

**The 13-th International Conference in Central Europe on Computer  
Graphics, Visualization and Computer Vision 2005**

in co-operation with

**EUROGRAPHICS**

**W S C G ' 2005**

**FULL PAPERS "**

University of West Bohemia  
Plzen  
Czech Republic

*Honowrary Chair*

M.L.V. Pitteway, Brunel University, Uxbridge, United Kingdom

*Co-Chairs*

Tosiyasu L. Kunii: Kanazawa Institute of Technology, Tokyo, Japan  
Vaclav Skala, Univ. of West Bohemia, Plzen, Czech Republic

*Edited by*

Vaclav Skala

***WSCG'2005 Full Papers Conference Proceedings***

Editor-in-Chief: Vaclav Skala  
University of West Bohemia, Univerzitni 8, Box 314  
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 80-903100-7-9*

# WSCG 2005

## International Programme Committee

Alexa, Marc (Germany)  
Bajaj, Chandrajit (United States)  
Bartz, Dirk (Germany)  
Bekaert, Philippe (Belgium)  
Benes, Bedrich (Mexico)  
Bengtsson, Ewert (Sweden)  
Bouatouch, Kadi (France)  
Brodie, Ken (United Kingdom)  
Brunet, Pere (Spain)  
Brunnet, Guido (Germany)  
Clapworthy, Gordon (United Kingdom)  
Coquillart, Sabine (France)  
Debelov, Victor (Russia)  
Deussen, Oliver (Germany)  
du Buf, Hans (Portugal)  
Ertl, Thomas (Germany)  
Ferguson, Stuart (United Kingdom)  
Floriani, Leila De (Italy)  
Flusser, Jan (Czech Republic)  
Goebel, Martin (Germany)  
Haber, Jörg (Germany)  
Harris, Mark (United Kingdom)  
Hauser, Helwig (Austria)  
Hege, Hans-Christian (Germany)  
Chen, Min (United Kingdom)  
Chrysanthou, Yiorgos (Cyprus)  
Jansen, Frederik,W. (The Netherlands)  
Jorge, Joaquim (Portugal)  
Kakadiaris, Ioannis (United States)  
Kalra, Prem (India)  
Kjell Dahl, Lars (Sweden)  
Klein, Reinhard (Germany)  
Klosowski, James T. (United States)  
Kobbelt, Leif (Germany)  
Kruijff, Ernst (Germany)  
Magnor, Marcus (Germany)  
Margala, Martin (United States)  
Moccozet, Laurent (Switzerland)  
Mudur, Sudhir,P. (Canada)  
Mueller, Klaus (United States)  
Muller, Heinrich (Germany)  
Myszkowski, Karol (Germany)  
O'Sullivan, Carol (Ireland)  
Pasko, Alexander (Japan)  
Peroche, Bernard (France)  
Post, Frits H. (Netherlands)  
Puech, Claude (France)  
Puppo, Enrico (Italy)  
Purgathofer, Werner (Austria)  
Rauterberg, Matthias (Netherlands)  
Rheingans, Penny (United States)  
Rokita, Przemyslaw (Poland)  
Rossignac, Jarek (United States)  
Rudomin, Isaac (Mexico)  
Sbert, Mateu (Spain)  
Shamir, Ariel (Israel)  
Schaller, Nan,C. (United States)  
Schneider, Bengt-Olaf (United States)  
Schumann, Heidrun (Germany)  
Skala, Vaclav (Czech Republic)  
Slusallek, Philipp (Germany)  
Sochor, Jiri (Czech Republic)  
Stuerzlinger, Wolfgang (Canada)  
Sumanta, Pattanaik (United States)  
Szirmay-Kalos, Laszlo (Hungary)  
Taubin, Gabriel (United States)  
Teschner, Matthias (Switzerland)  
Theoharis, Theoharis (Greece)  
Trahanias, Panos (Greece)  
Velho, Luiz (Brazil)  
Veltkamp, Remco (Netherlands)  
Weiskopf, Daniel (Germany)  
Westermann, Ruediger (Germany)  
Wuethrich, Charles Albert (Germany)  
Zara, Jiri (Czech Republic)  
Zemcik, Pavel (Czech Republic)



## WSCG 2005 Board of Reviewers

Adzhiev, V. (United Kingdom)  
Alexa, M. (Germany)  
Ammann, C. (Switzerland)  
Anan, H. (United States)  
Andreadis, I. (Greece)  
Artusi, A. (Italy)  
Aspragathos, N. (Greece)  
Avenueau, L. (France)  
Bajaj, C. (United States)  
Bartz, D. (Germany)  
Bekaert, P. (Belgium)  
Benes, B. (Mexico)  
Bengtsson, E. (Sweden)  
Bieri, H. (Switzerland)  
Bilbao, J. (Spain)  
Bischoff, S. (Germany)  
Bottino, A. (Italy)  
Bouatouch, K. (France)  
Bourdin, J. (France)  
Brodie, K. (United Kingdom)  
Brunet, P. (Spain)  
Brunnet, G. (Germany)  
Buehler, K. (Austria)  
Callieri, M. (Italy)  
Clapworthy, G. (United Kingdom)  
Coleman, S. (United Kingdom)  
Coombe, G. (USA)  
Coquillart, S. (France)  
Daniel, M. (France)  
de Aquiar, E. (Germany)  
De Decker, B. (Belgium)  
de Geus, K. (Brazil)  
Debelov, V. (Russia)  
del Rio, A. (Germany)  
Deussen, O. (Germany)  
Diehl, S. (Germany)  
Dingliana, J. (Ireland)  
Dmitriev, K. (Germany)  
Doleisch, H. (Austria)  
Dong, F. (United Kingdom)  
Drakopoulos, V. (Greece)  
du Buf, H. (Portugal)  
Duce, D. (United Kingdom)  
Durupina, F. (Turkey)  
Egges, A. (Switzerland)  
Eibl, M. (Germany)  
Erbacher, R. (United States)  
Ertl, T. (Germany)  
FariaLopes, P. (Portugal)  
Faudot, D. (France)  
Feito, F. (Spain)  
Ferguson, S. (United Kingdom)  
Fernandes, A. (Portugal)  
Fischer, J. (Germany)  
Flaquer, J. (Spain)  
Floriani, L. (Italy)  
Flusser, J. (Czech Republic)  
Gagalowicz, A. (France)  
Galo, M. (Brazil)  
Geraud, T. (France)  
Giannini, F. (Italy)  
Gudukbay, U. (Turkey)  
Gutierrez, D. (Spain)  
Haber, J. (Germany)  
Hadwiger, M. (Austria)  
Haro, A. (United States)  
Harris, M. (United Kingdom)  
Hast, A. (Sweden)  
Hauser, H. (Austria)  
Havran, V. (Germany)  
Hege, H. (Germany)  
Hladuvka, J. (Slovakia)  
Horain, P. (France)  
Hornung, A. (Germany)  
Chen, M. (United Kingdom)  
Chin, S. (Korea)  
Chover, M. (Spain)  
Chrysanthou, Y. (Cyprus)  
Iwanowski, M. (Poland)  
Jaillet, F. (France)  
Jansen, F. (Netherlands)  
Jeschke, S. (Germany)  
JoanArinyo, R. (Spain)  
Kalra, P. (India)  
Kjelldahl, K. (Sweden)  
Klosowski, J. (United States)  
Kobbelt, L. (Germany)  
Kolcun, A. (Czech Republic)  
Koutek, M. (Netherlands)  
Krolupper, F. (Czech Republic)  
Kruijff, E. (Germany)  
Larsen, B. (Denmark)

Leopoldseeder, S. (Austria)  
Lewis, J. (United States)  
Lintu, A. (Germany)  
Loizides, A. (Cyprus)  
Loizides, A. (Cyprus)  
Magnor, M. (Germany)  
Maierhofer, S. (Austria)  
Mandl, T. (Germany)  
Mantler, S. (Austria)  
Margala, M. (United States)  
Marinov, M. (Germany)  
Maughan, C. (USA)  
McAllister, D. (USA)  
McMenemy, K. (United Kingdom)  
Mertens, T. (Belgium)  
Moccozet, L. (Switzerland)  
Mokhtari, M. (Canada)  
Molledo, L. (Italy)  
Montrucchio, B. (Italy)  
Moreton, H. (USA)  
Mudur, S. (Canada)  
Mueller, K. (United States)  
Muller, H. (Germany)  
Myszkowski, K. (Germany)  
Neubauer, A. (Austria)  
Nielsen, F. (Japan)  
O'Sullivan, C. (Ireland)  
Ozguc, B. (Turkey)  
Pan, Z. (China)  
Pandzic, I. (Croatia)  
Pasko, A. (Japan)  
Pedrini, H. (Brazil)  
Perez, M. (Spain)  
Peroche, B. (France)  
Plemenos, D. (France)  
Post, F. (Netherlands)  
Prakash, E. (Singapore)  
Pratikakis, I. (Greece)  
Prikryl, J. (Czech Republic)  
Puppo, E. (Italy)  
Purgathofer, W. (Austria)  
Rauterberg, M. (Netherlands)  
Ravyse, I. (Belgium)  
Renaud, C. (France)  
Revelles, J. (Spain)  
Rheingans, P. (United States)  
Rodrigues, M. (United Kingdom)  
Rokita, P. (Poland)  
Rossignac, J. (United States)  
Rudomin, I. (Mexico)  
Sahli, H. (Belgium)  
Sainz, M. (USA)  
Sbert, M. (Spain)  
Segura, R. (Spain)  
Shamir, A. (Israel)  
Schaller, N. (United States)  
Schneider, B. (United States)  
Scholz, V. (Germany)  
Schumann, H. (Germany)  
Sijbers, J. (Belgium)  
Sips, M. (Germany)  
Sirakov, N. (United States)  
Sitte, R. (Australia)  
Slusallek, P. (Germany)  
Snoeyink, J. (United States)  
Sochor, J. (Czech Republic)  
Sorel, M. (Czech Republic)  
Sroubek, F. (Czech Republic)  
Stuerzlinger, W. (Canada)  
Stylianou, G. (Cyprus)  
Suarez Rivero, J. (Spain)  
Sumanta, P. (United States)  
Szekely, G. (Switzerland)  
Szirmay-Kalos, L. (Hungary)  
Tang, W. (United Kingdom)  
Taubin, G. (United States)  
Teschner, M. (Germany)  
Theobald, C. (Germany)  
Theoharis, T. (Greece)  
Theußl, T. (Austria)  
Tobler, R. (Austria)  
Torres, J. (Spain)  
Trahanias, P. (Greece)  
Traxler, A. (Austria)  
Van Laerhoven, T. (Belgium)  
Velho, L. (Brazil)  
Veltkamp, R. (Netherlands)  
Vergeest, J. (Netherlands)  
Vuorimaa, P. (Finland)  
Weiskopf, D. (Germany)  
Weiss, G. (Germany)  
Westermann, R. (Germany)  
Wu, S. (Brazil)  
Wuethrich, C. (Germany)  
Yilmaz, T. (Turkey)  
Zach, C. (Austria)  
Zachmann, G. (Germany)  
Zara, J. (Czech Republic)  
Zemcik, P. (Czech Republic)  
Zhu, Y. (United States)  
Zitova, B. (Czech Republic)

# WSCG 2005

## Contents

### Honourary Chair

Pitteway, M.L.W.: Welcome to the 13<sup>th</sup> International Conference in Central Europe on Computer graphics, Visualization and Computer Vision 2005! (U.K.)

### Invited speakers

Ferguson, S.: Adapting Computer Game technology to Build a Surgical Simulator (U.K.)

Klosowski, J.T.: Scalable Visualization using Commodity Clusters: Challenges and Solutions (USA)

Hubo, E., Bekaert, P.: A Data Distribution Strategy for Parallel Point-Based Rendering 1

### Regular papers

|   |     |
|---|-----|
| Lario, R., Pajarola, R., Tirado, F.: Cached Geometry Manager for View-dependent LOD Rendering   | 9   |
| Herout, A., Zemcik, P.: Hardware Pipeline for Rendering Clouds of Circular Points   | 17  |
| Kanodia, R.L., Linsen, L., Hamann, B.: Multiple Transparent Material-enriched Isosurfaces   | 23  |
| Knuth, M., Fuhrmann, A.: Self-Shadowing of Dynamic Scenes with Environment Maps using the GPU   | 31  |
| Loviscach, J.: Paving Procedural Roads with Pixel Shaders   | 39  |
| Bleser, G., Pastarmov, Y., Stricker, D.: Real-time 3D Camera Tracking for Industrial Augmented Reality Applications (Germany)           | 47  |
| Guizatdinova, I., Surakka, V.: Detection of Facial Landmarks from Neutral, Happy, and Disgust Facial Images (Finland)                   | 55  |
| Fournier, G., Péroche, B.: Multi-mesh Caching and Hardware Sampling for Progressive and Interactive Rendering (France)                  | 63  |
| Geimer, M., Abert, O.: Interactive Ray Tracing of Trimmed Bicubic Bézier Surfaces without Triangulation (Germany)                       | 71  |
| Lintu, A., Haber, J., Magnor, M.: Realistic Solar Disc Rendering (Germany)  | 79  |
| Mora, F., Aveneau, L., Meriaux, M.: Coherent and Exact Polygon-to-Polygon Visibility (France)   | 87  |
| Murotani, K., Sugihara, K.: New Spectral Decomposition for 3D Polygonal Meshes and its Application for Watermarking (Japan)             | 95  |
| Reitinger, B., Bornik, A., Beichel, R.: Constructing Smooth Non-Manifold Meshes of Multi-Labeled Volumetric Datasets (Austria)          | 227 |
| Beets, K., Claes, J., Van Reeth, F.: A Subdivision Scheme to Model Surfaces with Spherelike Features (Belgium)                          | 103 |
| Jin, C., Fevens, T., Li, S., Mudur, S.P.: Feature Preserving Volumetric Data Simplification for Application in Medical Imaging (Canada) | 235 |
| Levet, F., Hadim, J., Reuter, P., Schlick, Ch.: Anisotropic Sampling for Differential Point Rendering of Implicit Surfaces (France)     | 109 |
| Sugisaki, E., Yu, Y., Anjyo, K., Morishima, S.: Simulation-Based Cartoon Hair Animation (Japan)   | 117 |
| Ge, Ch., Chen, Y., Yang, Ch., Yin, B., Gao, W.: Motion Retargeting for the Hand Gesture (China)   | 123 |
| Fiorentino, M., Uva, A.E., Monno, G.: The SenStylus: A Novel Rumble-Feedback Pen Device for CAD Application in Virtual Reality (Italy)  | 131 |

|  |     |
|--|-----|
| Froehlich,B., Blach,R., Stefani,O., Hochstrate,J., Hoffmann,J., Klueger,K., Bues,M.: Implementing Multi-Viewer Stereo Displays (Germany)   | 139 |
| Miyazaki, T, Kaneko, T, and S. Kuriyama: Virtual Destruction of a 3D Object with a Stick (Japan)   | 147 |
| Somol,P., Haindl,M.: Novel Path Search Algorithm for Image Stitching and Advanced Texture Tiling (Czech Republic)  | 155 |
| Kartasheva,E., Adzhiev,V., Comninos,P., Pasko,A., Schmitt,B.: Construction of Implicit Complexes: A Case Study (United Kingdom)  | 219 |
| Klein,J., Zachmann,G.: Interpolation Search for Point Cloud Intersection (Germany)   | 163 |
| McDonald,J., Wolfe,R., Alkoby,K., Brzezinski,J., Carter,R., Davidson,M.J., Furst,J., Hinkle,D., Kroll,B., Lancaster,G., Smallwood,L., Toro,J., Ougouag,N., Schnepf,J.: Achieving Consistency in a Combined IK/FK Interface for a Seven Degree-of-Freedom Kinematic Chain (United States) | 171 |
| Somchaipeng,K., Erleben,K., Sporning,J.: A Multi-Scale Singularity Bounding Volume Hierarchy (Denmark)   | 179 |
| Frau,S., Roberts, J.C., Boukhelifa,N.: Dynamic Coordinated Email Visualization (United Kingdom)  | 187 |
| Semwal,S.K., Chandrashekhar,K.: Cellular Automata for 3D Morphing of Volume Data (United States)   | 195 |
| Schulze-Wollgast,P., Tominski,C., Schumann,H.: Enhancing Visual Exploration by Appropriate Color Coding (Germany)  | 203 |
| Wan,T.R., Chen,H., Earnshaw,R.A.: A Motion Constraint Dynamic Path Planning Algorithm for Multi-Agent Simulations (United Kingdom)   | 211 |

# A Data Distribution Strategy for Parallel Point-Based Rendering

Erik Hubo  
Expertise Center for Digital Media  
Limburgs Universitair Centrum  
Universitaire Campus  
B-3590 Diepenbeek Belgium  
erik.hubo@luc.ac.be

Philippe Bekaert  
Expertise Center for Digital Media  
Limburgs Universitair Centrum  
Universitaire Campus  
B-3590 Diepenbeek Belgium  
philippe.bekaert@luc.ac.be

## ABSTRACT

During the last couple of years, point sets have emerged as a new standard for the representation of largely detailed models. This is partly due to the fact that range scanning devices are becoming a fast and economical way to capture dense point clouds. Traditional rendering systems are impractical when a single polygonal primitive contributes less than a pixel during rendering. We present a data distribution strategy for parallel point-based rendering, using a cluster of PCs as target platform. We describe a data-structure and a system architecture, which allows for decoupling the point-data from the computational work. This strategy enables both a balanced workload as well as no full data replication on each node. We exploit frame-to-frame coherence to make our system scalable. The system renders high-resolution images from high complex data sets at interactive frame rates. To our knowledge parallel point-based rendering has not been investigated in the past. Our results indicate the feasibility of sort-first parallelization applied to point-based rendering.

## Keywords

Cluster Computing, Parallel Rendering, Point-Based Rendering

## 1. INTRODUCTION

A recent trend in computer graphics is the shift towards sample-based rendering. Today's range sensing devices are capable of producing highly detailed and massive point clouds, which do not fit in the main memory of a single commodity PC. Point-based rendering can be more efficient than traditional rendering for these complex models if triangles occupy a small screen region. Processing many small triangles leads to bandwidth bottlenecks and excessive floating point and rasterization requirements [DeeM93]. Because of the absence of topology and relative positions, point-clouds are well suited for spatial subdivision and distribution between different PC's. One way of visualizing these enormous data sets is the use of expensive multiprocessor graphics servers with a huge main memory. A reasonable less expensive alternative of

these dedicated graphics machines is a cluster of commodity PC's, linked by a high bandwidth network. The main challenge is to develop efficient parallel rendering algorithms that scale well within the processing, storage and communication characteristics of a PC cluster. Using this system architecture has many advantages: price-performance ratio, modularity, flexibility, storage capacity and scalability. Processing power, storage and memory capacity grow linearly with the number of PCs. A drawback to the traditional, tightly-integrated parallel computers is the fact that there is no fast access to a shared virtual memory space, and that the bandwidth and latencies of inter-processor communication are significantly higher. The challenge is to develop algorithms that evenly divide workload among PCs, do not introduce extra work due to parallelization and scale well as more PCs are added to the system.

In this paper we propose a data and work distribution scheme for parallel point-based rendering on a PC cluster.

This paper is organized as follows: first we discuss previous work in section 2. Next, we give a short system overview in section 3. In section 4 we present our implementation, data structures and system architecture. Finally, sections 5 and 6 discuss our results and conclusions.

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 2005 Conference proceedings ISBN 80-903100-7-9  
WSCG'2005, January 31-February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency – Science Press*

## 2. PREVIOUS WORK

### Point-Based Rendering

During the last couple of years, there has been an increased interest of the computer graphics community in point based rendering techniques. Point-based rendering dates back as far as 1985, the year in which Levoy and Whitted [LeMa85] proposed the use of points to model and render 3D continuous surfaces. In 1989, Westover [WeLe89], introduced splatting for interactive volume rendering. Splatting algorithms handle volume data as a set of particles that absorb and emit light. Westover's basic splatting algorithm suffers from considerable artifacts due to inaccurate visibility determination when composing the splats from back to front. More recently, image-based rendering [McLe95] has become popular because the rendering time is proportional to the number of pixels (points) warped from the source to the output images. This contrasts with the scene-dependant time complexity for more traditional rendering techniques. Later on, the Lightfield [LeMa96] and Lumigraph [GoSt96] techniques were developed. These algorithms describe the radiance of a scene as a function of position and direction in a four-dimensional space, however, at the price of storage overhead.

One of the first point based rendering systems was QSplat [RuSz00]. In QSplat, a multi-resolution hierarchy, based on bounding spheres, is employed for the representation and progressive visualization of large models. The system is able to handle large meshes at constant frame rate. Pfister and Zwicker introduced surfels [PfHa00], short for surface elements. Surfels are a powerful paradigm for efficiently rendering complex geometric objects at interactive frame rates. Surfels can handle complex shapes; introduce low rendering cost and high image quality. Three orthogonal LDI's [ShJo98] are used to sample objects and image space filters are employed to achieve hole-free rendering. Later Zwicker et al. presented a framework for direct volume rendering [ZwMA01] using a splatting approach based on elliptical Gaussian kernels, superior to the footprints of Westover [WeLe89]. This results in high-quality anti-aliased rendering without excessive blurring. Botsch et al. proved that a pure software implementation could render up to 14 million Phong shaded samples per second by using a quantization of splat shapes [BoMa02]. However the models used to achieve these rendering times are not complex in terms of memory requirement. Their quantized hierarchical data representation is very compact with a memory consumption of less than 2 bits per point position. Software-based point-based rendering algorithms have proven to be superior to polygon-based rendering algorithms for highly complex

scenes. High quality results can be achieved but their rendering speed is limited. Recent algorithms use graphical hardware to overcome this problem. This idea was first introduced in [RuSz00]. In [CoLi02] the authors avoid using the z-buffer by sorting an octree from back to front each frame similar to McMillan [McLe95]. In [BoMa03] the authors provide high quality as well as efficient rendering based on a two-pass splatting technique with Gaussian filtering. Finally, in their most recent publication the authors propose to base the lighting of a splat on a linearly varying normal field associated with it, resulting in a visually high quality image [BoMa04]. Dachsbacher et al. [DaCa03] present a hierarchical LOD structure that is suitable for GPU implementation. They can process 50M low quality points per second

A main drawback of all the GPU algorithms is that they only perform well on rather simple models with a low screen resolution. This is due to the fact that, although extremely fast, a GPU's on-board memory is currently rather limited in terms of data storage. To overcome this limitation we use a PC cluster to speed up the rendering. Since PC clusters have a scalable memory capacity, they are well suited for the interactive rendering of high-resolution images of complex models.

A short overview of parallel rendering is presented next.

### Parallel rendering

Parallel rendering systems have long been used for ray tracing [WaIn01], radiosity and global illumination [FuTh96, ZaDa95, ReEr98]. These systems can often be classified by the stage in the graphics pipeline in which the primitives are partitioned: sort-first, sort-middle or sort-last [MoSt94]. In sort-first systems, screen space is partitioned in non-overlapping 2D tiles, each of which is rendered independently. The final image is obtained by composing all 2D tiles. The main advantage of this method is the low communication cost. The efficiency of sort-first algorithms is limited by redundant rendering due to overlapping tiles [SaRu01]. In general, since the overlap factors grow with increasing numbers of processors, the scalability of sort first systems is limited [MuCa95]. Sort-middle, the most straightforward approach, is commonly used in traditional systems. Primitives are redistributed in the middle of the rendering pipeline, between geometry processing and rasterization. This approach is not well suited for a cluster of PC's due to its high communication requirements. Finally sort-last methods defer sorting until the end of the rendering pipeline. The main advantage of sort-last is its scalability [MoSt94].

In the last few years, there has been a growing interest in PC clusters for interactive rendering tasks. Humphreys and Hanrahan presented a sort-first system designed for 3D graphics called WireGL [HuGr99, HuGr00]. WireGL was used to achieve scalable display size with minimal impact to the application's performance. Unlike sort-middle, sort-first can use retained-mode scene graphs to avoid most data transfers for graphics primitives between processors [MuCa95]. In [SaRu00] a hybrid sort-first sort-last approach for parallel polygon rendering is presented. A specific algorithm for dynamic, view-dependent and coordinated partitioning is used of both the 3D model and the 2D image, which has positive results in terms of both performance and scalability.

Continual growth in typical dataset size and network bandwidth has made stream-based analysis a hot topic for remotely stored 3D models [RuSz01]. Streams are appropriate computational primitives, because large amounts of data arrive continuously, and it is impractical or unnecessary to retain the entire dataset. Chromium [HuGr02] is another a stream-processing framework based on WireGL. Its stream filters can be arranged to create sort-first and sort-last parallel graphics architectures.

Since we are interested in high-resolution images, we prefer a PC cluster method to the recently popular GPU methods because of its scalable memory capacity. High-resolution images require complex models with many point samples, which cannot be accommodated by the memory of the graphical hardware. We believe our sort-first parallelization is scalable because the overlap factor is negligible in point-based rendering. To the authors' knowledge parallel point-based rendering has not been investigated in the past.

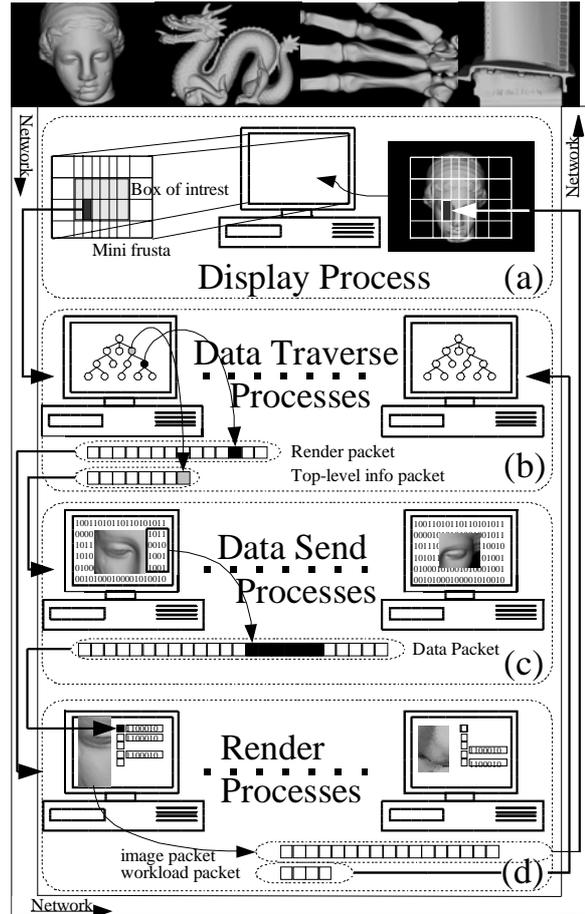
### 3. SYSTEM OVERVIEW

Our system operates in two stages:

**Preprocessing Stage:** The first stage serves as an offline preprocessing stage and is only performed once per 3D model. Details are provided in section 4.1. The input for the first stage is a point-cloud. The system creates a multi-resolution hierarchical spatial subdivision structure, optimized for fast data traversal.

