vector has been addressed via several ways. Ravani and Wang [Rav91] introduced a method that utilized the “centre point” of a line to define a line segment. They used two additional parameters on the line coordinates to do so. However, the location of the centre point cannot be controlled during the interpolation. Ding [Din02] used the dual De Boor algorithm for the on screen representation of screws. Sprott and Ravani [Spr97] proposed a method that used a point on the first control line as a reference in order to define a line through that point. Then they derive a surface construction interpolating the control screws with the reference line. These methods mainly utilize a line-segment scheme or approximation algorithms in order to “pass” from the dual number “world” to the screen. The technique proposed in this paper, is not based on an approximation algorithm such as De Boor or De Casteljau but on the exact calculation of the striction points.

The striction points always lie on the common perpendicular among successive rulings and their locus forms a curve of minimal length that meets all the rulings. Using this property, we draw each one of the rulings of the surface by plotting the straight line segments in the neighborhood of the striction curve. More precisely, we project onto the computer screen a specific number of pixels from the striction curve along a ruling. Thus, the need for the two point definition of a line segment is eliminated since the only necessary parameter is the distance from the striction curve. In the following illustrated examples, we demonstrate the difference between two projections of the same ruled surface and of alternative striction curve neighbourhoods. In Fig. 6 and 7 a dual NURBS ruled surface is shown along with the striction curve. It is obvious that in Fig. 7 only the pixels in one side of the striction curve are projected. Finally, two complete shaded models of two ruled surfaces via the dual NURBS definition are represented in Fig. 8 and 9.

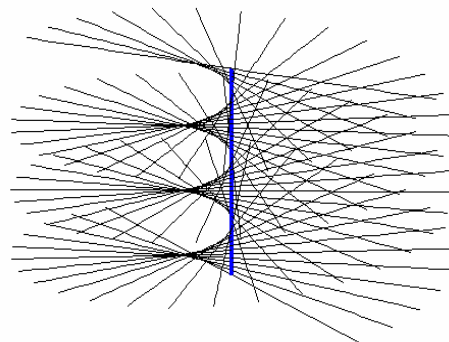


Figure 6 Dual NURBS ruled surface (500 pixels for the first ruling) and striction curve $S_p(u)$.

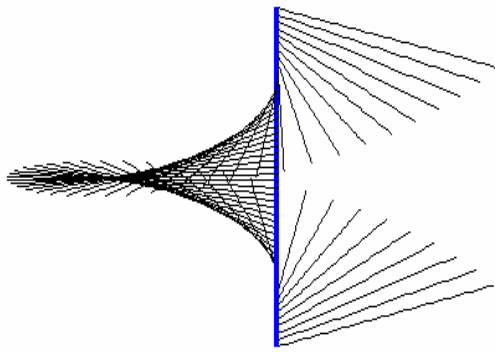


Figure 7 Dual NURBS ruled surface (250 pixels for the first ruling) and striction curve $S_p(u)$.

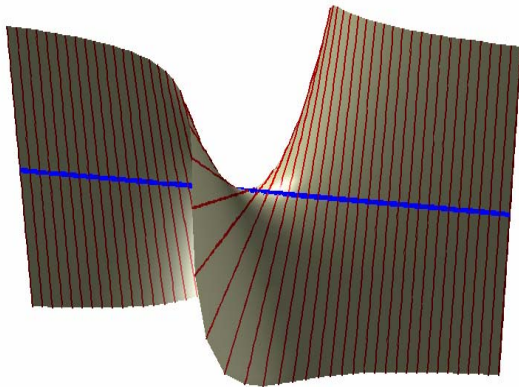


Figure 8 Dual NURBS ruled surface and striction curve $S_p(u)$.

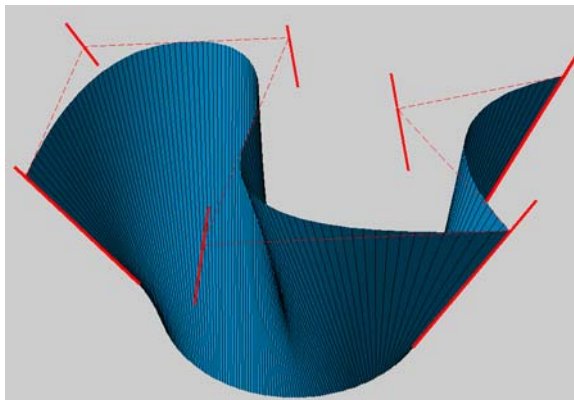


Figure 9 Dual NURBS ruled surface and control lines

6. CONCLUSIONS

In this paper, a new method for the representation of ruled surfaces using line geometry has been proposed. We have developed an alternative way for the definition of ruled surfaces utilizing dual

unit vectors, the NURBS basis functions and the striction curve. The problem of projecting straight lines represented by dual unit vectors is resolved by the utilization of the striction curve. We provide a way for the calculation of the surface's striction points. The calculation is based upon the dual definition of the surface and is performed utilizing the richness of the dual vector representation. Moreover, the definition of the striction points is used to address another problem of the "dual world": the on screen projection of dual unit vectors representing lines. Our representation, can find application in the areas of CAD and CAGD.

In the last decade, line geometry and dual vector calculus has attracted the interest of many researchers and has been studied from the design point of view. However, point-based representations are still dominating CAD and CAGD. This is due to the fact that modern computer graphics systems can only produce 3D to 2D projections point wise. Future research should include the utilization of line geometry by modern APIs. Moreover, the study of the transformations of these surfaces presents great interest. Dual quaternions and dual matrices that have been proved to be smooth transformation operators of dual vectors, are two natural candidates for future investigation.

7. REFERENCES

- [Aza01] Azariadis P, Aspragathos A.N. Computer graphics representation and transformation of geometric entities using dual unit vectors and line transformations, *Computers & Graphics* 25(2): 195-209, 2001.
- [Che99] Chen, H. Y., and H. Pottmann Approximation by ruled surfaces, *J. Comput. Appl. Math.* 102, 143-153, 1999.
- [Cli73] Clifford WK. Preliminary sketch of biquaternions, *Proceedings of the London Mathematical Society*, 1s.,4(64/65):381-95, 1873.
- [Din02] Ding, R. Drawing Ruled Surfaces Using The Dual De Boor Algorithm. In *Proceedings Of Computing: The Australasian Theory Symposium Cats 2002*. Jan. Melbourne, Australia, Elsevier Science B.V, 2002.
- [Far97] Farin G *Curves and surfaces for computer aided geometric design* (4rth ed.): a practical guide, Academic Press Professional, Inc., San Diego, CA, 1997
- [Gra97] Gray A. *Modern Differential Geometry of Curves and Surfaces* (2/E), CRC Press, 1997.
- [Pot95] Pottmann H and Farin G. Developable rational Bezier and B-spline surfaces. *Comput. Aided Geom. Design*, 12:513-531, 1995.

- [Pot96] Pottmann H, Lü W and Ravani B. Rational Ruled Surfaces and Their Offsets, Graphical Models and Image Processing, Vol. 58, No. 6, Nov., pp. 544-552, 1996.
- [Rav91] Ravani B. and Wang J. Computer aided geometric design of line constructs, ASME J. Mech. Design 113, pp. 363–371, 1991.
- [Roo78] Rooney J. A comparison of representations of general spatial screw displacement, Environment and Planning, B,5, 45-88, 1978.
- [Sch98] Schaaf, J. A. and Ravani, B. Geometric continuity of ruled surfaces. Comput. Aided Geom. Des. 15, 3, 289-310, 1998.
- [Spr97] Sprott, K. and Ravani, B. (1997) Ruled surfaces, Lie groups, and mesh generation. 1997 ASME Engineering Technical Conferences, Sep. 14-17, Sacramento, California, USA, 1997.
- [Xia00] Xia, J., and Ge, Q. J. Kinematic Approximation of Ruled Surfaces Using NURBS motions of a Cylindrical Cutter. Proc. 2000 ASME Design Automation Conference, Baltimore, MD, Paper No. DETC2000/DAC-14280, 2000.
- [Yan96] Yang M, Lee E. NC verification for wire-EDM using an R-map. Computer Aided Design;28:733–740, 1996.

APPENDIX

Dual numbers can be defined as:

$$\hat{a} = a + \varepsilon \cdot a^0, \quad a, a^0 \in \mathfrak{R}$$

where ε is the dual unit with the properties of:

$$\varepsilon \neq 0, \quad 0 \cdot \varepsilon = 0, \quad 1 \cdot \varepsilon = \varepsilon \cdot 1 = \varepsilon, \quad \varepsilon^2 = 0$$

The multiplication and addition operations of two dual numbers

$$\hat{a}_1 = a_1 + \varepsilon \cdot a_1^0, \quad \hat{a}_2 = a_2 + \varepsilon \cdot a_2^0 \text{ are defined as:}$$

$$\text{Addition: } \hat{a}_1 + \hat{a}_2 = (a_1 + \varepsilon \cdot a_1^0) + (a_2 + \varepsilon \cdot a_2^0) = (a_1 + a_2) + \varepsilon \cdot (a_1^0 + a_2^0)$$

Multiplication:

$$\hat{a}_1 \cdot \hat{a}_2 = (a_1 + \varepsilon \cdot a_1^0) \cdot (a_2 + \varepsilon \cdot a_2^0) = a_1 \cdot a_2 + \varepsilon \cdot (a_1 \cdot a_2^0 + a_1^0 \cdot a_2)$$

$$\text{External product: } \hat{a}_1 \times \hat{a}_2 = \underline{a}_1 \times \underline{a}_2 + \varepsilon \cdot (\underline{a}_1 \times \underline{a}_2^0 + \underline{a}_1^0 \times \underline{a}_2)$$

Human Skeleton Modeling from 2D Uncalibrated Monocular Data

En Peng

Department of Computing
Curtin University of Technology
GPO Box U1987 Perth
6845, Perth, Australia
perry@cs.curtin.edu.au

Ling Li

Department of Computing
Curtin University of Technology
GPO Box U1987 Perth
6845, Perth, Australia
ling@cs.curtin.edu.au

ABSTRACT

A novel method which is simple but effective is proposed to estimate human skeleton ratios from 2D uncalibrated monocular data. Unlike the existing skeleton estimation methods where pre-defined models are used or identification of body segments with certain attributes is necessary, the proposed method utilizes only the 2D joint locations as the input source without posture estimation. In addition, the proposed method uses a real perspective camera model instead of the popularly-used scaled orthographic camera model. The proposed method is tested on monocular data from different camera motions and the estimation result is satisfactory. The reconstructed human skeleton model can be used in further research for full body reconstruction or motion reconstruction from monocular data.

Keywords

Modeling, human skeleton proportion, monocular data

1. INTRODUCTION

Applications involving virtual human, such as digital movies, computer games and video surveillance, have become more and more popular with the rapid development of computer technology. There are many approaches to create a virtual human body. Common ways include employing 3D body scanners or using 3D modeling techniques based on understandings of human anatomy. Detailed 3D human body model can be acquired easily using such methods. However, 3D body scanning is expensive and clumsy, and more seriously, the person to be modeled might not be available for scanning. Meanwhile, human models based on human anatomy generally fall short in representing personalized individuals.

Another approach recovers the human model using image(s). This approach can be divided into two groups: (1) reconstruction from static human figure and (2) reconstruction from human figure in motion.

Most researchers in the first group employ multi-view images of a human figure in a specific posture [Hil98a; Lee00a; Vil03a]. There are strict requirements on the images which must capture certain specific views of the immobile human figure. The human figure is required to appear in certain standard immobile postures. Body reconstruction from image sequences containing human motion is another group in image-based human model reconstruction. Some researchers require the human figure to perform specified motions [Che03a; Kak95a] or limited arbitrary motions [Coh02a; Dap00a; Sta03a] under multiple cameras. No matter whether the static human figure or human motion is used, methods using multiple cameras share the same drawback: the person has to pose for the cameras at a specific location, normally in a fully-equipped laboratory or studio. In contrary, monocular video is conveniently available in various formats (e.g. DVD, VHS video tape, mpeg). However, monocular video is the most difficult source for body reconstruction, especially when the camera is uncalibrated which is mostly the case. Most reconstruction work based on monocular images ([Bar03a; Rem03a; Tay00a]) use

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.

SHORT COMMUNICATIONS proceedings
ISBN 80-86943-05-4
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

the so-called "scaled-orthographic" camera model, which amounts to a parallel projection, with a scale parameter added to mimic the effect that the image of the object shrinks with the distance. Obviously the scaled-orthographic camera model is very different from the real camera used to capture the video. The only exception is [Pen05a] who used a perspective camera model to handle the monocular images. Among these research effort, ([Bar03a], [Pen05a]) tried to reconstruct the human skeleton from uncalibrated monocular images, while others ([Rem03a; Tay00a]) attempted to reconstruct the human motions with a pre-defined human skeleton model.

Human body reconstruction from monocular image sequence is a very attractive idea due to the fact that monocular video is so conveniently available. For the appearance of the whole human body, the skeleton proportion is a dominant feature. Recovering the proper skeleton model is therefore the most important component for the reconstruction of human body. Due to the vast possibilities of human postures and camera settings, skeleton proportions usually cannot be simply calculated from the projected segments. To estimate such proportion, [Bar03a] requires the user to manually identify the body parts almost parallel to the image plane, or the skeleton segments having similar orientation with respect to the camera. From these manual identifications and under the scaled orthographic model, their algorithms are able to choose a suitable stick model from a group of pre-defined models. It is very troublesome to identify the skeleton segments with certain properties. When the input source contains large number of frames, such identification becomes impossible. Further more, analysis without using real perspective camera model results in serious estimation error especially when the perspective effect is significant in the images. [Pen05a] proposed a method to extract the human skeleton model from a perfect synthesized data sequence with a perspective camera model. They assume the input data are perfect without any noise and the camera is fixed with no motions.

It is clearly desirable to develop algorithms to estimate the skeleton proportions from only the landmark joints using a real perspective camera model which is more flexible. In this paper, we propose a novel method to extract the human skeleton proportion from uncalibrated monocular data. The human motion is unrestricted and a perspective camera model is used. Superior to [Pen05a], the proposed method allows the source data captured by the camera performing some simple motions, e.g., rotating, zooming.

The rest of this paper is organized as follows: Section 2 provides the overview of the system; Sections 3 and 4 discuss the major parts of the proposed system; Section 5 demonstrates the results with discussion; and Section 6 concludes the paper.

2. SYSTEM OVERVIEW

Human skeleton system can be treated as skeleton segments connected at joints as shown in Figure 1. It can be represented hierarchically as a skeleton tree if one joint is taken as the root. To minimize the number of tree levels, the joint J1 is considered as the root joint. It is clearly impossible to derive the absolute lengths of any body segments from monocular images without the knowledge on camera setting and the distance of the human to camera. Hence the skeleton proportion of every skeleton segment is used to represent the skeleton model which is defined as the ratio of the individual length of every skeleton segment to the sum of the lengths of all segments.

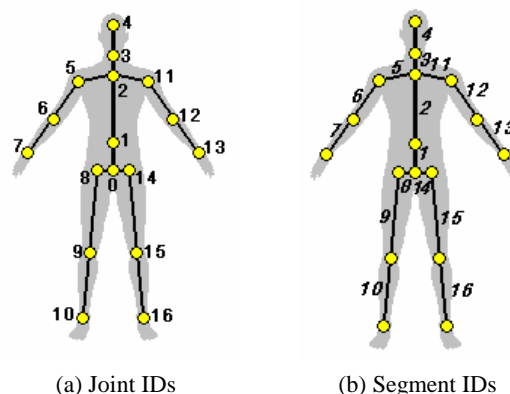


Figure 1. Skeleton joint IDs and segment IDs

Due to the considerable diversity of camera types and media types, the images recording the real world may be different in image size and aspect ratio. The conventional film is 35mm film with the size of 36mm×24mm. Camera parameters based on 35mm film, the basic film format, are commonly used as the standard. Therefore, without much generality loss, we scale any input image based on its original aspect ratio to fit and centering into a 35mm film for normalization.

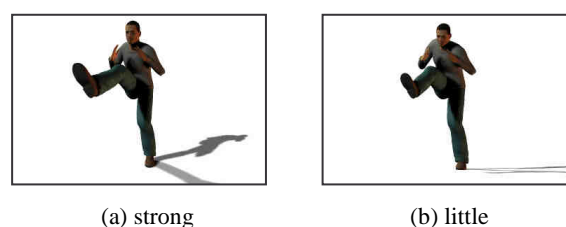


Figure 2. Perspective effects

To reconstruct the human skeleton from uncalibrated monocular images, [Bar03a] uses a scaled-orthographic camera model. In that work, the user is required to identify the perspective effects in the image(s), and only those with very little perspective effect can be chosen as the input. Such limitation greatly reduces the range of usable input video. In contrary, the system proposed here accepts input video with any levels of perspective effect. The user only needs to give an initial focal length to the system by identifying the perspective effects in the first few images. A reference guideline is provided for strong to little perspective effects, indicating the focal length range from 20mm (wide angle) to 300mm (telephoto). Figure 2(a) gives an example image displaying the strong perspective effect while Figure 2(b) shows an example with little perspective effect. If the user is unable to determine the perspective effects in the first few images, a default initial focal length is given as 50mm, which has the perspective effects closest to those seen by normal human eyes [Ele05a].

Many video available are recorded by a mounted camera on a stationary tripod. The mounted camera may be static or performing certain motions such as zooming, panning etc. during the video capturing. The proposed system estimates the camera motions from the source input data and “undo” these motions to produce the modified 2D data. The modified 2D data can then be considered as always taken by a static camera.

A human skeleton modeling algorithm is then used to estimate the proportions of the human skeleton from the 2D monocular data taken by a static camera.

3. CAMERA MOTION AND STANCE FOOT

Before applying the skeleton modeling algorithm for 2D monocular data taken by a static camera, a system is proposed to estimate motions such as zooming, panning etc, of a mounted camera. Such information is then used to “undo” the camera motions to produce the modified 2D monocular data as if they are taken by a static camera.

To estimate the camera motions from one frame to another, at least 3 pairs of corresponding points are required [Tan95a]. Each pair of corresponding points must be projected from the same stationary point. In fact, about 10 such pairs are needed for satisfactory estimation results [Tan95a]. Therefore, if the image background is featureless and no assistant stationary object is available, it is almost impossible to estimate the camera motions between two frames, since it is hard to find even one pair of stationary points.

In this project, the only input information is the projections of the 17 skeleton joints. As the human is dynamic, none of these skeleton joints can be considered stationary at any time. Without a pair of corresponding points between two frames, it is impossible to calculate the camera motion. Fortunately in most human motion, there is always a foot to support the weight of the body at any moment. This foot is called the stance foot and usually remains at the same position for a short period of time. We make use of this small piece of information to estimate the camera motions.

3.1 2D Data Rectification

Human motions and camera motions usually exhibit certain coherence in time. Thus, the projection trajectory of each human skeleton joint within a few neighboring frames is usually smooth. Any inconsistencies are deemed to be brought in by the sampling error in data extraction. Therefore, the 2D data are first rectified by a smoothing filter.