**Rendering stage:** The second stage is the render stage. We use four types of processes in our system architecture to decouple the data from the computation in order to achieve an optimal load balance. We briefly describe these processes of the rendering pipeline below (Details are provided in section 4.2 to 4.5):

*Display process:* This process executes the first and last stage of the rendering pipeline. In the first stage,



**Figure 1: System overview of the rendering pipeline: (a) Display process: Frustum subdivision according to a box of interest and display. (b) Data traverse processes: traversing data and gathering render information. (c) Data send processes: sending point-data. (d) Render processes: Caching and rendering the incoming data and sending the rendered images back to the display node.**

the display process divides the view frustum into a set of smaller mini view frusta, according to a box of interest, and sends them together with camera data to the data traverse processes. After computation in the final stage, the display process receives the images corresponding to these mini frusta and loads them into the framebuffer for display.(see figure 1 (a)).

*Data traverse process:* A data traverse process requests a mini frustum from the display process. While traversing the octree data structure, the data traverse process clips the octree cells against the mini frustum, and decides which octree cells are suitable for rendering. For each mini frustum the data traverse process maintains, together with the list of useful octree cells, a list of used top-level octree cells. These are hierarchically higher octree cells (see figure 2). Depending on the workload and the

available data on the render nodes (see section 4.5), the data traverse process can correctly determine the render node the data should be sent to. (see figure 1(b)).

*Data send process:* The data traverse processes inform the data send processes what point-data should be sent to which render node (see section 4.5). (see figure 1(c)).

*Render process:* Render processes receive packets from data send processes (data packets) and from data traverse processes (render packets). Data packets contain point-data of a top-level octree cell. Render packets contain pointers to the data that has to be rendered, camera and mini frustum data. Received data packets are temporarily stored on the render node (see section 4.5). A render node creates one image per received render packet, assuming all necessary data packets are available. This image is sent back to the display process. (see figure 1 (d)).

## 4. IMPLEMENTATION

In this section we describe the implementation of our distributed point-based-rendering system in detail, and comment on the applied data structures and algorithms.

### Preprocessing

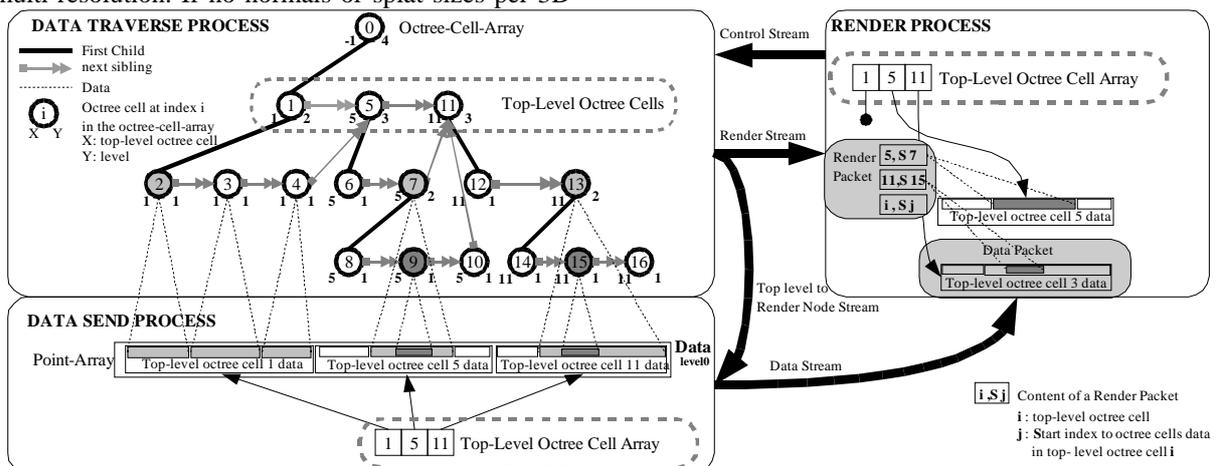
The preprocessing stage is the first stage in the algorithm and has to be executed only once for any given input point cloud. Like other point-based rendering algorithms [RuSz00, BoMa02], an octree based hierarchical spatial subdivision structure is created from an input point cloud. The advantages of this data structure are: (1) fast data traversal: frustum and backface-culling, optimal succession of octree cells cache coherence [ChTr99] (2) immediate access to all data in an octree cell (for data sending) (3) multi-resolution. If no normals or splat sizes per 3D

position are included in the point cloud, these data can be simply derived from sample neighborhoods.

### 4.1.1 Octree

We construct the octree data-structure using a two-step procedure. First, we create an ordinary axis-aligned octree. Since we are working with large datasets, special care has to be taken to limit the octree recursion, which could adversely affect the algorithms efficacy. The leaf octree cells contain the actual point-data.

In the second step the heavy loaded octree is rewritten to a fast, compact and memory-coherent octree. Initially, we split the point-data from the octree. The algorithm recursively creates the *point-array*. This array is sorted in such a way that every octree cell has a start index and a size to access its point-data in this *point-array* (see figure 2). This is useful when we need fast data-access to a non-leaf octree cell. Besides a start index and size to its data, each octree cell contains location, normal, normal cone and bounding box information. Each octree cell has some structural information: a level (section 4.1.2), an index to its sibling, and an index to its top-level octree cell (see figure 2). All the data of the octree cell is aligned in 64 bytes for cache-performance reasons. If an octree cell has no siblings it has a recursive index to its parent's sibling (see figure 2: octree cell 10's sibling). A top-level octree cell is a uniform parent at a low depth in the octree: it shares the same point-data as any octree cell beneath it. Each octree cell has an index to the top-level octree cell that contains its data (see figure 2). To align the data structure and avoid cache trashing [ChTr99] we write the octree down to an array, the *octree-cell-array*, by traversing the octree in depth-first order (the same order as the data traverse



**Figure 2: The octree data structure: Data Traverse Process: octree written down to an array, all information available except the point-data. Data Send Process: Top-level octree cell array pointing to point-array. Render Process: Top-level octree cell array pointing to received data packets. Render packets show what has to be rendered.**

processes use (see figure 2)) This way we do not need to save a pointer to the first child of an octree cell.

#### 4.1.2 Multi Resolution

It is not necessary to use the full point-data for a model far from the camera. It is better to use a compact version of the data to save processing and network resources. Other algorithms, e.g. [RuSz00], use the information in their spacial subdivision scheme to create a multi-resolution model. Since we decouple the data structure from the point-data, we cannot introduce multi-resolution point-data in the data-structure. Therefore *level-splats* are introduced. As we mentioned in the previous section, every octree cell has a level (see figure 2). Data-points have level zero, leaf octree cells have level one, and the levels of all other octree cells is one more than the maximum level of their children (see figure 2). To create level(n) splats we build for each level(n) octree cell a spatial subdivision data-structure on its level(n-1) splats. We use this data structure together with a covariance analysis [PaMa02] (Mahalanobis distance [JoIT]) to cluster level(n-1) splats to level(n) splats.

#### Display process

The system contains only one display process, which provides the user-interaction. The display process dynamically divides the view frustum into mini view frusta. This is a sort first approach [MoSt94]. The dimensions of these mini frusta are computed considering a box of interest. Typically this box is the bounding box of the point-data. The display process sends these mini frusta together with camera data and a timestamp to data traverse processes that reported to be idle. The display process keeps a queue of incoming images and sequentially displays these.

#### Data traverse process

A data traverse process only loads the octree-cell-array (see section 4.1.1) into its main memory. This implies that the data traverse processes can work on the entire data set without loading the massive point-data. This way the computational work can be decoupled from the data, resulting in a well-balanced workload. Each idle data traversing process asks the display process a new mini frustum and creates a render packet associated with it. This render packet is filled during the traversal of the octree as described below:

```

 TraverseOctreeCellArray(){
   int index = 0;
   do
     if(whole array[index] in mini frustum)
       AddToPacket(index); index = siblingindex
     else if( part of array[index] in mini frustum)

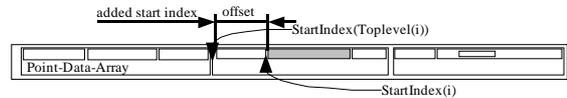
```

```

 if(array[index] benefit of subdivision is high)
   index++
 else
   AddToPacket(index); index = siblingindex
 else if(array[index] out mini frustum)
   index=siblingindex
 while(index exists) }

```

Where *array* is the *octree-cell-array*, *index* is the position in this array of the octree cell that we are using and *siblingIndex* is the position of the sibling of this octree cell in the *octree-cell-array*. This function exploits the structure of the *octree-cell-array* and avoids cache trashing [ChTr99]. Furthermore it uses frustum culling and decides whether the benefit of examining the children of the octree cell is sufficient. The *AddToPacket* function works on the octree cell at position *index* in the *octree-cell-array*. First we try to backface cull the octree cell, considering its normal and normal cone. If the top-level octree cell of the octree cell does not exist, we are too high in the octree and need to examine the children of the octree cell. The algorithm decides which data resolution it should use depending on the screen resolution, the octree cells distance to the camera and the available data resolutions for this octree cell. The size and the start index of the octree cells data are added to the render packet. The added start index is the offset from the octree cells data to the top-level octree cells data (see figure 3).



$$\text{Added Start index} = \text{StartIndex}(i) - \text{StartIndex}(\text{toplevel}(i)).$$

Where *i* is an octree cell.

**Figure 3: Added start index is the offset from the octree cells data to the top-level octree cells data.**

The indices of the used top-level octree cells are also added to the render packet. When the octree traversal is finished, the render packet is ready. Every data traverse process has information concerning the current workload and the available data on each

render node (see section 4.5). The render node with the smallest cost is chosen to receive and render the render packet. The cost is computed as described next:

$$\text{Cost}(i) = \text{Render Cost}(i) + \text{Network Cost}(i)$$

$$\text{Render Cost}(i) = \text{workload on render process}(i) * T_s$$

$$\text{Network Cost}(i) = \text{unavailable data on render process}(i) * T_n$$

Where *i* is a render node, *T<sub>s</sub>* is the time to render one splat and *T<sub>n</sub>* is the inverse network speed. Finally the data traverse process informs all data send processes what unavailable point-data they need to send to the chosen render node.

## Data send process

A data send process loads the *point-array*, or a part of it, grouped per top-level octree cell in its main memory (see figure 2).

Data send processes receive their instructions from the data traverse processes; they inform the data send processes to which render node which top-level octree cells data should be sent (see figure 2). Data send processes always send the entire point-data of a top-level octree cell.

## Render process

In [MoSt94], the authors state that a sort first approach is only scalable if the frame-to-frame coherence is exploited. Therefore, we introduce top-level octree cells. These are regular octree cells at a low depth in the octree (depth three, four or five depending on the size of the model). Combined, all top-level octree cells mutually exclusive enclose the entire *point-array* (see figure 2). When using top-level octree cells we avoid both redundant data in the cache of our render processes and high network traffic. Furthermore, we exploit the frame-to-frame coherence, by sending more data than directly needed.

A render node is a separate workstation running four render processes that share the same memory place. A render node receives two kinds of data streams, one from the data traverse processes and one from the data send processes. Initially, each render node contains an empty array with all top-level octree cells. The point-data in this array is filled each time point-data of a top-level octree cell is received from a data send process. Render packets, sent by the data traverse processes, contain pointers to the point-data of the octree cells that lie in the mini frustum. Each pointer is an offset in the point-data of the top-level octree cell where the pointers octree cell belongs to (see figure 2 and 3). Render packets also indicate which top-level octree cells point-data should be available to render this packet. If all requested point-data is available, an idle render process will render the packet. As long as the requested data is not available, the packet will be queued. To avoid

running out of memory, a least recently used caching scheme is applied. The least recently used point-data of a top-level octree cell will be deleted after a time-out period has expired. All data traverse processes will be informed about this, so they can recompute the cost of sending data to that render node. For the same reason, render nodes inform the data processes about their current workload, this is the amount of points they still need to render. The rendered image is sent back to the display process for composition and display.

In our current framework we use a simplified EWA [ZwMA01] splatting algorithm that could be easily replaced by a more advanced splatting algorithm if required.

## 5. RESULTS

The PC cluster used for our experiments consist of 9 workstations. Each node has two 2.4 Ghz Intel Pentium IV Xeon processors, 2 GB DDR Ram, and is running Suse Linux 9.1. The nodes communicate with the LAM MPI implementation through a gigabit network. Since we are using a purely software based implementation, we exploit the computational power of each workstation and run several processes simultaneously. In our test setup the system runs as many Data Traverse as Render Processes (please note that there is not a one-to-one mapping between these processes.)

### Scalability

#### 5.1.1 Model Complexity

We first consider the scalability of our system with regards to the model complexity. We have two test cases: (1) three dragon point sets with 0.3M, 1,2M and 4,2M points. (2) Different models with different complexities: Dragon 4,2M points, Turbine Blade 10M points, Hand 5M points and Venus 3M points.

##### 5.1.1.1 Splats Per Second

Our experiments showed that if we use only one render node, we are able to splat an average of 1.5 Million Splats per Second, if all necessary data is available on the render node. Figure 4 shows the scalability of the splats per second. If the model

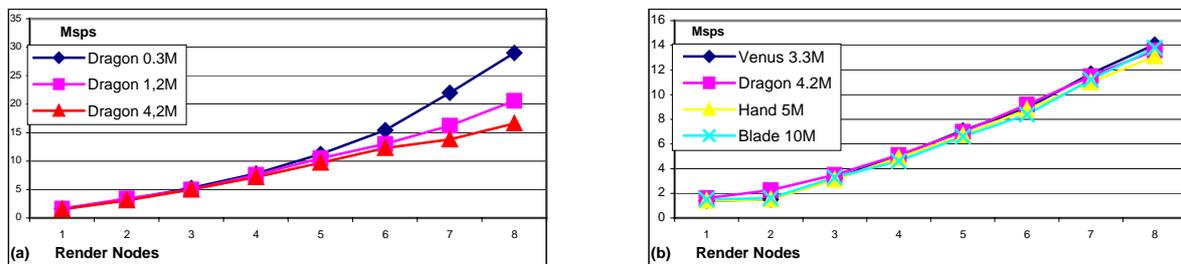
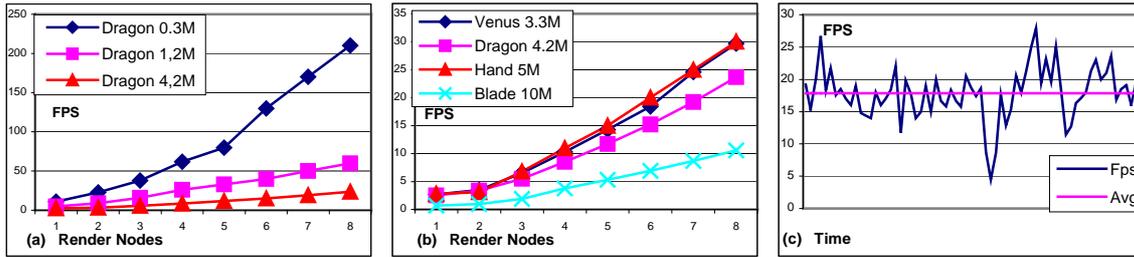


Figure 4:(a)(b) We averagely splat 1.5 Million Splats per Second per render node. If the model grows in complexity the splat rate could drop a little because the cost of traversing the octree increases.



**Figure 5:**(a)(b) If the model grows in complexity more point are needed each frame. Since the splat rate is rather constant, the frame rate will drop. For non-complex models we could render up to 210 fps while very complex models still result in 11 fps.(c) the frame rate is rather constant

grows in complexity, the global splat rate could drop a little because the cost of traversing the octree increases (the difference between figure 4 (a) Dragon 0.3M and (b) Blade 10M).

### 5.1.1.2 Frames per Second

If the model grows in complexity, more points need to be rendered each frame. Since the splat rate is more or less constant (see figure 4), the frame rate will drop for these complex scenes (see figure 5 (a)). However, it is not necessary to render more points than dictated by the screen resolution. This means the frame rate does not entirely depend on the complexity of the model, it also depends on the screen resolution and the available model resolutions. We could speed up the frame rate by choosing the optimal model resolution for each octree cell, depending on its distance to the camera and the screen resolution (see figure 5(b)). This results in a scalable frame rate.

Figure 5 (c) shows us that the frame rate is rather constant. If the frame rate drops, point-data packets are sent.

### 5.1.2 High Resolution

A small part of the computational power is spent on sending images to the display process that loads them to the graphics board. This implies that the performance of our system is not very sensitive to the screen resolution, if the number of splats stays constant. As we can see on figure 6(a) the frame rate only drops if the resolution becomes too high. This is

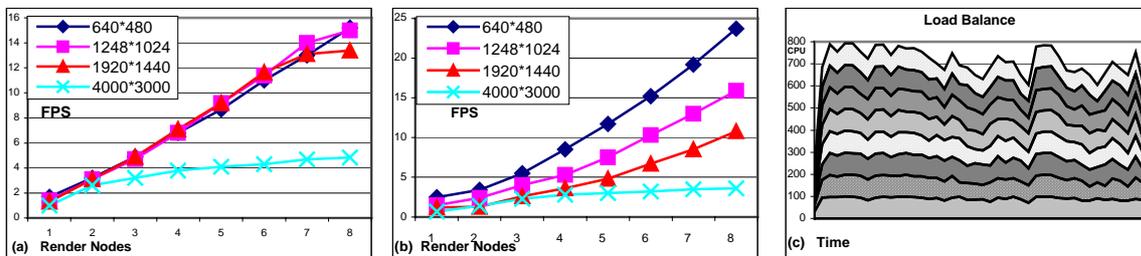
a result of the high communication costs associated with sending high-resolution images. However, if the number of splats increases with the resolution, as we described section 5.1.1, the frame rate will drop faster (see figure 6(b)), because more points need to be rendered. However, the quality of these images will be higher.

### Load Balance

Each render node has a cost to render a given render packet. The correct choice of the render node with the smallest cost (see section 4.3 data traverse node) is vital for good load balancing. In figure 6(c) the workload for 8 render nodes is depicted, during the rendering of the Turbine Blade point set (10M points). When our process starts, the workload is low because many point-data packets are sent to the render nodes. Figure 6(c) clearly indicates that our cost function and system architecture is well chosen, because all render nodes are almost equally loaded and the global workload does not drop too much. When the workload drops, point-data packets are sent.

## 6. CONCLUSION

This paper presents a scalable data distribution strategy for parallel point-based rendering on a PC cluster architecture. Since the used data-structure and the algorithm's architecture decouple the data from the computational work, the system achieves a well balanced workload and each data traverse process can work on the entire data without a full replication



**Figure 6:**(a) Tests are done with the dragon 4M point set. The system is, if the number of splats stays constant, only sensitive to the screen resolution if the overhead of sending the images back to the display node is too high (b) the frame rate drops faster because the number of splats increases with the resolution (c) the workload for 8 render nodes, during the rendering of the Blade point set (10M points).

of the data. The algorithm dynamically partitions the screen into smaller mini frusta (a sort-first approach). Our technique exploits the sort-first properties of the algorithm, by sending more data than is directly needed. Large data sets at high screen resolution can be rendered at interactive frame rates. Point-Based rendering is well suited for a sort-first parallel rendering approach because the overlap factor is negligible.

Topics for further study include faster software point-splatting algorithms with higher quality, using low-level processor instructions. Also, combining clustered CPU and GPU rendering might be an interesting research venue.

## 7. ACKNOWLEDGEMENTS

We would like to thank everybody who helped us with this publication and the Stanford Computer Graphics Laboratory for sharing the models used in our experiments.

## 8. REFERENCES

- [BoMa02] M. Botsch, A. Wiratanaya L. Kobbelt, Efficient high quality rendering of point sampled geometry, 13<sup>th</sup> Eurographics workshop on Rendering, pp 53-64, 2002.
- [BoMa03] M.Botsch, L.Kobbelt,High-Quality Point-Based Rendering on Modern GPUs, 11th Pacific Conference on Computer Graphics and Applications, pp 335,2003.
- [BoMa04] M.Botsch,M.Spernat,L.Kobbelt,Phong Splatting, pp 25-32, Symp.on Point-Based Graphics,2004.
- [CaEd74] E. E. Catmull, A subdivision algorithm for computer display of curved surfaces.1974.
- [ChTr99] T.M. Chilimbi, M. D. Hill, J. R. Larus, Cache-Conscious Structure Layout, Programming language design and Implementation SIGPLAN99, pp 1 –12, 1999.
- [CoLi02] L. Coconu, H Hege, Hardware-accelerated point-based rendering of complex scenes, 13th Eurographics workshop on Rendering, pp 43- 52,2002.
- [DaCa03] C. Dachsbacher, C Vogelgsang and Marc Stamminger, Sequential point trees, Trans. Graph.,pp 657-662, 2003.
- [DeeM93] Data Complexity for virtual reality: where do all the triangles go?, IEEE Virtual Reality Annual International Symposium, pp 357-363, 1993
- [FuTh96] T. A. Funkhouser, Coarse-Grained Parallelism for Hierarchical Radiosity Using Group Iterative Methods, Computer Graphics,pp 343-352, 1996.
- [GoSt96] S.J.Gortler, R. Grzeszczuk,R. Szeliski,M. F. Cohen, The Lumigraph,SIGGRAPH96, pp 253–262, 1996
- [HuGr99]G. Humphreys, P. Hanrahan, A distributed graphics system for large tiled displays, Proceedings of the conference on Visualization '99,pp 215-224,1999.
- [HuGr00]G. Humphreys, I. Buck, M. Eldridge,P. Hanrahan, Distributed rendering for scalable displays, ACM/IEEE conference on supercomputing,pp.30,2000.
- [HuGr02]G. Humphreys, M. Houston, R. Ng,R Frank, S. Ahern,P. D. Kirchner,J. T. Klosowski, Chromium: a stream-processing framework for interactive rendering on clusters, Computer graphics and interactive techniques,pp 693-702 ,2002.
- [JoIT]I.T. Jolliffe, Springer's Series in Statistics, Principal Component Analyse, second edition, pp 92- 93. ISBN 0-387-95442-2
- [LeMa85] M. Levoy, T. Whitted, The use of points as display primitive. Tech. Rep. TR 85-022, University of North Carolina at Chapel Hill.
- [LeMa96] M. Levoy, P. Hanrahan, Light Field Rendering,SIGGRAPH96,pp 31 – 42,1996
- [McLe95] L. McMillan, G. Bishop, Plenoptic Modeling: An Image-Based Rendering System, pp 39-46, 1995.
- [MoSt94] S. Molnar,M. Cox, D. Ellsworth,H. Fuchs, A Sorting Classification of Parallel Rendering, IEEE Computer Graphics and Algorithms, p23-32, 1994.
- [MuCa95] C. Mueller, The sort-first rendering architecture for high-performance graphics, symposium on Interactive 3D graphics, pp 75 - end, 1995.
- [PaMa02] M. Pauly, M. Gross, L.P. Kobbelt, Efficient simplification of point-sampled surfaces, IEEE Visualization pp 136- 170,2002.
- [PfHa00] H. Pfister, M. Zwicker,J.v. Baar, M. Gross, Surfels: Surface Elements as Rendering Primitives, SIGGRAPH00, pp 335-342,2000
- [ReEr98] E. Reinhard, A. Chalmers,F. W. Jansen, Overview of Parallel Photo-realistic Graphics, nr CS-EXT-1998-147, 1998.
- [RuSz00] S. Rusinkiewicz, M. Levoy, QSplat: A Multiresolution Point Rendering System for Large Meshes, pp 343-352, Siggraph00, 2000.
- [RuSz01] S. Rusinkiewicz, M. Levoy,Streaming QSplat: a viewer for networked visualization of large, dense models, symposium on Interactive 3D graphics,pp 63-68, 2001.
- [SaRu00] R. Samanta,T. Funkhouser,K. Li, J. Pal Singh, Hybrid sort-first and sort-last parallel rendering with a cluster of PCs, Eurographics workshop on Graphics hardware, pp 97-108, 2000.
- [SaRu01] R. Samanta,T. Funkhouser,K., Parallel Rendering with K-way Replication, IEEE 2001 symposium on parallel and large-data visualization and graphics, pp 75 –84, 2001
- [ShJo98] J.Shade, S. Gortler,L. He R. Szeliski, Layered depth images, Computer graphics and interactive techniques, pp 231 –242, 1998.
- [WaIn01] I. Wald, P. Slusallek, C. Benthin, Interactive Distributed Ray Tracing of Highly Complex Models, EUROGRAPHICS, Workshop on Rendering, pp 277-288, 2001,
- [WeLe89] L. Westover, Interactive volume rendering, Chapel Hill workshop on Volume visualization pp 9-16, 1989.
- [ZaDa95] D. Zareski, B.Wade, P. Hubbard,P. Shirley, Efficient Parallel Global Illumination Using Density Estimation,IEEE/ACM 1995 Parallel Rendering Symposium (PRS '95),pp 47- 54, 1995.
- [ZwMA01] M. Zwicker, H. Pfister, J.v.Baar, M.Gross, Ewa volume splatting, IEEE Visualization 2001, pp 29-36, 2001.

# Cached Geometry Manager for View-dependent LOD Fendering

Roberto Lario  
Universidad Complutense  
Madrid, Spain  
rlario@dacya.ucm.es

Renato Pajarola  
University of California Irvine  
USA  
pajarola@acm.org

Francisco Tirado  
Universidad Complutense  
Madrid, Spain  
ptirado@dacya.ucm.es

## ABSTRACT

The new generation of commodity graphics cards with significant on-board video memory has become widely popular and provides high-performance rendering and flexibility. One of the features to be exploited with this hardware is the use of the on-board video memory to store geometry information. This strategy significantly reduces the data transfer overhead from sending geometry data over the (AGP) bus interface from main memory to the graphics card. However, taking advantage of cached geometry is not a trivial task because the data models often exceed the memory size of the graphics card. In this paper we present a dynamic Cached Geometry Manager (CGM) to address this issue. We show how this technique improves the performance of real-time view-dependent level-of-detail (LOD) selection and rendering algorithms of large data sets. Alternative caching approaches have been analyzed over two different view-dependent progressive mesh (VDPM) frameworks: one for rendering of arbitrary manifold 3D meshes, and one for terrain visualization.

## 1. INTRODUCTION

The functionality and speed of graphics hardware has increased significantly in last few years, making the GPU a programmable stream processor with sufficient power and flexibility to perform intensive calculations. Despite advances in the graphics hardware, the data transfer from main memory to the graphics card remains the major bottleneck [HCH03]. This restriction prevents the full exploitation of the potential computational horsepower of the GPU and introduces significant overhead in short data transfers [THO02].