3.2 Sequence Segmentation

When recording video with a mounted camera, the camera can either be static (motionless) or perform the following motions: (1) zooming: the camera's focal length changes; (2) panning: the camera rotates about the Y-axis; (3) tilting: the camera rotates about the X-axis; (4) swinging: the camera rotates about the Z-axis. (See Figure 3)

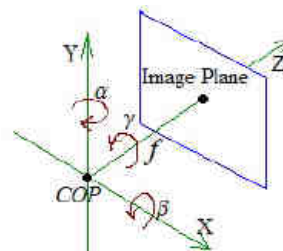


Figure 3. Camera motions

In practice, a mounted camera often remains static, occasionally performing one or both of the following motions: (1) directional rotating: a combination of panning and tilting in one direction, usually used when tracking a moving person; (2) zooming: usually used when keeping the full body of a person in the film range, if the person moves closer or further to the camera. This project assumes the camera performs only one of these motions at a time, and the camera would be motionless for at least 1 second in between any change of its motion. It is also assumed that the duration of each camera motion is over 0.5 second. On the other hand, each stance foot can be reasonably assumed to be static for more than 1 second. Under these assumptions, it is possible to find the frames when there is no camera motion, since

the projection of the stance foot would be fixed during those periods.

For the input data with frame rate p fps, the projection of a stance foot would remain fixed for at least $(p/2)$ frames under the assumptions that the camera remains still for at least 1 second before any motion and the stance foot remains still for at least 1 second. The extreme case is that, within 1.5 seconds, the camera moves during the first 0.5 second and remains still for the next 1 second, while the foot is only still for the first 1 second. In this case, the projection of the foot is static for the period from 0.5 to 1 second, i.e. $p/2$ to p frames. Therefore, the accumulated value of the movement of each foot in every neighboring $p/2$ frames can be calculated. If the accumulated movement of any one foot in these frames is below a given threshold, these frames are considered to be taken by a static camera.

In this way, the input 2D data is segmented into different sequence segments. These segments can be classified into two categories: data from static camera, and data from camera in motion.

3.3 Camera Motion Estimation

For video segments that come from dynamic camera, camera motion needs to be estimated in order to produce the modified 2D data. If the stance foot is known in each frame within the data segment, the camera motion estimation is pretty easy. However, most of the time it is not known which foot is the stance foot in these frames. An iteration method is hence proposed to determine the stance foot in each frame.

It is assumed that the stance foot usually remains motionless for at least 1 second. For the input with frame rate p fps, the stance foot would not lift off within p frames after it first touches the ground. Therefore, for the video segment of n frames, there are minimum 0 time and maximum $(n/p+1)$ times of stance foot swapping.

Every possible case of foot swapping is tested to calculate the motion smoothness under the two possible camera motions mentioned in the previous subsection. The “motion smoothness” is calculated based on the standard deviations in the motion parameters: For “directional rotating”, the motion parameters are the panning and tilting angles between two neighboring frames; for “zooming”, it is the scaling difference in focal lengths between two neighboring frames. The smoother a camera motion is, the more likely that motion is correct.

After testing all the cases, two possible cases can be discovered: one with the largest motion smoothness for “directional rotating”, another with the largest

motion smoothness for “zooming”. The case with the large (above the threshold) motion smoothness is chosen as the solution. If both motion-smoothness are large (above the threshold), both camera motions are possible. In this case, the best way is to “undo” the camera motions in both cases. Significant enlargement/shrinking effects will appear in the new data if the camera motion is “undone” mistakenly and the right camera motion can be chosen correspondingly.

3.4 Modified 2D Data Production

After the camera motions are estimated for each segment from the dynamic camera, it is possible to “undo” the camera motions for the whole input 2D data segment by segment. The resulted modified data can be treated as taken by a static camera. This static camera also has the initial focal length chosen by the user.

4. SKELETON MODELING

The skeleton modeling process is based on the 2D data from static camera and the initial focal length selected by the user. It consists of two major steps: (1) virtual scale parameter estimation; (2) length of skeleton segment calculation.

4.1 Virtual Scale Parameter

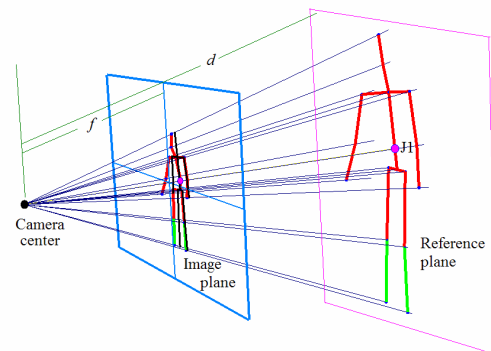


Figure 4. Scaled human figure

Any image can be considered as the projection of a virtual human figure whose root joint lies on the image plane. As shown in Figure 4, if the plane passing through the root joint J1 of the actual human figure has the distance d to the camera center, the scaling parameter of the human on the image can be calculated as $s=f/d$, where f is the camera’s focal length. However, the scaling parameter in each frame is impossible to establish since the actual distance d is unknown. Assume that such distance for every frame in the sequence are d_1, d_2, \dots, d_n respectively. Assuming that the first frame has the unit scaling parameter ($d_1'=f$), we just need to find the relative distance to derive d_1', d_2', \dots, d_n' . If the distance

relationships are correctly found, the distances d' calculated would have the same trend as the actual d : $d_1'/d_1 = d_2'/d_2 = \dots = d_n'/d_n$. The virtual scale parameter is taken as $s' = f/d'$, with the first frame having $s_1' = 1$.

To find the distance relationship for the joint J1, the standard perspective projection theory can be utilized. It is based on the fact that the projected length of any line segment parallel to the image plane has a linear relationship with its distance from the image plane - the further the segment is from the image plane, the shorter its projected length is. Obviously such knowledge cannot be directly used for joint J1, since it does not necessarily belong to a line segment parallel to the image plane.

For this purpose, the projected length of the calf connected to the stance foot is studied. It is difficult to gauge the posture of the actual calf only from its projection. However, the frame containing the longest projection of that calf can be treated as taken at the moment when the calf is parallel to the image plane. Searching through the entire sequence, the projected lengths of the possible parallel calf from different stance foot can be obtained from the input.

Such information can then be used to derive the distance relationship of the joint J1. During the swapping of the stance foot, the joint J1 is usually located on the same plane that passes through the two feet and is perpendicular to the ground plane. If a reference plane passing through J1 is parallel to the image plane, any point on the intersection line between this plane and the ground can be utilized for calculating the virtual distance of J1. One straightforward way to approximate such a point is to find the intersection between the vertical line passing through J1 and the line segment from one stance foot to the other. With this point, the virtual distance in this frame can be calculated. However, not all frames show the moment of stance foot swapping. The virtual distance in other frames are then interpolated linearly. The virtual scale parameter can then be derived from these virtual distances.

4.2 Virtual Length Estimation

Figure 5 shows the projection of a branch of the connected skeleton segments from the root joint. The virtual scale parameter has been derived for this image, and the image is captured by a camera with a known focal length f . P_1' is the projection of the root joint; line segment $P_1'P_{i+1}'$ represents the projection of the skeleton in the i^{th} level of the skeleton tree; and the projection center is denoted as O_c . It is assumed that this frame is projected from a virtual human figure whose root joint P_1 locates on the image plane, as discussed in "Virtual Scale Parameter" part.

The estimation of the length of the virtual skeleton is preceded hierarchically. The estimation in the first level of the tree includes two steps: (1) lower boundary estimation for all possible virtual skeleton length in each frame and (2) virtual skeleton length selection.

As shown in Figure 6(a), $P_1'P_2'$ is the projection of a skeleton segment in the first level of the tree while the root joint P_1 of the virtual human figure is located on the image plane. $P_1'P_2'$ can be projected from an infinite number of possible skeleton segments with different lengths, due to the uncertainty of the actual joint that projects onto P_2' .

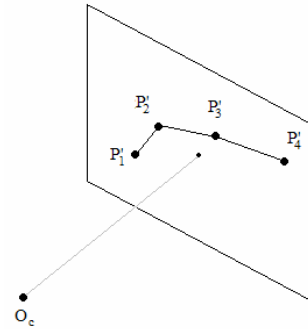
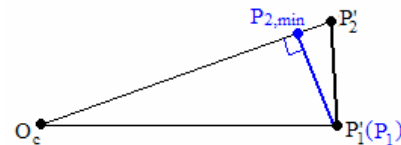
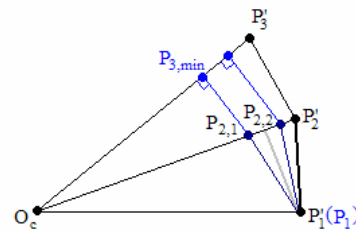


Figure 5. Projection of connected segments



(a) first level



(b) second level

Figure 6. Estimation of lower boundary

Fortunately, the lower boundary of these possible skeleton lengths can be calculated, e.g., $P_1P_{2,min} \perp O_cP_2'$, where $|P_1P_{2,min}|$ is the shortest possible virtual skeleton length with end point P_1 that has the projection $P_1'P_2'$. This shortest possible virtual length will always be shorter than the actual virtual skeleton length in this frame, and is called the lower boundary for this frame.

With the virtual scale parameter in every frame established, the lower boundaries from all frames are comparable. Denote the virtual scale parameter in all

frames as: s_1', s_2', \dots, s_n' , the lower boundary of the virtual skeleton length in each frame as: l_1, l_2, \dots, l_n respectively, and the skeleton length of the scaled human figure at the normalized size as L . According to the calculation of lower boundary in each frame, it is known that $l_i/s_i' \leq L$ ($1 \leq i \leq n$). Therefore, the largest value of these normalized lower boundaries in all frames $\text{Max}\{l_i/s_i', 1 \leq i \leq n\}$ is the best approximation of the virtual length L of the actual human at normalized size.

Unlike the segments in the first level where one end joint is assumed to be on the image plane, both end joints of the segments in higher levels are undetermined. Hence, the position of one end joint must be estimated first in order to calculate the shortest possible virtual length. If the virtual lengths of the segments in the lower levels of the skeleton tree are known, there are at most 2^i possible skeleton poses in the i^{th} level. For example in Figure 6(b), $P_{2,1}$ and $P_{2,2}$ are two possible positions that give the projection at P_2' in 2^{nd} level. As a result, 2^i shortest possible relative lengths in level i^{th} can be calculated. The minimum value among them is selected as the shortest possible virtual length of this skeleton in this frame. Similar to the case in the first level, the largest value of the normalized shortest possible virtual lengths among all frames has the minimum difference from the actual length of that segment at the normalized size.

In this way, the skeleton length for each segment can be estimated under the given focal length.

5. RESULTS

One of the most challenging tasks in any 3D reconstruction attempts from images is the 2D feature extraction from images. Up to date, the image processing techniques are still unable to provide sufficiently accurate 2D feature information from any images. In fact the extraction of 2D feature information and the 3D model reconstruction are two independent modules. They should be addressed separately and in parallel for the ultimate development of the complete system. In this project, the image processing step is not an emphasis. 2D extracted projections of human skeleton joints are assumed available as the input of our system.

Computer synthesized input video is currently used to test our algorithm. The synthesized video is generated by computer with a 3D virtual skeleton model and a virtual camera. The 3D virtual skeleton model has 17 joints with 16 skeleton segments. It is driven by BVH motion files [Ani05a]. The virtual camera is located at a fixed 3D position when capturing any frame of 720*480 pixels, where the fixed 3D position of the camera is not restricted as long as the camera can

capture the whole human motion. It can perform one of the following motions in between motionless periods: directional rotating (panning and/or tilting, but excluding swinging since it is hardly used in video production) and zooming. Each camera motion, including the motionless period, should be smooth and lasts for at least 0.5 second. Noises are added to the 2D data to make the application closer to actual practice.

The proposed system first estimates the camera motions in the source 2D data. The modified 2D data is then produced which can be considered as taken by a static camera. The human skeleton model in different frame has different projected size. A virtual scale parameter is needed to resize the human model to a standard size. The supporting foot determined in each frame can help to estimate the virtual scale parameter. After that, the length of the same skeleton segment in each frame is estimated. By using the virtual scale parameter, these lengths can be compared and the length for that skeleton segment can be determined for the human model at the standard size.

5.1 Camera Motion

To estimate the camera motion, the data segments from a dynamic camera are detected from the source data. Two possible camera motions can be estimated: “directional rotating” and “zooming”.

Figure 7 shows some sample frames comparing the source data and the modified data. The source data includes three camera motions: panning, tilting and zooming. With the initial focal length specified by the user, the proposed system successfully identified the data segments that taken by a dynamic camera. The camera motion for each of such segments is also automatically determined.

5.2 Virtual Scale Parameter

The proposed system utilized the estimations of stance foot and the recovered 2D data to calculate the relative distance for the position of each stance foot. With this relative distance, the relative distance of joint J1 in each frame is determined. The virtual scale parameter is estimated based on this relative distance.

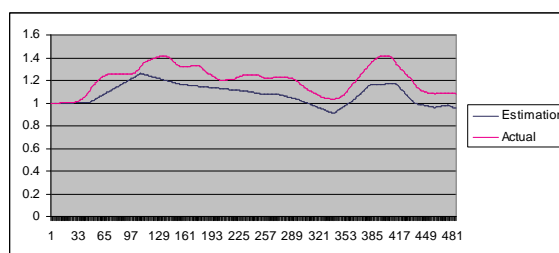


Figure 8. Virtual scale parameter estimation

Figure 8 shows the result comparing the estimated virtual scale parameter to the actual one. The vertical axis represents the virtual scale parameter while the horizontal axis represents the frame id. The red dots indicate the actual virtual scale parameter at each frame while the blue dots indicate the estimation. From the comparisons, it can be seen that the trend of scaling up and down can be successfully estimated.

5.3 Skeleton Model

The estimation of the skeleton proportion is shown in Table 1 which compares the actual and the estimated skeleton proportion in both numerical and graphical ways. The first column indicates the skeleton segment ID. The second column and third column show the skeleton proportion of that skeleton segment. The skeleton proportion is calculated by comparing its length to the sum of all skeleton lengths. The fourth column shows the difference between the estimated and the actual proportion. The last column compares the skeleton proportion in an intuitive way: the red skeleton with blue joints is the actual skeleton model while the green skeleton with black joints shows the estimation.

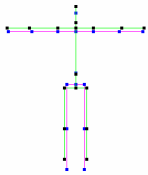
Skeleton	Actual	Estimation	Error	Comparison
1	4.90%	5.97%	1.07%	
2	17.75%	19.86%	2.11%	
3	1.55%	2.28%	0.73%	
4	6.44%	6.89%	0.44%	
5	7.54%	7.69%	0.15%	
6	11.19%	12.17%	0.98%	
7	10.08%	9.58%	0.50%	
8	3.80%	4.66%	0.86%	
9	19.05%	17.50%	1.55%	
10	17.69%	13.40%	4.29%	

Table 1. Skeleton proportion estimation

It can be easily seen from Table 1 that the estimated skeleton proportion resembles the actual skeleton proportion very closely. The errors between the estimation and the actual skeleton model are very small in lower hierarchical levels but increase in higher levels. In skeleton S10, the estimated error reaches its peak. The estimation accuracy of the skeleton in higher level decreases due to the algorithm's hierarchical nature.

6. CONCLUSION

This paper presents a simple but effective method to estimate the human skeleton proportion from uncalibrated monocular data. The proposed method can estimate the human skeleton proportion without using any pre-defined skeleton information. It is not required to manually identify skeletal segments with certain properties, and no posture estimations are

necessary. The proposed method is tested on the computer synthesized data captured by a mounted camera with motions. The reconstructed result is satisfactory. Future work includes the improvement on the estimation of skeletons in higher hierarchical levels. More cues are to be investigated in order to minimize the error propagated to higher hierarchical levels. The algorithm will also be tested on real video to verify its applicability.

7. REFERENCES

- [Ani05a] Animazoo.BVH Library.
http://www.bvhfiles.com, accessed 1 Jul, 2005.
- [Bar03a] Barron, C., and Kakadiaris, I.A. On the improvement of anthropometry and pose estimation from a single uncalibrated image. *Mach. Vision Appl.*, 14(4):229-236. 2003.
- [Che03a] Cheung, K. M., Baker, S., and Kanade, T. Shape-From-Silhouette of Articulated Objects and its Use for Human Body Kinematics Estimation and Motion Capture. *CVPR*. 2003.
- [Coh02a] Cohen, I., and Lee, M. W. 3D Body Reconstruction for Immersive Interaction. 2nd International Workshop on Articulated Motion and Deformable Objects. 2002.
- [Dap00a] D'Apuzzo, N., Gruen, A., Plankers, R., and Fua, P. Least Squares Matching Tracking Algorithm for Human Body Modeling. XIX ISPRS Congress, Amsterdam, Netherlands. 2000.
- [Ele05a] Elert, G. (editor). Focal Length of a Camera Lens, *The Physics Factbook*.
http://hypertextbook.com, accessed 1 Jul, 2005.
- [Hil98a] Hilton, A., and Gentils, T. Popup people: capturing human models to populate virtual worlds. *SIGGRAPH*. 1998.
- [Kak95a] Kakadiaris, I. A., and Metaxas, D. 3D human body model acquisition from multiple views. *ICCV*, pp. 618. 1995.
- [Lee00a] Lee, W-S., Gu, J., and Magnenat-Thalmann, N. Generating Animatable 3D Virtual Humans from Photographs. *Computer Graphics Forum (Eurographics 2000)*, 19(3). 2000.
- [Pen05a] Peng, E., and Li, L. Estimation of Human Skeleton Proportion from 2D Uncalibrated Monocular Data. *CASA*. 2005.
- [Rem03a] Remondino, F., and Roditakis, A. Human figure reconstruction and modeling from single image or monocular video sequence. *3DIM 2003*, pp. 116-123. 2003.
- [Sta03a] Starck, J., and Hilton, A. Model-Based Multiple View Reconstruction of People. *ICCV*, pp.915. 2003.
- [Tan95a] Tan, Y-P., Kulkarni, S.R., and Ramadge, P.J. A New Method for Camera Motion Parameter Estimation, *ICIP*. 1995.

[Tay00a] Taylor, C. J. Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *Comput. Vis. Image Underst.*, 80(3):349-363. 2000.

[Vil03a] Villa-Uriol, M., Sainz, M., Kuester, F., and Bagherzadeh, N. Automatic creation of three-

dimensional avatars. *Videometrics VII, Proceedings of the SPIE*, 5013:14-25. 2003.