View-dependent level-of-detail (LOD) algorithms can significantly reduce the amount of data transfer as the geometric scene complexity is adaptively minimized using a view-dependent error metric [LRC03]. The adaptive nature of such methods introduces constant but infrequent and small geometric changes between consecutive frames. Our goal is to take advantage of this fact using the video memory of modern consumer graphics hardware as geometry cache. The rendering performance can greatly be improved if the geometric data of a given scene is stored in video memory. However, the limited size of available video memory restricts the complete caching of big data models. The use of view-dependent LOD algorithms can provide a

solution to this problem because the geometric information required for rendering a scene at a certain LOD is in general only a small fraction of the full resolution model. This visible portion of geometry information can be cached on the graphics card using video memory (see Figure 1) and is updated every frame when the viewpoint location of the camera or the resolution is changing. In order to efficiently handle the constantly occurring video memory updates, a *Cached Geometry Manager* (CGM) is needed. The continuous adaptive LOD changes guarantee that only a small amount of the cached geometry in the video memory has to be updated between consecutive frames.

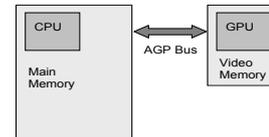


Figure 1: CPU/GPU communication diagram.

In this paper we describe several strategies to implement an efficient geometry-cache manager. Two *view-dependent progressive mesh* (VDPM) frameworks are used to test the proposed techniques and to show the speed-up in rendering performance when applied to a general view-dependent LOD algorithm. The first framework is *FastMesh* [Paj01], it uses an efficient view-dependent and adaptive LOD method for rendering arbitrary 3D meshes in real-time. The general concepts of this framework are common to most similar VDPMs, e.g. such as [XV96], [Hop97], [LE97], [DMP97] or [KL01]. The second framework is *QuadTIN* [PAL02], an efficient quadtree-based triangulation approach for irregular terrain height-fields that provides fast quadtree-based adaptive triangulation, view-dependent LOD-selection and real-time rendering. Many interactive terrain

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 2005 conference proceedings, ISBN 80-903100-7-9  
WSCG 2005, January 31-February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency – Science Press

visualization systems, e.g. such as [SS92], [LKR96], [DMP96], [Pup96], [Paj98], [BAV98] or [EKT01], exhibit a similar top-down LOD triangulation and rendering approach.

The remainder of the paper is organized as follows: Section 2 presents a very brief overview of related work. Section 3 describes the Cached Geometry Manager. In Section 4 the two VDPM frameworks are presented to test the CGM approach. Experimental results are presented in Section 5 and Section 6 ends the paper with some conclusions.

## 2. RELATED WORK

Despite the extensive work on level-of-detail (LOD) techniques [LRC03], only very few methods that use cached geometry have been proposed recently. One possible reason for this lack is that only recent generations of graphics cards allow the application program to manage large amounts of video memory systematically and dynamically for storing geometry. In [Lev02] a terrain rendering algorithm is presented that operates on clusters of cached geometry called aggregate triangles. The dynamically generated aggregate triangles are kept in the geometry cache for several frames to improve rendering performance. A similar concept is followed in [CGG03a], [CGG03b] where a LOD hierarchy of simplified height-field triangle patches is generated in a pre-process. At run-time the appropriate LOD triangle patches are selected for rendering and a LRU strategy is used for caching. In [LPT03] square patches of a quadtree-based hierarchical terrain triangulation are used for fast rendering and caching in video memory. A common limitation of the above methods is that they are restricted special-purpose solutions for terrain rendering and not applicable in general to other VDPM frameworks. In contrast, the concepts presented in this paper are directly applicable to a wide range of VDPM frameworks. A remarkable approach to provide seamless geometric LODs is provided by GLOD [CLD03]. It allows advanced users to define discrete LOD objects as well as specify the use of video memory for patches of the geometry. In contrast to GLOD, the proposed Cached Geometry Manager interacts directly with VDPM frameworks that dynamically generate continuously adaptive LOD meshes and provides transparent use of video memory.

## 3. CACHE GEOMETRY MANAGER

Most view-dependent simplification frameworks represent the geometry in a hierarchical data structure called vertex hierarchy (see Figure 2). The nodes located near the root correspond to low-resolution vertices while those located farther away represent high-resolution detail vertices. The vertex hierarchy is dynamically queried to perform a view-dependent LOD simplification for each frame. A front of active

nodes divides the current nodes used to generate the simplified scene from the rest. This frontier can continuously and incrementally be updated between rendered frames.

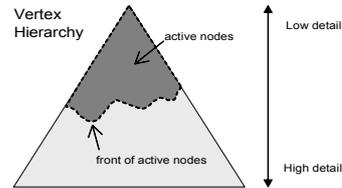


Figure 2: Vertex hierarchy diagram.

One can observe that by far most of the vertices of a scene remain active between consecutive frames and just a small fraction changes its state. Hence most vertices can be stored and kept continually in video memory in order to improve rendering performance. Each frame only few vertices require a read operation from main memory, to transfer to video memory, when they change their state from inactive to active. Note that a remove operation is needed when the video memory is full and new vertices have to be added. Inactive but cached vertices are the prime candidates to be deleted from video memory in this case. A video memory manager is required to carry out these operations.

## Vertex Arrays

Indexed vertex buffers or indexed vertex arrays (IVA) in OpenGL, are the best way to take advantage of modern graphics accelerator (see section 11.4.5 in [MH02]). The application puts the data into specific buffers and gives the pointers to the driver, which accesses the data directly. Hence vertex arrays need much fewer OpenGL function calls for rendering than the classic immediate mode vertex submission (using `glBegin()/...glVertex()/...glEnd()` blocks). In [Mar00], several methods to optimize submission of vertex data in OpenGL are described. Our CGM takes advantage of vertex arrays in combination with the OpenGL extension `NV_vertex_array_range` [Kill99]. The order and positions of vertices is different in the cached IVA from the main memory IVA. Thus the vertex indices of an indexed triangle mesh must be remapped accordingly for rendering. However, the rendering speedup will compensate for this extra re-indexing required by a dynamic CGM.

## CGM Strategies

In this and the following section we describe three basic caching strategies to implement the video memory manager. These three strategies are going to be discussed for two variants of VDPM frameworks in order to cover the range of applications: those which calculate an explicit front of active nodes by incremental updates between consecutive frames, and those which implicitly define the active front by selecting the active nodes top-down for each frame (see Figure 2). In this section we first discuss the more

general case of implicit active front VDPM frameworks. Note that a non-explicit front does not mean it does not exist, in fact the front always implicitly exists in any view-dependent LOD framework. The implicitly-defined refers to the behavior of the VDPM framework that has no other information than if a vertex is selected or not for each rendered frame. From here on we will refer to both video memory and geometry cache as equivalent concepts. The basic two tasks of the cache manager are: (1) to determine that a vertex is already resident (cached) in the video memory, and (2) to find and use an open slot in the cache to store a new vertex. Task (1) can efficiently be determined by a cross indexing: each vertex in main memory has a field that indicates the cache index where it was last stored, and each cache slot has an index field indicating which vertex it stores. Hence if both indices coherently cross-link the same vertex then it is already cached and ready for use. More complicated is task (2) for which we describe viable strategies below.

**First-Available Strategy (FA):** This simple strategy uses the video memory as a linear list of slots with flags. This list is incrementally traversed from the beginning to the first non-used slot (First Available) every time a new vertex must be cached. Then this slot is marked as used. The process continues while there are vertices to cache, and a pointer is moved from the head to the end to search for the next available open slot. Owing to the fact that the list of slots is sequentially traversed this strategy can be implemented using a simple array, as illustrated in Figure 4a). Each slot is considered used when it stores a vertex used in the current or last frame. This policy considers the fact that it is very likely that a vertex used in frame  $i$  will also be required in frame  $i+1$ . Hence each slot flag is an integer counter which stores the last frame in which that cached vertex was used. This strategy is simple to implement, but has one potential drawback: unused slots near the beginning of the list will immediately be overwritten when a new vertex has to be cached while unused slots at the end may cache an unused vertex for a long time. This bias of reusing cache slots based on their position is not necessarily the best solution. At the expense of more complexity, the next strategy addresses this problem.

**LRU Strategy:** As mentioned above, the FA strategy considers any empty slot in the cache as equally good. If a slot has not been used in the current or last frame it is considered available. However, there is an intuitive reason that more recently used vertices are more likely to be used again than vertices that have not been used for a long time. Hence a more refined policy is to take into account the age of the unused slots and use a last-recently-used (LRU) strategy. The LRU parameter is directly obtained from the frame

counter associated with each slot. One possible data structure to make use of this strategy is a doubly-linked-list. Two pointers (head and tail) are needed for the proposed implementation as shown in Figure 4b). The head points to the youngest slot, and the tail points to the oldest slot. New vertices are cached in the slot pointed to by tail which is then moved to the head. Reused slots of rendered vertices already in cache are simply moved from their current position in the linked list to the head. Consequentially, unused slots automatically move towards the tail which always points to the oldest slot entry. Note that these operations do not imply a displacement of the actual vertex data in video memory, it is just a mechanism for the cache manager to maintain access to the last-recently-used open slot. Each slot in this linked list corresponds to a fixed memory location in the cache.

**LRU + Error-PriorityQueue Strategy:** Figure 2 shows clearly that the vertices near the top of the hierarchy are more significant as they correspond to coarser LOD information. Consequently, these vertices are included in the mesh representation before any vertices of finer LODs. Therefore, for a new vertex it is more suitable to choose among the empty slots the one that corresponds to an old vertex which represents a fine level-of-detail. In order to add this new feature to the CGM we propose to categorize the age of the unused slots and introduce a priority-queue for the oldest-category vertices. The oldest category vertices are naturally and compactly stored at the end of the LRU list as described above. Hence as shown in Figure 4c) we only manage this last section of the LRU list in a priority-queue with the LOD error-metric parameter as key. Note that it is not advisable to choose a big priority queue size since this data structure is more costly than the doubly-linked-list of the simple LRU approach.

As with the LRU approach, reused slots are moved from the current location to the head and unused slots slowly sink towards the tail. The tail marker also indicates the bounds of the oldest-category. Thus elements at the tail are moved to the priority-queue as soon as their age has reached a certain limit and the priority-queue is not at maximal capacity. When a new vertex has to be inserted into the cache, the top slot of the priority-queue is used and moved to the head.

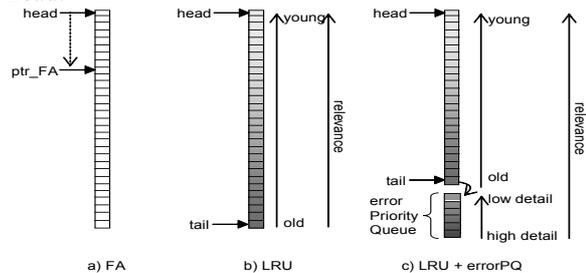


Figure 3: Data structures for the CGM strategies.

## CGM Strategies for Front-Frameworks

The strategies described in the previous section can be refined if the VDPM framework has explicit knowledge of which vertices have been removed from and which vertices have been added to the current LOD triangle mesh. Thus if the change from active to inactive, and vice-versa in Figure 2, is explicitly observable by the application. This feature is typical in LOD systems that maintain an explicit active front for the current frame and update this front incrementally as illustrated in Figure 5. For a new frame, the newly activated vertices are called added (+) vertices, and those deactivated are called removed vertices (-). For each frame the added vertices have to be inserted into video memory, if not already cached from previous frames, while the removed vertices (may) remain cached but change their slot flag to be unused. Note that the removed vertices have always just been active in the previous frame.

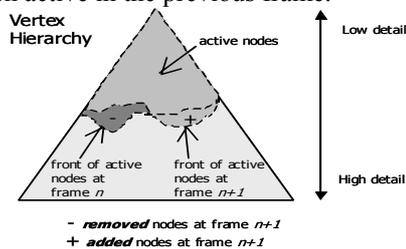


Figure 4: Vertex hierarchy of a front-framework.

**FA Strategy for front-framework:** This strategy, while obviously suboptimal when information about both added and removed vertices is explicitly provided by the LOD system, applies without changes to explicit-front frameworks.

**LRU Strategy for front-framework:** The LRU policy described previously can be improved using a third pointer, called frontier in Figure 6 that divides the active slots from the inactive ones.

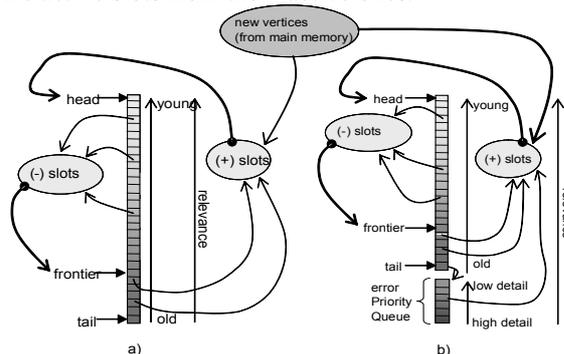


Figure 5: a) LRU for Front-Frameworks. b) LRU + errorPQ for Front-Frameworks. (-) slots of removed vertices. (+) slots of added vertices.

The slots of removed vertices are moved to just below the frontier while slots of added vertices change their position from the tail to the head. Advantage can be taken for vertices that were already active and cached in the previous frame because their corresponding slot

in the LRU list is not affected by any move operation in the linked list. Note that these reused vertices are by far the largest fraction of active vertices. Therefore, compared to the basic LRU cache algorithm, linked-list operations are limited to the few removed and added vertices in front-frameworks.

**LRU + Error-PriorityQueue Strategy for front-framework:** Following the same idea expressed above, a priority queue may consider the LOD error-metric to choose the least relevant available slot. As with the LRU strategy also the LRU + priority-queue strategy can be refined if the sets of removed and added vertices are explicitly known and a frontier pointer separates the active from the inactive slots. As illustrated in Figure 7, the slots of removed (-) vertices are moved to behind the frontier pointer, while the slots of added (+) vertices are moved from the priority queue to the head. The slots between the frontier and the tail pointer are candidates to be transferred to the priority-queue if it is not at maximum capacity. Again, advantage is taken by not touching any of the slots of vertices that remain active between consecutive frames.

## 4. TARGET FRAMEWORKS

Two different view-dependent LOD frameworks have been analyzed for testing the proposed Cached Geometry Manager: FastMesh [Paj01] and QuadTIN [PAL02]. As mentioned in the introduction, the former is a VDPM system for rendering arbitrary manifold 3D meshes, and the latter is for rendering terrain height-fields. Both frameworks are briefly explained in the following sections before we provide experimental results.

### Arbitrary Mesh Render System: FastMesh

FastMesh [Paj01] is an efficient hierarchical multiresolution triangulation framework based on a half-edge triangle mesh data structure and edge-collapse operations. Optimized computation of view-dependent error metrics within the framework provide conservative LOD error bounds. FastMesh is efficient both in space and time cost, and it spends only a fraction of the time required for rendering to perform the view-dependent LOD error calculations and dynamic triangle mesh updates. Conceptually it follows exactly the diagram shown in Figure 5 as many other similar approaches such as [XV96], [Hop97], [LE97], [DMP97], [KL01] do.

One of the main features of FastMesh is the explicit calculation of the front of active nodes, which is obtained for any frame by incremental changes to the previous frame. Hence FastMesh directly provides information about vertices added or removed from the front for every frame and falls into the category of front-frameworks described above in Section 3.3. The initial implementation of FastMesh rendered the mesh

as a list of active triangles in immediate mode vertex submission which has been changed for this project to an indexed vertex array (IVA) rendering mode. Experimental results using the front-framework CGM strategies described in Section 3.3 are given for this VDPM framework in Section 5.1.

### Terrain Rendering System: QuadTIN

QuadTIN [PAL02] is an efficient quadtree-based terrain triangulation approach. It provides fast quadtree-based adaptive triangulation, view-dependent LOD-selection and real-time rendering. Its fundamental quadtree-based triangulation method and top-down vertex selection and rendering approach is similar to many other terrain visualization systems such as [SS92], [LKR96], [Paj98], [BAV98], [EKT01]. In contrast to other approaches, however, QuadTIN presents an efficient quadtree-based triangulation approach over irregular input point sets with feature adaptive sampling resolution while preserving a regular quadtree multiresolution hierarchy over the irregular input data set. Although the resulting quadtree hierarchy is not balanced, it conforms to the restricted quadtree constraints [SS92]. Additional information such as geometric approximation error, bounding spheres and normal cones are used for view-dependent LOD-triangulation and rendering. Like most other terrain visualization systems, QuadTIN selects the active vertices for a LOD of a particular viewpoint in a recursive top-down traversal for each frame. Hence QuadTIN belongs to the category of VDPMs with implicitly defined active front and does not provide explicitly the removed or added vertices between two consecutive frames. Experimental results using the basic CGM strategies given in Section 3.2 are reported in Section 5.2 for this VDPM framework.

## 5. EXPERIMENTAL RESULTS

Experimental results were performed on a 3.0 GHz Pentium 4 with 1GB RAM using an NVIDIA GeForceFX 5200 graphics card. For all scenes a 45° vertical field-of-view camera followed several test trajectories as described below.

### FastMesh Results

The models used with the FastMesh rendering system are given in Table 1. The rendering experiments were averaged over 1000 frames in a window of 800 x 800 pixels using an error tolerance equal to one and a half pixels (projective tolerance of geometric error projected on screen).

|          |   |   |   |
|----------|---|---|---|
|          |  |  |  |
| model    | hand  | dragon  | happy   |
| vertices | 327323  | 437645  | 543652  |
| faces    | 654666  | 871414  | 1087716   |

Table 1: 3D models used with FastMesh.

Three different camera trajectories have been analyzed to examine the impact of the Cached Geometry Manager within the FastMesh framework as illustrated in Figure 8:

- Circular camera trajectory.
- Small camera rotations.
- Straight line camera trajectory.

These camera movements are very common as they are typical movements observers normally execute to explore a 3D object. The camera is pointing to the center of the model in all the trajectories.

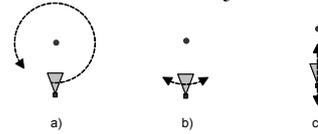
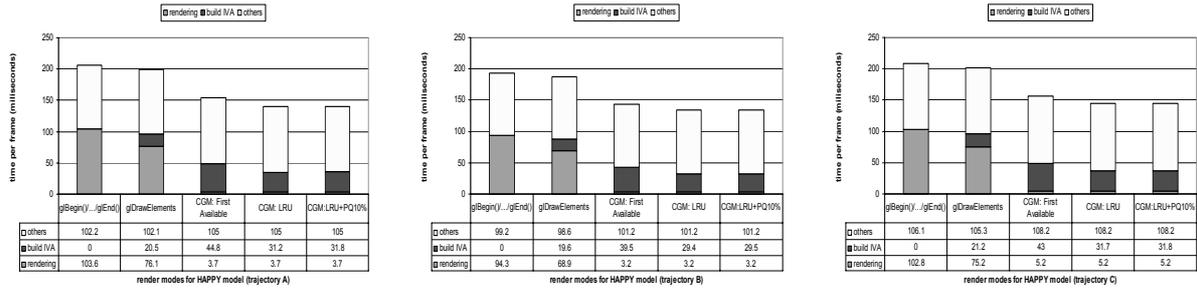


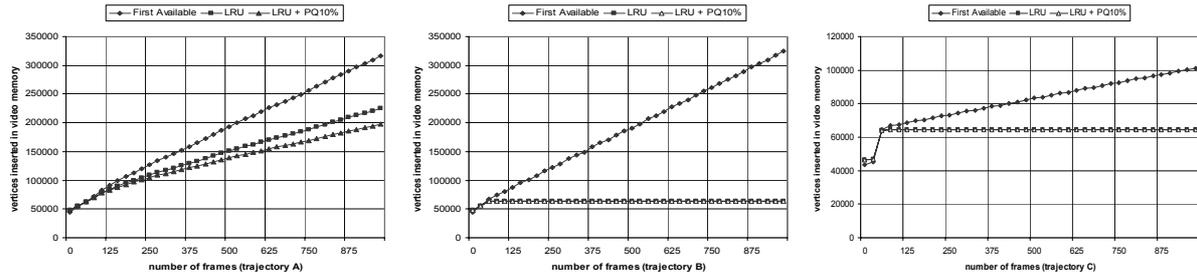
Figure 6: 3D object rendering camera trajectories: a) circular trajectory. b) small rotations. c) straight line trajectory (zoom in/out).

The chosen CGM size (slot-list length) was  $216 = 65536$  because all models required at least  $215 = 35768$  vertices to render from every tested viewpoint. Of course, in this context no LOD mesh can have more vertices than available in the geometry cache, and the application program must make sure that the best LOD for a limited number of vertices is selected. If this mesh exceeds the cache size then the application should disable the CGM and render the mesh in normal mode, or possibly render the mesh using the CGM in multiple passes. We are only studying the effect of the CGM in this paper and do not address the latter issues in this work. In the experiments, the size of each vertex element is 36 bytes, consisting of: 3 floats (position) + 3 floats (normal) + 3 floats (color).

We have focused the numerical results on the biggest model (happy) to avoid excessive and repeated information. Statistical data for the other models is given in Table 3. Figure 9 shows the per-frame timing results (in milliseconds) of happy model for the three different camera trajectories. The total time per frame has been divided into three parts: the rendering time, the time needed to construct the vertex array (build IVA), and the time required to perform the error metrics and updating the LOD mesh (others). Note that a brute-force rendering of this model using a standard immediate mode vertex submission achieved less than 4 frames per second, or equivalently required more than 250ms per frame. Our CGM techniques only affect the vertex array construction and rendering time but not any other tasks of the VDPM framework. In particular, the LOD-computation and vertex selection is not affected by the CGM and thus limits the overall speed-up with respect to the observable frame-rate. Hence we focus on the speed-up achieved only within the rendering part of a VDPM framework which is the sole target of the CGM.



**Figure 7:** Per frame timing results (in milliseconds) for the Happy model for: a) circular camera trajectory, b) small camera rotations and c) straight line camera trajectory. The build IVA time contains the time consumed by the cache memory manager in modes where the CGM is enabled.



**Figure 8:** Vertices inserted into video memory for the Happy model for: a) circular camera trajectory, b) small camera rotations and c) straight line camera trajectory.

As we mentioned in section 4.1, the initial version of FastMesh traverses a linked-list of triangles to render the model in immediate mode vertex submission. The standard IVA mode is still faster than the previous despite the transformation from a linked-list of triangles to an indexed triangle array. The LRU+errorPQ%10 strategy employs an error priority queue size equal to 10 percent of the total cache size. As expected, the rendering improvement is significant. The build-IVA time for these modes increases the workload of the CGM. However, the global speedup obtained for the three strategies easily compensates the extra CGM cost. In fact, the actual rendering cost, which is the only cost affected by the CGM, is improved by a factor of up to 3 (including the IVA build time) as shown in Table 3.

Despite three different camera trajectories, all show almost the same behavior. More information may be obtained taking into account the number of vertices inserted in video memory. It allows to understand which strategy makes better use of the cached geometry since more inserted vertices involves more data transfer from main to video memory. This information is given in Table 2.

| CGM TYPE        | Trajectory A | Trajectory B | Trajectory C |
|-----------------|--------------|--------------|--------------|
| First Available | 171873       | 302288       | 290747       |
| LRU             | 169928       | 286331       | 66453        |
| LRU+PQ10%       | 169972       | 281707       | 66436        |

**Table 2:** Vertices inserted into video memory for the three CGM modes over 1000 frames (Happy model).

The First Available strategy clearly makes the worst use of cached geometry, especially for small camera

rotations. In contrary to our initial expectations, in most cases the simple LRU strategy outperforms the LRU + Error-PriorityQueue strategy. The latter gives the best result only for the circular camera trajectory, and even in this case, the lower data transfer rate does not compensate for the more expensive priority queue operations. Figure 10 b) and c) show the inefficiency of the First Available strategy for the last two camera trajectories, where the insertion of new vertices is unnecessary after a certain number of frames. Recall the most expensive frame is always the first because the cache stores no vertices at that time.

Table 3 lists the rendering speed-ups achieved by the different CGM strategies with respect to the original immediate mode vertex submission FastMesh version. The first column corresponds to the variant with a standard main-memory IVA but no CGM. The last three columns indicate the speedup factors for the three implemented CGM strategies. The individual speed-ups for the rendering stage reach factors up to 3 which shows the real impact of the Cached Geometry Manager on the rendering phase of a VDPM framework.

| Model  | camera trajectory | CGM render modes |                 |      |             |
|--------|-------------------|------------------|-----------------|------|-------------|
|        |                   | std IVA          | First Available | LRU  | LRU + PQ10% |
| happy  | A                 | 1.07             | 2.14            | 2.97 | 2.92        |
|        | B                 | 1.07             | 2.21            | 2.89 | 2.88        |
|        | C                 | 1.07             | 2.13            | 2.79 | 2.78        |
| dragon | A                 | 1.06             | 1.97            | 2.69 | 2.66        |
|        | B                 | 1.05             | 2.12            | 2.83 | 2.8         |
|        | C                 | 1.05             | 2.12            | 2.76 | 2.76        |
| hand   | A                 | 1.07             | 2.25            | 3.07 | 3.06        |
|        | B                 | 1.05             | 2.29            | 2.92 | 2.91        |
|        | C                 | 1.05             | 2.31            | 2.95 | 2.95        |

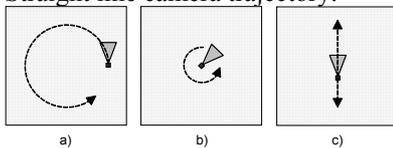
**Table 3:** FastMesh speed-ups (just rendering stage) for different CGM strategies.

## QuadTIN Results

The height-field model used for the QuadTIN rendering experiment is the well known Puget Sound data set (2563548 vertices, after QuadTIN-preprocess error tolerance = 6 meters, [PAL02]). The results were averaged over 3000 frames in a window of 1024 x 768 pixels using an error tolerance equal to one pixel (projective tolerance of geometric error projected on screen).

The chosen CGM size was 216 = 65536 following the same criteria applied as for the experiments with FastMesh. The size of each vertex element in this case is 32 bytes: 3 floats (position) + 3 floats (normal) + 2 floats (texture coordinate). The camera trajectories tested to perform the CGM analysis with the QuadTIN rendering system are the following (see Figure 11):

- Circular camera trajectory.
- Camera rotation with fixed eye.
- Straight line camera trajectory.



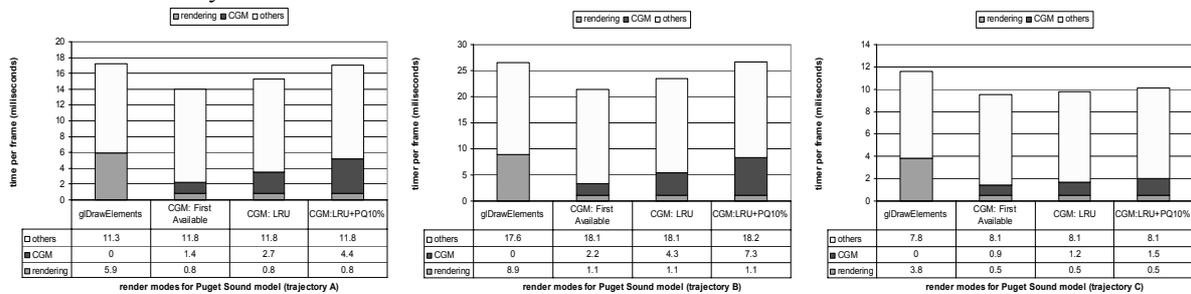
**Figure 9:** Terrain rendering camera trajectories: a) circular trajectory. b) camera rotation with fixed eye. c) straight line trajectory.