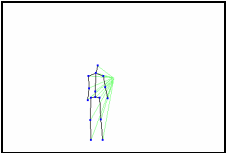
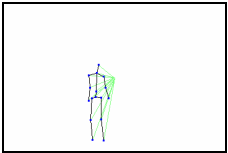
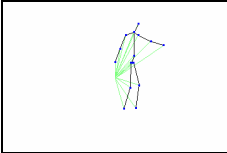
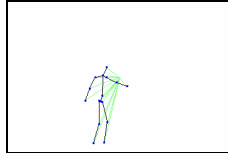
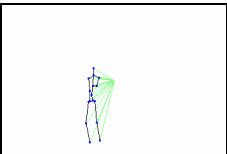
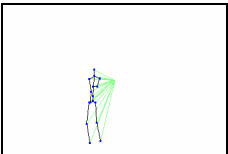
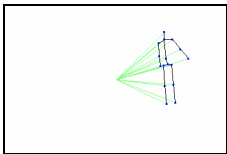
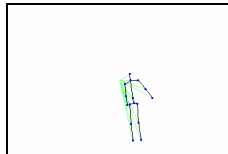
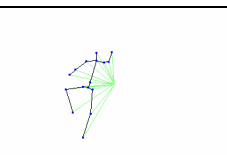
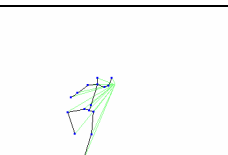
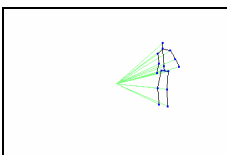

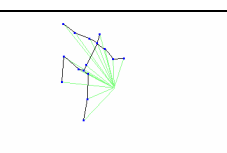
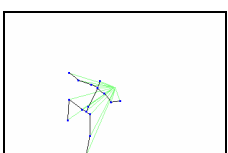
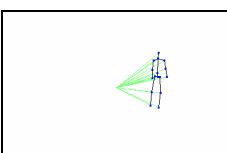
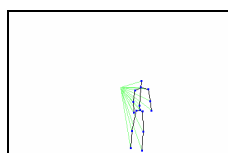
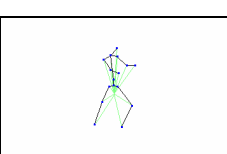
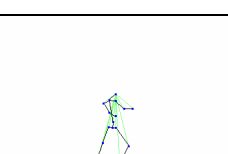
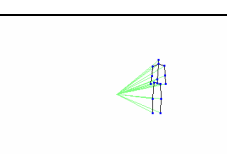
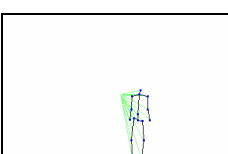
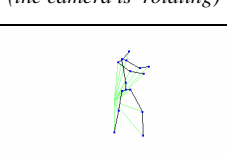
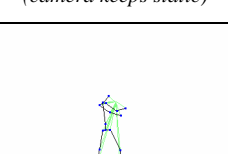
Frame	Source	Recovered	Frame	Source	Recovered
#1			#271		
...	(camera keeps static)	(camera keeps static)	...	(camera keeps static)	(camera keeps static)
#31			#453		
...	(the camera is rotating)	(camera keeps static)	...	(the camera is zooming)	(camera keeps static)
#56			465		
...	(the camera is rotating)	(camera keeps static)	...	(the camera is zooming)	(camera keeps static)
#81			#476		
...	(camera keeps static)	(camera keeps static)	...	(camera keeps static)	(camera keeps static)
#231			#485		
...	(the camera is rotating)	(camera keeps static)			
#251					
...	(the camera is rotating)	(camera keeps static)			

Figure 7. Camera motions and the modified 2D data

A Mesh Data Structure for Rendering and Subdivision

Robert F. Tobler
VRVis Research Center
Donau-City Str. 1/3
1120 Wien, Austria
rft@vrvis.at

Stefan Maierhofer
VRVis Research Center
Donau-City Str. 1/3
1120 Wien, Austria
sm@vrvis.at

ABSTRACT

Generating subdivision surfaces from polygonal meshes requires the complete topological information of the original mesh, in order to find the neighbouring faces, and vertices used in the subdivision computations. Normally, winged-edge type data-structures are used to maintain such information about a mesh. For rendering meshes, most of the topological information is irrelevant, and winged-edge type data-structures are inefficient due to their extensive use of dynamical data structures. A standard approach is the extraction of a rendering mesh from the winged-edge type data structure, thereby increasing the memory footprint significantly.

We introduce a mesh data-structure that is efficient for both tasks: creating subdivision surfaces as well as fast rendering. The new data structure maintains full topological information in an efficient and easily accessible manner, with all information necessary for rendering optimally suited for current graphics hardware. This is possible by disallowing modifications of the mesh, once the topological information has been created. In order to avoid any inconveniences due to this limitation, we provide an API that makes it possible to stitch multiple meshes and access the topology of the resulting combined mesh as if it were a single mesh. This API makes the new mesh data structure also ideally suited for generating complex geometry using mesh-based L-systems.

Keywords

Mesh, Subdivision, Rendering.

1. INTRODUCTION

Subdivision Surfaces [Cat78], [Doo78] have recently been established as a very popular method for generating smooth geometrical objects in computer graphics [Cav91], [DeR98], [Kob00]. With the rise of cheap graphics hardware for consumer PCs, real-time rendering of subdivision surfaces becomes ever more important.

In order to generate the necessary geometry for real-time rendering of subdivision surfaces, complete topology information is necessary for a given mesh of input polygons. Over the years a number of mesh representations have been developed that provide this

topology information, given an arbitrary input mesh. Most of these mesh representations are based on the winged-edge representation introduced by Bruce Baumgart [Bau72].

In order to facilitate access to the complete topology of a polygon mesh, winged-edge representations maintain lists of edges around each vertex, and lists of edges for each face. Thus each edge partakes in four lists: the two lists for its end vertices, and the two lists for the faces it separates (see figure 1).

This type of data-structure is very elegant from a topological point of view: as an example, the dual mesh, where each vertex is replaced by a face, and each face is replaced by a vertex, can be easily created, since vertices and faces are structurally equivalent in this representation. Also, topological modifications to the mesh can be easily implemented by changing the corresponding lists. However, this flexibility comes at a price: due to the dynamic nature of the list data structures, access to specific edges, faces or vertices often results in following a number of elements of the described lists, reducing the access performance. In addition to that, dynamic

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.

*Short Communications proceedings ISBN 80-86943-05-4
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press*

memory management is needed to maintain these structures.

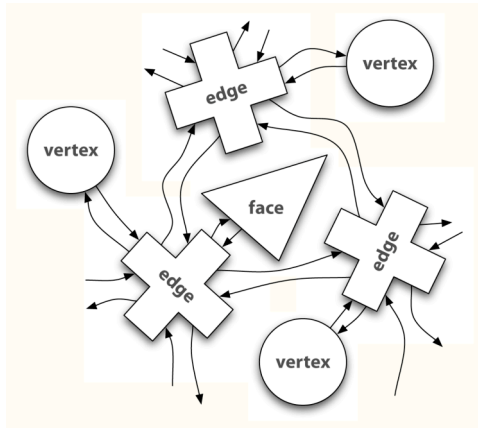


Figure 1: Winged-Edge data structure with complete topological information.

For real-time rendering, the topological information is not needed, therefore the standard approach of a number of packages is to generate a triangle representation optimized for rendering when it is needed. Although this results in optimal rendering performance, the memory cost is increased, since some information needs to be duplicated. Additionally the generation of this rendering representation requires some computation.

2. MESH REPRESENTATIONS

In order to improve the performance of the winged-edge data structure for a number of algorithms that are usually performed on meshes, various mesh representations have been developed. In the following section we will highlight two of these representations on which we have based our additional ideas.

Directed Edges

A useful adaptation of the winged-edge that avoids the original pointer data structures has been developed by Campagna et al. [Cam98]. The idea of this structure is the representation of each edge between two faces as two directed edges, each belonging to one of the faces. This is equivalent to the half edge structure described by Kettner [Ket98]. Each directed edge is part of a circle of directed edges around a face, and references its corresponding directed edge in the neighbouring face (see figure 2). In order to avoid lists in the representation, directed-edges are stored consecutively in an array, and only the base index needs to be stored in the face data structure.

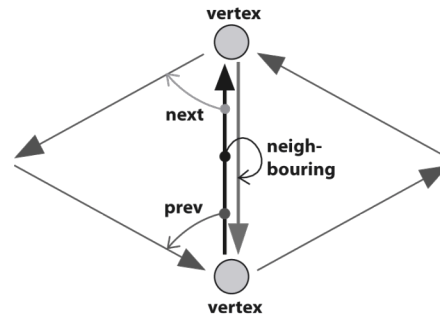


Figure 2: Directed edges representing triangles.

OpenMesh

OpenMesh [Bot02] is a C++ implementation of the ideas presented in the directed edge data structure paper. Due to the use of templates, the exact mesh data-structure that is used by an application can be adapted to the algorithms that need to be applied to the mesh. With the help of the template mechanism, arbitrary attributes can be attached to each geometric object (face, vertex, edge) in the mesh. On first glance, this data structure seems to solve all possible needs, however due to the use of templates, it is necessary to instantiate specific mesh data-structures for each algorithm that needs private attributes. Therefore, if two algorithms with different attribute needs are run on the same mesh, the mesh data structure has to be cloned.

3. MESH REQUIREMENTS

The two main algorithms that our mesh data structure has been designed for, are *subdivision* and *real-time rendering*. Although this seems limiting for a data structure, it turns out, that our resulting design is still very general and can be used for a wide variety of algorithms. The following section summarizes the base requirements that are necessary for efficient implementation of our two main algorithms.

Requirements for subdivision

As already mentioned, full topology information is necessary for subdivision algorithms. In order to optimize these algorithms it is beneficial if topology information is not stored in lists, that have to be traversed, but in arrays that can be directly indexed. Since the resulting meshes of subdivision algorithms can be stored in different meshes, it is not necessary to modify the topological information once it has been created.

Requirements for real-time rendering

Current graphics hardware supports a number of ways to supply geometric data. Among these are: sending the coordinates of each triangle separately, sending so-called triangle strips, where shared coordinates of neighbouring triangles are not retransmitted, or indexed structures, with separate

coordinate arrays and triangle arrays containing vertex indices.

Since triangle strips or indexed structures are normally more efficient than sending each triangle separately, they are the methods of choice for sending geometric data. Triangle strips require an additional preprocessing step for finding the longest strips in a given mesh of triangles. Thus the indexed structures often represent the optimal choice for storing geometric information.

In real-time rendering applications a specific triangle mesh is normally rendered multiple times in subsequent frames. In order to prevent this, it is possible to explicitly store the transmitted data directly in the memory of the graphics hardware using so called *vertex buffer objects*. By using the indexed structures that separate coordinates and vertex indices, some of the information stored in the graphics hardware can be reused between different meshes.

For these reasons the use of indexed structures is often the preferred method for storing geometric data and sending the data to the graphics hardware.

4. THE NEW DATA STRUCTURE

Our data structure for rendering and subdivision maintains rendering information and topological information separately. The topological information can be created if it is needed, but need not be present for rendering.

Geometric information for rendering

In order to facilitate fast rendering, the memory layout of all geometric information has been chosen to correspond to the indexed face set data structure as used in VRML and Inventor. As current graphics hardware supports explicit commands for supplying triangles and quadrangles, we reflect this by maintaining separate arrays for these geometric primitives. Some popular types of geometric data, such as heightfields use quadrangles and therefore our explicit support in noticeable memory savings (as opposed to splitting everything into triangles). Figure 3 shows the data structure with additional normal and colour attributes for each vertex. In order to simplify our first implementation we do not explicitly support general polygons, and directly split these into triangles.

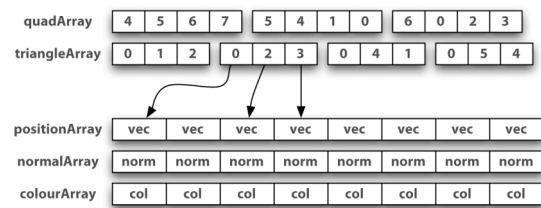


Figure 3: Mesh data structure for rendering. Shown with normal and colour attributes for each vertex.

Topology information for rendering and subdivision

As the data structure for geometric information already supplies references from each face to its vertices we designed the topological information in such a way that this information is reused. The general structure of our topological information is shown in figure 4. Note that almost all other mesh data structures (e.g. [Hop98]) do not explicitly store the references from faces to vertices. Thus for rendering all these other data structures, either these references have to be created, or a somewhat slower traversal of the mesh data structure, together with the creation of a chaced representation with additional memory requirements, needs to be performed.

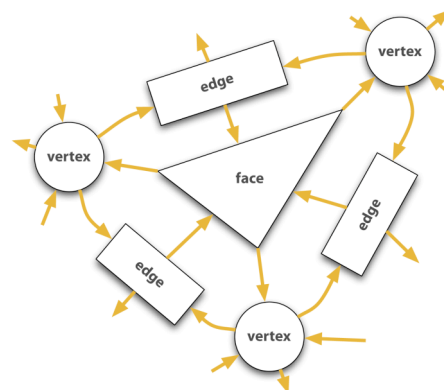


Figure 4: General structure of topological information in our mesh data structure. Note that the references from faces to vertices are already part of the rendering information.

In order to store this information as tightly as possible we use some of the ideas presented by Campagna et al. [Cam98]. A representation of the resulting memory layout can be seen in figure 5. Note that the face array shown in this figure is not stored explicitly, but consists of the concatenated triangle and quad arrays that are part of the rendering information.

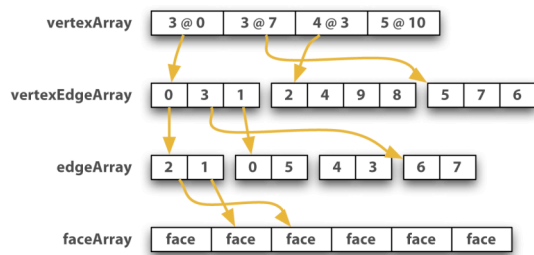


Figure 5: Memory Layout of the topological information.

To complete the topological information, as shown in figure 4 and 5, each of the references must contain more than just the index of the target object: it is necessary to know the exact orientation of the referenced object as well (see figure 6) If e.g. a face is indexed, the index of the face alone does not include the information, which side of the face is referenced. Thus for each array that contains indices of faces, vertices, or edges we maintain a parallel array that contains the orientation (or side) of the referenced face, vertex or edge. For edge references this orientation array could be packed to only contain one bit per reference. For face and vertex references, the memory consumption depends on the maximal number of vertices per face, and edges meeting in a vertex. In order to simplify our implementation we chose to use the same size for this orientation array for all three object types: 8-bit integers. Assuming 32-bit integers for all indices, this results in a tolerable memory increase of 25% for all references.



Figure 6: For each reference the orientation (or side) of the target object has to be maintained.

As the topological information for a mesh is created, the mesh is split into multiple 2-manifold sheets. Each vertex in a mesh is only allowed to be member of a single sheet, so vertices are replicated for representing non-manifold objects.

5. ADVANCED CONCEPTS

As we use indexed arrays to maintain vertices, edges, and faces, modifying the topological information would entail writing customized memory management algorithms for these arrays. Although this is possible in principle, and we may want to implement this in future, for the sake of simplicity we currently do not allow modification of the topological information, once it has been created. This sounds like a severe restriction, however any algorithm that modifies the topology of a mesh can

easily be implemented in such a way, that it creates a new mesh with the modified topology.

Per object attributes

As all geometric objects (faces, vertices, and edges) can be identified by their index within a mesh, additional attributes can be maintained as parallel attribute arrays. Thus multiple algorithms that need different sets of attributes can be run on the same mesh without replicating the geometric and topological information. Based on the frequency of each attribute different implementation strategies can be chosen for each attribute:

- **dense attributes:** ie. attributes that are present for nearly every object are stored in standard arrays
- **sparse attributes:** ie. attributes that are present for very few objects are stored in hash tables

As the attributes for different algorithms can be maintained in parallel for the same geometric and topological mesh structure, and chosen to be represented according to their density, the proposed data structure achieves a near optimal memory utilization.

Stitching of multiple meshes

In order to alleviate the restriction that the topology of a mesh cannot be modified, we introduce a different concept – *stitching* of multiple meshes. Again we try to keep the concept very simple, so that it can be easily used to build complex meshes out of simpler constituting part meshes: we allow that single faces of a mesh can be set to be equivalent with single faces of other meshes (see figure 7). As we assume, that only few of the faces of a mesh are stitched with other meshes, we maintain this information as a sparse attribute of the mesh faces.

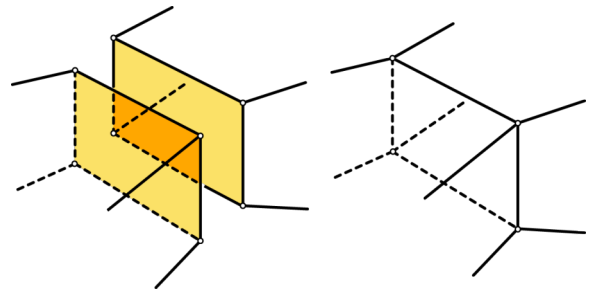


Figure 7: Stitching two meshes by setting faces of different meshes to be equivalent.

Topology API

One important aspect of our mesh data structure is an API to access topology across multiple stitched meshes, as if they were one single mesh. This API is implemented by creating simple handle objects for faces, vertices, and edges. These handle objects that are depicted in figure 8 contain a pointer to the mesh

that contains the respective object, its index within that mesh, and its orientation.

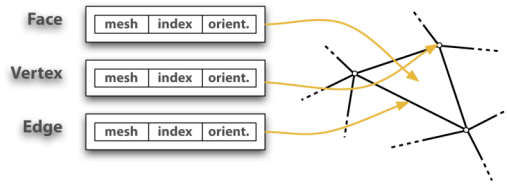


Figure 8: Handle objects are general references to faces, vertices, and edges.

The API of the handle objects allows access to all connected geometric objects of a given object, and returns handles to these connected objects. These handle objects are normally allocated on the stack, and filled with the reference to the object upon creation.

- **face handle objects:** allow access to all vertices and edges of a face, as well as all neighbouring faces
- **vertex handle objects:** allow access to all faces and edges meeting in a vertex, as well as to all neighbouring vertices
- **edge handle objects:** allow access to the two end vertices and the two coinciding faces of an edge

6. RESULTS

Memory consumption

In order to analyze the memory consumption of our mesh data structure we computed the memory requirements for the geometric and topological information of an infinite regular triangle mesh as well as an infinite regular quad mesh. We assume that 32-bit IEEE floats are used for the vertex coordinates and normals, and that for each vertex the position, the normal and a 32-bit colour is stored. Table 1 shows the resulting memory consumption per triangle/quad.

	geometric information	topological information
infinite triangle mesh	26 bytes	37 bytes
infinite quad mesh	44 bytes	52 bytes

Table 1. Per triangle/quad memory consumption for infinite triangle/quad meshes.

Example meshes

We used our data structure for subdivision and rendering of a few example meshes as seen in figure 9 and 10.

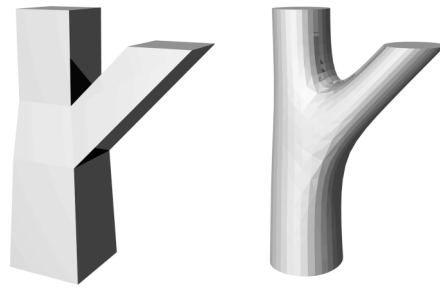


Figure 9: The input and subdivided mesh of a branching structure.



Figure 10: A chair built from multiple sheets created using Catmull-Clark subdivision [Cat78].

7. APPLICATION OF THE NEW MESH DATA STRUCTURE

The mesh data structure we presented is suitable for a number of applications, however our main application is real-time rendering of large vegetation scenes. One important aspect of this application is, that a typical scene contains more geometry than can be explicitly represented in memory.