The CGM strategies applied to QuadTIN are the ones described in detail in Section 3.2. The QuadTIN system constructs an indexed triangle strip as required for a standard IVA approach. Note that due to the implicit definition of the active front, no information about incrementally added or removed vertices

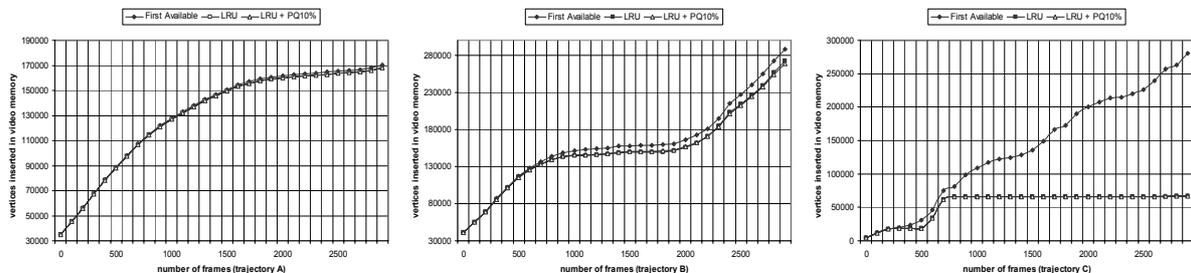
between consecutive frames is provided. QuadTIN only reports which vertices are selected for a particular frame and LOD. Note again that the CGM extension only affects the rendering time but not the LOD-selection and meshing parts of the VDPM framework, and hence we focus on the achievable rendering speed-up which is the sole target of the CGM.

Figure 12 shows four columns, one for each rendering strategy, for each camera trajectory: the first column for the standard IVA mode, and the three others for the different CGM strategies. In this case, the best result is achieved by the First Available (FA) strategy. Despite the fact that the FA strategy still makes the worst use of the geometry cache (see Table 4), its simple and fast data structures are still advantageous over the doubly-linked list of slots in the two different LRU CGM strategies. This result is not completely surprising as the minor data transfer overhead of FA is amortized by the simple and fast array data structure for slots.

The straight line camera trajectory deserves a special discussion (see Figure 13c) since the FA strategy remains the fastest despite its bad reuse of cached geometry. The LRU and LRU+PQ10% strategies require more computation time but much less data transfer. Depending on the CPU speed in relation to the AGP bus bandwidth this result may slightly change, and the LRU strategies may win over the FA strategy for certain configurations. The relation between CPU speed and AGP bus bandwidth of the system will decide which is the best strategy.



**Figure 10:** Per frame timing results (in milliseconds) for the Puget Sound data set for: a) circular camera trajectory, b) camera rotation with fixed eye and c) straight line camera trajectory.



**Figure 11:** Vertices inserted into video memory for the Puget Sound data set for: a) circular camera trajectory, b) camera rotation with fixed eye and c) straight line camera trajectory.

| CGM TYPE        | Trajectory A | Trajectory B | Trajectory C |
|-----------------|--------------|--------------|--------------|
| First Available | 322353       | 329214       | 102097       |
| LRU             | 228694       | 63465        | 64688        |
| LRU+PQ10%       | 200126       | 63465        | 64690        |

**Table 4:** Vertices inserted in video memory for the CGM modes over 3000 frames (Puget Sound model).

The QuadTIN rendering speed-up factors are shown in Table 5. The speedup of the rendering stage itself, which is the only stage affected by the CGM, reaches factors up to 2.7. This dramatically shows the potential of using a CGM in a view-dependent LOD rendering system.

| camera trajectory | CGM render modes |      |             |
|-------------------|------------------|------|-------------|
|                   | First Available  | LRU  | LRU + PQ10% |
| A                 | 2.68             | 1.69 | 1.13        |
| B                 | 2.7              | 1.65 | 1.06        |
| C                 | 2.71             | 2.24 | 1.9         |

**Table 5:** QuadTIN speedups (just rendering stage) for different CGM modes.

## 6. CONCLUSION

This paper presents several strategies to implement an efficient Cached Geometry Manager that takes advantage of on-board video card memory for caching vertex data. It provides effective solutions to manage the video memory as a geometry cache in order to dramatically reduce the vertex data transfer rate from main to video memory for each rendered frame. The proposed techniques can be applied to a wide range of view-dependent LOD rendering frameworks, and allow the efficient reuse of cached geometry information stored on the video graphics card.

The presented approaches significantly improve the rendering performance of view-dependent LOD rendering applications with little extra implementation effort. Experimental results on two different VDPM frameworks have confirmed the suitability and the effectiveness of our approach to dramatically accelerate the rendering stage. Overall performance speed-up of observable frame rates heavily depends on the application-side view-dependent LOD-selection and meshing framework. More recent and improved VDPM frameworks – compared to the tested FastMesh and QuadTIN systems – with significantly lower LOD-selection and meshing cost will exhibit significantly higher overall frame rate performance if combined with a dynamic CGM as demonstrated in this paper.

## 7. ACKNOWLEDGMENTS

This research was supported by the Spanish research grant TIC 2002-750 and the New Del Amo award UCDM-33657.

## 8. REFERENCES

[BAV98] BALMELLI L., AYER S., VETTERLI M.: Efficient algorithms for embedded rendering of terrain models. IEEE Inter. Conference on Image Processing ICIP 98 (1998), pp. 914-918.  
 [CGG03a] CIGNONI P., GANOVELLI F., GOBBETTI E., MARTON F., PONCHIO F., SCOPIGNO R.: BDAM - Batched

Dynamic Adaptive Meshes for High Performance Terrain Visualization. EG/IEEE TCVG Symp. on Visualization 2003.  
 [CGG03b] CIGNONI P., GANOVELLI F., GOBBETTI E., MARTON F., PONCHIO F., SCOPIGNO R.: Planet-Sized Batched Dynamic Adaptive Meshes (P-BDAM). IEEE Visualization 2003, pp. 147-154.  
 [CLD03] COHEN J., LUBKE D., DUCA N., SCHUBERT B.: GLOD: Level of Detail for the Masses (2003). URL <http://www.cs.jhu.edu/~graphics/TR/TR03-4.pdf>.  
 [DMP96] DE FLORIANI L., MARZANO P., PUPPO E.: Multiresolution models for topographic surface description. The Visual Computer (Aug. 96), pp. 317-345.  
 [DMP97] DE FLORIANI L., MAGILLO P., PUPPO E.: Building and traversing a surface at variable resolution. IEEE Visualization 97 (1997), pp. 103-110.  
 [EKT01] EVANS W., KIRKPATRICK D., TOWNSEND G.: Right-triangulated irregular networks. Algorithmica (March 2001), pp. 264-286.  
 [HCH03] HALL J. D., CARR N. A., HART J. C.: Cache and Bandwidth Aware Matrix Multiplication on the GPU. Technical Report UIUCDCS-R-2003-2328. University of Illinois at Urbana-Champaign Computer Science Department. April 2003.  
 [Hop97] HOPPE H.: View-dependent refinement of progressive meshes. SIGGRAPH 97 (1997), pp. 189-198.  
 [Kil99] KILGARD M. J.: NVIDIA OpenGL Extension NV\_vertex\_array\_range. URL: [http://www.nvidia.com/dev\\_content/nvopengl/specs/GL\\_NV\\_vertex\\_array\\_range.txt](http://www.nvidia.com/dev_content/nvopengl/specs/GL_NV_vertex_array_range.txt).  
 [KL01] KIM J., LEE S.: Truly selective refinement of progressive meshes. Graphics Interface 2001, pp. 101-110.  
 [LE97] LUEBKE D., ERIKSON C.: View-dependent simplification of arbitrary polygonal environments. SIGGRAPH 97 (1997) pp. 199-208.  
 [Lev02] LEVENBERG J.: Fast view-dependent LOD rendering using cached memory. IEEE Visualization 2002, pp. 259-265.  
 [LKR96] LINDSTROM P., KOLLER D., RIBARSKY W., HODGES L. F., FAUST N., TURNER G. A.: Real-time, continuous level of detail rendering of height fields. SIGGRAPH 96 (1996), pp. 109-118.  
 [LPT03] LARIO R., PAJAROLA R., TIRADO F.: Hyperblock-QuadTIN: Hyper-block quadtree based triangulated irregular networks. IASTED International Conference on Visualization, Imaging and Image Processing (VIIP 2003), pp. 733-738.  
 [LRC03] LUEBKE D., REDDY M., COHEN J., VARSHNEY A., WATSON B., HUEBNER R.: Level of detail for 3D graphics. Morgan Kaufman. 2003.  
 [Mar00] MARSELAS H.: Optimizing Vertex Submission for OpenGL. Game Programming Gems, pp. 353-360. Charles River Media. 2000.  
 [MH02] MÖLER T., HAINES E.: Real-time rendering. 2nd edition. A K Peters. 2002.  
 [Paj98] PAJAROLA R.: Large scale terrain visualization using the restricted quadtree triangulation. IEEE Visualization 98 (1998), pp. 19-26 and 515.  
 [Paj01] PAJAROLA R.: FastMesh: Efficient View-dependent Meshing. Pacific Graphics 2001, pp. 22-30.  
 [PAL02] PAJAROLA R., ANTONIJUAN M., LARIO R.: QuadTIN: quadtree based triangulated irregular networks. IEEE Visualization 2002, pp. 395-402.  
 [Pup96] PUPPO E.: Variable resolution terrain surfaces. 8th Canadian Conference on Comput. Geometry (1996), pp. 202-210.  
 [SS92] SIVAN R., SAMET H.: Algorithms for constructing quadtree surface maps. 5th International Symposium on Spatial Data Handling (August 1992), pp. 361-370.  
 [THO02] THOMPSON C. J., HAHN S., OSKIN M.: Using modern graphics architectures for general-purpose computing: a framework and analysis. 35th annual ACM/IEEE international symposium on Microarchitecture (2002), pp. 306-317.  
 [XV96] XIA J. C., VARSHNEY A.: Dynamic view-dependent simplification for polygonal models. IEEE Visualization 96 (1996), pp. 327-3.

# Hardware Pipeline for Rendering Clouds of Circular Points

Adam Herout  
Faculty of Information Technology  
Brno University of Technology  
Božetěchova 2  
612 00 Brno, Czech Republic  
herout@fit.vutbr.cz

Pavel Zemčík  
Faculty of Information Technology  
Brno University of Technology  
Božetěchova 2  
612 00 Brno, Czech Republic  
zemcik@fit.vutbr.cz

## ABSTRACT

This paper presents an algorithm for image rendering using FPGA (Field-Programmable Gate Arrays). The image is rendered by an FPGA chip coupled with a DSP (Digital Signal Processor) on an experimental board. The graphical data is 3D point-clouds – sets of particles that are from the geometrical point of view oriented ellipses in 3D space. Such scene representation seems to be more suitable for potentially many purposes than the most commonly used triangle meshes. The actual experimental implementation which verifies the concept and shows promising results is described.

## Keywords

point clouds, rendering, FPGA, hardware acceleration

## 1. INTRODUCTION

The developers of graphics applications can rely on the presence of accelerated graphics engines in the computers. However, it is quite unfortunate from the point of view of choice of graphics and imaging algorithms that the function of the graphics accelerators is usually quite strictly limited to rendering of planar triangles/polygons and limited choice of shading and texture algorithms and it is usually impossible to use them for implementation of any other algorithms. At the same time, the real research of such high-performance graphics subsystems is being done by the manufacturers and by only a limited number of affiliated institutions, such as research laboratories and universities.

A reasonable way forward was offered by the recent development of Field Programmable Gate Arrays (FPGAs). Current technological progress allows implementation of even very complex devices in the programmable logic devices and achieving

good results even with architectures and algorithms that are not supported by the traditional computer graphics manufacturers.

This paper presents a hardware architecture for real-time high quality rendering of point-based graphical scenes [Gro02, Pfi00, Zwi01]. By a particle we mean a surface element (also referred to as surfel or point, element) defined by  $x, y, z$  coordinates,  $n_x, n_y, n_z$  normal, size, and color. The design is based on an FPGA chip, hosted on a multi-purpose board featuring the FPGA chip, DSP (Digital Signal Processor), DRAM and SRAM memory. Common graphical accelerators (designed to efficiently render polygon-based entities) are unsuitable for this purpose since they do not offer any good way of transferring simple point/particle data. Transfer of triangle vertex data is effective enough (rasterization algorithms are far more time consuming than the transfer itself) but the process of rendering points using this common hardware faces the bottleneck of data stream bandwidth [Gro02]. Some manufacturers of the graphics hardware are accepting the above mentioned trend and are already experimenting with the particles and programmable logic [Mit03].

Probably the most feasible geometrical representation of the scene element (particle) is an oriented circle whose projection is an ellipsis. The rendering algorithm can be subdivided into several principal parts:

1. Projection of the particles' positions into 2D screen space and Z co-ordinate and computation

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 2005 conference proceedings ISBN 80-903100-7-9  
WSCG'2005, January 31-February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency – Science Press

of the corresponding particles' projected normal and radius. This is merely 3D projection that is implemented using a transformation matrix multiplication (see e.g. [Wat93]).

2. Evaluation of the particles' color (lightness) based on the projected normal vector, local lighting model (material), and the light sources' and observer's parameters. This task is implemented through a precalculated color (lightness) table indexed by the quantized normal vector.
3. Rendering of the particles into the image frame buffer (one-by-one with visibility solved using depth-buffer). This task is done through specialized circuits programmed into the FPGA and is described in more details below.

In the proposed approach, parts 1 and 2 are performed through the host processor (DSP) while the part 3 which is the actual rendering is performed through programmable hardware in the FPGA and supported by the host DSP only in terms of data flow organized through the host processor's DMA (direct memory access) channels.

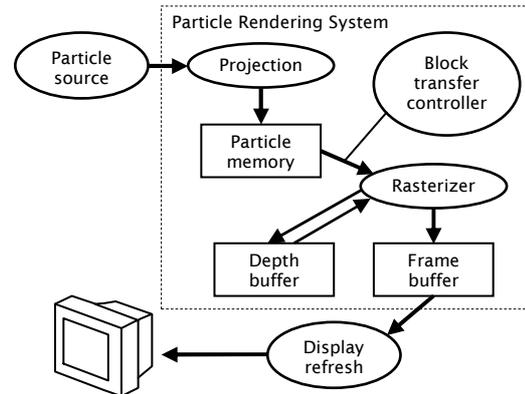
The particle rendering engine being described is the "simplistic" implementation of the proposed rendering architecture. It should be seen rather as a "proof of concept" than as the full-scale implementation. For this reason, maximum possible rendering subtasks were left on the host DSP processor. (However, they can eventually be moved into the FPGA.)

The block diagram of the rendering engine is shown in Figure 1. It is based on the above constraints and uses the Texas Instruments C6711 DSP [TMS01] as the host processor that handles the particles and performs the above rendering subtasks. The DSP then transfers the particle data into the Xilinx Virtex E-300 FPGA [Vir01] through DMA block memory transfer. The particle data comprises the coordinates, encoded shape (described below), and color information.

## 2. RENDERING ALGORITHM

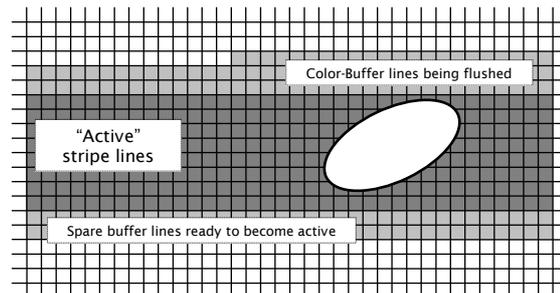
As each particle generally affects large number of pixels, it is desirable to have the frame and depth buffers distributed in several memory banks that can be accessed in parallel in order to parallelize the rendering process. To achieve efficient parallelization of rendering, it must be ensured that the particles affect minimum possible number of words in the memory banks and at the same time that the affected words are as uniformly as possible distributed in all memory banks. The constraints in the FPGA led us to the decision to use 8 memory banks with 8 bits for

color and 8 bits for the depth value (with the possible extension to 32 bits for four 8-bit pixels).



**Figure 1. Rendering engine block diagram**

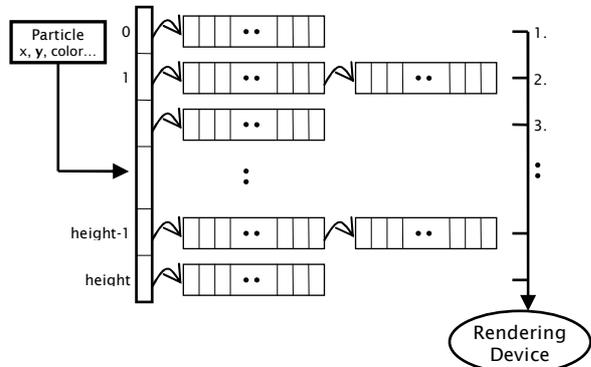
Our research [Her04, Zem03] performed up till now resulted in the following concept of "striping". The basic idea of this algorithm is to let the FPGA handle one horizontal stripe (a portion of the frame-buffer covering several subsequent scan-lines) and render particles coming from the particle source into it. The particle source ensures particles come in a predefined order – with increasing y (i.e. vertical) coordinate. The stripe is then moving vertically across the frame-buffer by one line, rendering all particles and covering the whole frame-buffer area. Each move-down of the stripe consists of two steps: a) flushing the top-most line to the global frame-buffer, and b) re-using it as a fresh bottom line for the next stripe position. To avoid delays caused by flushing of the finished lines, more color-buffer lines are allocated, allowing the rendering to proceed continually – see Figure 2.



**Figure 2. The striping algorithm**

The presumption of the particles being sorted into groups by one coordinate and in a particular order is not too restrictive. The particles can be sorted into this form easily, provided the system contains a memory buffer large enough to store all the particles in the scene. Such memory does not necessarily have to provide high-bandwidth random access, and it does not need to be connected to the FPGA closely as each of the particles is needed by

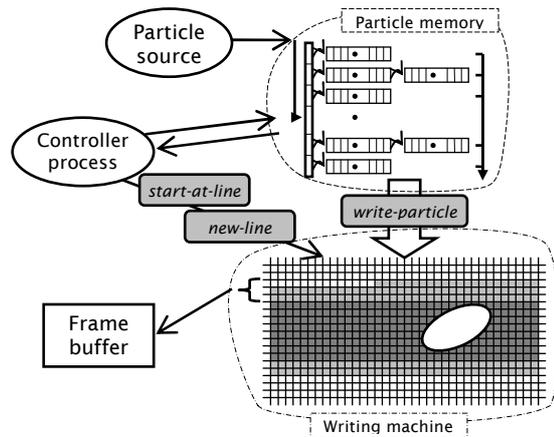
the FPGA during the rendering only once and in a defined order. This memory can contain particle lists for each line of the frame buffer and sort incoming particles into them (see Figure 3). The process of sending the particles into the rasterizing FPGA will be started after receiving all particles of the scene.



**Figure 3. Sorting particles by their y-coordinate**

Incorporating these mechanisms together with a rasterization pipeline rasterizing the shapes of the

particles (ellipses) may result in an architectural design similar to the one shown in Figure 5, based on a “particle writing machine” in Figure 4.

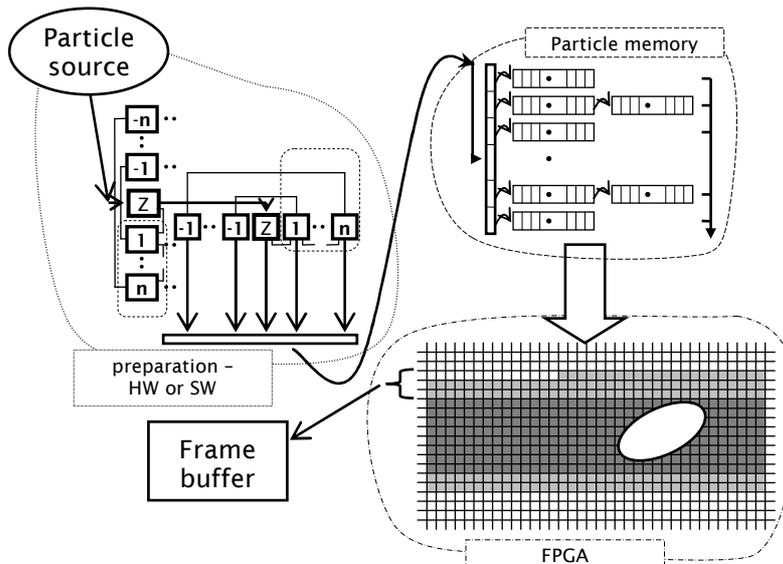


**Figure 4. Particle writing machine utilizing the striping**

The particle writing machine embodies the following operations:

| operation             | arguments             | description   |
|-----------------------|-----------------------|---|
| <i>write-particle</i> | x, d,<br>color, shape | Writes particle of given properties (d=depth, shape – encoded into a small number of bits by a suitable algorithm).<br>Note that the y coordinate is determined by the state of the writing machine – the number of calls to the new-line operation.                  |
| <i>new-line</i>       |                       | Disposes the “oldest” line of the stripe, starts flushing into the frame-buffer, and activates the next free spare color-buffer line.   |
| <i>start-at-line</i>  | y                     | Starts a new frame, skips y first lines without particles (filled with background color).<br>This operation may well exist without the argument, only starting a new frame – it would be then followed by appropriate number of subsequent calls to <i>new-line</i> . |

**Table 1. Basic operations of the particle writing machine**



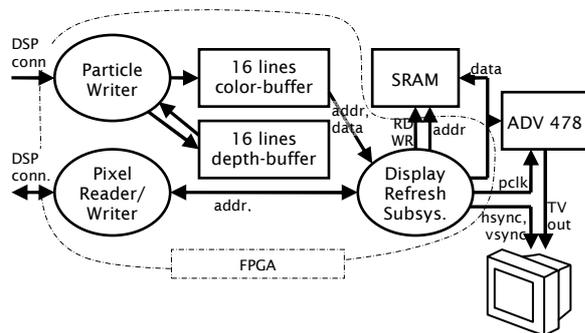
**Figure 5. Over-all rasterizer design, consisting of a particle source incorporating the rasterization process, of the particle memory sorting particles by their y-coordinate and the striping writer**

### 3. EXPERIMENTAL HARDWARE IMPLEMENTATION

For the hardware implementation, FPGA Xilinx Virtex E 300 has been used. In the future, Virtex II is planned to be used instead. Hardware design programmed in the FPGA consists of particle reader/writer, pixel reader/writer, frame and depth buffers and the viewing engine. FPGA input frequency is 100 MHz, but for the major part of the design, 50 MHz is used. Accessing time to the SRAM (used as a video-RAM) is 15 ns. This memory and ADV 478 chip (D/A converter and palette memory) are placed outside the FPGA. There is also 16 MB SDRAM placed at the board used by the DSP. This SDRAM runs at 100 MHz.

The **Display Refresh Subsystem** takes care about correct viewing of an image placed in the SRAM and its writing into this memory.

Every pixel clock period, data are read from the memory (address is the counter automatically incremented every pixel clock cycle). Meanwhile, shared data bus is put to the third state at the side of FPGA, so that data from the memory could be read by the ADV 478 chip. Pclk rising edge ensures the data on this bus to be converted to analog format suitable for a TV or a monitor.



**Figure 6. Experimental implementation block scheme**

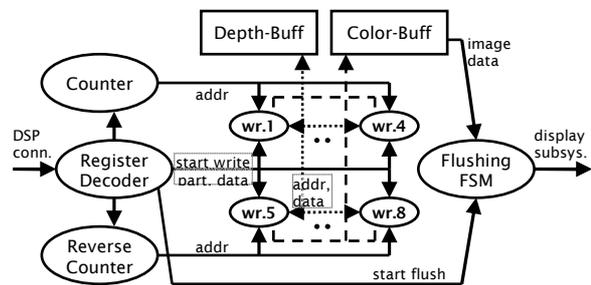
Between every two pixels read for the TV out, it is possible to place one read or write cycle from/to SRAM. Such cycle is used for reading the image to the DSP and writing pixels into the RAM. The writing requests come from two sources – pixel writer (pixels written directly from the DSP) and frame buffer of the particle writer.

SRAM is organized as a two-bank video RAM in order to implement double-buffering: while writing an image to one bank, the second bank is being shown on the screen and vice versa.

An extension to the striping particle writer concept as presented in section 2 is the **Pixel Reader/Writer** unit, which allows accessing the

SRAM frame-buffer directly from the DSP. It simply gets data and address from the DSP and writes to the SRAM through the viewing engine. This operation may be used for writing additional information to the screen. It is also possible to read the data from the memory, and e.g. store the image in the DSP controlled memory for future use. However, this means of access to the frame-buffer is meant primarily for debugging and testing purposes, it is not very fast, since any request through this port waits to be synchronized with monitor refresh and the particle writer.

Functional description of the **Particle Writer** unit is described in the theoretical part of this article. Hardware implementation consists of few state machines using two groups of block RAMs – one for the frame buffer and one for the depth-buffer.



**Figure 7. Particle Writer**

When a particle description is sent from the DSP, it is processed by the register decoder. Base horizontal coordination is set to its position, and two counters are running to define the exact position of the current processed pixel. One counter runs from zero to maximum and displays the upper part of the particle. Second counter runs from maximum to zero and displays the lower mirrored part. Writing engines start writing reacting to the start write signal. While processing, data are read from the depth-buffer, and writing engines decide whether to write (both to the frame and the depth-buffer) or not by comparing the actual depth with the depth from the depth-buffer. Reading from the depth-buffer must start some cycles before the writing process due to the memory read latency.

When the special code word is written by the DSP, writing is moved to the next line and flushing of the processed line is started – data from the Block-RAM are written to the SRAM through the viewing engine.

### 4. ACHIEVED RESULTS

The proposed algorithm was fully implemented on an experimental setup shown below in Figure 8, that uses the Camea DX6 board [Cam03]. Current maximal number of particles rendered by the FPGA is 5 million per second. This number comes out from

the clock period which is 20 ns, and the number of cycles required for showing one particle. One column of a particle is written in one period, and two periods are required for pre-reading the depth-buffer data. Totally 10 periods are 200 ns per particle.

Of course, possibilities exist to improve the performance. Using more advanced FPGA chip (for example Virtex II) would lead into higher possible frequency (we assume at least 100 MHz). Another

possibility is to parallelize writing to the memory by setting the width of the data bus to the Block-RAMs from 8 bits to 32 bits. Extra logic for treating this situation would be needed, but speed-up ratio would be up to four. We could also avoid the depth-buffer reading latency by pipelining. Finally, we could show one particle in 2 clock cycles (10 ns), which means speedup up to 10 times from the current state to 50 million particles per second, still using standard off-the-shelf components.