Levels of detail

In order to overcome this problem, a level of detail approach is employed in our application. Due to the nature of current graphics hardware, the so called *chunked level-of-detail* approach has been chosen, i.e. the granularity of the geometry for which a level-of-detail is chosen, is rather coarse. Thus we represent a plant or a tree as a collection of meshes for branches, twigs, leaves and trunk parts, that are stitched to represent a single large mesh. For each part-mesh, the topology API is used to generate smoother, more detailed variants of the geometry using subdivision. In the rendering application, the currently optimal mesh is sent to the graphics hardware.

Generating complex geometry

For generating the vegetation models in our application, the mesh data structure is highly useful due to its efficient use of memory. We used the mesh-based L-system [Tob02a], [Tob02b] that employs generalized subdivision in order to introduce detail at finer subdivision levels. Some results of our implementation can be seen in figures 11 and 12.



Figure 11: A grown structure consisting of stitched meshes, and a smoothed result using subdivision.



Figure 12: A complete rubber tree built using stitched meshes and subdivision.

8. CONCLUSION AND FUTURE WORK

We presented a new mesh data structure that has been optimized for both real-time rendering and subdivision. Due to the widely differing requirements of these two applications the new data structure is sufficiently general for a wide variety of algorithms.

For our prototype implementation a number of simplifications in our implementation have been made, and the only subdivision scheme we have implemented so far is the one by Catmull-Clark. In the future we expect to generalize our implementation and test our mesh data structure by implementing additional subdivision schemes, as well as other mesh based algorithms such as quadric-based surface simplification [Hec99].

9. REFERENCES

- [Bau72] Baumgart B. Winged edge polyhedron representation. Artificial Intelligence Project Memo AIM-179 (CS-TR-74-320), Stanford University 1972.
- [Bot02] Botsch M., Steinberg S., Bischoff S., Kobbelt L. Openmesh – a generic and efficient polygon mesh data structure. OpenSG Symposium, 2002.
- [Cam98] Campagna S., Kobbelt L., Seidel H.-P., Directed edges – A scalable representation for triangle meshes. Journal of Graphics Tools, JGT 3,4, 1-12, 1998..
- [Cat78] Catmull E., Clark J., Recursively generated B-spline surfaces on arbitrary topological meshes. Computer Aided Design 10, pp. 350-355, Sept. 1978.
- [Cav91] Cavaretta A., Dahmen W., Micchelli C., Subdivision for Computer Aided Geometric Design. Memoirs Amer. Math. Soc. 93, 1991.
- [DeR98] DeRose T., Kass M., Truong T., Subdivision surfaces in character animation. Computer Graphics 32, Annual Conference Series, pp. 85-94, Aug. 1998.
- [Doo78] Doo D., Sabin M., Behaviour of recursive division surfaces near extraordinary points. Computer-Aided Design 10, pp. 356-360, Sept. 1978.
- [Hec99] Heckbert P.S., Garland M., Optimal triangulation and quadric-based surface simplification. Computational Geometry 14, pp. 49-65, 1999.
- [Hop98] Hoppe, H., Efficient implementation of progressive meshes, Computers & Graphics, Elsevier, Vol. 22, No. 1., pp. 27-36, Jan-Feb 1998.
- [Ket98] Kettner L., Using generic programming for designing a data structure for polyhedral surfaces. 14th Annual ACM Symp. On Computational Geometry, 1998.
- [Kob00] Kobbelt L., Sqrt(3) Subdivision. In SIGGRAPH 2000, Computer Graphics Proceedings, Akeley K. (Ed.), Annual Conference Series, ACM SIGGRAPH, pp. 103-112, 2000.
- [Tob02a] Tobler R. F., Maierhofer S., Wilkie A. Mesh-based parametrized L-Systems and generalized subdivision for generating complex geometry. Journal on Shape Modelling 8, 2, pp. 173-191, Dec. 2002.
- [Tob02b] Tobler R.F., Maierhofer S., Wilkie A., A multiresolution mesh generation approach for procedural definition of complex geometry. Shape Modeling International, Banff, Canada, pp. 35-42, 2002.

An Approach to Convert 4D Geometry into a 4D CT Scan

P.F. VILLARD M. BEUVE B. SHARIAT
L.I.R.I.S : Lyon Research Center for Images and Intelligent Information Systems
Bâtiment Nautibus, 8 boulevard Niels Bohr
69622 Villeurbanne Cedex, FRANCE
pierre-frederic.villard@iris.cnrs.fr

ABSTRACT

We present here an approach to convert the geometrical information produced by a physical simulation of soft-organ motion into a 3D+time CT scan. The paper describes how we calculate matter density at mesh points and how we produce dynamic 3D CT scan using the convolution parameters of medical scanners. The aim of this work is to provide physicians with standard images useful to appreciate organ motions and to incorporate them into a treatment planning platform.

Keywords

Radiotherapy, Hadrontherapy, CT Scan, Organ Motion, Simulation, Convolution, mesh, interpolation functions.

1 INTRODUCTION

A crucial problem in radiotherapy and hadrontherapy stems from organs motion due, in the case of lung tumours, to patient's breathing. Moving tumours are indeed hard to be well targeted. An improvement can be achieved with 4D CT scan providing necessary information on organs displacements.

In our latter works, we developed a pulmonary motion simulation using the well-known continuous mechanics techniques. Results consist of a "time-dependent" mesh of bulk and surface, describing various geometrical states of lung inflating [VBS+05]. We propose now to convert the displacement information into CT-scan images because of three reasons. First, we need to compare our model simulated image with real clinical CT-scan images. Second, physicians are used to proceeding only with CT scan images. Last, 4D CT scans, i.e. time varying densitometric data matrix, are to be used in a treatment planning software for dosimetry calculation. In our approach, the only inputs we need are one

initial CT scan and the displacement at mesh nodes for a series of organ-motion stage. A state of the art describing the CT scan technology as well as image reconstruction techniques is presented in §2. We learn in particular that to transform motion information into CT scan two problems should to be considered: 1-Since CT scan data, namely the Hounsfield units, are obtained from the matter density, we shall see in §3 how to compute this density from the mass conservation equation. 2-Since CT scan images are the results of complex operations, we shall see in section §3 how to use matter density information to compute realistic CT scan images. The global principle of our approach is illustrated on Figure.1. The "numerical experiments" section exhibits a numerical validation.

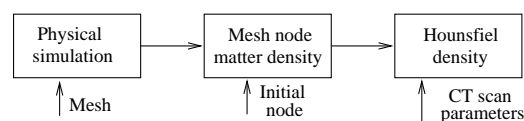


Figure 1: Illustration of the different steps of our work

2 STATE OF THE ART

CT scan device

CT scan devices aim at measuring electronic density of tissues by X-ray absorption. It is computed from the extraction of a set of attenuation coefficient μ defined for homogeneous materials by: $I = I_0 \cdot \exp^{-\mu x}$ where I is the intensity of the X-ray beam after crossing a thickness x of tissues

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 2006 conference proceedings, ISBN 80-86943-05-4
WSCG'2006, January 30 – February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

and I_o is the initial intensity of the x-ray beam (Cf Figure.2).

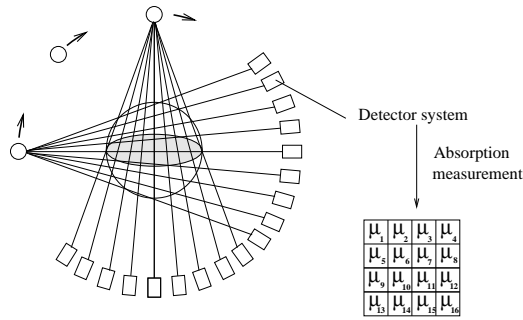


Figure 2: CT scan measurement principle

CT scan slices are reconstructed from a number of global attenuation coefficient measurements taken from various incidences. For each angle, the non absorbed beam dose hits then several detectors, producing 2D "density profiles" at several projection angles. To convert this set of 2D density profiles into attenuation coefficients, and then into 3D density, one has to "voxelize" the space. Each voxel is supposed to be homogeneous, isotropic and characterised by one attenuation coefficient. To constitute the voxels [G.T80] two successive operations are executed:

1. Retro-projection: it corresponds to a 3D reconstruction, i.e. the inverse projection of the numerical values obtained on the detector plane, to the 3D space coordinates [Rad17];
2. The Convolution or filtering: it consists in improving the quality of the reconstruction. We focus on this operation in the next section.

Convolution algorithm

The convolution algorithm (with filter or kernel) is defined by a mathematical process used to compensate the measurement errors related to the physics of the device (beam hardening, ...) and to the reconstruction operation. In the majority of CT-scan modules, several filtering algorithms are available. Aspect and characteristics of the resulting CT scan images depend strongly on the selected algorithm. According to clinical needs, it can be necessary to choose an algorithm that provides a higher space resolution, for the detailed representation of the bone and other volumes of high contrast such as pulmonary parenchyma. For example, algorithms developed for GE, Philipps, Siemens and Toshiba scanners are described in [Kea05]. Two kinds of deconvolution algorithms are commonly used: the bone kernel and the soft organs kernel. Bone kernel as presented in [SL74] allows sharp edges and accurate resolution,

but is sensitive to noise. For the clinical applications where resolution is of less importance than contrast, for example, in lung CT scan, soft-organ kernels are used. The kernel described in [RL71] gives smooth edges, high contrast and is robust to noise. To summarise, image noise decreases with flattened convolution kernel, simultaneously reducing the space resolution and giving more details to low contrast zones (Cf Figure.3).

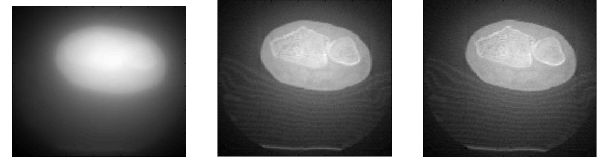


Figure 3: Filter influence: none (a.) soft tissue (b.) and bone (c.)

Scanner convolution modelling

In [JWS⁺03], the authors show how 3D CT scan data can be modelled as a linear system where an arbitrary cross section in a 3D image can be approximated by the following convolution: $g(x) = p(x) \otimes \lambda(x)$, where $p(x) = G_\sigma(x)$ is a 2D Gaussian function with σ as standard deviation, $g(x)$ is the selected cross section and $\lambda(x)$ is the real cross section, i.e., the true image. The optimal σ to produce the optimal image is found by statistical methods. Similar methods could be found in [WVS⁺98].

DRR image generation

Digitally Reconstructed Radiographs generation is a method to simulate CT scan slices. They are generated by ray-tracing techniques through a 3D volume extracted from CT scan but with a modified beam's eye view. The aim is to convert each voxel Hounsfield density into attenuation coefficient, x-ray energy specific. Several works describe DRR construction processes ([BDZ⁺99], [MBG⁺00]). Nevertheless, such techniques are only available when the geometry undergoes Euclidean transformations (rotations or translations) and cannot be applied to our case where solids are completely deformed, i.e. when the voxel density varies.

3 FORMALISATION

Our aim is to calculate the matter density in order to produce a 4D CT scan using the scanner convolution parameters associated to the clinical scanner devices. To do this, we assume that the density is a continuous function well represented by the interpolation functions of the mesh-points density values. We shall see that 4D CT scan can be built up from these density values.

Mesh node density computation

The only input we need are the displacement and the knowledge of the initial density (extracted from one CT scan) at each mesh node. In the following we denote \mathbf{P}_j a mesh node, \mathbf{U} a displacement and ρ_M the matter density. The key point of our approach is the mass conservation (Equation.1):

$$\frac{\partial \rho_M}{\partial t} + \text{div}(\rho_M \cdot \mathbf{V}) = 0 \quad (1)$$

The interest of this equation, is the presence of the velocity \mathbf{V} that could be replaced by displacement after integration. Let us then integrate Equation.1 over a small time step $[t_i, t_i + \Delta t]$: The *divergence* operator being independent of time, previous equation reads:

$$\rho(\mathbf{P}, t_i + \Delta t) - \rho(\mathbf{P}, t_i) + \text{div} \int_{t_i}^{t_i + \Delta t} (\rho \cdot \mathbf{V}) dt = 0 \quad (2)$$

Let $\Delta \rho = \rho(\mathbf{P}, t_i + \Delta t) - \rho(\mathbf{P}, t_i)$ be the required quantity. We note in Equation.2 the presence of ρ in $\int \rho \cdot \mathbf{V} dt$ avoids a direct link with displacement. A Taylor series development of $\rho(p, t_i - \Delta t)$ about t_i gives:

$$\begin{aligned} \rho(\mathbf{P}, t_i - \Delta t) = \\ \rho(\mathbf{P}, t_i) + (\Delta t) \frac{\partial \rho(\mathbf{P}, t_i)}{\partial t} + O((\Delta t)^2) \end{aligned} \quad (3)$$

Finally due to the time integration Equation.2 reads :

$$\begin{aligned} \Delta \rho = \\ - \text{div} \left(\int_{t_i}^{t_i + \Delta t} \rho(\mathbf{P}, t_i) \cdot \mathbf{V}(\mathbf{P}, t) dt \right) + O((\Delta t)^2) \end{aligned} \quad (4)$$

$\rho(\mathbf{P}, t_i)$ is time independent then:

$$\Delta \rho \approx - \text{div} \left(\rho(\mathbf{P}, t_i) \cdot \int_{t_i}^{t_i + \Delta t} \mathbf{V}(\mathbf{P}, t) dt \right) \quad (5)$$

If we denote $\mathbf{U}_i(\mathbf{P})$ the displacement of the point \mathbf{P} from the time t_i to the time $t_i + \Delta t$, velocity definition gives:

$$\Delta \rho \approx - \text{div} (\rho(\mathbf{P}, t_i) \cdot \mathbf{U}_i(\mathbf{P})) \quad (6)$$

One can now use the interpolation functions often considered in Finite Element Method [PFT⁺92]. In this method, the considered solid is

subdivided into elements E_i composing a mesh. For any point $P \in E_i$ we have :

$$\begin{cases} \rho(\mathbf{P}, t_i) = \sum_{j \in E_i} N_j(\mathbf{P}) \cdot \rho(\mathbf{P}_j, t_i) \\ \mathbf{U}_i(\mathbf{P}) = \sum_{j' \in E_i} N_{j'}(\mathbf{P}) \cdot \mathbf{U}_i(\mathbf{P}_{j'}) \end{cases} \quad (7)$$

where $N_j(\mathbf{P})$ is the interpolation function at the node j . Then Equation.6 can be written for any point P in element E_i :

$$\Delta \rho \approx - \sum_{j, j'} \rho(\mathbf{P}_j, t_i) \cdot \text{div} (N_j(\mathbf{P}) \cdot N_{j'}(\mathbf{P}) \cdot \mathbf{U}_i(\mathbf{P}_{j'})) \quad (8)$$

Let us develop the *divergence* term considering now the space independence of \mathbf{U}_i :

$$\begin{aligned} \text{div}(N_j(\mathbf{P}) \cdot N_{j'}(\mathbf{P}) \cdot \mathbf{U}_i(\mathbf{P})) = \\ \frac{\partial N_j \cdot N_{j'}}{\partial x} \cdot U_x + \frac{\partial N_j \cdot N_{j'}}{\partial y} \cdot U_y + \frac{\partial N_j \cdot N_{j'}}{\partial z} \cdot U_z \end{aligned} \quad (9)$$

Finally for any \mathbf{P} , it is possible to calculate the divergence and then $\Delta \rho$ from the knowledge of:

- The node position \mathbf{P}_j ;
- The node initial densities $\rho(\mathbf{P}_j, t_i)$;
- The node displacements $\mathbf{U}_i(\mathbf{P}_{j'})$.

Scanner convolution

To compute the Hounsfield density of each voxel it could be possible to take only into account the contribution of each deformed element to the voxel, weighting this contribution by the percentage of voxel volume, within the deformed element volume (Cf Figure.4). Then it would be easy to transform the average density into Hounsfield density but the most tedious would be to evaluate the intersection volume. However it would not take into account the aspects of CT scan convolution process. Instead we propose to proceed to a convolution.

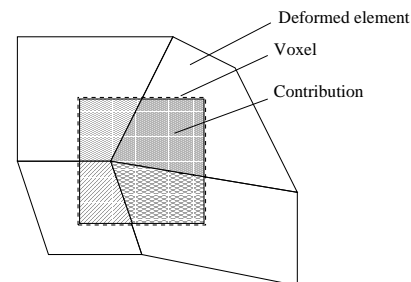


Figure 4: volume of deformed elements inside a considered voxel

The Hounsfield density $\rho_h(P)$ is an affine function of the matter density $\rho_M(P)$ [MBG⁺00]. Let $g(x) = ax + b$ be this function:

$$\rho_H(P) = a\rho_M(P) + b \quad (10)$$

From the previous section we get at each point the matter density $\rho_M(P)$. It is computed in any solid point by interpolation (Equation.7) of the node values $\rho_M(P_j)$ belonging to the mesh element E_i with the functions N_j :

$$\rho_M(P) = \sum_{j \in E_i} N_j(P) \rho_M(P_j) \quad (11)$$

By injecting (11) in (10) we obtain the Hounsfield density expression at any point P in an element E_i . Let us $\rho'_H(P')$ be the Hounsfield number at point P' . Its expression is given by a convolution of the mesh element density $\rho_H(P)$ with a kernel filter as the ones explained in the state of the art (Cf Figure.5).

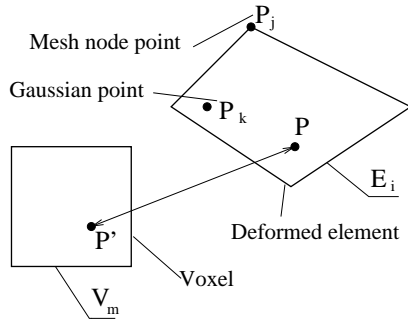


Figure 5: Mesh element influence on a voxel

$\rho'_H(P')$ is then given by:

$$\begin{aligned} \rho'_H &= f_\sigma \otimes \rho_H \\ &= f_\sigma \otimes (g \circ \rho_M) \end{aligned} \quad (12)$$

which reads:

$$\rho'_H(P') = \sum_{E_i} \int_{E_i} f_\sigma(\|P - P'\|^2) \cdot g(\rho_H(P)) dP^3 \quad (13)$$

If one defines the densitometric value $D_H(V_m)$ at voxel V_m , as the average of the Hounsfield density function over the voxel, then the expression 13 has to be integrated over the voxel (Cf Figure.6). If we denote \mathcal{V}_m the voxel volume:

$$D_H(V_m) = \frac{1}{\mathcal{V}_m} \int_{V_m} \rho'_H(P') dP'^3 \quad (14)$$

Injecting Equation.13 in Equation.14, we obtain:

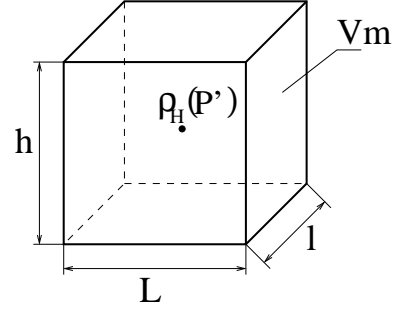


Figure 6: Voxel parameters

$$\begin{aligned} D_H(V_m) &= \frac{a}{\mathcal{V}_m} \\ &\sum_{E_i} \int_{E_i} \rho_M(P) \int_{V_m} f_\sigma(\|P - P'\|^2) dP'^3 \cdot dP^3 + b \end{aligned} \quad (15)$$

The latter simplification comes from the fact that the space integral of f_σ is equal to 1. The expression $I(P) = \int_{V_m} f_{\sigma}(\|P - P'\|^2) dP'^3$ is independent of the function $\rho_h(P)$. Since the geometry of V_m is known and constant, an analytical expression for $I(P)$ can be derived. As it is often convenient we could consider for f_σ a truncated filter as presented in [THG00]. But the function must be integrated over the field given by the intersection of a voxel with a sphere.