**Figure 8. Experimental setup displaying a medical data set**

## 5. CONCLUSION

In this paper, a rendering system based on Xilinx Virtex E-300 FPGA and Texas Instruments C6711 DSP was described. The system implements a modern 3D point-cloud rendering algorithm and is fully functional. 3D point cloud graphics seems to be a concept of close future for visualization and realistic rendering, partially replacing the most common approach at the moment – triangle meshes.

The proposed rasterization algorithm solves the rendering task including the visibility issues between the particles inside the FPGA in order to achieve high performance. A part of the projection phase is left to the host task being performed by the DSP. While this solution leaves space for further hardware acceleration, it was chosen as the best possible approach to test the concept.

Hardware implementation in the FPGA contains control subsystems treating read and write cycles of the video-RAM, pixel writer and the particle writer. The particle writer unit consists of eight pixel writers that write the data to the internal Block RAMs, the flushing unit that transfers the image to the video SRAM, and of the DSP bus interface.

Current speed of particle drawing is 5 million per second. Changes that could increase this number up to 50 million still using currently available general purpose components are proposed. However, this implementation is considered to be rather a proof-of-concept than a final graphics acceleration solution. The Virtex II Pro Xilinx FPGA that is to come should allow further optimizations and may be ground for a graphical hardware challenging graphics equipment of desktop computers.

## 6. ACKNOWLEDGMENTS

This work was partly supported by the “Rapid prototyping tools for development of HW-accelerated embedded image- and video-processing applications”, GA AVČR, T400750408 grant.

## 7. REFERENCES

- [Cam03] “DSP Accelerator Boards”, CAMEA, Ltd., (available at <http://www.camea.cz/products/accelerators.cz.htm>)
- [Her04] Herout, A, Zemcik, P: Animated Particle Rendering in DSP and FPGA. In: SCCG 2004 Proceedings, Bratislava, SK, 2004, pp 237-242, ISBN 80-223-1918-X
- [Gro02] Gross, M: “Point Based Computer Graphics”, Spring Conference of Computer Graphics 2002, Budmerice, Slovakia, 2002
- [Mit03] Mitsubishi Electric Research Laboratories: “SURFELS - Surface Elements as Rendering Primitives, (available at <http://www.merl.com/projects/surfels/>)
- [Pfi00] Pfister, H, Zwicker, M, van Baar, J, Gross, M: Surfels: Surface Elements as Rendering Primitives. Proceedings of SIGGRAPH 2000, pp 335-342
- [Ree83] Reeves, WT: “Particle Systems – A Technique for Modeling a Class of Fuzzy Objects”, ACM Transactions on Graphics, Vol. 2, No. 2, April 1983
- [Rus01] Rusinkiewicz, S: “QSplat: A Multiresolution Point Rendering System for Large Meshes”, Proceedings of SIGGRAPH 2001, USA, 2001
- [TMS01] TMS3B0C6711, TMS320C6711B Floating point Digital Signal Processors, Texas Instruments, SPRS088B, September 2001, USA, 2001, (available at <http://www.ti.com>)
- [Vir01] VirtexTM 2.5V Field Programmable Gate Arrays, Xilinx, DS003-1 (v2.5), April 2, 2001, USA, 2001, (available at <http://www.xilinx.com>)
- [Wat93] Watt A.: 3D Computer Graphics, Addison-Wesley, Wokingham, UK, 1993
- [Zem02] Zemcik, P: “Hardware Acceleration of Graphics and Imaging Algorithms Using FPGAs”, SCCG 2002, Budmerice, Slovakia, 2002
- [Zem03] Zemcik, P, Tisnovsky, P, Herout, A: “Particle Rendering Pipeline”, SCCG2003, Budmerice, Slovakia, 2003
- [Zwi01] Zwicker, M, Pfister, H, van Baar, J, Gross, M: “Surface Splatting” In: Proceedings of SIGGRAPH 2001, ACM SIGGRAPH, Los Angeles 2001

# Multiple Transparent Material-enriched Isosurfaces

Ravindra L. Kanodia\*

Lars Linsen\*†

Bernd Hamann\*

\* Institute for Data Analysis and Visualization (IDAV)  
Department of Computer Science  
University of California, Davis  
Davis, CA 95616, U.S.A.

† Department of Mathematics and Computer Science  
Ernst-Moritz-Arndt-Universität Greifswald  
Greifswald, Germany

## ABSTRACT

Isosurface extraction is a standard method for visualizing scalar volume data that can be used to render a specific material boundaries inherent in multi-material data sets. Multiple transparent isosurfaces can thus be used to visualize multiple material boundaries, but still fail to capture any data in between the boundary layers. We describe how isosurfaces can be “enriched” with surrounding material information. By visualizing surrounding material, both material boundary information and gradient - or change in density - information of the scalar field are represented. Visualizing multiple transparent material-enriched isosurfaces leads to a fairly effective volumetric impression. Thus, our approach approximates results obtained from direct volume rendering.

The visualization of multiple transparent isosurfaces requires a back-to-front rendering of the typically triangulated surface components. The order of the surfaces’ triangles is imposed by the location of the convex cells they are extracted from, which supports fast rendering of multiple isosurface.

## Keywords

Multiple Transparent Isosurfaces, Volume Rendering, Material-enriched Isosurfaces.

## 1 Introduction

Isosurface extraction and direct volume rendering are the most common techniques used for visualizing scalar-valued volume data. The volumetric data sets typically result from numerical simulations or from 3D scanning and imaging processes. The computed or measured scalar fields represent material properties

such as pressure, temperature, density, etc.

Isosurface extraction methods explicitly compute the three-dimensional geometry of material boundaries in the form of two-manifold surfaces. Once an isosurface has been extracted, surface rendering is efficient, especially when exploiting acceleration mechanisms using graphics hardware. Moreover, the rendered surface is of high quality, as a sharp material boundary is computed and photo-realistic shading algorithms can be applied. On the other hand, one isosurface represents only one material boundary and fails to capture most of the volumetric information.

Direct volume rendering methods provide a “richer” visualization, as the whole range of material information is incorporated into the rendered image. Transfer functions are used to control color and opacity for the depiction of the various materials present in the volume data. Direct volume rendering results provide a strong sense of the overall content of a data set. Unfor-

\*ravi@cyberman.com, hamann@cs.ucdavis.edu

†linsen@uni-greifswald.de

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

Conference proceedings ISBN 80-903100-7-9  
WSCG 2005, January 31-february 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency - Science Press

unately, the computational costs are high. Hardware-accelerated approaches exist and are commonly used, but are limited in their applicability to large data sets.

Multiple transparent isosurface rendering can be used to overcome the single isosurface’s drawback by adding more information to rendered images, while maintaining the advantages in speed and quality. In Section 3, we describe how a standard isosurface extraction algorithm can be extended to extract multiple isosurfaces into a data structure that allows for fast and correct rendering. We exploit the properties of marching isosurface-extraction algorithms to compute the occlusion properties of the multiple isosurfaces “on-the-fly”, i. e., without applying an expensive post-processing step for polygon sorting.

Unfortunately, visualization of multiple transparent isosurfaces do not quite establish the same-quality volumetric impression of the visualized data set as direct volume rendering techniques do. Multiple transparent isosurface rendering can only approximate direct volume-rendered images when using “spiky” transfer functions, i. e., transfer functions with vanishing opacity everywhere except for a finite number of small separated intervals. The spikes or peaks in the opacity function correspond to individual isosurfaces. The individual isosurfaces are colored with respect to the color information of the respective material, which can be looked up in the transfer function. In between opacity peaks, however, there are gaps, which represent material that cannot be captured using multiple isosurfaces.

We present an approach to “fill” these gaps to a certain extent. We enrich the multiple transparent isosurfaces with information about the material in-between the isosurfaces. In Section 4, we describe how multiple material-enriched isosurfaces can be used to approximate volume rendering. We present and discuss our results in Section 5.

## 2 Related Work

Isosurfaces are commonly extracted from volumetric scalar fields using marching-cell algorithms. These algorithms go back to the marching-cubes approach [LC87], which operates on uniform rectilinear grids. The algorithm “marches” through the grid considering each cell of the grid once. Intersections of the isosurface with the edges of each cell are computed using linear interpolation of the values at the cell’s corners. The intersection points are connected forming a triangular surface mesh. Many extensions and improvements have been made to the original approach including the solution of ambiguous cases [Ham91],

better triangulations [Nie03], and a generic algorithm combining previous approaches [BL03]. When splitting the hexahedral cells into tetrahedral ones, ambiguous cases are resolved (while making certain assumptions though), and isosurfaces can be extracted using a marching-tetrahedra algorithm [GH95].

Multiple isosurfaces are extracted by the method of Wan et al. [WTTL96]. Gerstner [Ger02] developed a multiresolution approach for fast multiple isosurface extraction from adaptively refined tetrahedral grids. For a fast rendering of the multiple isosurfaces, Gerstner deployed a binary tree-based sorting method. The implementation of our multiple isosurface extraction algorithm is based on a marching-tetrahedra approach, but any other isosurface extraction approach that marches through a grid of convex polyhedra could be used instead.

Gerstner’s method [Ger02] renders different isosurfaces with different colors and opacities, which leads to results similar to those one can obtain using direct volume rendering with spiky transfer functions. Information about the scalar field in between the isosurfaces is not captured. In particular, the transformation from one material, whose boundary is represented by one isosurface, to another material, whose boundary is represented by another isosurface, is not visualized, as it would be when using direct volume rendering with smoother transfer functions.

Direct volume rendering goes back to the work by Levoy [Lev88], who introduced the concept of volume ray casting for uniform rectilinear grids. Data values within a cell are computed using trilinear interpolation. Ray casting in an improved form is still a widely used approach for direct volume visualization, as it produces high-quality results. Another direct volume rendering approach with high-quality results is splatting [Wes90], where the trilinear interpolation is replaced by a Gaussian function. The shear-warp approach [LL94] is targeted toward higher speed at the expense of lower-quality results. Recent developments in graphics hardware made the 3D-texture-mapping approach favorable in terms of speed and, with floating-point precision, even in terms of quality. The uniform rectilinear grid is mapped to the 3D texture memory and trilinear interpolation is performed by the hardware components [CCF94]. One obvious limitation is the restriction to the size of the 3D texture memory.

When comparing isosurface extraction with direct volume rendering, isosurface rendering is still faster and/or produces higher-quality images in terms of smoothness and illumination. The advantage in speed is given by the reduction of the volume data set to a two-manifold surface, whose triangular mesh repre-

sensation allows for fast rendering. Moreover, many fast and realistic lighting techniques for triangular mesh rendering exist, which allow for smooth shading without rendering artifacts. Also, the explicit computation of material boundaries facilitates quantitative analyses, which is particularly important for biomedical applications such as clinical measurements. On the other hand, the reduction of a volumetric model to a surface model obviously can lead to significant loss of information. Direct volume rendering incorporates all of the volume data and shows overlaying and internal features in a realistic fashion. We enrich multiple transparent isosurfaces with surrounding material information, such that the volumetric impression of a direct volume rendering is approximated while maintaining the advantages of isosurface rendering with respect to speed and quality.

### 3 Fast Multiple Isosurface Rendering

Let  $f : D \rightarrow R$  be a trivariate scalar function with domain  $D \subset \mathbf{R}^3$  and range  $R \subset \mathbf{R}$ . The values  $f(\mathbf{x}) \in R$  of function  $f$  are known at discrete sample points  $\mathbf{x} = (x, y, z) \in D$ . Let the sample points be organized in a three-dimensional grid with convex grid cells. Moreover, let  $v_0, \dots, v_{n-1} \in R$  be an arbitrary number  $n$  of isovalues. The multiple isosurfaces are defined by  $f(\mathbf{y}) = v_i$  for  $i = 0, \dots, n-1$  and  $\mathbf{y} \in D$ . Let  $F_i$  be the isosurface with respect to isovalue  $v_i$ ,  $i = 0, \dots, n-1$ .

A marching isosurface extraction algorithm determines an isosurface  $F_i$  by iterating through the three-dimensional grid once, i. e., each cell is considered once, and determining an isosurface component within each cell independently, if existent.

Typically, uniform rectilinear or tetrahedral grids are used for discrete data representation and a linear interpolation model for the determination of the isosurface components within each cell. The approach presented in this paper does not depend on the grid structure and the interpolation method used. It only requires the cells of the grid to be convex, which can easily be achieved in the unlikely case that they are not. For implementation purposes, we used uniform tetrahedral grids and linear interpolation within each tetrahedron, following the ideas in [GH95].

Multiple isosurfaces can be extracted just as single ones are, i. e., by iterating through the grid once and extracting for each cell all existent isosurface components within the cell for all isosurfaces  $F_i$ ,  $i = 0, \dots, n-1$ . Thus, the algorithm is still linear in the number of cells. Each cell stores up to  $n$  isosurface components. When using a linear interpolation

method, the isosurface components are represented using a polygonal surface model.

When rendering multiple transparent isosurfaces, the isosurface components must be sorted in a back-to-front or front-to-back order according to depth from the view-point or viewing plane. We employ a fast yet simple back-to-front sorting method. It essentially exploits a loophole for depth-sorting: The isosurface components in a scene do not have to be truly sorted according to depth from viewing plane, as long as the rendering algorithm guarantees that no isosurface component is drawn after another isosurface component which occludes it.

Under the assumption that all grid cells are convex and non-overlapping polyhedra, we can uniquely determine whether one cell is in front of another cell with respect to a given viewing vector, or the other way round. Each isosurface component lying entirely in the back cell must be rendered before any isosurface component lying entirely in the front cell. Moreover, the “in-front-of” relation defines a partial order, as it fulfills the property of being transitive. Thus, the isosurface components can be sorted by sorting the cells.

The sorting of the cells can be done hierarchically by grouping neighbored cells to form larger convex and non-overlapping cells. For uniform rectilinear or uniform tetrahedral grids, cells can be grouped to rows of cells, and rows can further be grouped to slabs. Obviously, rows and slabs are, again, convex and non-overlapping.

Exploiting this three-step hierarchy, cells can be sorted in constant time. We only have to determine for each of the three axes of the grid’s coordinate system whether high or low values are closer to the viewing plane, i. e., we have to determine the orientation of the grid with respect to the viewing direction. The sorting of the cells is implicitly given by grid order.

It remains to sort the isosurface components within each cell. If isosurface components are represented using a polygonal surface model, the number of triangles per cell is bounded by the maximum number  $t$  of triangles per cell generated by the single isosurface extraction algorithm. Therefore, for multiple isosurface extraction, the total number of triangles per cell is bounded by  $n \cdot t$ . As this number is small, we employ a standard depth-sort algorithm to sort the triangles within each cell, which does not slow down the performance of our multiple isosurface rendering algorithm noticeably.

## 4 Material-enriched Isosurfaces

Rendering multiple isosurfaces obviously exhibits additional information compared to a visualization using a single isosurface. However, this additional information partially gets lost when displaying the surfaces using same color and opacity. Since for performance reasons multiple isosurfaces are extracted simultaneously, they are also stored together and cannot be easily separated afterwards. Therefore, already during the extraction phase, we have to tag each isosurface component and even each triangle. The tag can be an identifier of the material whose boundary is represented by the isosurface. Multiple isosurfaces are rendered by assigning color and opacity to each material [Ger02].

By performing this assignment step, a transfer function, as known from direct volume rendering, is approximated. However, this use of multiple transparent isosurfaces to approximate direct volume rendering is rather limited. When using direct volume rendering, the transfer function allows for color and opacity assignments for every value  $v \in R$ . When using multiple isosurface rendering, color and opacity for only  $n$  values  $v_0, \dots, v_{n-1} \in R$  are assigned. Any other material with value  $v \neq v_i, i = 0, \dots, n - 1$ , is not visualized.

To add more material information to a visualization, we introduce the concept of material-enhanced isosurfaces. During isosurface extraction, we first compute the surface normal  $\mathbf{n}$  at point  $\mathbf{p}$  as the normalized gradient, which needs to be computed anyway if smooth shading algorithms are applied. We determine and store the material value  $v_{near}$  found at a short distance  $\lambda$  from point  $\mathbf{p}$  along the normal direction  $\mathbf{n}$ , i. e.,

$$v_{near} = f(\mathbf{p} + \lambda \mathbf{n}).$$

During rendering, this additional material information can be used to color the isosurface. The color associated with material value  $v_{near}$  is obtained from the used transfer function. Also, the distance  $\lambda$  can be determined from the transfer function: The opacity function has peaks at isovalues  $v_0, \dots, v_{n-1}$ ; the wider these peaks are, the greater is distance  $\lambda$ .

We have implemented two versions to color material-enriched isosurfaces. Let  $F_i$  be the isosurface to be rendered. The first option is to render the isosurface with the color of material value  $v_{near}$ . The second option is to render the isosurface by blending the color of material value  $v_{near}$  with the color of isovalue  $v_i$ . The former leads to a stronger volumetric impression, as more emphasis is placed on the material around each isosurface, while the latter leads to a more realistic impression, as the color is closer to the color assigned to the isovalue.

## 5 Results and Discussion

In Figures 1 and 2, we compare visualizations using material-enriched isosurfaces with standard isosurface visualizations in the context of single isosurface rendering. The isosurfaces are extracted using an extended marching-tetrahedra approach. The colors used in the renderings are obtained from a user-defined transfer function.

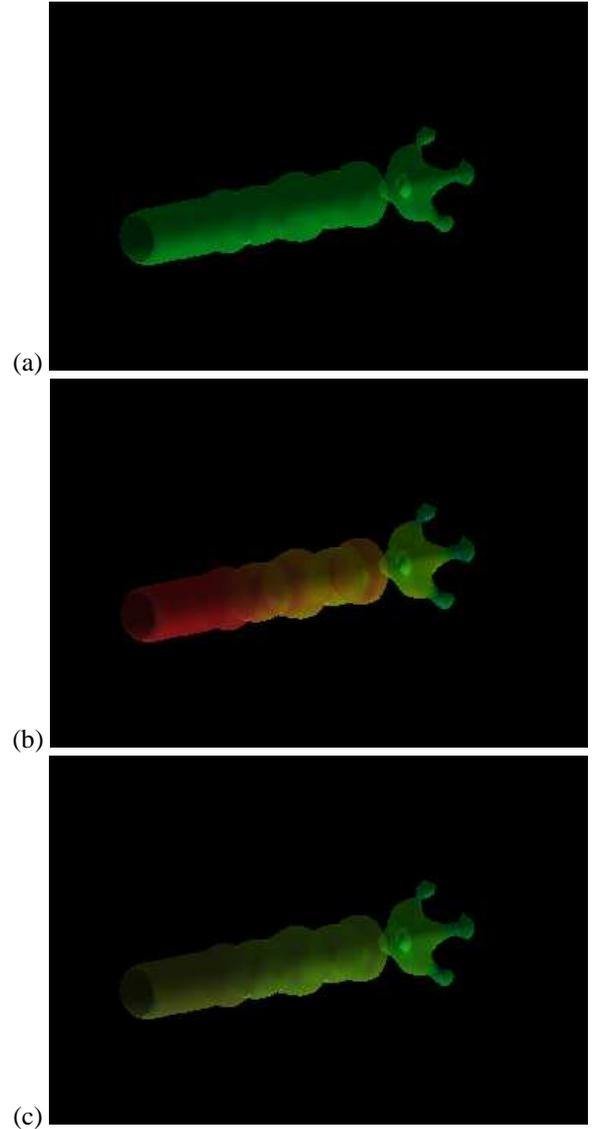


Figure 1: Single isosurface rendering applied to “fuel injection” data set: (a) standard isosurface; (b) material-enriched isosurface using color of material  $v_{near}$ ; (c) material-enriched isosurface using color blending.

Figure 1 shows a simulation data set of fuel injected into a combustion chamber. The higher the density

value is, the less air is present.<sup>1</sup> Figure 1(a) shows a standard isosurface, while Figures 1(b) and 1(c) show the rendering of material-enriched isosurfaces. In Figure 1(b), the assigned color is defined by a transfer-function look-up table for material with value  $v_{near}$ , i. e., material near the surface. In Figure 1(c), the assigned color is a blending of the colors used in Figures 1(a) and 1(b).

The visualizations with material-enriched isosurfaces exhibit more information in the data set. A single material-enriched isosurface suffices to understand that the density field has the steepest gradient perpendicular to the extracted isosurface where the fuel is injected (left side in the figures), which decreases smoothly with increasing distance.

In Figure 2, we provide another example for single transparent material-enriched isosurface rendering. The data set represents a simulation of a two-body probability distribution of a nucleon in the atomic nucleus  $^{16}\text{O}$ , where the position of a second nucleon is known.<sup>2</sup> The material-enriched isosurface renderings in Figures 2(b) and 2(c) exhibit a high-density region (reddish color) - an information that cannot be perceived from the standard isosurface visualization in Figure 2(a).

In Figure 3, we show multiple transparent isosurfaces extracted from the “fuel injection” data set. In Figure 3(a), the multiple isosurfaces are rendered with the same color and opacity. It is difficult to distinguish the different layers of material boundary. In Figure 3(b), the visualization of the different isosurfaces uses colors and opacities obtained from an assigned transfer function. This visualization leads to results similar to the ones shown in [Ger02]. The different material layers are easy to identify, but no volumetric impression as known from direct volume rendering techniques can be achieved. In Figure 3(c), we use material-enriched isosurfaces, where color is assigned with respect to near material values. A strong volumetric impression is achieved. The visualization with multiple transparent material-enriched isosurfaces can indeed approximate a visualization using direct volume rendering. Figure 3(d) shows a visualization with material-enriched isosurfaces, where the color of the “iso-material” is blended with the color of the near material. The volumetric impression is not as strong as in Figure 3(c), but the visualization is more realistic, as the colors are close to the colors in Figure 3(b). For comparison, we also show a direct volume-rendered image in Figure 3(e).<sup>3</sup>

---

<sup>1</sup>Data set courtesy of SFB 382 of the German Research Council (DFG).

<sup>2</sup>Data set courtesy of SFB 382 of the German Research Council (DFG).

<sup>3</sup>A visualization of this data set using a more sophis-

When comparing the multiple transparent material-enriched isosurface renderings with the direct volume rendering, we observe that the surfaces can approximate the direct volume rendering in terms of volumetric impression, but we also recognize differences in the resulting images. When using direct volume rendering, material boundaries are not as clearly visible. For example, the outer-most material boundary visible in the renderings in Figures 3(a)-(d) is hardly visible in Figure 3(e). This is probably due to the fact that this outer shell is a very thin layer of the chosen material. Even though we slightly “drift away” from our initial goal to approximate direct volume rendering as closely as possible, we consider the capability to add information about these thin outer layers as advantageous. On the other hand, we believe that these layers could also be visualized using direct volume rendering with appropriate higher-dimensional transfer functions and/or higher sampling rates.

Another example of multiple transparent isosurface rendering is given in Figure 4. The isosurfaces are extracted from the “nucleon” data set. Standard isosurfaces with uniform 4(a) or material color 4(b) are compared to material-enriched surfaces with colors for near material 4(c) or blended colors 4(d).<sup>4</sup> Although the geometry does not change, material-enhanced isosurfaces clearly can reveal the different rates of change around an isosurface. Thus, a more general transfer function can be implemented.

One disadvantage of multiple isosurfaces compared to direct volume rendering is their vulnerability to noise. In a volume rendering, noise appears as “dust” or “fog”, while in a multiple-isosurface method, noise manifest itself as jarring, jagged triangle groups. A noise reduction algorithm can solve this problem, but would also affect the data. We plan to explore whether a feature detection employed in a preprocessing step can take care of this problem instead.

We have tested our implementation on a standard PC with an Athlon 733 MHz processor. Our unoptimized prototype achieved a rendering performance of three to five frames per second. Our triangle sorting method is extremely fast, but the triangle drawing method could be improved. The rendering algorithm steps through all cells of the triangle-storing grid one at a time, regardless whether they are filled or not. In practice, this does not turn out to be a major issue; rendering time is dominated by the very high number of material shifts that occur when rendering multiple material-enhanced isosurfaces - three per triangle.

---

ticated direct volume rendering technique can be found at <http://www.volvis.org>.

<sup>4</sup>A visualization using direct volume rendering can be found at <http://www.volvis.org>.

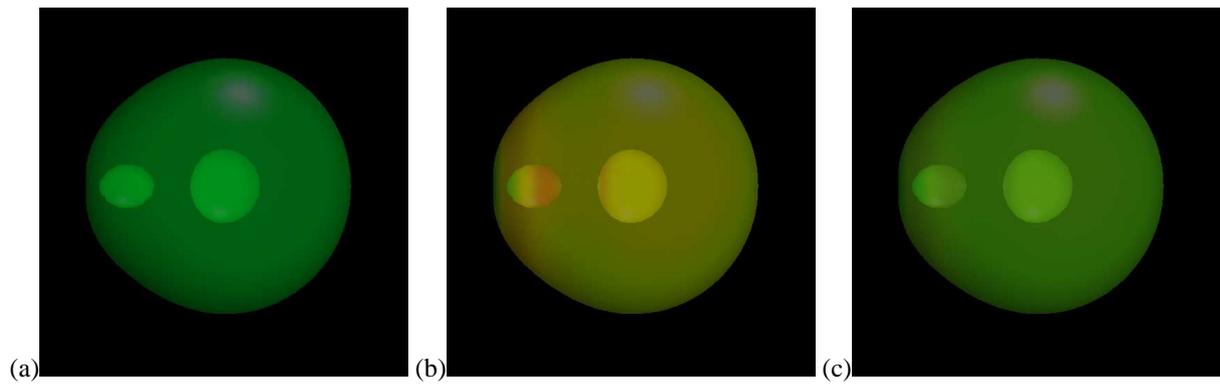


Figure 2: Single isosurface rendering applied to “nucleon” data set: (a) standard isosurface; (b) material-enriched isosurface using color of material  $v_{near}$ ; (c) material-enriched isosurface using color blending.