This integration is tedious and a filter defined on an infinity space is thus used. We used the 3D Gaussian function with σ the standard deviation.

As previously seen, the $\rho_h(p)$ function can be expressed as a sum of interpolation functions of the node values, which gives to $D_H(V_m)$ the expression:

$$\begin{aligned} D_H(V_m) &= \frac{a}{\mathcal{V}_m} \sum_{E_i} \sum_{j \in E_i} \int_{E_i} N_j(P) \rho_H(P_j) I(P) dP^3 + b \end{aligned} \quad (16)$$

The integral $\int_{E_i} N_j(P) \rho_H(P_j) I(P) dP^3$ can be integrated numerically by a Gaussian Quadrature [Gau94], which consists in a numerical estimations of an integral by picking optimal points called the Gaussian Points (P_k). Due to the element distortion, the integration domain is rather complex. It is then convenient to apply a reference change from a reference element of simple geometry to a real element. One has only to include the associated Jacobian matrix J :

$$\begin{aligned} & \int_{E_i} N_j(P) \rho_H(P_j) I(P) dP^3 \\ &= \sum_{k \in E_i} w_k N_j(P_k) \rho_H(P_j) I(P_k) \det_k \end{aligned} \quad (17)$$

Where \det_k represents the Jacobian determinant and w_k are weights associated with the Gaussian Points. Finally, the expression of the Hounsfield density on a voxel V_m of the scanner is given by the equation (18).

$$\begin{aligned} D_H(V_m) = & \\ \frac{a}{V_m} \sum_{E_i} \sum_{j \in E_i} \sum_{k \in E_i} & w_k N_j(P_k) \rho_H(P_j) I(P_k) \det_k + b \end{aligned} \quad (18)$$

4 RESULTS

This result step aims at validating our approach. Two computing phases are here qualitatively and quantitatively checked : matter density on mesh nodes and scanner convolution. The most critical stage is the node density computation because the mass conservation must be ensured. We shall test not only the mathematical validity but also the numerical validity.

Trial definition

The trial consists of pulling a cube fixed on one of its faces. Its edge length is 10 and the mesh is composed of small hexahedra with edge length $dx = dy = dz$ (Cf Figure.7).

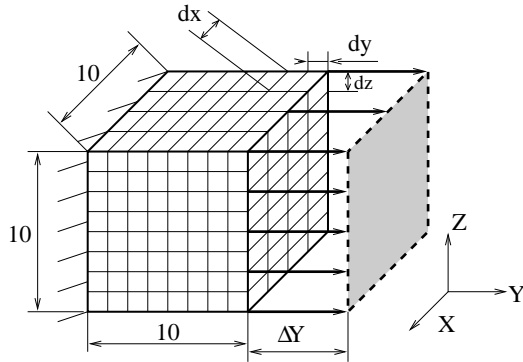


Figure 7: Trial cube geometry

The density in any point within the cube is set to 1 at initial state. The test consists in completely fixing one of the cube faces (plan \mathbf{XZ} with $y = 0$) and in pulling the opposite face (plan \mathbf{XZ} with $y = 10$) with a ΔY length in the direction of \mathbf{Y} leaving free the \mathbf{X} and \mathbf{Z} displacements (Cf

Figure.7). The cube stretching will be accompanied by a axial thinning in \mathbf{X} and \mathbf{Z} .

The interest of such kind of tests is the simplicity of this system in terms of mechanical behaviour. We can thus qualitatively evaluate if the calculations are realistic and self-consistent.

Figure 8 shows the displacements obtained by finite element method (FEM) computed with the software Code-Aster [Ast]. The displacements scale is voluntarily exaggerated to highlight the stretching in \mathbf{Y} and the contraction in \mathbf{X} and \mathbf{Z} . The result is calculated with n computing steps.

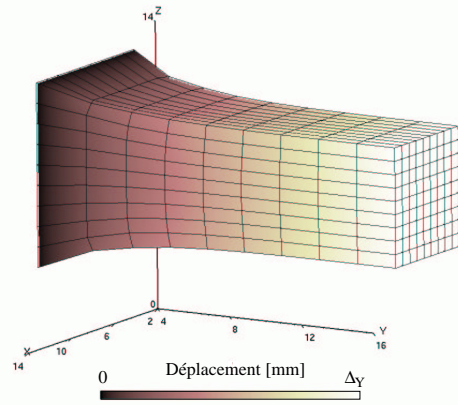


Figure 8: Displacements computed with FEM. The grey scale represents the displacement amplitude.

Numerical validation

The validation stages focus both on matter conservation and on CT scan imaging validations.

A first stage consists in a qualitative validation of the spatial evolution of the matter density . Figure 9 shows a continuous representation of density values computed on mesh nodes. It is a smoothing with the Gouraud method, i.e. colour interpolations. The transversal slice points out a density reduction in the centre of the cube due to pulling. As expected, in the periphery, as the volume decreases, the density increases. In between, the density remains close to 1. The longitudinal slice shows out a low density where the cube is fixed. Indeed at the vicinity of this zone, the volume increases. At the opposite face, free x and z displacements result in less shear and then less density changes.

All these observations are in full agreement with those that must be obtained. In other words we can conclude a qualitative validation. We can now study a more quantitative aspect by checking if the

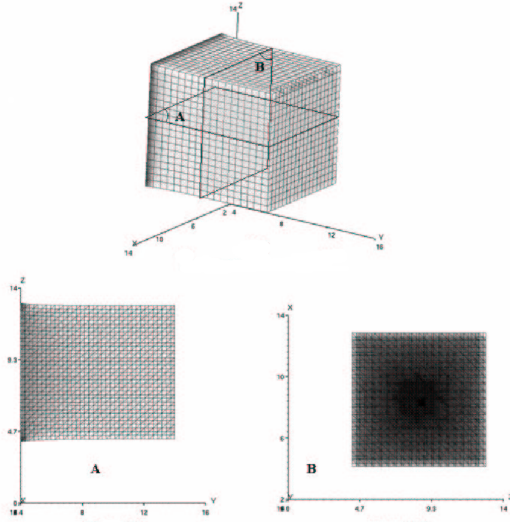


Figure 9: Continuum representation of matter density. **A** : longitudinal slice. **B** : transversal slice.

system mass - i.e. the integral over the volume of the density - remains constant.

To evaluate the level of conformity we defined a mass loss error at each computing step:

$$\begin{aligned} \text{Error}[\%] \\ = \frac{\text{mass}_{\text{theoretical}} - \text{mass}_{\text{computed}}}{\text{mass}_{\text{theoretical}}} \times 100 \end{aligned} \quad (19)$$

which gives:

$$\begin{aligned} \text{Error}[\%] \\ = \frac{m_0 - (\sum_{E_i} \int_{E_i} \rho_M^i(P) dP)}{m_0} \times 100 \end{aligned} \quad (20)$$

where m_0 is the initial mass.

The results of these tests are gathered in the following Tables. The influence of several parameters on this error is shown. Table.1 deals with the number of computing steps n , which monitors the level of Taylor series approximation seen in (3). Table.2 deals with the applied-displacement amplitude, which fixes the level of distortion in geometry and Table.3 deals with mesh resolution and therefor with the accuracy of interpolating-function approximation.

The results of Tables 1, 2 and 3 show that the computing precision varies according to the computing number of steps (if it is multiplied by two, the error is approximately divided by two). The mesh resolution plays a less significant role, which

Computing step	Total computing step number			
	1	5	10	100
0.1n			0.008	0.001
0.2n		0.033	0.017	0.002
0.3n			0.025	0.004
0.4n		0.066	0.034	0.006
0.5n			0.043	0.008
0.6n		0.098	0.052	0.01
0.7n			0.061	0.013
0.8n		0.131	0.070	0.016
0.9n			0.079	0.02
n	0.796	0.163	0.089	0.0223

Table 1: n influence on mass conservation error with $\Delta Y = 10$ and $dx = 0.5$

Computing step	Applied displacement		
	1%	10%	100%
0.1n	0	0.008	0.796
0.2n	0	0.017	1.460
0.3n	0	0.025	2.052
0.4n	0	0.034	2.610
0.5n	0	0.043	3.162
0.6n	0.001	0.052	3.724
0.7n	0.001	0.061	4.312
0.8n	0.001	0.070	4.935
0.9n	0.001	0.079	5.602
n	0.001	0.089	6.322

Table 2: ΔY influence on mass conservation error with $dx = 0.5$ and $n = 10$

computing step	Mesh resolution		
	1	0.5	0.25
0.1n	0.008	0.008	0.008
0.2n	0.017	0.017	0.017
0.3n	0.025	0.025	0.025
0.4n	0.034	0.034	0.034
0.5n	0.043	0.043	0.042
0.6n	0.052	0.052	0.051
0.7n	0.061	0.061	0.060
0.8n	0.071	0.070	0.070
0.9n	0.081	0.079	0.079
n	0.091	0.089	0.089

Table 3: dx influence on mass conservation error with $\Delta Y = 10$ and $n = 10$

is a good point since on the other sides the computing time increases considerably with the number of elements. The last important observation is that a very significant displacement gives unacceptable errors. However this can be compensated by increasing the number of computing steps. We even conclude that a good criterion is to fix $\Delta Y/n$ small enough. For example, in the case of the 100% pulling (Cf Table.2), if calculation is carried out on

200 computation steps, the maximum error falls down to 0.461%.

The following test consists in checking if our defined convolution function is correct. To separate any possible errors in density calculation, we arbitrarily fixed the density to 1 and show the result on Figure.10.



Figure 10: CT scan convolution with constant node density and $n = 10$. a) $t=0$, b) $t=5$, c) $t=10$

We observe a low grey level at the border due to the smoothing effect of the convolution. No artefact can be noted. The result given on Figure.11 includes the calculated density.



Figure 11: CT scan convolution with computed node density and $n = 10$. a) $t=0$, b) $t=5$, c) $t=10$

The results obtained are very satisfactory.

5 CONCLUSION

The method presented in this work converts mesh displacement into 4D CT scan. The appealing aspect of our approach lies in the fact that the simulated-displacement data could be calculated with any kind of physical modelling resulting from 3D CT scan.

The module we built up needs only one parameter: the standard deviation σ of the CT scan device used for the signal treatment. Thus it remains the only data to obtain in order to have an effective 4D CT scan.

The success of our qualitative and quantitative validation now allow to go further: application of this module to a real organ motion and real density for clinical validation and finally a dynamic dosimetry.

REFERENCES

[Ast] Code Aster. <http://www.code-aster.org/>.
 [BDZ⁺99] M.L. Bahner, J. Debus, A. Zabel, S. Levegrun, and G. Van Kaick. Digitally reconstructed radiographs from abdominal CT scans as a new tool for radiotherapy planning. *Invest. Radiol.*, 34(7):643, 1999.

[Gau94] C.F. Gauss. Methodus nova integralium valores per approx. inveniendi. *Werke*, 3:163, 1794.
 [G.T80] G.T.Herman. *Image Reconstruction from Projections*. Academic Press, New York, 1980.
 [JWS⁺03] M. Jiang, C. Wang, M. Skinner, J. Rubinstein, and M. Vannier. Blind deblurring of spiral CT images. *IEEE Trans. Med. Imaging*, 22(7):837–845, 2003.
 [Kea05] N. Keat. CT scanner automatic exposure control systems. Technical Report 05016, IMPACT, February 2005.
 [MBG⁺00] N. Milickovic, D. Baltas, S. Giannouli, M. Lahanas, and N. Zamboglou. CT imaging based digitally reconstructed radiographs and their application in brachytherapy. *Physics in Medicine and Biology*, 45:2787–2800, 2000.
 [PFT⁺92] W.H. Press, B.P. Flannery, S.A. Teukolsky, , and W. T. Vetterling. *Numerical Recipes in FORTRAN: The Art of Scientific Computing, 2nd ed.* Cambridge University Press, Cambridge, England, 1992.
 [Rad17] J. Radon. Über die bestimmung von funktionen durch ihre integralwerte längs gewisser mannigfaltigkeiten. *Mathematisch-PhysikalischeKlasse*, 69(77), 1917.
 [RL71] G.N. Ramachandran and A.V. Lakshminarayanan. Three dimensional reconstructions from radiographs and electron micrographs: Application of convolution instead of fourier transform. *Proc. Nat Acad Sci*, 68:2236–2240, 1971.
 [SL74] L.A. Shepp and B.F. Logan. The fourier reconstruction of a head section. *Trans. on Nuclear Science*, 21:21–43, 1974.
 [THG00] T. Theußl, H. Hauser, and M.E. Gröller. Mastering windows: Improving reconstruction. Technical report, Institute of Computer Graphics and Algorithms, Austria, 2000.
 [VBS⁺05] P-F. Villard, M. Beuve, B. Shariat, V. Baudet, and F. Jaillet. Simulation of lung behaviour with finite elements : Influence of biomechanical parameters. *IEEE, Mediviz, Conference on Information Visualization*, pages 9–14, 2005.
 [WVS⁺98] G. Wang, M. W. Vannier, M. W. Skinner, M. G. P. Cavalcanti, and G. Harding. Spiral CT image deblurring for cochlear implantation. *Trans. Medical Imaging*, 17:251 – 262, 1998.

Bending Models for Thin Flexible Objects

B. Thomaszewski*

WSI/GRIS,

Sand 14, D-72076 Tübingen, Germany,

and

M. Wacker†

HTW university of applied sciences Dresden,

Friedrich-List-Platz 1, 01069 Dresden.

ABSTRACT

Textiles usually exhibit much larger resistance to in-plane deformation than to bending deformation. However, the latter essentially determines the formation of folds and wrinkles which in turn govern the overall appearance of the cloth. The resulting numerical problem is inherently stiff and hence susceptible to instability. This overview is devoted to a closer investigation of bending deformation. Approaches known from the field of engineering can describe the problem of bending in a physically accurate way. However, the nature of the governing equations is such that they cannot be discretised with the standard methods currently used in cloth simulation. Since curvature is a central variable, we introduce related concepts from differential geometry and describe the transition to the discrete setting. Different approaches are discussed and demands on an approach for correctly modelling the bending behaviour of cloth are formulated.

Keywords cloth simulation, physically based simulation, bending

1 Introduction

The most salient characteristic of thin flexible objects is their bending behaviour. Typically, objects from this category show a relatively large resistance to in-plane deformations such as stretching and shearing while the forces due to out-of-plane deformation, i.e. bending, are small. However, this does not mean that the treatment of bending is less important. On the contrary, due to these different reactions to deformation the characteristic folds and wrinkles that we associate with garments are actually formed. Especially in the case of compressive in-plane deformation, i.e. when buckling occurs, the bending behaviour is crucial. Despite its importance, bending has rather

been neglected in the physically based simulation of clothes. In contrast, for modelling the in-plane properties of fabrics sophisticated and accurate models are used [BHW94, EWS96]. We think that this discrepancy should be addressed. While this paper provides an introduction to the problem of bending for thin flexible objects, we report on a related implementation and give examples of cloth for illustration in [TWS05].

This overview starts with a look at how existing techniques for cloth simulation deal with bending, including the way curvature is approximated. Subsequently, approaches to bending dominated problems from the field of engineering will be examined more closely. These methods allow a description in a physically accurate way. At the same time however, the governing equations cannot be discretised with the standard approaches currently used in cloth simulation. We will therefore consider alternative ways. Since curvature is a central variable, we will briefly review some related concepts from differential geometry. The next section shows how a general transition to the discrete setting can be made. This work concludes with a discussion of the presented material and an outlook on how a physically accurate model for bending can be devised.

2 Bending

This section starts with an overview of bending models used in physically based simulation. Although 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.

SHORT COMMUNICATIONS proceedings, Vol.9, No.1., ISBN 80-86943-05-4

WSCG 2006, January 30 - February 3, 2006, Plzen, Czech Republic.

Copyright UNION Agency – Science Press.

exact form depends on the actual method, the general steps to set up a physically based model for elastic deformations on a discrete representation of an object can be summarised as follows: first, an elastic energy E resulting from deformation has to be determined. This involves a material law relating strain to stress. In the case of bending energy, this strain is a change in curvature. By differentiation of E with respect to nodal positions, the force acting on a vertex is obtained. Lastly, if an implicit integration scheme is used for stepping forward the system over time, the Jacobian of the nodal forces has to be computed, too. Therefore, second order derivatives of all terms contributing to the energy have to be available. Since the computation of these derivatives can become very complex, some of the following methods model bending forces directly without relation to energy.

2.1 Existing Bending Models

In the seminal work of Terzopoulos et al. [TPBF87] a model for the animation of elastically deformable surfaces based on continuum mechanics is presented. The authors derive an elastic strain energy depending on nonlinear differential quantities, namely the metric and curvature tensors. The associated partial differential equations are discretised in space using finite differences on a regular quadrilateral grid. Although this approach is based on a physically sound theory, it was not widely adopted in the computer graphics community, due to its significant computational complexity. In the following years, approaches for the simulation of deformable surfaces like cloth mainly relied on particle and mass-spring systems. For modelling forces due to bending deformation, most of these methods use some kind of angular measure to approximate curvature. Breen et al. [BHW94] were among the first to use a coupled particle system in cloth simulation. The authors present an approach based on energy potentials for modelling the static drape of cloth. Departing from linear beam theory (see section 3), they first derive the bending energy between two successive edges in a rectangular discretisation. Curvature is approximated by fitting a circle through the three points involved. For large bending deformations, a different measure is used which in sum yields a biphasic curvature expression. Breen et al. model bending energy using data obtained from measurements with the *Kawabata Evaluation System* (KES) [Kaw80]. The corresponding nonlinear stress-strain curves are approximated numerically with quadratic fits. Once energies are set up at the nodes, the gradients have to be computed to obtain nodal forces. The approach presented by Eberhardt et al. [EWS96] extends this work to the dynamic range. Computation times are greatly reduced using sophisticated integration schemes. Eberhardt et al. do not explicitly approximate curvature but directly

use the angle as a deformation measure.

Volino et al. [VCMT95] use a mass-spring system inspired by continuum mechanics. The basic bending element is formed by two adjacent triangles from the underlying unstructured grid. To determine curvature, a circle fitting inside the two triangles is found using the dihedral angle. The curvature over an element is then obtained as the inverse of the circle's radius. The authors point out that the curvature has to be limited to a certain maximum to prevent bending forces from growing to infinity. The actual forces are deduced from the geometry of the involved triangles using linear beam theory. Baraff et al. [BW98] use the same basic bending element as in [VCMT95]. Following their proposed computational framework, a constraint expression for bending energy is derived. This essentially corresponds to an energy term which depends quadratically on the dihedral angle.

A different approach to cloth simulation was proposed by Eischen et al. [EDC96, EB00]. Their method is based on the nonlinear shell theory derived by Simo et al. [SFR89]. A four node bilinear element with nodal displacements and director rotations as the primary unknowns is used for discretisation (see [SFR89]). In the context of shell theory, curvature is directly accessible through bending strains and does not need not be approximated otherwise. Like in [BHW94] Eischen et al. use measured data obtained from the KES and approximate the curves with a 5th-order polynomial fit. In sum, the approach leads to highly nonlinear equilibrium equations which have to be solved, for example, with the Newton-Raphson procedure. This solver is coupled with an adaptive arc length control to account for limit or bifurcation points in the solution due to buckling instabilities. The proposed method is limited to the static case and does not account for dynamic effects. For subsequent comparison, Eischen et al. [EB00] present a particle-based approach based on principles found in continuum mechanics. Like Breen et al. they use a regular quadrilateral discretisation along with linear elasticity theory. However, they derive forces directly without explicit resort to energy potentials. Bending forces are computed using linear beam theory which again results in a linear moment-curvature relationship. The angle formed by two consecutive edges is taken as a direct measure for curvature. The authors state that the results of the two methods cannot be visually distinguished on the scale of the images they produced.

More recently, Choi et al. [CK02] proposed a bending model based on assumptions on the buckling behaviour of fabric. Departing from a quadrilateral mass-spring system, the basic bending element consists of an interleaved spring. The authors advocate that compressive in-plane forces on textiles lead to large out-of-plane deflections once a critical loading is reached. For the notoriously unstable post-buckling

state the buckled shape is predicted as a circular arc of constant length and curvature. With this assumption, the curvature can be computed analytically without angular expressions appearing. Hence, linear beam theory can be applied to derive the bending energy. Lastly, the authors derive expressions for force vectors and Jacobians at the nodes required in an implicit time integration scheme.

Bridson et al. [BMF03] proposed another derivation of bending forces for cloth animation. Again, two adjacent triangles form the basic bending element. With the requirement that bending forces should neither cause in-plane deformation of the fabric nor lead to rigid body motions, they derive the directions and relative magnitudes for the four bending force vectors of an element. These vectors are then scaled with a bending stiffness constant and the sine of the dihedral angle. An additional scaling factor accounts for anisotropy of the mesh. For the numerical time integration, Bridson et al. suggest to use a mixed implicit-explicit integration scheme in which the (comparably small) bending forces can be handled in an explicit manner while viscous damping forces are treated implicitly. Thus, the computation of the complicated derivatives of the bending forces is avoided.

While most of the previous approaches use a rather rough curvature approximation, Grinspun et al. [GH⁺03] presented a method which is based on a sound curvature derivation. Their work extends existing cloth simulators to the range of objects with a strong resistance to bending deformation. To this end, a discrete flexural energy potential is established using differential geometry (see section 5). Again, the basic bending element consists of two adjacent triangles. The energy derives from an approximation to the squared difference of mean curvature in the current and initial configuration. A drawback of this approach is that the derivatives of the bending energy are intricate to compute. Because of this complexity, the authors suggest the use of an automatic differentiation system.

Yet another way to treat bending was proposed by Eitzmuß et al. [EKS03]. They use a discrete approximation of the surface Laplacian to model curvature. This has some aspects in common with the discrete mean curvature computation described in section 5. In the context of a linear finite element approach the Laplacian is computed for each element and projected onto the corresponding vertex normals. The element contributions are summed up to give the pointwise value for every vertex.

2.2 The Concept of Bending

Common deformation modes in 3D continuum mechanics are stretching and shearing. These modes are orthogonal to each other: pure stretching does not

lead to shear deformation and *vice versa*. The pointwise view of (3D) continuum mechanics does not account for bending since it is indifferent of shape. The

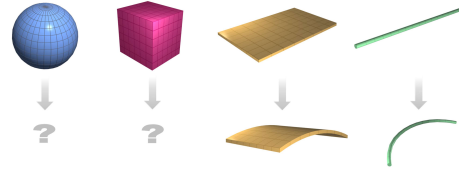


Figure 1: Whether an object can be bent depends mainly on its shape.

ability to bend an object is, however, closely related to its shape, or more precisely, to the proportion of its extents in the different dimensions. Consider e.g. a thin plate as shown in Figure 1. Here, one direction can be distinguished, in which the lengths are clearly inferior to those in the orthogonal directions. In this case, the intuitive bending deformation is such that it causes as little in-plane deformation as possible – a pure change in curvature. For a parametric surface which can be thought of as an infinitely thin plate this bending deformation can be determined analytically (see section 4). We examine the notion of bending for objects with finite thickness subsequently.

2.3 Bending with Finite Thickness

For a cylindrically bent plate (see Fig. 1), we can – without loss of generality – restrict our investigations to a thin slice. Thus, we arrive at a geometry corresponding to the classical beam element. A cross-sectional view of such a beam is shown in Figure 2. It can be seen that during deformation the bottom layer is stretched while the top layer is compressed (cf. [Kee99]). We can reasonably assume that the maximum values of tension and compression occur on the boundary layers. If we further assume that the induced stresses vary monotonically between these maxima we arrive at an axis with zero stress, the so called *neutral axis* (see Figure 2). These ge-

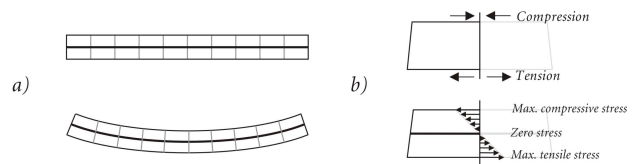


Figure 2: Beam geometry. a) Cross-sectional view of bending deformation. b) Linearly varying stresses through the thickness and *neutral axis*.

ometric relations motivate an analytic treatment of the problem in which the neutral axis is the primary parametrisation domain. This dimension reduction is the starting point for the theory of beams and plates which is introduced next.

3 Linear Elasticity of Beams and Plates

This section presents models for bending dominated problems known from engineering sciences. The simplest model corresponding to our interests is the 1D linear elastic beam. As we have seen in section 2, many existing approaches to bending in cloth simulation rely on this model. Because the stretching deformation of the neutral-axis is assumed to be negligible, the central unknown is the lateral deflection w of the neutral axis. The kinematic constraints leading to the common *Euler-Bernoulli* beam derive from the Kirchhoff-Love Assumptions: lines that are initially normal to the neutral axis remain straight, normal, and unstretched. The deformed state of the beam can be described by the displacements u_0 and w_0 of the neutral axis and a rotation θ of the normal (see Figure 3). The horizontal and vertical displacements

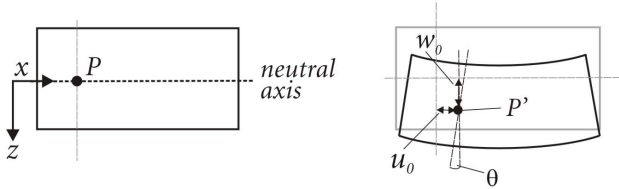


Figure 3: Displacements u_0 and w_0 of the neutral axis and cross-sectional rotation θ for a deformed beam element.

ments of any material point in the beam are given by $u(x, z) = u_0(x) - z\theta(x)$, $w(x, z) = w_0(x)$, and the generalised strain follows as $\varepsilon_x = \frac{\partial u}{\partial x} = \frac{\partial u_0}{\partial x} - z \frac{\partial \theta}{\partial x}$ (see [ZT00a]). Because normal lines are assumed to remain unstretched, the strain ε_z in this direction can be neglected. Using the second assumption, the transverse shear strain ε_{xz} equally vanishes.

With the strain defined, the stress now follows by the use of an appropriate constitutive law. For a linear elastic material law the stress is

$$\sigma_x = \frac{E}{1 - \nu^2} \varepsilon_x, \quad (1)$$

where E is Young's modulus and ν is Poisson's ratio. The bending moment around the horizontal axis is obtained as

$$M = D \frac{\partial \theta}{\partial x} = \frac{Eh^3}{12(1 - \nu^2)} \frac{\partial^2 w_0}{\partial x^2}. \quad (2)$$

Note the term $\frac{\partial^2 w_0}{\partial x^2}$ which, for small deflections w_0 , is actually the curvature κ of the beam. Thus, equation (2) can be written in a clearer manner as $M = D\kappa$. This linear moment-curvature relationship is exploited by some approaches in cloth simulation to directly model bending forces (e.g. [VCMT95]).

The governing equations are established by considering the forces acting on a differential beam element (Fig. 4). The beam is in equilibrium if the transverse

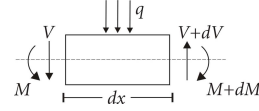


Figure 4: Distributed lateral forces q , transverse shear force V , and bending moment M acting on a differential beam element dx .

internal force V (or shear resultant) and the external distributed load q are in balance. Along with the moment equilibrium this leads to the equilibrium equation of the Euler-Bernoulli beam

$$\frac{Eh^3}{12(1 - \nu^2)} \frac{\partial^4 w}{\partial x^4} = -q. \quad (3)$$

The above formulations directly carry over to cylindrically deformed plates. They can as well be translated to the general case of (doubly curved) thin plates (see [ZT00b]). In engineering, thin plate elements are used to support lateral loads. Because curvature now occurs in both transverse directions, one speaks of the *neutral surface*, or simply *mid-surface*, in analogy to the neutral plane. Again, it is assumed that the stretch deformations of the mid-surface are negligible. Hence, the primary unknown is again the lateral deflection w . However, the deflection now varies in both x and y direction which renders the problem two-dimensional. For thin plates, the governing equation turns out to be

$$\frac{Eh^3}{12(1 - \nu^2)} \left(\frac{\partial^4 w}{\partial x^4} + 2 \frac{\partial^4 w}{\partial x^2 \partial y^2} + \frac{\partial^4 w}{\partial y^4} \right) = -q. \quad (4)$$

This is a biharmonic equation involving fourth order partial derivatives. Investigating the corresponding strains it can be seen, that second order derivatives of the (lateral) displacement field are required. The thin plate equations have been used in computer graphics, too. For instance, they appear in a common minimisation problem from variational design (see e.g. [WW98]). Despite its demands on continuity, the thin plate approach can be used in physically based simulation. Since this theory does not take into account in-plane deformations it has to be augmented by an appropriate membrane model for this purpose. This conjunction can be found in the class of *Kirchhoff-Love* thin shell theories.

As already mentioned in the introduction, in most of the existing techniques for the simulation of thin flexible objects, the displacement basis required in thin plate analysis is not available. Nevertheless, many approaches make use of the Euler-Bernoulli beam equations to model bending, for example, directly along the edges of the underlying mesh. It is also possible to set up bending energies for the discrete setting, say, in terms of mean-curvature (see [GH⁺03]). All of the methods that rely on such a physical model must necessarily use some kind of

curvature measure. Therefore, it is worth investigating what properties a reasonable curvature measure should have. To this end, we will proceed with some relevant material from differential geometry in the next section.

4 Curvature from Differential Geometry

The following concepts from differential geometry are mainly based on [Opr97] and [MDSB03]. The reader interested in a more detailed discussion is referred to the original work. Let \mathcal{S} be a surface (2-manifold) embedded in 3D space with a parametric description

$$\mathbf{r}(\theta^1, \theta^2) = (x(\theta^1, \theta^2), y(\theta^1, \theta^2), z(\theta^1, \theta^2)) , \quad (5)$$

where θ^1 and θ^2 are surface coordinates. At any point \mathbf{p} on the surface, the two tangent base vectors are spanned by the partial derivatives of the mapping (5) with respect to the surface coordinates

$$\mathbf{a}_\alpha = \frac{\partial \mathbf{r}}{\partial \theta^\alpha} . \quad (6)$$

The surface normal at this point is simply the cross-product. Curvature expressions for \mathcal{S} at a point \mathbf{p} are derived from lines on the surface defined as follows: for every direction $\hat{\mathbf{e}}_\varphi$ in the tangent plane let c_φ denote the curve that results from the intersection of \mathcal{S} with the plane spanned by the surface normal \mathbf{n} at \mathbf{p} and $\hat{\mathbf{e}}_\varphi$ (see Figure 5).

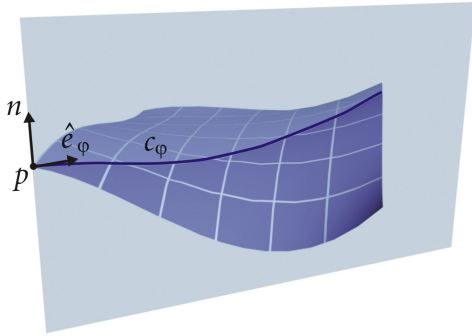


Figure 5: The surface curve c_φ is the intersection of the surface \mathcal{S} with the plane spanned by the normal \mathbf{n} and the unit direction vector $\hat{\mathbf{e}}_\varphi$.

The curvature of c_φ is called the *normal curvature* κ_N of \mathcal{S} in the direction $\hat{\mathbf{e}}_\varphi$. The minimum and maximum of these values are called the *principal curvatures* κ_1 and κ_2 . By integration over all unit directions the *mean curvature* at \mathbf{p} is obtained as

$$\kappa_H = \frac{1}{2\pi} \int_{2\pi} \kappa_N(\varphi) d\varphi = \frac{\kappa_1 + \kappa_2}{2} , \quad (7)$$

where the last equality follows from $\kappa_N(\varphi) = \kappa_1 \cos^2(\varphi) + \kappa_2 \sin^2(\varphi)$. A different expression for

mean curvature is given by the *mean curvature normal*

$$\kappa_H \mathbf{n} = \lim_{\text{diam}(\mathcal{A}) \rightarrow 0} \frac{\nabla \mathcal{A}}{2\mathcal{A}} , \quad (8)$$

where \mathcal{A} is an infinitesimal area around a point and $\text{diam}(\mathcal{A})$ is its diameter (see [MDSB03]).

Another important local property of a surface is the *Gaussian curvature* κ_G which is defined as the product of principal curvatures $\kappa_G = \kappa_1 \cdot \kappa_2$. The Gaussian curvature is independent of the surrounding embedding of the surface: it measures only *intrinsic* curvature and is not affected by pure bending deformations as shown e.g. in Figure 1. Therefore, Gaussian curvature is not an appropriate measure for bending deformation. In contrast, normal and mean curvatures measure inextensional, extrinsic deformation and thus reflect changes due to pure bending. The following section gives an idea of how these quantities can be transferred to the discrete setting of triangle meshes.

5 Discrete Curvature

Almost every reasonable bending model used in the simulation of flexible materials incorporates some measure of curvature. Of course, every such model must necessarily result in a discrete formulation - whether it is derived from a partial differential equation or not. Hence, the interest in a discrete curvature measure is obvious. In the continuous case the curvature tensor provides a scalar value for every unit direction at every point of a surface. A discrete counterpart should give these data at distinct features (say vertices or edges) of the mesh as an average over the pointwise values of its attributed surface part. This value can then be plugged into the desired bending energy equation, from which forces are derived for every vertex in the (triangle) mesh in the usual way. Furthermore, it is desirable for computational aspects that the operator is easy to evaluate. It should have minimal support, i.e. only require information from a small local neighbourhood. Lastly, the operator should be independent of the actual discretisation of the surface. Of course, a central question is what kind of curvature should be measured? In the previous section we have seen that Gaussian curvature is inappropriate. For simple isotropic materials the mean curvature is sufficient and we will focus on this quantity. However, if anisotropic material behaviour is desired, the full curvature tensor will most likely be needed.

There has been abundant work on defining and computing discrete differential quantities, e.g. [PP93, Tau95a] and more recently [CSM03]. A concise and sound derivation of a complete set of discrete differential operators for triangulated 2-manifolds can be found in the work of Meyer et al. [MDSB03]. We take this work as a basis for the following overview.

5.1 The Laplacian and Derived Operators

Loosely spoken, curvature is related to second order derivatives. A commonly known differential operator based on second order derivatives is the *Laplace Operator* which in 2D Euclidean space is defined as

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} . \quad (9)$$

The discrete approximation of this operator on a regular quadrilateral grid can be expressed by the 5-point star

$$\mathbf{L}_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} , \quad (10)$$

which is the result of taking second order finite differences in both dimensions. The generalisation of the Laplace operator from Euclidean space to 2-manifolds with Riemannian metric is called the *Laplace-Beltrami* operator. This is of interest here, because its discretisation leads to the mean curvature normal operator [Pol02a]. The question that arises is how this operator can be discretised on unstructured triangle meshes. Taubin [Tau95b] suggested to use an approximation which was later called the umbrella operator

$$\mathbf{L}_U = \frac{1}{m} \sum_{j \in N(i)} \mathbf{x}_j - \mathbf{x}_i , \quad (11)$$

where $N(i)$ is the set of neighbours of vertex i and m is the sum of the neighbour's valences. The advantage of this formulation is that it is linear in the vertex positions, just as the Laplacian on a regular quadrilateral setting. The drawback, however, is that it requires a specific parametrisation of the surface to be valid [KCVS98]. An extensions that accounts for irregularities is available but then the formulation is no longer linear [Fuj95]. As discussed subsequently, a more accurate approach is available in this case.

5.2 Discrete Differential Quantities

Generally, properties at a vertex of a mesh can be defined as spatial averages on the continuous surface around this vertex. Meyer et al. point out that with a consistent definition, the spatial averages will converge to the pointwise definition of the quantity in the limit. To this end, an appropriate area \mathcal{A} has to be chosen, first. For a continuous function f defined on a surface \mathcal{S} , an average value f_{avg} over the area \mathcal{A} can be obtained as

$$f_{avg} = \frac{1}{\mathcal{A}} \int_{\mathcal{A}} f \, du \, dv . \quad (12)$$

If such an average is to be assigned to every vertex of a triangle mesh \mathcal{M} resulting from a discretisation of \mathcal{S} , an appropriate definition for the area \mathcal{A} is important. It is desirable that the choice of \mathcal{A} results in a

(disjoint) partition of \mathcal{M} . Therefore, \mathcal{A} must lie inside the 1-ring neighbourhood of \mathbf{x}_i and the borders $\partial\mathcal{A}$ must cross the edges in their midpoints. It remains to choose which point the borders should pass through in the interior of a triangle. As two possibilities the barycenter or the circumcenter of the triangle can be chosen (cf. Figure 6). Selecting the circumcenter leads to a partitioning of \mathcal{M} into *Voronoi* regions. Meyer et al. show that Voronoi regions are preferable

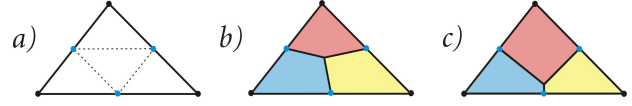


Figure 6: Alternative partitionings for a triangle. a) The border of any tiling must cross the mid-points of the edges. b) The barycenter is taken as interior point. c) Tessellation resulting from choosing the circumcenter as interior point.

over barycentric tilings, since they minimise the approximation error. Additionally, this partition is also useful for computing vertex masses needed for simulation (cf. [EKS03]). They derive a simple formula for the area \mathcal{A}_V of the Voronoi region for vertex \mathbf{x}_i as

$$\mathcal{A}_V = \frac{1}{8} \sum_{j \in N(i)} (\cot \alpha_{ij} + \cot \beta_{ij}) \|\mathbf{x}_i - \mathbf{x}_j\|^2 , \quad (13)$$

where α_{ij} , β_{ij} are the inscribed angles as shown in Figure 7. In case there is an obtuse triangle in the 1-ring neighbourhood, additional adjustments are necessary.