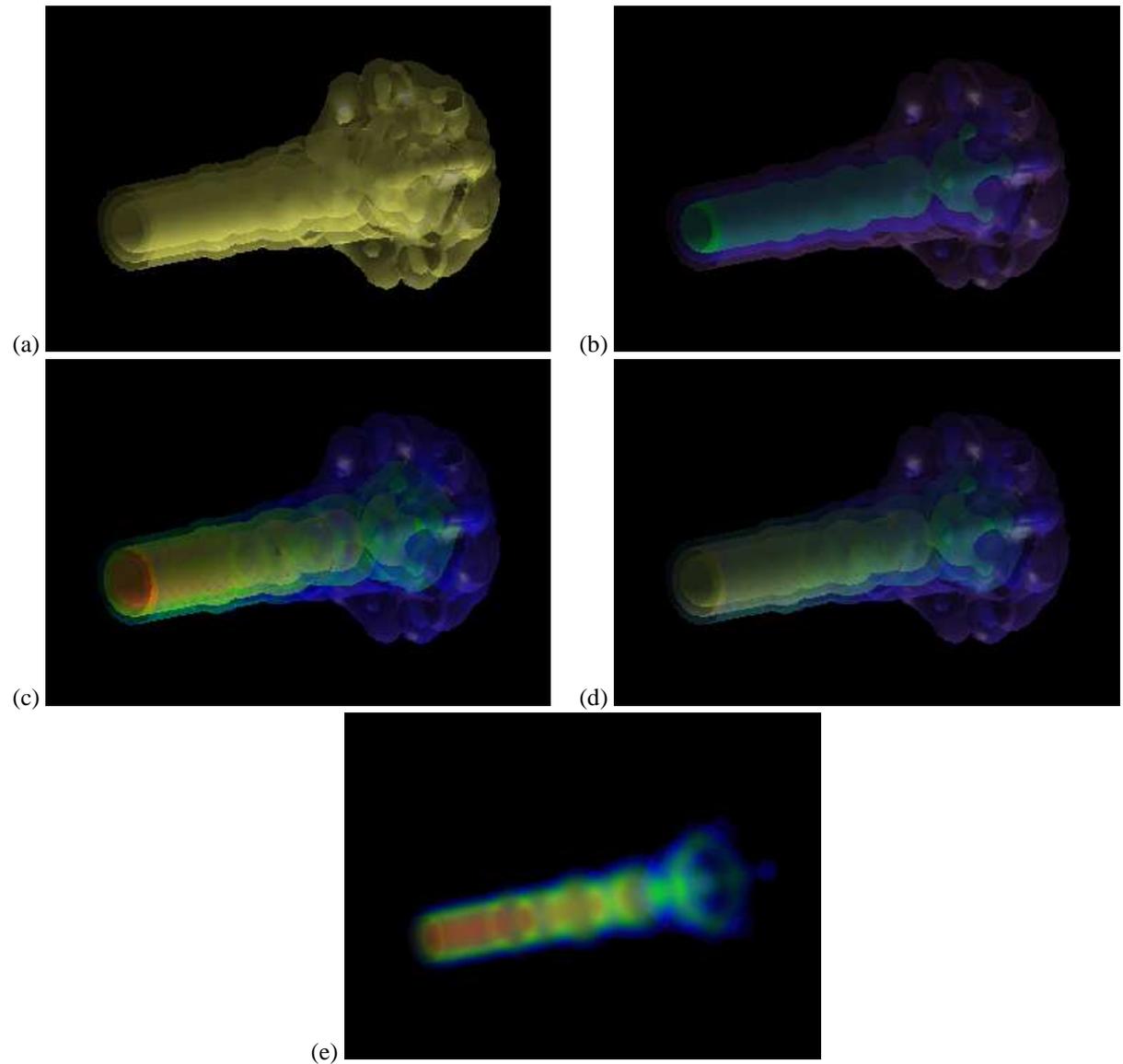


Figure 3: Multiple transparent isosurface rendering and direct volume rendering applied to “fuel injection” data set: (a) standard isosurfaces; (b) standard isosurfaces with material color; (c) material-enriched isosurfaces using color of material  $v_{near}$ ; (d) material-enriched isosurfaces using color blending; (e) direct volume rendering.

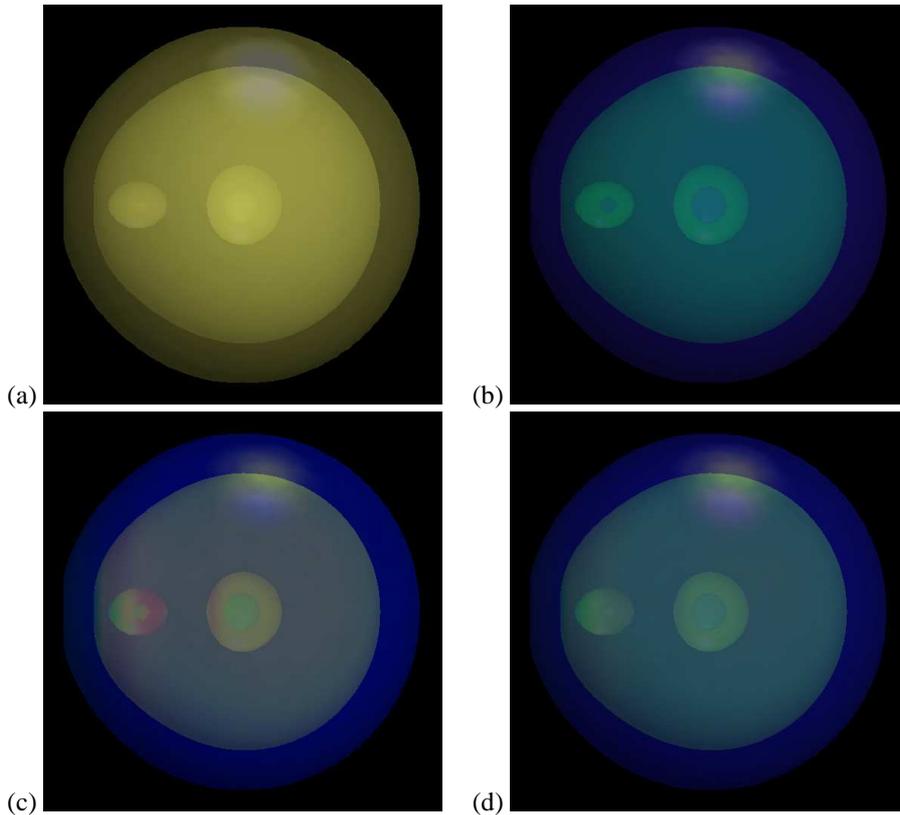


Figure 4: Multiple transparent isosurface rendering and direct volume rendering applied to “nucleon” data set: (a) standard isosurfaces; (b) standard isosurfaces with material color; (c) material-enriched isosurfaces using color of material  $v_{near}$ ; (d) material-enriched isosurfaces using color blending.

Still, there is room for improvement. A run-length encoding would allow large empty spaces to be skipped. After isosurface extraction, we would determine for each cell how far the next non-empty cell is. Then, during rendering, instead of stepping through the cells one-by-one, we would use the stored information to skip all empty cells. However, the run-length encoding would have to be done six times, storing how many cells to be skipped for each of the possible six drawing orders. OpenGL display lists could also work, but constructing six of them would be necessary to cover all possibilities. Alternatively, one could use a hierarchical volumetric data organization such as an octree data structure.

## 6 Conclusions and Future Work

We have presented an enhancement of isosurface rendering based on coloring isosurfaces with respect to material information at a short distance from the surface in surface normal direction. Already a single isosurface visualization can benefit from the enhancement. A single well-chosen material-enriched isosur-

face can provide a fairly good insight into the nature of the underlying scalar field. When using multiple transparent material-enriched isosurfaces, a volumetric impression of volume data can be achieved similar to a visualization using direct volume rendering.

Multiple transparent material-enriched isosurfaces diminish the drawback of standard isosurfaces, which do not represent any information of the volume data apart from a few selected material boundaries. In particular, standard isosurfaces do not capture any gradient information of a scalar field. Adding more and more isosurfaces would eventually lead to a banding effect. By visualizing material information close to the isosurfaces, material-enriched isosurfaces capture both material boundaries and gradient information, leading to a more complete visual depiction of the overall data set.

One idea to expand upon would be to weight the blending of the color of the “iso-material” and the color of the near material. We plan on basing the weight coefficients on the opacity values of the two materials obtained from the transfer function. Our approach should easily support such a change, which may produce an even better approximation to direct volume rendering

with arbitrary transfer functions.

While diminishing drawbacks of standard isosurfaces rendering, multiple transparent material-enriched isosurfaces still benefit from the advantages of isosurface rendering: Rendering is still fast (due to our fast sorting method), lighting is easy, fast and of high quality, and material boundaries are defined explicitly, if required, e. g., for quantitative analyses.

## Acknowledgments

This work was supported by the National Science Foundation under contract ACI 9624034 (CAREER Award), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, through the National Partnership for Advanced Computational Infrastructure (NPACI) and a large Information Technology Research (ITR) grant; the National Institutes of Health under contract P20 MH60975-06A2, funded by the National Institute of Mental Health and the National Science Foundation; and the U.S. Bureau of Reclamation. We thank the members of the Visualization and Computer Graphics Research Group at the Institute for Data Analysis and Visualization (IDAV) at the University of California, Davis.

## References

- [BL03] David Banks and Stephen Linton. Counting cases in marching cubes: Toward a generic algorithm for producing subtopes. In *Proceedings of IEEE Conference on Visualization 1998*. IEEE Computer Society Press, 2003.
- [CCF94] Brian Cabral, Nancy Cam, and Jim Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings of the 1994 symposium on Volume visualization*, pages 91–98. ACM Press, 1994.
- [Ger02] Thomas Gerstner. Multiresolution Extraction and Rendering of Transparent Isosurfaces. *Computers & Graphics*, 26(2):219–228, 2002.
- [GH95] André Guéziec and Robert Hummel. Exploiting triangulated surface extraction using tetrahedral decomposition. *IEEE Transaction on Visualization and Computer Graphics*, 1(4):328–342, 1995.
- [Ham91] Bernd Hamann. *Visualization and Modeling Contours of Trivariate Functions*. PhD thesis, Arizona State University, Tempe, Arizona, 1991.
- [LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques - SIGGRAPH 1987*, pages 163–169. ACM Press, 1987.
- [Lev88] Marc Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988.
- [LL94] Philippe Lacroute and Marc Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques - SIGGRAPH 1994*, pages 451–458. ACM Press, 1994.
- [Nie03] Gregory Nielson. MC\*: Star functions for marching cubes. In *Proceedings of IEEE Conference on Visualization 2003*. IEEE Computer Society Press, 2003.
- [Wes90] Lee Westover. Footprint evaluation for volume rendering. In Forest Baskett, editor, *Proceedings of the 17th annual conference on Computer graphics and interactive techniques - SIGGRAPH 1990*, pages 367–376. ACM Press, 1990.
- [WTTL96] Ming Wan, Long Tang, Zesheng Tang, and Xinyou Li. Pc-based quick algorithm for rendering semi-transparent multi-isosurfaces of volumetric data. In *Proceedings of the 1996 Conference on Computer Graphics International*, page 54. IEEE Computer Society, 1996.

# Self-Shadowing of Dynamic Scenes with Environment Maps using the GPU

Martin Knuth  
Fraunhofer IGD  
Fraunhoferstr. 5  
64283 Darmstadt, Germany  
mknuth@igd.fhg.de

Arnulph Fuhrmann  
Fraunhofer IGD  
Fraunhoferstr. 5  
64283 Darmstadt, Germany  
afuhr@igd.fhg.de

## ABSTRACT

In this paper we present a method for illuminating a dynamic scene with a high dynamic range environment map with real-time or interactive frame rates, taking into account self shadowing. Current techniques require static geometry (pre-computed light transport), are limited to few and small area lights or are limited in the frequency of the shadows. We facilitate importance sampling of the environment map and GPU based shadow calculation in an efficient way. The shadows are calculated per pixel, so no highly tessellated models are necessary in opposition to other techniques. Our method provides a novel and highly efficient way for using shadow maps as data structure for visibility computations done entirely on the GPU. We achieve real-time frame rates for moderate sized models on current graphics hardware. Since we evaluate the light transport of the scene per frame, complex dynamically animated models can be rendered efficiently.

**Keywords** Shadow Algorithms, Environment Mapping, GPU Programming

## 1. INTRODUCTION

Shadows reveal information about spatial object relation within a scene. Hence, using shadows in computer graphics allows a better immersion into a scene. Enhancing the quality and dynamics of the shadows will result in a more efficient comprehension of the image. For that task we present a system for rendering shadows caused by an environment map. The system evaluates the self shadowing of the scene at interactive or real time frame rates on current GPUs. The shadows are evaluated per pixel and are not limited to low frequencies. Furthermore, objects are allowed to be non-manifold. All this is done for fully dynamic scenes without prior knowledge of the animation.

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 2005 conference proceedings,  
ISBN 80-903100-7-9*

*WSCG'2005, January 31-February 4, 2005, Plzen, Czech Republic. Copyright UNION Agency - Science Press*

Existing systems used for rendering such scenes lit by an environment map are limited to vertex lighting, do not allow shadows or are not interactive.

The presented method makes use of importance sampling to create a light setup, which approximates the environment map. The light visibility is determined with shadow buffers. Since the rendering is entirely done on the GPU, no time expensive read backs of data towards the CPU is needed.

In section 2 we refer to existing work related to our approach. Then, we explain our algorithm and the ideas behind it. Additionally some issues are presented which have to be considered when implementing the algorithm on a GPU. In section 4 an implementation of the system is shown and discussed, followed by a summary and a look into the future.

## 2. RELATED WORK

There exists a lot of literature addressing the problems of shadowing 3D scenes in real time. In [Has03] several methods for generating soft shadows are compared. Unfortunately, they are either limited to small area sources or too slow to be useful for our approach. Rendering real time shadows of dynamic geometry on today's graphics hardware is mainly done by only two approaches: shadow volumes [Cro77] and shadow

maps [Wil78]. Both methods are capable of rendering shadows of directional- and point lights, creating hard shadow boundaries. Shadow maps are fast to compute and need less fill rate than shadow volumes. On the other hand, care has to be taken of sampling artefacts. An approach for minimizing shadow buffer artifacts can be found in [Ree87].

Direct illumination of a scene with an environment map is given by a group of environment mapping methods (like [Gre86, Ram01, Hei99]). They do not support shadows, which is a big disadvantage.

In [Deb98] Paul Debevec presents image based lighting done with high dynamic range (HDR) environment maps, to bring together synthetic and real scenes under natural lighting conditions.

Ray tracing [Wal03b, Wal03a, Pur02] is another approach for visibility determination. It has reached interactive and real time frame rates by using a PC-cluster, the GPU or dedicated ray tracing hardware. Unfortunately, there is still development needed until this hardware is out of prototype state. The software solution needs the power of an efficient cluster system. Ray tracing on the GPU at last needs fast read back towards the CPU, which is a bottleneck.

Global illumination at interactive frame rates can be done, utilising a real time raytracer. A direct approach towards global illumination on the GPU is presented in [Coo04], achieving interactive frame rates for small scenes. Another method to achieve fast global illumination computations was presented by Keller et. al in [Kel97]. The method accumulates images of the scene illuminated by single shadowcasting lights.

Other methods make use of occlusion or precomputed lighting information to allow the use of an environment map as light source. The information is mostly stored per vertex. Nevertheless a lot of additional information has to be stored. Since directions have to be mapped to data, structures like spherical harmonics are used. Since these methods are based on pre computations they normally can not be used for animated geometry. This flaw is faced by Kautz et al. in [Kau04]. They presented a method for speeding up the pre-calculation to interactive frame rates on small models. In opposition to our method they need a model hierarchy which can have high preprocessing cost - if animated. Due to its Spherical Harmonics/vertex based character, the method processes only low frequency shadows.

Many methods use directional- or point-lights as an approximation of the environment map. Our algorithm belongs to this kind of methods. For the pre-calculation of ambient occlusion NVIDIA [Pha04] presented an algorithm, which uses accumu-

lation of shadow maps in a preprocessing step to light the scene. As a disadvantage their method needs several seconds of preprocessing time for calculating the occlusion, with a reasonable visual quality.

Another approach for calculating ambient occlusion is presented in [Sat04]. It takes into account the colour of the environment map. Their method is mainly different in three points to our approach: Occlusion is evaluated per vertex, lights are generated with spherical distribution and at last occlusion data is read back from GPU.

Since the approximation of an environment map with directional lights is a difficult task, several methods addressing this problem exist. The easiest way is to sample the environment map homogeneously. But since most environment maps have varying areas of interest, it is advantageous to take the importance of these areas into account [Aga03, Kol03, Ost04, Sze04]. This is done by analysing the image to figure out more important areas of the image to place lights in.

### 3. OUR METHOD

#### Importance sampling

If we directly used the environment map for lighting a surface point, we would have to solve the problem of integrating over a hemisphere to get the amount of incoming radiance. Since the used environment maps are discrete, we could use a sum over  $n$  texels of the environment map for that task, but this would be still too much work to do.

Hence, we reduce the visibility problem to  $k$  directional lights, which are computed from the environment map. This is done by using structured importance sampling [Aga03]. This algorithm creates a distribution of the lights on the environment map according to the importance of the respective region. The importance of a region is determined by its extend and its light intensity. Roughly speaking, the method creates many lights in bright areas and only a few in darker ones, as can be seen in Figure 1. For more details on the importance metric and a derivation of it we refer the interested reader to the original paper [Aga03].

The computed point lights accumulate the radiance of their surrounding region. This method works considerably well, so a teapot scene inside Galileo's Tomb using only 300 lights rendered by Agarwal et al. shows no significant difference to a reference image computed with 100,000 samples using standard Monte Carlo sampling.

In difference to [Sat04] we use importance sampling in our system, since the rotation of lights is decou-

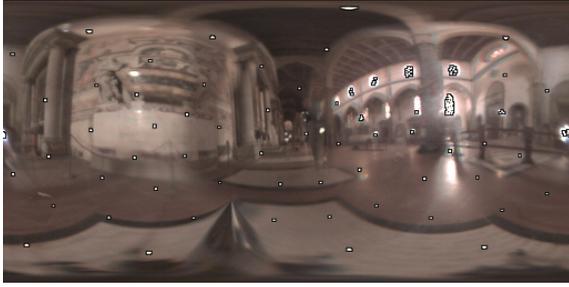


Figure 1: Environment map with 128 importance sampled lights shown as white dots.

pled from the rotation of the geometry. Rotation of the environment is followed by equal rotation of the lights. Although importance sampling of an environment map as described in [Aga03] takes some time, it either can be pre-computed and stored or done once at start up.

### Scene Rendering

Conventional real-time algorithms render a shadowed scene light after light in several passes. So, a shadow map is calculated for each light and used directly. Since all triangles have to be rasterized in each pass, this approach generates a lot of redundant calculations, which slows down the rendering. Our idea is to get rid of most of these redundant calculations, by doing something similar to ray tracing: Take a pixel and calculate all lighting for it, then take the next one. Calculations for a given pixel which are independent of the light position need to be done only once. The algorithm looks like this:

```

Calculate  $k$  lights from environment map.
for all Frames do
    Calculate  $k$  shadow maps.
    for all pixel do
        Compute visibility of  $k$  lights using the shadow maps.
    end for
end for

```

Algorithm 1: The shadowing algorithm.

Since rendering the scene taking all lights into account at once, a lot of redundant calculations are prevented. The rendering of the scene into the frame buffer becomes a single pass operation. But the storing and handling of the  $k$  shadow maps raises problems addressed in the next section.

### Shadow Map Management

Normally, a shadow map is used for one light at a time only. Since shadow maps are usually represented by

textures, this causes a lot of state changes. To minimize these state changes we render several shadow maps into one texture to fulfill the requirements set by our algorithm (See Figure 2).

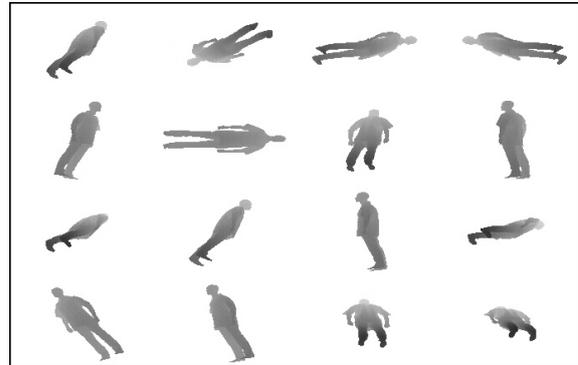


Figure 2: Texture containing several shadow maps, packed side by side.

This approach concentrates first completely on shadow map generation for all light sources and then on rendering the actual scene. Shadow map generation and scene rendering are completely separated.

### Further Reduction of the Number of Lights

The structured importance sampling reduces the number of lights necessary to approximate the lighting of an environment map. But, there are still too many lights needed to emulate soft shadows caused by lights on the environment map. Reducing the number of lights further will result in clearly distinguishable shadows with hard boundaries, due to under sampling (See Figure 3).

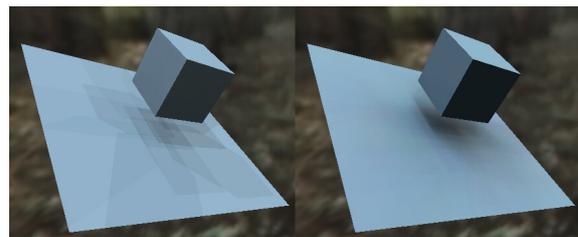


Figure 3: An example for under sampling the environment map: On the left 32 lights are used and the single shadows are clearly distinguishable. On the right figure 2048 light sources have been used.

These hard boundaries arise from sharp shadow edges of the individual light sources. More reduction needs a method to avoid these artefacts. The shadow maps are stored as plain depth information within a texture. So we can use simple 2D image manipulation functions to solve the problem by using a softening filter function.

This allows a blending of shadow boundaries. By this, a quality similar to images rendered with much more lights is achieved (See Figure 4).

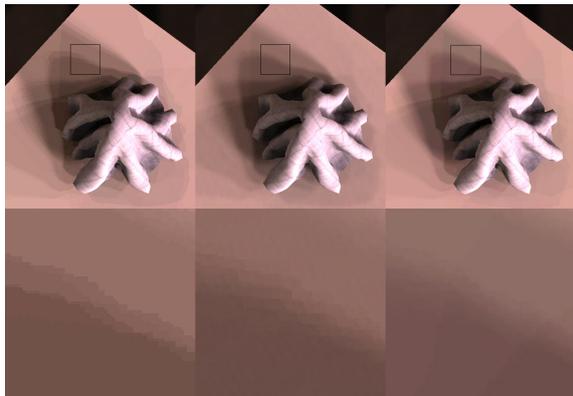


Figure 4: Using smoothing to avoid shadow artifacts: The left image is illuminated by 64 lights. Single shadows are well visible. The center image is illuminated by the same number of light sources, but with smoothed shadows. As reference, the right image is illuminated with 192 lights.

In [Aga03] jittering is used to reduce these artefacts. This is done by randomly choosing a light direction pointing inside the stratum of the light. This takes into account the distance of the occluder to the surface point: With increasing distance of the occluder the shadow gets more and more blurry.

In our approach a shadow map contains the visibility for a constant light direction. As a consequence we can only jitter the position within the shadow map. A shadow boundary will be equally thick regardless of the distance to the occluder. So our method cannot be interpreted as a quick alternative to soft shadow algorithms. Taken alone it just blurs a shadow boundary. The soft shadow effect is caused mainly by the large number of lights used.

### GPU-based visibility

Current GPUs behave like a dataflow machine. This has severe consequence when designing algorithms for GPUs. Changing the GPU state for example will stall the pipeline and if this happens often the overall performance decreases considerably. In our system we take care of this by clearly dividing shadow map calculation and scene rendering.

Unfortunately, it is necessary to split Algorithm 1 into several parts, since the GPU has a limited program length and not all lights can be computed in one pass. In order to be able to map our shadowing algorithm onto a programmable GPU, we modified Algorithm 1 into a multi-pass algorithm. The lights are packed into

clusters of size  $c$ . The size depends on the maximum number of lights supported by the fragment program of the GPU. The modified algorithm is shown in Algorithm 2 and can be implemented on current GPUs.

```

Calculate  $k$  lights from environment map.
for all Frames do
  for all Light clusters do
    Calculate  $c$  shadow maps.
    for all pixel do
      Compute visibility of  $c$  lights using the
      shadow maps.
    end for
  end for
end for

```

Algorithm 2: The modified shadowing algorithm.

Also, all data and intermediate data used should be stored within the GPU memory to prevent wait states. So, intermediate data should simply reside on the GPU. By using a GPU which is able to render into a texture the shadow maps fulfill this criteria. The geometry data can be stored inside the GPU memory for one frame of animation, since we are using a multi-pass operation this speeds up the algorithm.

## 4. IMPLEMENTATION AND DISCUSSION

We implemented our algorithm on a Radeon 9700 using OpenGL. The workload was divided between CPU and GPU. The CPU handles constants and the control flow of the algorithm. The vertex processor computes parameters which then can be interpolated over a triangle. The fragment shader does the per pixel work.

In order to map the algorithm efficiently to graphics hardware we used several extensions:

- `GL_ARB_vertex_program` for vertex program support.
- `GL_ARB_fragment_program` for fragment program support.
- `GL_ARB_vertex_buffer_object` for geometry storage inside GPU memory.
- `WGL_ARB_pbuffer` to be able to render to texture.

We have implemented shadow maps via the `PBuffer` extension of OpenGL. So using shadow map information is simply a texture lookup. As described above, we are trying to put as many shadow maps in the `PBuffer` as possible. Unfortunately, the `PBuffer` has a

maximum resolution which limits the number of containable shadow maps. But since textures are usually coloured there is another way to put more shadow maps inside one PBuffer. Every colour channel is used separately. This multiplies the capacity by four without decreasing the shadow map resolution (See figure 5). The shadow buffer information is interpreted in-

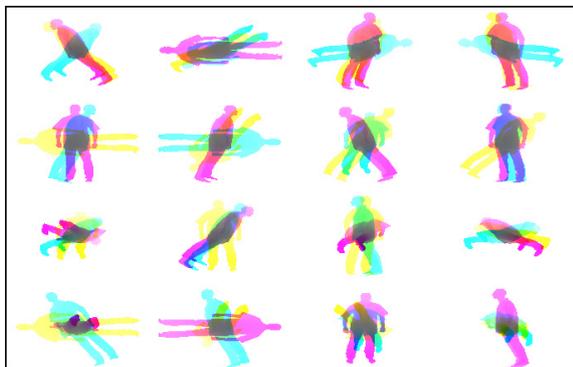


Figure 5: RGBA-texture as shadow buffer (alpha channel not shown): Additionally several buffers were packed side by side.

side the fragment program. Additional filtering (simple smoothing or percentage closer filtering ([Ree87])) is done here, too.

Storing the geometry data inside the GPU is mandatory, since we have a multi-pass algorithm. So the bus between CPU and GPU is free for control operations and is not a bottleneck.

### Vertex/Fragment Load Balancing

Load balancing is done by changing the number of lights calculated simultaneously. By calculating one light per pass, most work of the vertex program is done by transforming vertex coordinates. The fragment program has less work to do by handling one light. Nevertheless, it is more often called due to more passes are needed. Calculating several lights per pass increases the load inside the fragment program. Since fewer passes are needed the vertex program has to do less work.

### Results

In order to analyse the behaviour of the load balancing we implemented shaders for calculating shadows of one, four and eight lights simultaneously inside the fragment program. The shaders have shown a boost of frame rates as more lights were rendered per frame, since the number of passes decreases. All methods for reducing fill rate and vertex count have shown direct consequences towards higher frame rates. The implementation also has shown that careful detection of

bottlenecks and several exploitations of redundancies created a system, able to reach real time frame rates for moderate polygon count models. In practical use, bottlenecks tend to wander between fragment program and vertex program, dependent on the amount of objects covering the screen.

The evaluation of the shadows within the fragment program allows user defined filtering functions. Observing the visual results of our system, shadow smoothing is not always necessary to be convincing. It depends on the environment map and the number of lights used.

## 5. CONCLUSIONS

We presented a method for self-shadowing of dynamic scenes with environment maps using the GPU. Our algorithm allows the creation of interactive systems, which are capable of rendering scenes taking into account self shadowing caused by an environment map. We evaluate the lighting condition of the geometry on the fly by using current graphics hardware and their shadow mapping features. Our algorithm achieves interactive frame rates for large dynamic models, without prior knowledge of the animation. The implementation is flexible enough to allow an easy load balancing between vertex and fragment program, by controlling the number of lights rendered per pass. In order to raise the visual quality of the shadows we use smoothing and percentage closer filtering.

The implementation of the algorithm has shown it's ability to achieve real-time frame rates for models with moderate polygon count with plausible looking self shadowing of the scene, realistically illuminated by the HDR environment map.

## 6. FUTURE WORK

One direction for future research would be to consider more information about the light source. The directional lights created by importance sampling describe actually areas of the environment map and not just a singular point. If we took the shape and size of the light source into account during the visibility computations, it would be possible to reduce the number of light sources needed for realistic images even further.

The rendered images would reach a next grade towards photo realism if inter-reflections are taken into account. Since this requires visibility calculation between faces of the geometry, inter-reflections are not handled by our scheme yet. It will take several generations of GPUs and further algorithmic improvements until this vision will be reality.

## 7. ACKNOWLEDGEMENTS

The high dynamic range environment maps used in this paper were made by Paul Debevec [Deb98]. Textiles shown were created with the prepositioning and cloth simulation methods described in [Fuh03b, Fuh03a, Gro03].

## 8. REFERENCES

- [Aga03] Agarwal, S., Ramamoorthi, R., Belongie, S., and Jensen, H. W. (2003). Structured importance sampling of environment maps. *ACM Trans. Graph.*, 22(3):605–612.
- [Coo04] Coombe, G., Harris, M. J., and Lastra, A. (2004). Radiosity on graphics hardware. In *GI '04: Proceedings of the 2004 conference on Graphics interface*, pages 161–168. Canadian Human-Computer Communications Society.
- [Cro77] Crow, F. (1977). Shadow algorithms for computer graphics. *J-COMPGRAPHICS*, 11(2):242–248.
- [Deb98] Debevec, P. (1998). Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 189–198. ACM Press.
- [Fuh03a] Fuhrmann, A., Gross, C., and Luckas, V. (2003a). Interactive animation of cloth including self collision detection. *Journal of WSCG*, 11(1):141–148.
- [Fuh03b] Fuhrmann, A., Gross, C., Luckas, V., and Weber, A. (2003b). Interaction-free dressing of virtual humans. *Computers & Graphics*, 27(1):71–82.
- [Gre86] Greene, N. (1986). Environment mapping and other applications of world projections. *IEEE Comput. Graph. Appl.*, 6(11):21–29.
- [Gro03] Gross, C., Fuhrmann, A., and Luckas, V. (2003). Automatic pre-positioning of virtual clothing. In *Proceedings of the Spring Conference on Computer Graphics*, pages 113–122.
- [Has03] Hasenfratz, J.-M., Lapierre, M., Holzschuch, N., and Sillion, F. (2003). A survey of real-time soft shadows algorithms. In *Eurographics*. Eurographics, Eurographics. State-of-the-Art Report.
- [Hei99] Heidrich, W. and Seidel, H.-P. (1999). Realistic, hardware-accelerated shading and lighting. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 171–178. ACM Press/Addison-Wesley Publishing Co.
- [Kau04] Kautz, J., Lehtinen, J., and Aila, T. (2004). Hemispherical rasterization for self-shadowing of dynamic objects. In *Proceedings Eurographics Symposium on Rendering 2004*.
- [Kel97] Keller, A. (1997). Instant radiosity. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 49–56. ACM Press/Addison-Wesley Publishing Co.
- [Kol03] Kollig, T. and Keller, A. (2003). Efficient illumination by high dynamic range images. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 45–50. Eurographics Association.
- [Ost04] Ostromoukhov, V., Donohue, C., and Jodoin, P.-M. (2004). Fast hierarchical importance sampling with blue noise properties. *ACM Transactions on Graphics*, 23(3):488–495. Proc. SIGGRAPH 2004.
- [Pha04] Pharr, M. (2004). Ambient occlusion. *Game Developers Conference (GDC) 2004*.
- [Pur02] Purcell, T. J., Buck, I., Mark, W. R., and Hanrahan, P. (2002). Ray tracing on programmable graphics hardware. *ACM Trans. Graph.*, 21(3):703–712.
- [Ram01] Ramamoorthi, R. and Hanrahan, P. (2001). An efficient representation for irradiance environment maps. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 497–500. ACM Press.
- [Ree87] Reeves, W. T., Salesin, D. H., and Cook, R. L. (1987). Rendering antialiased shadows with depth maps. *SIGGRAPH Comput. Graph.*, 21(4):283–291.
- [Sat04] Sattler, M., Sarlette, R., Zachmann, G., and Klein, R. (2004). Hardware-accelerated ambient occlusion computation. In Girod, B., Magnor, M., and Seidel, H.-P., editors, *Vision, Modeling, and Visualization 2004*, pages 331–338. Akademische Verlagsgesellschaft Aka GmbH, Berlin.
- [Sze04] Szecsi, L., Sbert, M., and Szirmay-Kalos, L. (2004). Combined correlated and importance sampling in direct light source computation and environment mapping. *Computer Graphics Forum (Eurographics 04)*, 23(3).
- [Wal03a] Wald, I., Benthin, C., and Slusallek, P. (2003a). Interactive global illumination in complex and highly occluded environments. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 74–81. Eurographics Association.
- [Wal03b] Wald, I., Purcell, T. J., Schmittler, J., Benthin, C., and Slusallek, P. (2003b). Realtime ray tracing and its use for interactive global illumination. In *Eurographics State of the Art Reports*.
- [Wil78] Williams, L. (1978). Casting curved shadows on curved surfaces. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 270–274. ACM Press.



Figure 6: The left image was rendered with standard OpenGL. The center image was illuminated by an irradiance map. The right image was rendered with our algorithm taking self-shadowing into account. The resolution was 421x711 pixel and 128 smoothed shadows were used. We achieved four frames per second. The model consists of 107K triangles.

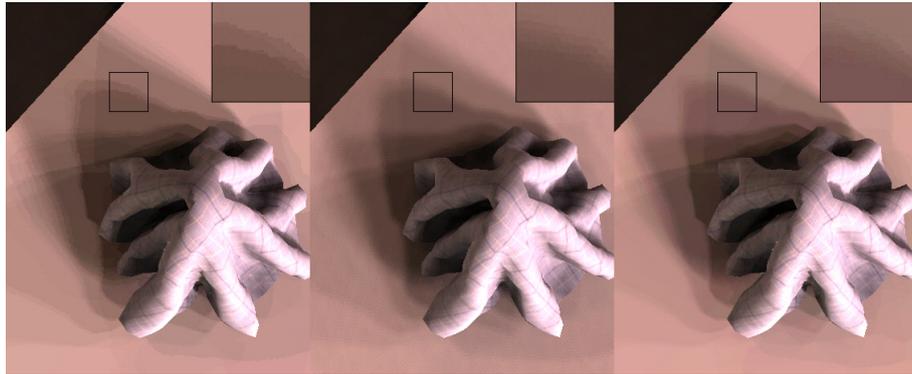


Figure 7: The left image was rendered with 64 light sources and 8 lights per pass at 21 FPS. The middle image was rendered with 64 light sources, shadow smoothing and 4 lights per pass at 8 FPS. The right image was rendered with 192 light sources and 8 lights per pass at 6 FPS.

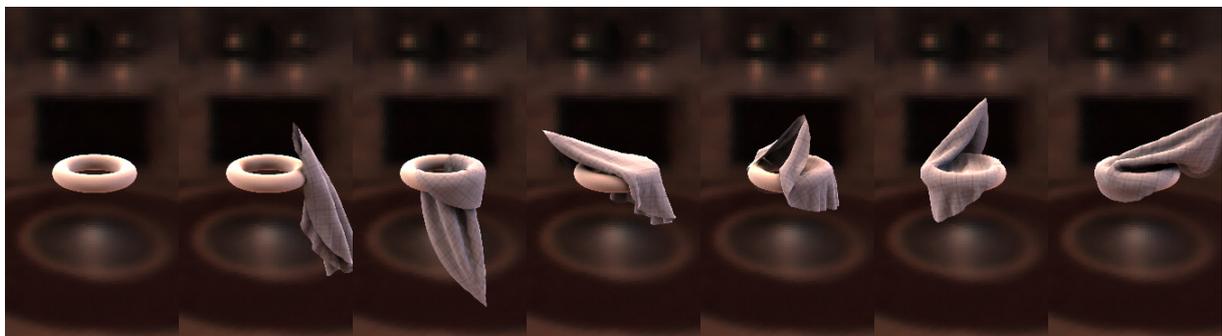


Figure 8: Some sample frames taken from a real-time animation and rendering of cloth.



# Paving Procedural Roads with Pixel Shaders

Jörn Loviscach

Hochschule Bremen  
Flughafenallee 10  
28199 Bremen, Germany

jlovisca@informatik.hs-bremen.de

## ABSTRACT

Modern graphics hardware can be used to create procedural geometry. Our proposal details an optimized method to form roads and similar 3D objects by cookie-cutting them from slightly oversized polygons. The roads follow spline-like curves on a plane. The curves and their offset variants are cast into an approximated, implicit description. This can efficiently be evaluated within a pixel shader to discard pixels that are part of the oversized polygons but not part of the roads. Our method guarantees smooth geometry and smooth texturing. To achieve comparable results with roads formed from polygons in the usual way requires level-of-detail or similar mechanisms which not only complicate development and scene management, but also add load on the CPU.

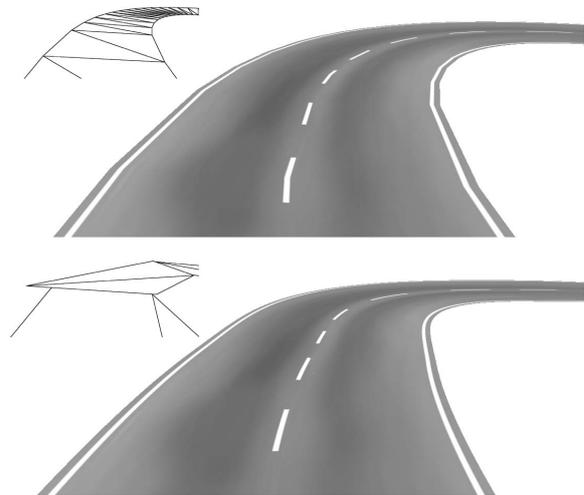
## Keywords

driving simulator, implicit curve, offset curve, pixel shader, clipping

## 1 INTRODUCTION

Roads are a prominent feature of virtual reality and gaming applications such as driving simulators. Many roads follow curved paths, in particular circles and spirals [AAS01], which are rendered with a large number of polygons. If this is not done, both the lateral borders of the roads and their textures such as medians show objectionable angles, see Figure 1.

Typical applications use large numbers of roads. To prevent a serious drop in the frame rate, these may not be rendered with a high polygons count. Thus, the number of polygons used has to be reduced for less visible or invisible roads or parts of them. This not only leads to additional development effort but also requires visibility estimation and a more sophisticated scene management to be done on the CPU.



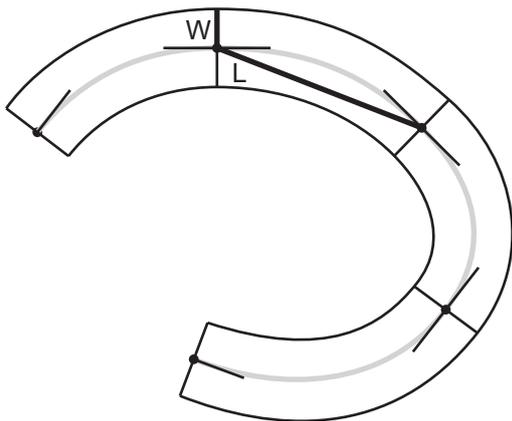
**Figure 1. A conventionally built road shows angular artifacts (upper image). Our method yields smooth shapes and textures (lower image). The insets show the polygons used.**

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 no copies are made or distributed for profit or commercial advantage and that all copies bear this notice and the full citation on the first page. To otherwise copy or republish, to post on servers or to redistribute to lists, a prior specific permission and/or a fee are required.

*Conference Proceedings ISBN 80-903100-7-9  
WSCG '2005, January 31-February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency – Science Press*

The main contribution of this paper is an efficient implicit description of fat planar curves: a centerline plus a family of offset curves. This implicit description is used to cookie-cut roads from large polygons using a pixel shader. These roads possess tangent continuous shape and texturing.

The input to our method is a set of anchor points—each equipped with a tangent direction—that represents the centerline (mostly marked by a median) of the road. Every two consecutive anchor points determine a road segment, which is to be treated separately. We assume that each segment lies in a plane. This is at least approximately valid for roads with slowly varying grade. Furthermore, we assume that no segment is strongly bent horizontally, so that its centerline can be parameterized using the projection onto the straight line that connects the two anchor points, see Figure 2.

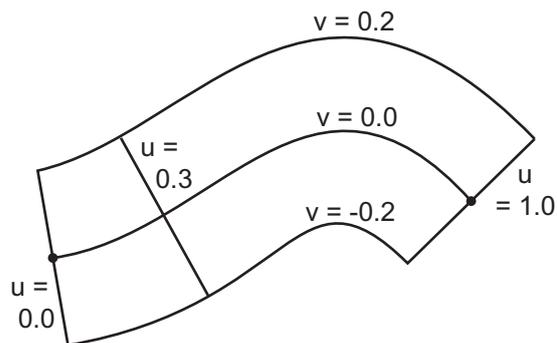


**Figure 2. A road is divided into segments defined by start and end points on its centerline together with tangent directions.**

Every road segment is covered with a quadrangle (to be rendered as two triangles under DirectX) computed from the road’s width and the spline-like curve that forms the central line. A pixel shader is used to discard the pixels of the quadrangle that do not belong to the road. To this end, the `clip` instruction of the HLSL shading language is used. It translates to the `texkill` instruction of DirectX pixel shader assembler. Furthermore, the pixel shader assigns texture coordinates to the pixels. A mapping  $\mathbf{x} \mapsto (u, v)$  is employed that ensures smoothness along every single road segment as well as tangent continuity at the transition from one segment to the next, see Figure 3.

All computations are offloaded from the CPU to the graphics hardware, excluding a short initialization routine to build vertex and index buffers. For optimization, the computation of all quantities that vary linearly is moved from the pixel shader into the vertex shader of the same (and only) rendering pass.

The proposed method does a substantial amount of work in the pixel shader. Roads close to the viewer incur a high computational cost, but are perfectly free from angular-looking defects. Distant roads, however, lead to only a small computational cost because they



**Figure 3. World space coordinates are converted to coordinates  $(u, v)$  along and across the road. These serve to define both geometry and texturing.**

consist of few pixels in screen space. A large percentage of invisible segments of the road will be discarded already at the vertex level through frustum clipping. Only a low number of polygons is needed to construct the roads using the shader. Thus, the efficiency for distant roads is close to that of level-of-detail or frustum culling approaches.

This paper is structured as follows: In Section 2 we outline related work; Section 3 describes the formulation of roads as fat curves. How these can be evaluated using a graphics chip is covered in Section 4. Section 5 presents and discusses results; Section 6 gives a summary and points out directions for further research.

## 2 RELATED WORK

Vertex-based procedural creation of geometry on graphics cards has been studied much in recent years. For instance, it can be found in the curved-PN-triangle subdivision offered by current ATI graphics cards [Vla01]. Bolz and Schröder [Bol03] propose a vertex-based method to evaluate subdivision surfaces.

Due to increasing computing power and improved functionality, pixel-based instead of vertex-based procedural creation of geometry is now becoming a viable option. Some works have already addressed this topic.

Hirche et al. [Hir04] render per-pixel displacement maps on the graphics chip. To this end, they extrude prisms from the triangles of a mesh. They render these prisms with a complex pixel shader, which employs ray casting to evaluate the displacement map using four samples per ray. Kanai and Yasui [Kan04] evaluate per-pixel positions and normals of subdivision surfaces in a pixel shader and use the results to fill a vertex buffer to render the surface from. Lovisich [Lov04] uses curved fins along the silhouette of a mesh to smooth the outline visually. The fins are painted by a pixel shader onto quadrangles that are ex-

truded from the silhouette of the mesh inside a vertex shader.

Rose and Ertl [Ros03] draw wire frames onto simplified polyhedra. ATI’s demo “Ruby: The Double Cross” [ATI04] employs pixel shaders to procedurally generate the ATI logo from lines and circles. This method does not actually produce geometry, but comes close in spirit.

In order to construct roads with a pixel shader, we use a parameterization that is related to offset curves:  $v = 0$  is the centerline of the road; a non-zero  $v$  leads to an offset curve. Offset curves are a classic topic of computer graphics; for surveys see [Elb97] and [Mae99].

Most of the work done on offset curves is concerned with explicit representations. In contrast to that, we are interested in the inverse mapping from world space to parameter space, which may be compared to an implicit representation of offset curves. This can for instance be achieved with the distance function that maps every point to its distance to the original curve, a mapping that can be used, for instance, to find the medial axis transform.

Pottmann et al. [Pot02] study local quadratic approximations of the squared distance to a curve in the Frenet frame. They employ this approximation to generate offset curves with active splines. However, it does not seem straightforward to use these results here, in particular due to the non-global nature of the approximation.

### 3 ROADS AS FAT CURVES

For simplicity we only show the construction for the 2D case in which the central line starts at the origin  $(x,y) = (0,0)$  and ends at  $(1,0)$ , see Figure 4. All other cases can be reduced to this by rotation and uniform scaling. Let the tangent direction at the origin be parallel to  $(1,a)^T$  and that at the end be parallel to  $(1,b)^T$ .

Then we can construct the centerline as the graph of a function  $y = f(x)$  with the following properties:  $f(0) = 0 = f(1)$ ,  $f'(0) = a$ , and  $f'(1) = b$ . We choose  $f$  to be the cubic function that fulfills those requirements:

$$f(x) := x(1-x)^2a - x^2(1-x)b$$

Given a point  $(x,y)$  near the curve, we want to find approximate values for the nearest position on the centerline (parametrized by  $x$ ) and the signed distance from the centerline. Call these two values  $(u,v)$ . The point  $(x,y)$  is a point on the road if and only if  $0 \leq u \leq 1$  and  $-W/L \leq v \leq W/L$ , where  $W$  is half the road’s width

and  $L$  the distance between its anchor points (before scaling). Furthermore,  $(u,v)$  serve as curved texture coordinates on the road.

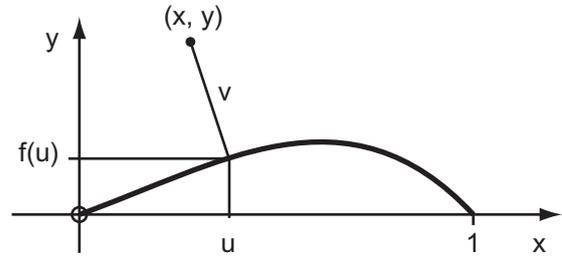
We assume that  $a$ ,  $b$ , and  $y$  are close to zero so that there are no problems concerning the uniqueness of a nearest point on the curve. In Section 5 we will show how large these values may be chosen in practice.

To convert  $(x,y)$  to  $(u,v)$ , we employ a basic idea from the theory of offset curves [Elb97]: The vector from  $(x,y)$  to the nearest point on the central line has to be perpendicular to a tangent vector to the curve, see Figure 4:

$$\begin{pmatrix} x-u \\ y-f(u) \end{pmatrix} \cdot \begin{pmatrix} 1 \\ f'(u) \end{pmatrix} = 0$$

This leads to

$$x - u + (y - f(u))f'(u) = 0. \quad (1)$$



**Figure 4.**  $(x,y)$  is converted to  $(u,v)$  using the nearest point  $(u, f(u))$  on the curve.

We are not going to solve Equation 1 (which in general is of degree five) precisely, but rather use it as a guidance to construct an approximately equal object with precisely identical properties regarding start point, end point, initial and final direction.

Note that  $f(u) \approx 0$  for a curve which is only weakly bent. Furthermore,  $f'$  can be computed using  $a$  and  $b$ . This yields an approximation  $u_1$  of  $u$  in Equation 1:

$$x - u_1 + y((3u_1^2 - 4u_1 + 1)a + (3u_1^2 - 2u_1)b) = 0 \quad (2)$$

This equation typically possesses two different solutions in  $u_1$ . We pick the solution close to  $x$ . This solution is guaranteed to exist for  $y$ ,  $a$ , and  $b$  sufficiently close to zero. It can be written

$$u_1 = \frac{2\gamma}{\sqrt{\beta^2 - 4\alpha\gamma} - \beta}, \quad (3)$$

where

$$\alpha := 3y(a+b), \quad \beta := -1 - (4a+2b)y, \quad \gamma := x + ya. \quad (4)$$

We write the solution of the quadratic equation in the untypical form of Equation 3 to prevent a division by

zero when  $y = 0$  and hence  $\alpha = 0$ . Note that  $\beta < 0$  if the bending is weak and the width is small enough.

The points  $(x, y)$  with  $u_1 = 0$  form the line through the start point  $(0, 0)$  perpendicular to the tangent of the curve at that point: From  $u_1 = 0$  follows  $x = -ay$ . Similarly, the points  $(x, y)$ , for which  $u_1 = 1$ , form a line through the end point  $(1, 0)$  perpendicular to the tangent of the curve there.

A simpler approximation with the same properties would be

$$u_2 = \frac{x + ya}{1 + (a - b)y}.$$

However, it turns out that this approximation—concerning its overall shape—does not perform well for strongly curved paths.

Now  $v$  remains to be computed. If we had solved Equation 1 precisely, the signed distance  $v$  could be found through the dot product of a normalized vector perpendicular to the curve and the difference vector between the point  $(x, y)$  on the plane and the nearest point  $(u, f(u))$ , see Figure 4:

$$v = \frac{\begin{pmatrix} x - u \\ y - f(u) \end{pmatrix} \cdot \begin{pmatrix} -f'(u) \\ 1 \end{pmatrix}}{\sqrt{f'(u)^2 + 1}} \quad (5)$$

Due to the dot product, this equation is robust under a small shift along the curve. Thus, it seems reasonable to use this equation in our framework with  $u_1$  in place of  $u$ . This completes an efficient algorithm to convert a point  $(x, y)$  near the curve to curved coordinates  $(u_1, v)$ .

Whereas the mapping  $(x, y) \mapsto (u_1, v)$  is only approximate, it possesses the same features as the exact solution of Equation 1: On the lines  $u_1 = 0$  and  $u_1 = 1$  the mapping equals the exact solution, what is crucial for the continuous transition from one road segment to the next.

On top of that, the transition from one segment to the next is not only continuous, but also *tangent* continuous. To prove tangent continuity, one can compute the gradient of  $v$  with respect to  $x$  and  $y$  at  $u_1 = 0$  and  $u_1 = 1$  for arbitrary  $v$  using basic mathematics. It turns out that the gradient equals  $(-a, 1)^T / \sqrt{a^2 + 1}$  and  $(-b, 1)^T / \sqrt{b^2 + 1}$ , respectively. All lines  $v = \text{const}$  must run perpendicular to the gradient field. Therefore, they have a slope of  $a$  and  $b$ , respectively, at  $u_1 = 0$  and  $u_1 = 1$ , what proves tangent continuity. This property of the construction does neither depend on the details of  $f$  nor on the approximation used to find  $u_1$ , as long as  $f(0) = 0 = f(1)$ ,  $f'(0) = a$ ,  $f'(1) = b$ ,  $u_1 = 0$  corresponds to  $x = -ay$ , and  $u_1 = 1$  corresponds to  $x = 1 - by$ .

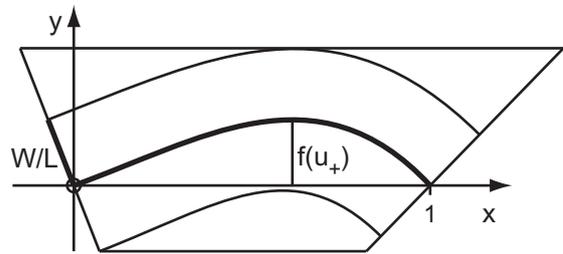
## 4 HARDWARE ACCELERATION

In the prototype, we have implemented the method using the following steps:

- Preprocessing:
  - Given a sequence of anchor points along the median of the road to be built, compute the initial and final slopes  $a$  and  $b$  for every segment using a Catmull-Rom spline that interpolates the anchor points. (The slope could also be defined arbitrarily.)
  - Create vertex and index buffers that describe one quadrangle per road segment.
- For every frame:
  - Draw the terrain.
  - Draw the quadrangles defined in the preprocessing step. Use shaders on them both to form the road through pixel clipping and to map a texture onto it.

Each quadrangle is chosen such that it covers the corresponding road segment completely with not much excess, see Figure 5. To create quadrangles with minimum area reduces rendering time. This construction employs T-junctions (see inset in the lower part of Figure 1), which may be objectionable in other circumstances. For a discussion see Section 5.

In order that the segments fit together, two of the sides of a quadrangle have to run through the start and end points, respectively, of the centerline, perpendicular to the corresponding tangents, see Figure 5. The two other sides of the quadrangle are parallel to the straight line  $y = 0$  that connects the start and the end point of the centerline. To position these two sides, we look for extremal values of  $f$  on  $[0, 1]$  by solving the equation  $f'(u) = 0$ , which in general is quadratic. If, for instance, there is a maximum at  $u = u_+$ , the upper side has to be shifted upward to  $y = f(u_+) + W/L$ .



**Figure 5. Every road segment is covered by a quadrangle.**

Note that this shape saves the test whether  $0 \leq u_1 \leq 1$ , because this is automatically true for any point on the

quadrangle: Along one of its sides  $u_1$  equals 0, along one other side it equals 1. Thus, only  $-W/L \leq v \leq W/L$  remains to be checked to determine if a point lies on the road.

For each of the four vertices of such a quadrangle we store the following attributes in the vertex buffer:

- 3D position
- $xy$  position in the rotated and scaled system according to Section 3
- $a, b$ , and the distance  $L$  between the anchor points
- $u_{\text{Start}}, u_{\text{Range}}$

The distance  $L$  is needed to adapt the road width (which is transmitted as a constant parameter), because the  $xy$  position is scaled to  $x \in [0, 1]$ . The values  $u_{\text{Start}}$  and  $u_{\text{Range}}$  are used to shift and rescale  $u$ , the texture coordinate along the road. Both are determined from arc length such that the textures of the road segments fit together seamlessly.