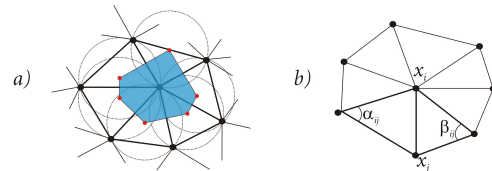


Figure 7: a) Voronoi region of a vertex. b) Angles for weighting edge $(\mathbf{x}_i - \mathbf{x}_j)$.

With an appropriate area for spatial averaging, the *mean curvature normal* operator \mathbf{H} can be established. It is related to mean curvature as

$$\mathbf{H}(\mathbf{x}) = 2\kappa_H(\mathbf{x})\mathbf{n}(\mathbf{x}) . \quad (14)$$

Using the induced triangle metric, the mean curvature normal operator can be expressed as

$$\int_{\mathcal{A}} \mathbf{H}(\mathbf{x}) dA = - \int_{\mathcal{A}} \left(\frac{\partial^2 \mathbf{x}}{\partial u^2} + \frac{\partial^2 \mathbf{x}}{\partial v^2} \right) du \, dv , \quad (15)$$

where u and v describe a *conformal* (i.e. angle preserving) space parametrisation. With the definition of the Voronoi area (13) and additional transformations (see [MDSB03]), equation (15) turns into

$$\mathbf{H}_v(\mathbf{x}) = \frac{1}{2\mathcal{A}_V} \sum_{j \in N(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(\mathbf{x}_i - \mathbf{x}_j) , \quad (16)$$

where the subscript v signifies that the operator is vertex-related. This expression provides a simple and accurate way to determine mean curvature on triangular meshes. The operator requires only information from a small local neighbourhood and can be evaluated efficiently. Note, however, that it is nonlinear in terms of the involved vertex positions. As well as for vertices the mean curvature can also be defined on the edges of the mesh (see [Pol02b]). This may be more convenient for implementation issues.

6 Discussion

As we have seen there are quite a lot of approaches to modelling bending energy for the simulation of thin flexible objects. Most of them use only a rough curvature approximation mainly derived from angular deformations while others use a more sophisticated approach based on discrete curvature measures.

With the expressions from the previous section it is possible to set up approximate bending energies on discrete surface representations. These can then be used to compute force vectors and Jacobians at the vertices, needed in a simulation context. A related implementation was demonstrated in [GH⁺03]. It must, however, be noticed that this model does not result from a discretisation of the thin plate (or shell) equations and is thus not as accurate as these methods known from engineering sciences. The corresponding energy expressions as well as first and second order derivatives are nonlinear in terms of the vertex positions. The computation of the derivatives bears additional complexity such that the overall costs will be higher than for standard approaches. Furthermore, using only mean curvature limits the application to isotropic materials. Computing the full curvature tensor would mean additional costs.

A question arising at this point is: what computational demands has an accurate approach based on physically more accurate models (e.g. the Kirchhoff-Love thin shell equations)? In order to reproduce the behaviour of thin flexible objects for a broad range of materials and independent of resolution, continuum mechanics are indispensable. Because of the shortcomings of finite difference schemes, the finite element approach will most likely be the method of choice for discretisation. Although these ingredients are commonly considered too costly for computer graphics, according to Hauth [Hau04] an efficient implementation leaves only a factor roughly between two and three when compared to standard approaches.

Basically, the thin plate equations impose certain smoothness requirements on the displacement field used in a finite element approach. More precisely, the displacement field has to be C^1 -continuous. As a direct consequence the linear finite element approach presented by Eitzmuß et al. cannot be simply extended to support the thin plate equations. The con-

struction of an element type which provides a C^1 -continuous displacement interpolation on its domain is not complicated. However, ensuring the continuity across elements is a major difficulty. Recently, a new paradigm for the finite element simulation of *thin shells* was introduced to the engineering community by Cirak et al. [COS00]. Using subdivision basis functions, they construct an element type with nodal displacements as only variables. The authors present a formulation of the thin shell equation which is linear in displacements. Hence, this is a promising way for the physically accurate modelling of bending in cloth simulation. In [TWS05] we therefore present an accurate and yet efficient approach to cloth simulation based on the work of Cirak et al.

ACKNOWLEDGEMENTS

Markus Wacker is supported by the Elitenförderung für Postdoktorandinnen und Postdoktoranden, Landesstiftung Baden-Württemberg. Bernhard Thomaszewski was supported by a grant from the Thomas-Gessmann-Stiftung.

References

- [BHW94] D. Breen, D. House, and M. Wozny. Predicting the drape of woven cloth using interacting particles. In *SIGGRAPH proceedings '94*, pages 365–372. ACM Press, 1994.
- [BMF03] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2003)*, pages 28–36. ACM Press, 2003.
- [BW98] D. Baraff and A. Witkin. Large steps in cloth simulation. In *SIGGRAPH proceedings '98*, pages 43–54. ACM Press, 1998.
- [CK02] K.-J. Choi and H.-S. Ko. Stable but responsive cloth. *ACM Transactions on Graphics (ACM SIGGRAPH 2002)*, 21(3):604–611, July 2002.
- [COS00] F. Cirak, M. Ortiz, and P. Schröder. Subdivision surfaces: A new paradigm for thin-shell finite-element analysis. *International Journal for Numerical Methods in Engineering*, 47, 2000.
- [CSM03] D. Cohen-Steiner and J.-M. Morvan. Restricted delaunay triangulations and normal cycle. In *Proc. of the 19th Annual ACM Symposium on Computational Geometry*, pages 237–246, 2003.

- [EB00] J. Eischen and R. Bigliani. Continuum versus particle representations. In D. House and D. Breen, editors, *Cloth Modeling and Animation*, pages 79–122. A. K. Peters, 2000.
- [EDC96] J. Eischen, S. Deng, and T. Clapp. Finite-element modeling and control of flexible fabric parts. *IEEE Computer Graphics and Applications*, 16(5):71–80, September 1996.
- [EKS03] O. Eitzmuß, M. Keckeisen, and W. Straßer. A Fast Finite Element Solution for Cloth Modelling. *Proc. of Pacific Graphics*, 2003.
- [EWS96] B. Eberhardt, A. Weber, and W. Straßer. A fast, flexible, particle-system model for cloth draping. *IEEE Computer Graphics and Applications*, 16(5):52–59, September 1996.
- [Fuj95] K. Fujiwara. Eigenvalues of laplacians on a closed riemannian manifold and its nets. In *Proc. of AMS*, pages 2585–2594, 1995.
- [GH⁺03] E. Grinspun, , A. Hirani, M. Desbrun, and P. Schröder. Discrete shells. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2003)*, pages 62–67. ACM Press, 2003.
- [Hau04] M. Hauth. *Visual Simulation of Deformable Models*. Phd thesis, Wilhelm-Schickard-Institut für Informatik, University of Tübingen, Germany, July 2004.
- [Kaw80] S. Kawabata. The standardization and analysis of hand evaluation. *The Textile Machinery Society of Japan*, 1980.
- [KCVS98] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. *SIGGRAPH '98 proceedings*, 20 April 1998.
- [Kee99] S. Keeler. The science of forming: A look at bending. *Metal Forming Magazine*, pages 27–29, October 1999.
- [MDSB03] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In H.-C. Hege and K. Polthier, editors, *Visualization and Mathematics III*, pages 35–57. Springer-Verlag, Heidelberg, 2003.
- [Opr97] J. Oprea. *Differential Geometry and its Applications*. Prentice-Hall, 1997.
- [Pol02a] K. Polthier. Computational aspects of discrete minimal surfaces. *Proc. of the Clay Summer School on Global Theory of Minimal Surfaces*, 2002.
- [Pol02b] K. Polthier. Polyhedral surfaces of constant mean curvature. *Habilitationschrift TU Berlin*, 2002.
- [PP93] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Math.*, 2(1), 1993.
- [SFR89] J. C. Simo, D. D. Fox, and M. S. Rifai. On a stress resultant geometrically exact shell model. part i: Formulation and optimal parametrization. In *Computational Methods in Applied Mechanics and Engineering*, volume 72, pages 267–302, 1989.
- [Tau95a] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. *Proc. of ICCV*, pages 902–907, 1995.
- [Tau95b] G. Taubin. A signal processing approach to fair surface design. *Computer Graphics Proc.*, pages 351 – 358, 1995.
- [TPBF87] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *SIGGRAPH proceedings '87*, pages 205–214. ACM Press, July 1987.
- [TWS05] B. Thomaszewski, M. Wacker, and W. Straßer. A consistent bending model for cloth simulation with corotational subdivision finite elements. Technical Report WSI-2005-19, Uni Tübingen, 2005.
- [VCMT95] P. Volino, M. Courchesne, and N. Magnenat-Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. In *SIGGRAPH proceedings '95*, pages 137–144. ACM Press, 1995.
- [WW98] H. Weimer and J. Warren. Subdivision schemes for thin plate splines. *Computer Graphics Forum*, 17(3):303–314, 1998. ISSN 1067-7055.
- [ZT00a] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method. Volume 1: The Basis*. Butterworth Heinemann, 5th edition, 2000.
- [ZT00b] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method. Volume 2: Solid Mechanics*. Butterworth Heinemann, 5th edition, 2000.

A Generalized Mandelbrot Set Based On Distance Ratio

Xizhe Zhang College of Computer Science and Technology, Jilin University No.2699, Qianjin street 130012, Changchun, Jilin, China zxzok@163.com	Tianyang Lv College of Computer Science and Technology, Harbin Engineering University raynor1979@163.com	Zhengxuan Wang College of Computer Science and Technology, Jilin University No.2699, Qianjin street 130012, Changchun, Jilin, China
-------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------

ABSTRACT

The iteration of complex function can generate beautiful fractal images. This paper presents a novel method based on the iteration of the distance ratio with two points, which generates a generalized Mandelbrot set according to distance ratio convergence times. This paper states the definition of distance ratio and its iteration. Then taking the complex function $f(z)=z^{\alpha}+c$ for example, it discusses the visual structure of generalized Mandelbrot with various exponent and comparing it with Mandelbrot set generated by escape time algorithm. When exponent $\alpha>1$, the outer border of DRM is same as Mandelbrot set, but has complex inner structure; when $\alpha<0$, the inner border of DRM is same as Mandelbrot set, DRM is the "outer" region and complement set of Mandelbrot set, the two sets cover the whole complex plane.

Keywords

Fractal; Distance Ratio; complex mapping; Mandelbrot set

1. INTRODUCTION

Since the introduction of fractal by Mandelbrot [Man77a], fractals have experienced considerable success in quantifying the complex structure exhibited by many natural patterns and have captured the imaginations of both scientists and artists [Spe03a]. Many researchers perform research on the generation method of fractal and draw many beautiful images [Fal90a].

Besides the escape time algorithm, many researchers propose other methods for generating fractal structure. [Hoo91a] proposes epsilon cross and star trails methods to render the interior structure of Mandelbrot set. [Cal96a] presents the algorithm of rendering pseudo-3D effects of fractal in complex plane, and on the basis of which presents two artistic rendering methods for Newton fractals [Cal99a]. [Roc00a] proposes a generalized Mandelbrot Set based on bicomplex number in 3D space and

discusses its properties. But seldom are these approaches used to generate fractal structure in convergent region of complex mapping.

This paper proposes a new method based on iteration distance ratio with two points, which renders fractal images by convergent times of distance ratio. Comparing with escape time algorithm, it consists of abundant details in the convergent region and more complex structure can be generated. It also can be used as a new method to render interior or exterior structure of escape time fractal. This paper states the visual characteristic of generalized Mandelbrot set generated by above method, and compares its image with escape time fractal.

The paper is organized as follows: Section 2 states the definition of distance ratio and rendering method; Section 3 generates a generalized Mandelbrot Set for $f(z)=z^{\alpha}+c$ and analyzes its visual characteristic; Finally, section 4 summarizes the paper.

2. ITERATION OF DISTANCE RATIO

This section states the definition of distance ratio and the method of generating fractal structure by using distance ratio convergent times.

Let $f: C \rightarrow C$ be an analytic function, for any $z_1 \neq z_2 \in C$, define the distance ratio L as follow:

$$L = \frac{|f(z_1) - f(z_2)|}{|z_1 - z_2|} \quad (1)$$

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.

*Uj qtv'Ego o wplecvkpu'rt qeggf kpi u'KDP': 2/: 8; 65/27/6
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.
Copyright UNION Agency – Science Press*

Perform the iteration on the initial points z_1 and z_2 , let $z_1^{(n+1)} = f(z_1^{(n)})$, $z_2^{(n+1)} = f(z_2^{(n)})$, $n = 0, 1, \dots$, and define the distance ratio after n -times iteration $L^{(n)}$ as:

$$L^{(n)} = \frac{|z_1^{(n+1)} - z_2^{(n+1)}|}{|z_1^{(n)} - z_2^{(n)}|} \quad (2)$$

If mapping f is contraction, the iteration of initial points will converge to the fixed-point. Thus, the distance ratio will also converge and we have following propositions:

Proposition 1: If mapping f has a fixed-point z^* and $A(z^*)$ is the attraction basin of z^* , for any z_1 and z_2 in $A(z^*)$, it follow $\lim_{k \rightarrow \infty} L^{(k)} = |f'(z^*)|$

Proposition 2: For a specified threshold ε , $\exists k^* \in N$, if $k > k^*$, $|L^{(k)} - |f'(z^*)|| < \varepsilon$

Proposition 1 states that distance ratio will converge by iterating; Proposition 2 states that distance ratio will converge within limited times for a certain threshold. According to these propositions, we can generate a new type fractal set. We call this kind of set the Distance Ratio Fractal, in short **DRF**.

If classify iterated points according to their distance ratio convergent times and render **DRF** in parameter space, we can give the definition of generalized Mandelbrot set based on Distance Ratio (**DRM**) as follows:

$$DRM = \{c \in C : \{\lim_{k \rightarrow \infty} L_c^{(k)} \rightarrow f'(z^*)\}\} \quad (3)$$

Supposing c is a complex number and ε is a small real number, we say that c is k -level if c satisfies the following condition:

$$\begin{aligned} |L_c^{(k-1)} - f'(z^*)| &> \varepsilon \\ |L_c^{(k)} - f'(z^*)| &\leq \varepsilon \end{aligned} \quad (4)$$

The set consisting of all the k -level points is termed **zone(k)**. The points in **zone(k)** approach the derivative after at least k iterations. The DRM can be divided into some **zone(k)**, like:

$$DRM = \bigcup_{k=1}^{\infty} \text{zone}(k) \quad (5)$$

Below is the rendering method of DRM. Its main idea is to travel the rendering region, so as to compute the convergent times of iterated points, then color these points according to their convergent times, and finally a fractal image is obtained.

Let rendering region D be a rectangle; top left corner be $(-1.5, 1.2)$; bottom right corner be $(1.5, -1.2)$; maximum iteration times $n=10000$; threshold $\varepsilon=0.0000001$. Steps of the algorithm are as follows:

1. Select point c in rendering region D ; construct mapping $f(z, c)$; let $z_1=(0,0)$ $z_2=(0.0001,0)$;
2. If $k < n$, perform the iteration $z_1^{(k)} = f(z_1^{(k-1)}, c)$, $z_2^{(k)} = f(z_2^{(k-1)}, c)$, else go to step 7;
3. Computing the distance ratio $L^{(k)} = \frac{|z_1^{(k+1)} - z_2^{(k+1)}|}{|z_1^{(k)} - z_2^{(k)}|}$;
4. If $|L^{(k)} - L^{(k-1)}| > \varepsilon$, $k=k+1$, go to step 3;
5. Color point c according to its iteration times;
6. Repeat steps 2-5, till all points in D are computed.

3. Image of DRM

Taking complex mapping $f(z)=z^\alpha+c$ as example, this section generates images of DRM and states conjectures on their visual characteristics.

A lot of research has been done about Mandelbrot set for $f(z)=z^\alpha+c$ generated by escape time algorithm. [Guj91a] generates many images of M-set, and states some conjectures on relationship between image structure and exponent value. [Wan00a] analyzes the image properties of negative exponent value and provides the strict mathematics prove. But previous work does not render images when $-1 < \alpha < 1$ (Escape time algorithm is not able to render fractal images when $-1 < \alpha < 1$). The following will analyze the images under different conditions, while $\alpha > 1$, $0 < \alpha < 1$, and $\alpha < 0$.

3.1 DRM for $\alpha > 1$

Fig.1 shows the DRM obtaining by setting $\alpha=2$. Its border is same as M-set generated by escape time algorithm, but there is very complex structure in its inner stable region. Its visual characteristic is listed as follows:

1. The border of DRM is exactly same as M-set generated by escape time algorithm.
2. There is layer structure in central quasi-circle and surrounding periodic buds.
3. Each layer is composed of some small areas like petals.
4. The inner petals are relatively in big size and small number; while outer petals are on the contrary, and the outer, the larger number, to infinity.

The layer structure is a typical characteristic of DRM. As points in same layer have same convergent times

k, each layer is a zone(k). The portion marked by a square in Fig.1 is zoomed in and shown in Fig.2. From it we can see clear layer and petal structure.

It is the convergence region of the mapping that is computed by the distance ratio iteration method. The border of DRM is the boundary between divergent

and convergent, so DRM and M-set have the same shape. Fig.3 and Fig.4 are generated when $\alpha=4$ and $\alpha=1.4$. Compared with images shown in [Guj91a], they have the same border. Therefore, distance ratio iteration method can be used as a new method for rendering inner structure of M-set when $\alpha>1$.

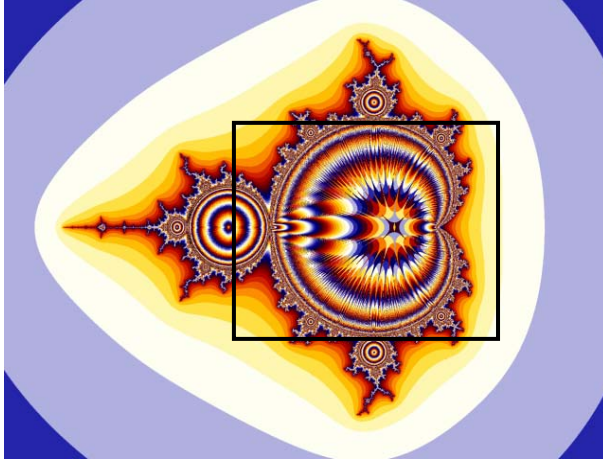


Figure 1. $\alpha=2$

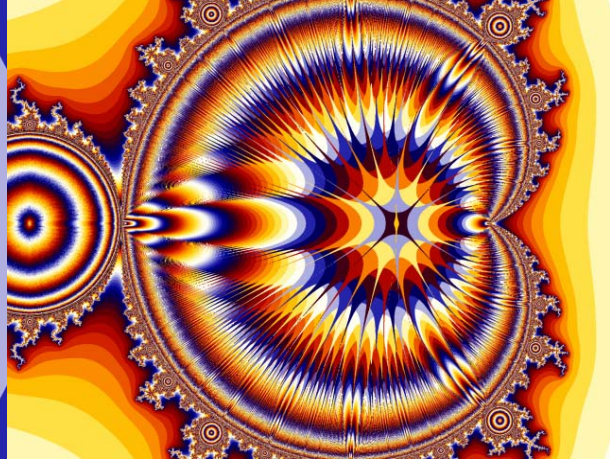


Figure 2. $\alpha=2$ zoomed

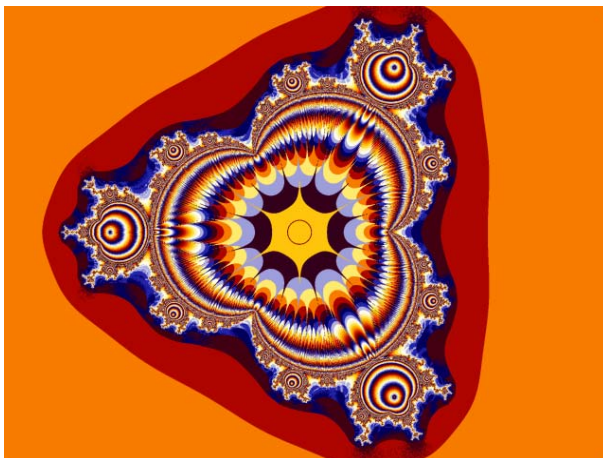


Figure 3. $\alpha=4$

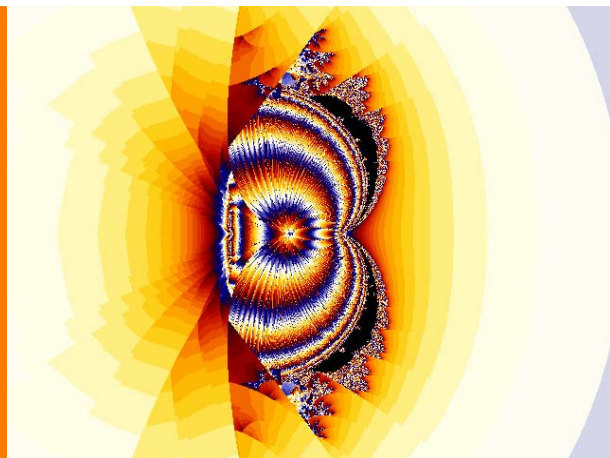


Figure 4. $\alpha=1.4$

3.2 DRM for $0<\alpha<1$

Fig.5 illustrates that M-set generated by escape time algorithm is a kind of simple geometry image, because the attractive region of the mapping changes when $-1<\alpha<1$. But DRM still has complex structure.

Fig.6 shows partly DRM obtained by setting $\alpha=0.1$. It looks like a flower with five petals. There are fewer petals at outer layers with the minimum of one petal. When it goes to inner layers, the number of petals

increases, so that a complex structure is formed. Fig.7-10 are gradually zoomed in images of a square area shown in Fig.6. From Fig.7 we can see the above-mentioned layer petal structure, in the center of which there is a spindle core like a "pistil", shown in Fig.8. When the right part of the spindle is zoomed in, we can see a pseudo-3D "spray hole" in Fig.10, which keeps spraying the petal structures out from the hole.

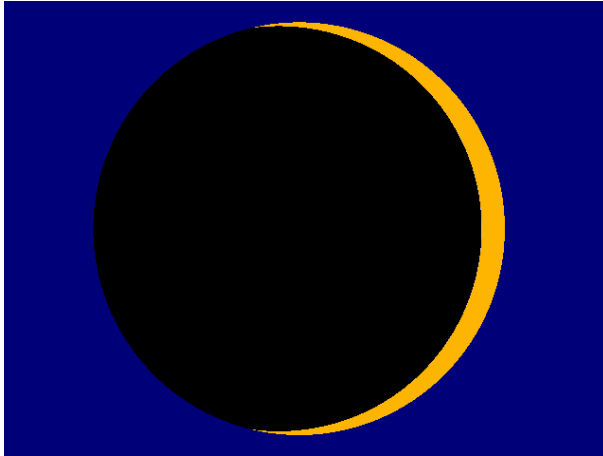


Figure 5. Mandelbrot set for $\alpha=0.1$

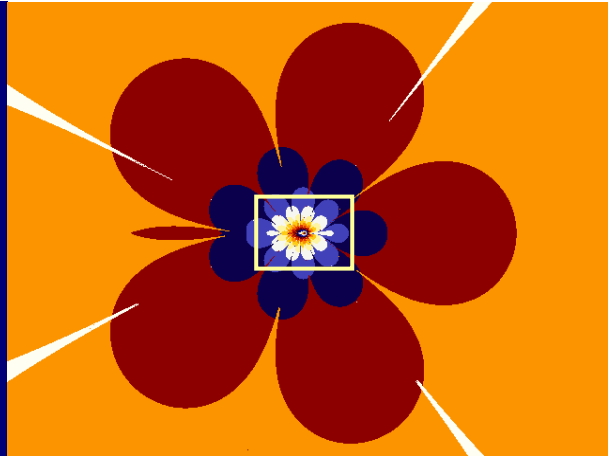


Figure 6. DRM for $\alpha=0.1$

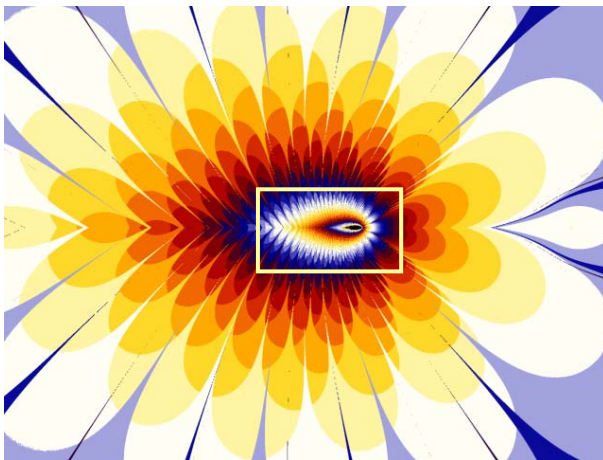


Figure 7. DRM for $\alpha=0.1$ zoomed

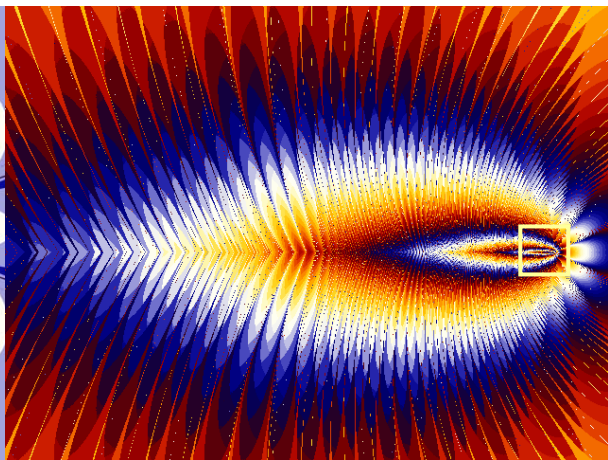


Figure 8. DRM for $\alpha=0.1$ zoomed

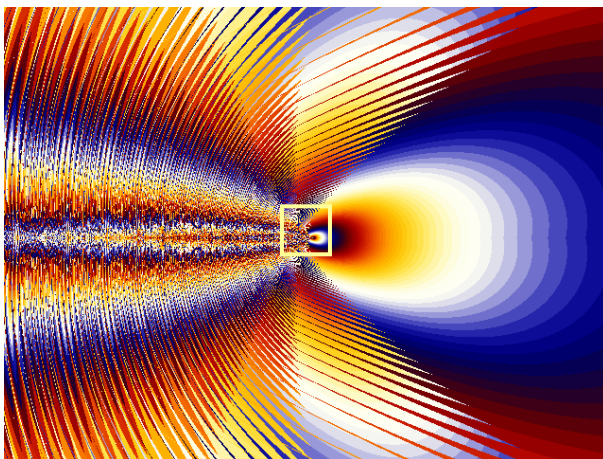


Figure 9. DRM for $\alpha=0.1$ zoomed

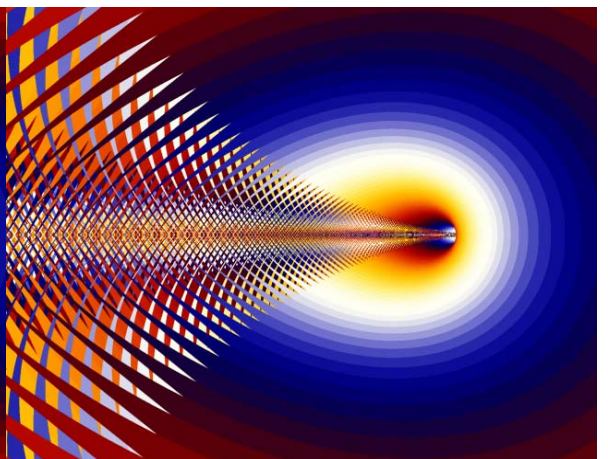


Figure 10. DRM for $\alpha=0.1$ zoomed

3.3 DRFM for $\alpha < 0$

There are different structures in DRM when $\alpha < 0$. Fig.13 shows the DRM by setting $\alpha = -2$. Fig.11 is $\alpha = -3$ and Fig.12 is $\alpha = -4$. Below characteristics are concluded from these images:

1. There is regular polygon structure with curve edge in the central of DRM.

2. Layer petal structure exists outside of the curve polygon, so is $0 < \alpha < 1$.

3. Central planet exists inside the curve polygon; the planet has some protuberances, which are tangent with curve edge of outside polygon in the middle points.

4. The planet is surrounded by many satellites. Each satellite is surrounded by further satellites, and so on.

Fig.13 is the DRM by setting $\alpha=-2$. Fig.14 is a partially zoomed in image of Fig.13; and so is Fig.15 and Fig.16. We can see the planet is surrounded by many satellites with buds near their boundaries and the satellites are surrounded by further satellites. The image has obvious self-similarity and complexity.

Fig.17 is the M-set generated by escape time algorithm for $\alpha=-2$ (M-set for other exponent see [Guj91a]). Compared with Fig.13, the border of Fig.17 is the same as curve polygon of DRM. If render the M-set and DRM in one image, we will obtain Fig.18. We can see they are a perfect match except the central planet is partly overlapped. If we increase the escape value of escape time algorithm, the size of central planet will reduce. Under this

condition, DRM is the complement set of M-set and they cover the whole complex plane. Therefore, distance ratio iteration method can be used to render “outer” of the M-set.

Based on above discussion, we have the following conjecture:

Conjecture: DRM for $\alpha<0$ is composed of two parts: outer curve polygon and inner constellation. The curve polygon has $|\alpha|+1$ edges and is covered by petal structures in outer layer; Inner constellation is composed of a central planet and surrounding satellites. The central planet has $|\alpha|+1$ protuberances and tangent with curve edge of outside polygon in the middle points. The DRM is the complement of M-set in complex plane.



Figure 11. DRM for $\alpha=-3$



Figure 12. DRM for $\alpha=-4$

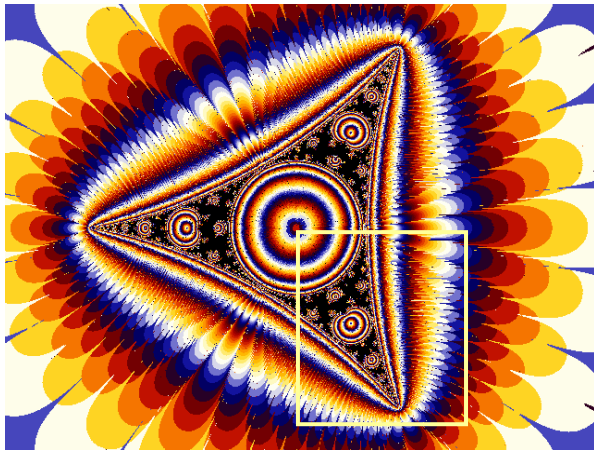


Figure 13. DRM for $\alpha=-2$

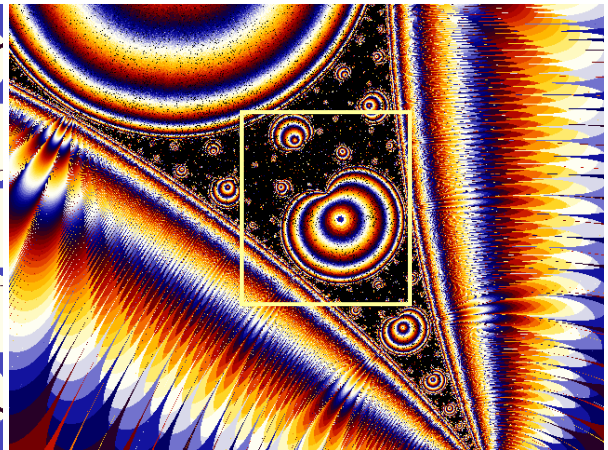


Figure 14. DRM for $\alpha=-2$ zoomed

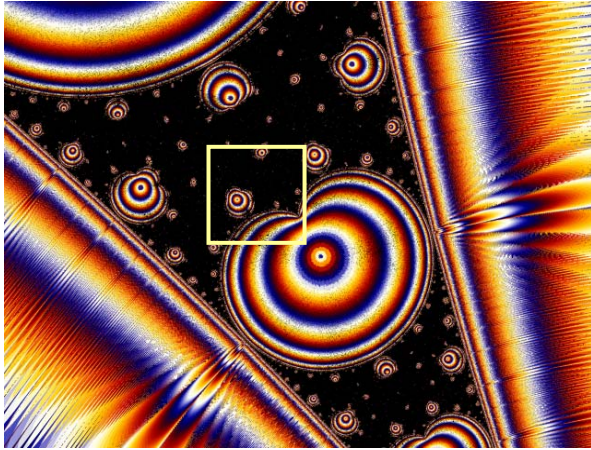


Figure 15. DRM for $\alpha=-2$ zoomed

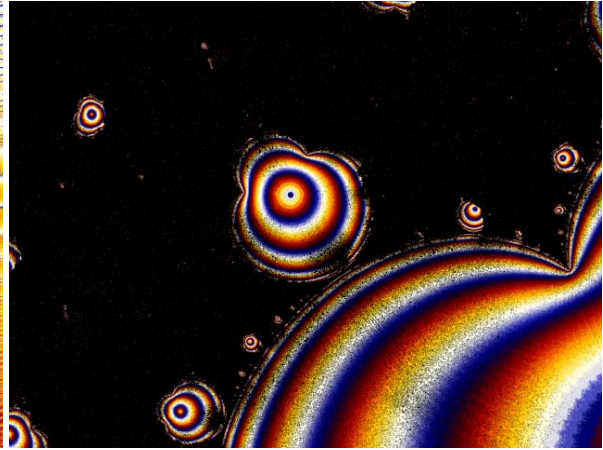


Figure 16. DRM for $\alpha=-2$ zoomed

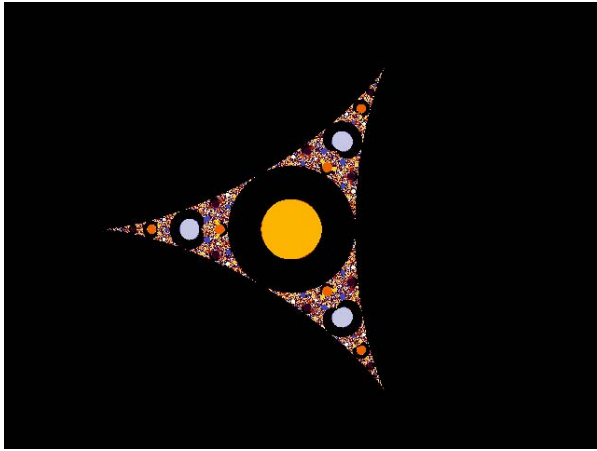


Figure 17. Mandelbrot set for $\alpha=-2$

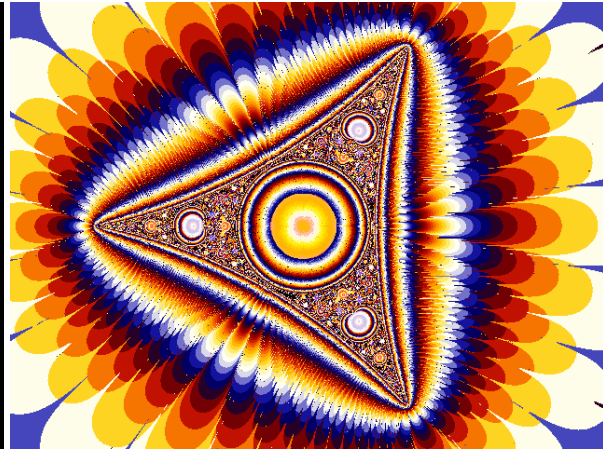


Figure 18. DRM and M-set for $\alpha=-2$

4. CONCLUSION

This paper presents a new method based on iteration of distance ratio, and renders a generalized Mandelbrot set by using it. From these images we can see there are both difference and relation between DRM and M-set. When $\alpha > 1$, DRM can generate the convergent region of mapping and be used as a method to render inner structure of M-set. When $0 < \alpha < 1$, distance ratio iteration method can render complex structure which escape time algorithm is not able to generate. When $\alpha < 0$, DRM is the “outer” region and complement set of M-set, the two sets cover the whole complex plane.

This paper discusses only DRM of $f(z)=z^\alpha+c$. The rendering method can be used for other analytic mappings, such as trigonometric function, $3x+1$ fractal [Pej04a]. Therefore, it is a new way to construct fractal images.

5. REFERENCES

[Cal96a] Carlson, P. W. Pseudo 3D rendering methods for fractals in the complex plane. Computers & Graphics 20(5), pp.751-758, 1996

- [Cal99a] Carlson, P. W. Two artistic orbit trap rendering methods for Newton M-set fractals." Computers & Graphics 23(6), pp.925-931, 1999
- [Fal90a] Falconer, K. Fractal Geometry – Mathematical Foundations and Applications, John Wiley & Sons. 1990.
- [Guj91a] Gujar, U. G., V. C. Bhavsar. Fractals from $z \leftarrow z^\alpha + c$ in the complex c-plane. Computers & Graphics 15(3), pp.441-449, 1991
- [Hoo91a] Hooper, K. J. A note on some internal structures of the Mandelbrot Set. Computers & Graphics 15(2), pp.295-297, 1991
- [Man77a] Mandelbrot, B. B. The fractal geometry of nature. New York, Freeman. 1977
- [Pej04a] Pe, J. L. The $3x+1$ fractal. Computers & Graphics 28(3): pp.431-435, 2004
- [Roc00a] Rochon, D. A Generalized Mandelbrot Set for Bicomplex Numbers. Fractals 8(4), pp.355-368, 2000
- [Spe03a] Spehar, B., C. W. G. Clifford, et al. Universal aesthetic of fractals. Computers & Graphics 27(5), pp.813-820, 2003
- [Wan00a] Wang XY, L.X., Zhu WY, Gu SS, Analysis of c-plane fractal images from for ($\alpha < 0$). Fractals, 8(3): pp. 307-314, 2000