The data  $a, b, L, u_{\text{Start}}$ , and  $u_{\text{Range}}$  are identical for all vertices of one quadrangle. Therefore, we use the Vertex Stream Frequency Divider offered by DirectX 9.0 to save memory bandwidth for this subset of the vertex attributes.

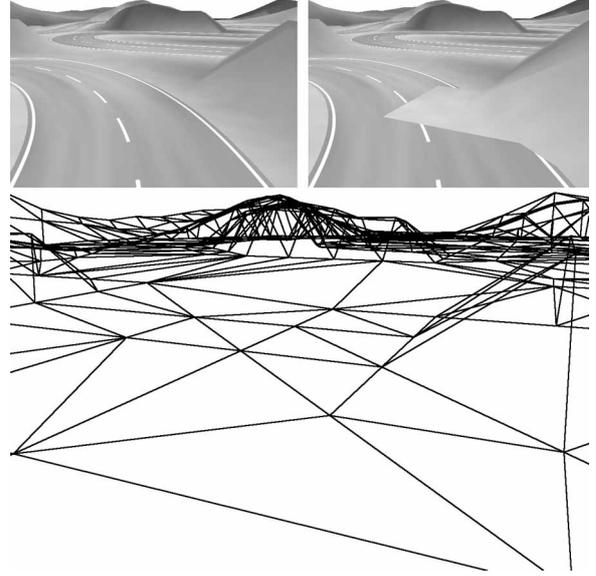
The value  $\alpha, \beta$ , and  $\gamma$  of Equation 4 depend linearly on the position of a pixel inside a triangle. For efficiency, we use a vertex shader to compute them per vertex, and rely on the automatic linear interpolation applied by the graphics chip to all values that are transmitted from the vertex shader to the pixel shader. The pixel shader then evaluates Equations 3 and 5.

For the implementation we chose Managed DirectX 9.0c using the language C# and Microsoft's Effect framework with HLSL. The vertex and the pixel shader compile to 23 and 35 instructions, respectively, of Shader Model 2.0. All computations are done using 16 bit floating point precision instead of the regular 32 bit floating point precision without visually objectionable roundoff errors.

In a typical virtual reality or gaming setting, the geometry of buildings and terrains can be much more angular than that of roads: Most buildings possess rectangular forms by construction; terrains can be covered with complex textures that help to hide large polygons. Smooth roads may, however, not be combined with a coarsely-tessellated terrain in a straightforward manner: The roads would be cut off in angular patterns.

To prevent this, the terrain in the vicinity of the road is composed of large level polygons, see Figure 6. Another possibility would be to introduce ditches. We

render the terrain before the roads and leave a visually unnoticeable small height gap between the road and the terrain below to prevent z-fighting.



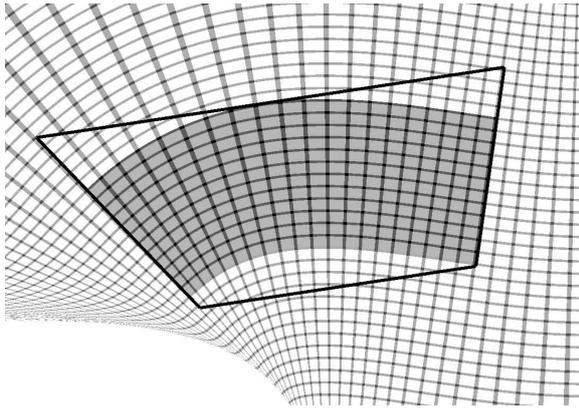
**Figure 6. Intersections between the roads and the terrain may reveal the coarse tessellation of the latter (top right). Hence, we create level geometry along the roads (bottom).**

## 5 RESULTS. DISCUSSION

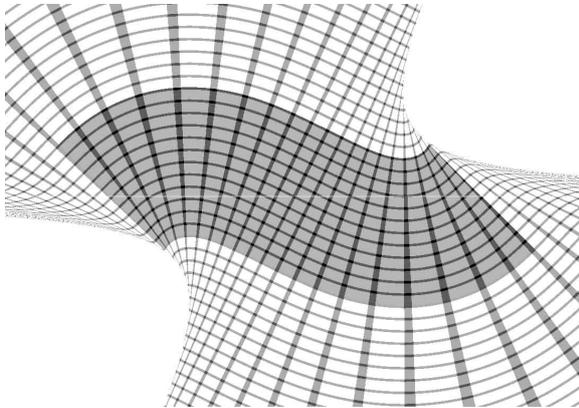
Even for relatively large values of  $a, b$ , and  $W/L$  the approximate mapping  $(x, y) \mapsto (u_1, v)$  yields useful results, and the quadrangle fits closely, see Figure 7. In our experiments, we found no visually objectionable deviations as long as  $|a| \leq 1$  and  $|b| \leq 1$ , and furthermore  $|W/L| \leq 0.3$  if  $a$  and  $b$  are of different sign and  $|W/L| \leq 0.2$  if they are of same sign. This range allows strongly curved segments, see Figures 8 and 9.

In situations with strong bending such as that of Figure 9 the viewer may realize that the  $u_1$  coordinate used for texturing deviates from arc length parameterization. This difference could be diminished through a corrective term. With typical road textures, however, this is not necessary.

To fit the quadrangles tightly around the road, we employ geometry with T-junctions. Thus, roundoff may lead to pixel-wide gaps between two quadrangles. However, in our experiments such defects did not turn up. One may also argue that the number of vertices could be cut by half by joining every two neighboring vertices on each side along the road. But this would enlarge the area of the quadrangles and thus lead to more invocations of the pixel shader. Furthermore, as described in Section 4, every vertex contains the values of  $x$  and  $y$  in its local coordinate frame. A shared



**Figure 7.** The parameterization of a road segment ( $a = 0.7$ ,  $b = -0.3$ ,  $W/L = 0.2$ ) and the quadrangle used for rendering show that the approximation in Eq. 2 does not lead to easily recognizable errors.



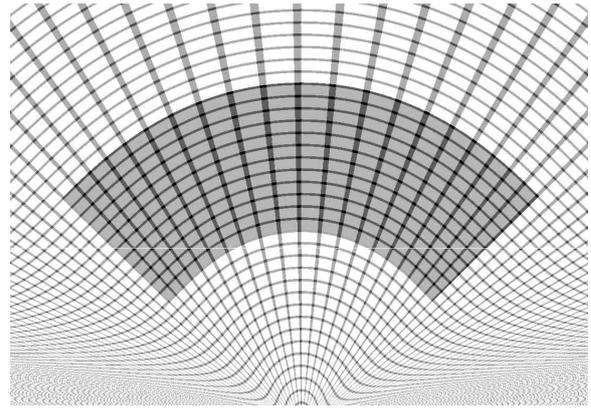
**Figure 8.** Strongly bent curves such as for  $a = 1$ ,  $b = 1$  must not be too wide. Here,  $W/L = 0.2$ , which is the allowable maximum for these values of  $a$  and  $b$ .

vertex would have to be equipped with two sets of these data—one for the previous quadrangle, one for the next. It is hard to see how the shader could switch between both sets.

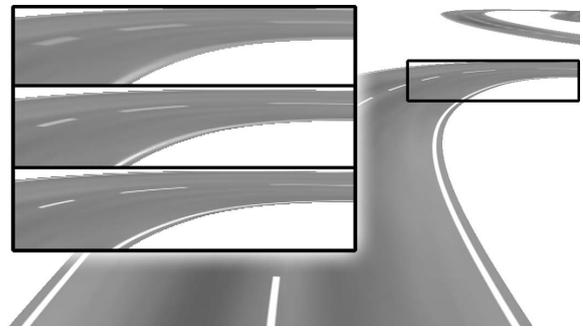
For the speed benchmarks we used an Nvidia GeForce FX 6800 graphics card in a PC equipped with an Intel Pentium-4 processor running at 2.5 GHz. The rendering was done in  $1280 \times 1024$  full screen mode without vertical synchronization.

Because roads are typically viewed under a very oblique angle, textures have to be filtered anisotropically. In our experiments, a setting of 4 for the maximum degree of anisotropy proved to be sufficient, see Figure 10.

To study the scaling behavior we used a base scene, see Figure 11, as a building block to create seven scenes



**Figure 9.** A quarter circle ( $a = 1$ ,  $b = -1$ ,  $W/L = 0.2$ ) is approximated with a peak error of 25 percent, which, however, is not immediately apparent.

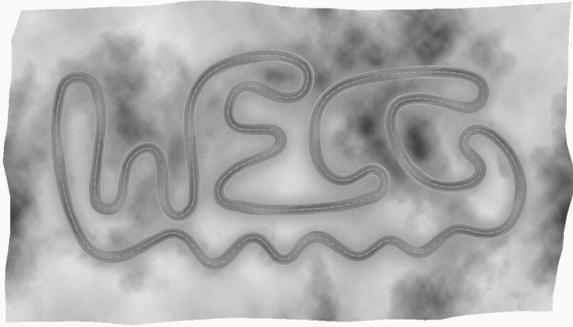


**Figure 10.** To avoid noticeable blurring, we set the maximum anisotropy level of texture filtering to 4. The inset on the left shows the portion outlined on the right with levels 1, 2, and 4 (top to bottom).

of different complexity ranging from one copy of the base scene to  $25 \times 25$  copies arranged side by side in a rectangular pattern. The road of the base scene is composed of 97 segments. In addition, we created a terrain consisting of 2868 triangles, rendered before the road. We used a field of view of  $45^\circ$  and a far plane distance of 1.5 times the longer side length of the base terrain.

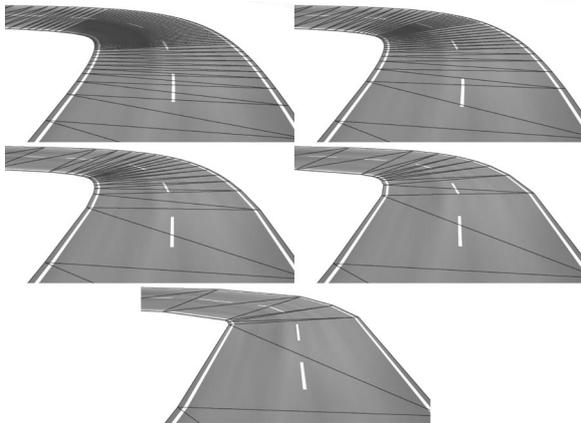
To compare the shader-based solution with a purely polygonal construction, we used the software package Maxon Cinema 4D to create a Catmull-Rom spline curve from the anchor points. (Note that the center-line of the road generated by our method is no such curve, but a visually close approximation.) The spline was extruded into a road, which was stored inside an .x mesh, imported and rendered inside our software prototype.

To have a basis for comparison, we generated a set of five differently tessellated versions using the



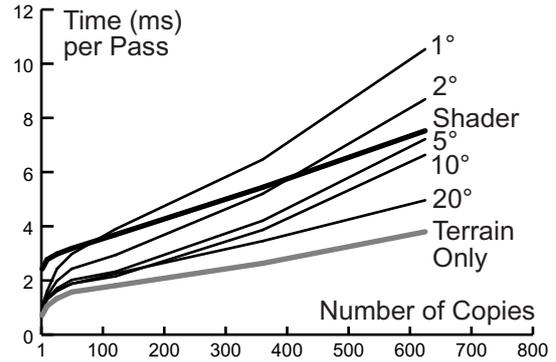
**Figure 11. The base scene for the benchmark comprises a road defined by 97 anchor points and tangents.**

curvature-adaptive setting of Cinema 4D with threshold angles of  $1^\circ$ ,  $2^\circ$ ,  $5^\circ$ ,  $10^\circ$ , and  $20^\circ$ , respectively, which led to polygonal versions of the road consisting of 5374, 2972, 1392, 772, 444, and 256 triangles, see Figure 12. Only the highest one of these resolutions could warrant that the shape and the texture of the road looked perfectly smooth from viewpoints such as that of Figure 12.



**Figure 12. To compare the cookie-cutting method with other approaches, we used five different tessellations at threshold angles  $1^\circ$ ,  $2^\circ$ ,  $5^\circ$ ,  $10^\circ$ , and  $20^\circ$  (from top to bottom).**

Whereas per-pixel procedural geometry in itself is more expensive than standard polygons, the proposed approach may outperform roads rendered from polygons in scenes with high complexity, see Figure 13. This is mainly due to the strongly reduced amount of polygons to be discarded during view frustum clipping. To achieve a similar effect, a purely polygon-based approach may switch to the “5°” or a coarser version based on distance (level of detail) or use some sort of hierarchical frustum culling.



**Figure 13. The benchmarks compares the rendering times for our shader-based method, standard renderings with varying degrees of tessellation, and the terrain without the road.**

In principle, our solution should also benefit greatly from early-z optimization. If the terrain is drawn before the roads, the graphics chip would be able to cull all pixels of roads that are hidden beneath terrain geometry. This could reduce the workload of the pixel shader drastically. However, currently the cookie-cutting approach (i.e., use of the `texkill` instruction in the pixel shader) will disable early-z optimization of typical graphics cards [Rig02, Nvi04].

## 6 CONCLUSION. OUTLOOK

We have presented a method to generate roads and similar 3D objects procedurally with a pixel shader. This approach generates smooth shapes and textures with little effort for initial setup and no runtime scene management overhead. In contrast to that, level-of-detail switching would have to involve countermeasures against popping artifacts or intersections between coarse versions of roads and the terrain.

For roads close to the viewer, the shader-based method leads to a perfect look but adds a noticeable computational load on the graphics chip. For distant roads and scenes of high complexity, the performance approaches that of the standard method of tessellating objects into fine triangles. Given the fast performance growth of graphics chips as opposed to that of CPUs, this may also become true for scenes of medium complexity in the near future.

Most applications of our method will need to combine the curved roads with intersections etc. The latter can be built using standard polygon-based geometry. The transition between such a crossing and a road cookie-cut from polygons can be constructed easily because the road conforms to precise boundary conditions concerning width and direction.

It is straightforward to add sidewalks to the method. One can use a quadrangle shifted upward by the sidewalk's height and cookie-cut this at the corresponding  $v$  values. The size of the quadrangle used can be adapted. However, sidewalks need curbs. Their surface is not contained in a plane, so that a different method than the one described here is needed if they are to be generated procedurally.

We have treated only level roads. If its grade varies only slowly, a road may be constructed from segments that form small angles to each other in the vertical direction. In addition to curvature in the vertical direction one may also try to reproduce such features as superelevation, which means a rotation about the centerline.

It seems plausible that smoothly bent tubes can be generated by a pixel-based method that is similar to the one described. However, such a method would have to employ billboard-type pseudo-geometry, which always faces the viewer. Furthermore, it would have to address shading, too. To this end, normal vectors can easily be derived from the curved coordinates.

## 7 ACKNOWLEDGMENTS

The author wishes to thank two of the anonymous reviewers for providing detailed comments, which proved very helpful for clarification.

## 8 REFERENCES

[AAS01] American Association of State Highway and Transportation Officials. A Policy on Geometric Design of Highways and Streets. AASHTO, 2001.

- [ATI04] ATI. Making of Ruby, [http://www.ati.com/developer/SIGGRAPH04/MakingOfRuby\\_Slides.pdf](http://www.ati.com/developer/SIGGRAPH04/MakingOfRuby_Slides.pdf), 2004.
- [Bol03] Bolz, J., Schröder, P. Evaluation of Subdivision Surfaces on Programmable Graphics Hardware. Submitted for publication, <http://www.multires.caltech.edu/pubs/GPUSubD.pdf>, 2003.
- [Elb97] Elber, G., Lee, I.-K., Kim, M.-S. Comparing Offset Curve Approximation Methods. IEEE Computer Graphics and Applications 17(3), pp. 62–71, 1997.
- [Hir04] Hirche, J., Ehlert, A., Guthe, S. Hardware Accelerated Per-Pixel Displacement Mapping. Proc. of Graphics Interface 2004, pp. 153–158, 2004.
- [Kan04] Kanai, T., Yasui, Y. Per-Pixel Evaluation of Parametric Surfaces on GPU. Surface Quality Assessment of Subdivision Surfaces on Programmable Graphics Hardware. Proc. Int'l Conf. on Shape Modeling and Applications 2004, pp.129-136, 2004.
- [Lov04] Loviscach, J. Silhouette Geometry Shaders. In: Engel, W., ed., ShaderX<sup>3</sup>: Advanced Rendering With DirectX and OpenGL, Charles River, pp. 49–56, 2004.
- [Mae99] Maekawa, T. An Overview of Offset Curves and Surfaces, Comp. Aided Design 31, 165–173, 1999.
- [Nvi04] Nvidia GPU Programming Guide, Version 2.2.0, [http://developer.nvidia.com/object/gpu\\_programming\\_guide.html](http://developer.nvidia.com/object/gpu_programming_guide.html), 2004.
- [Pot02] Pottmann, H., Leopoldseder, St., Hofer, M. Approximation with Active B-Spline Curves and Surfaces. Proc. Pacific Graphics 02, pp. 8–25, 2002.
- [Rig02] Rieger, G., Performance Optimization Techniques for ATI Graphics Hardware with DirectX 9.0, Revision 1.0, <http://www.ati.com/developer/dx9/ATI-DX9.Optimization.pdf>, 2002.
- [Ros03] Rose, D., Ertl, T., Interactive Visualization of Large Finite Element Models, Workshop on Vision, Modelling, and Visualization VMV '03, pp. 585–592, 2003.
- [Vla01] Vlachos, A., Peters, J., Boyd, C., Mitchell, J.L. Curved PN triangles. Proc. 2001 Symp. on Interactive 3D Graphics, pp. 159–166, 2001.

# Real-time 3D Camera Tracking for Industrial Augmented Reality Applications

Gabriele Bleser  
Koblenz-Landau University  
Universitätsstraße 1  
56070 Koblenz, Germany

Yulian Pastarmov  
Fraunhofer IGD  
Fraunhoferstraße 5  
64283 Darmstadt, Germany

Didier Stricker  
Fraunhofer IGD  
Fraunhoferstraße 5  
64283 Darmstadt, Germany

{gbleser, ypastarm, stricker} @ igd.fhg.de

## ABSTRACT

In this paper we present a new solution for real-time 3D camera pose estimation for Augmented Reality (AR) applications. The tracking system does not require special engineering of the environment, such as placing markers or beacons. The required input data are a CAD model of the target object to be tracked, and a calibrated reference image of it. We consider the whole process of camera tracking, and developed both an autonomous initialization and a real-time tracking procedure. The system is robust to abrupt camera motions, strong changes of the lighting conditions and partial occlusions. To avoid typical jitter and drift problems the tracker performs feature matching not only in an iterative manner, but also against stable reference features, which are dynamically cached in case of high confidence. We present experimental results generated with help of synthetic ground truth, real off-line and on-line image sequences using different types of target objects.

## Keywords

Computer vision, real-time marker-less camera tracking, automatic initialization, augmented reality

## 1. INTRODUCTION

Augmented Reality (AR) opens new perspectives for a lot of application areas [Azum95], such as maintenance of machines, design, medicine [Bock03, Wesa04], or cultural heritage [Vass02]. Nevertheless one major difficulty of AR is the user-tracking, which is often unstable or requires special infrastructure in the environment, and thus limits severely the application. Computer vision based methods provide the best accuracy, and represent the currently most developed approach. They rely on 2D/2D or 2D/3D correspondences between features (interesting points, edges, regions) of the image frames. This can be either artificially designed and positioned patterns (marker-based tracking) [Thom97] or natural characteristics of the scene (markerless tracking) [Lauc00, Poll99]. For industrial application, the preparation of the scene with markers is not economically viable, so

that only marker-less solutions are accepted.

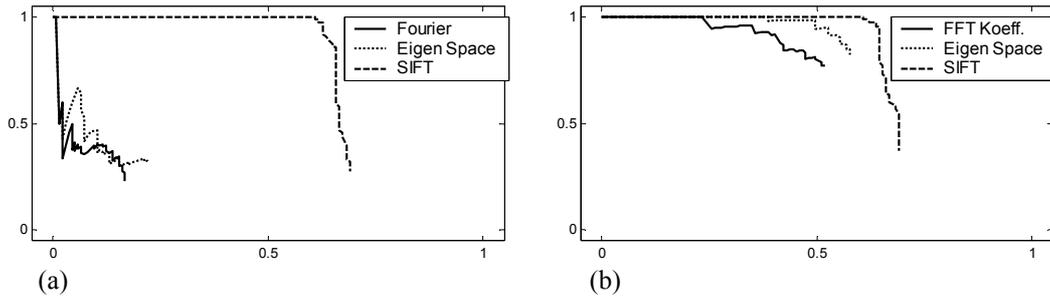
There are generally two approaches to this problem. The global image-based approach computes a 2D transformation, which registers the current frame as a whole on a reference pattern. Stricker [Stri01] uses the Fourier-Mellin Transform to retrieve an Euclidian transformation between the incoming frame and one from a set of calibrated reference images. The current pose is deduced from this transformation assuming the camera being fixed to a tripod and the viewer being far away from the scene. An advantage of the registration on reference images is that no error is accumulated over time (drift).

The local model-based approach fully solves the 3D problem estimating the current pose with 6 DOF. It is very similar to the marker-based approach searching for 2D/3D correspondences by using natural features instead of artificial ones. An interesting approach is presented in [Lepe03]. To avoid jitter and drift during tracking, he merges the information of subsequent frames with that of off-line calibrated reference images. Genc [Genc02] provides useful criteria of stable features. Comport [Comp03] uses edge features instead of points.

Interesting developments have been made in the field of feature extraction during the last years. Classical methods like KLT tracker [Shi94] or Harris detector

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 2005 conference proceedings, ISBN 80-903100-7-9*  
*WSCG'2005, January 31-February 4, 2005*  
*Plzen, Czech Republic.*  
Copyright UNION Agency – Science Press



**Figure 1: Precision-recall-plots showing the behavior of different descriptors concerning Gaussian blur and Euclidian transformations: (a) 10 degrees rotation, 1.1 scaling and (b) 30 degrees rotation, 1.4 scaling.**

[Harr88] with correlation assume small frame-to-frame difference and restrict the searching region of a feature in the near neighborhood of its previous location. Therefore they fail in case of wide baseline. SIFT [Lowe99] has been developed for that particular purpose. The so-called SIFT-keys are Euclidean invariant, robust against small affine and 3D projective transformations, and linear lighting changes. Moreover they can be stored and matched efficiently without the related images or patches.

## 2. APPROACH

The approach presented here provides a full solution of 3D pose estimation without any restrictions on camera motions or the scene configuration. Changes of the lighting conditions are also handled in a reasonable range. Our method consists in 3D-model-based tracking with SIFT features "lying" on a predefined CAD model of a target scene object. The system initializes autonomously and recalibrates itself in case of tracking failure. It is designed for small environments and because of its robustness, speed and fast automatic (re-)initialization, it is particularly usable in industrial processes. The approach is similar to Lepetit's one, but the feature extraction is different and enables to reduce the amount of offline data and to use a single reference image. The feature matching is done not only in iterative manner, but also against the features of the reference image in order to avoid drift. To be independent of occlusions, new features are taken into account by back-projecting them onto the CAD model. We thus generate in a dynamic way new 2D/3D correspondences after successful pose estimation and can handle large changes of the features.

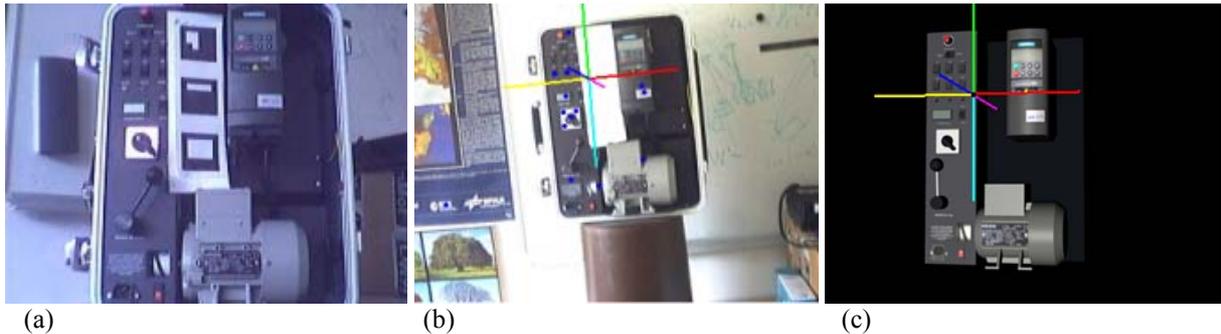
## 3. INITIALIZATION

The initialization method yields the initial camera pose within a global reference coordinate system using one calibrated "bootstrap" reference frame. It is based on the matching of the features and provides proper results even if the initial pose is rela-

tively far away from that related to the reference image.

Only a minimal preparation is required. It consists in taking one photograph of the target object and calibrating it manually. Extrinsic parameters are computed by choosing at least four correspondent points on the 3D model and the snapshot.

When the tracking system starts, the initialization procedure is invoked receiving one calibrated reference frame, a CAD model of the target object and the incoming frame as input. Firstly it performs feature extraction from the reference frame and back-projects all interesting points on the 3D model to obtain 3D coordinates. The back-projection is done by sending rays from the related camera position through the image plane and computing their intercept points with the 3D model. The intersection test is implemented with OpenSG, which employs bounding volume hierarchies for efficient ray tracing. All features that lie on the target object surface are kept ready for (re-)initialization during the whole system run-time. Now we have a reference set of 2D/3D corresponding points and features describing their local appearances within the reference image. The following steps consist in firstly detecting and matching features from the incoming frame with those of the reference image, removing spurious matches by applying geometric constraints and afterwards estimating the initial pose from the resulting 2D/3D correspondences if there are enough (usually more than 10). These steps are processed for each incoming frame until the initial pose could be determined adequately. If the initialization has been successful, the features and corresponding 2D and 3D points of the current frame are added to the reference data set, whereas those, which have not been matched, are back-projected from the initial pose to obtain 3D coordinates. This technique dynamically extends reference data for further (re-)initializations whereas the dynamically added reference features resemble current lighting conditions and camera parameters better than the bootstrap reference frame.



**Figure 2: Example of automatic initialization: (a) shows the reference frame, (b) the incoming frame. Note, that the viewpoints as well as the lighting conditions are quite different. The incoming frame is augmented by the matched feature points (blue) and the axis of the world coordinate system projected from the calculated pose. A synthetic image generated from the textured CAD model shows the correct projection of the coordinate axis (c). This allows for optical verification of the calculated pose. The matching yielded 17 point correspondences. 10 were accepted by RANSAC running 50 iterations.**

### Initial Feature Matching

Unlike iterative feature tracking, the initial matching between features of the reference image and the incoming frame cannot profit from temporal coherence. The initial camera pose can be relatively far away from that related to the reference frame. So there exists no initial guess of the current pose and, consequently, no meaningful assumption about feature displacements or the 2D location of the target object within the current frame. The whole incoming frame has to be processed in terms of feature extraction solving the subsequent matching as problem with quadratic costs. Moreover, a drastic change of lighting conditions often occurs in praxis. Ideally, viewpoint and illumination insensitive feature extraction and characterization methods yielding small and distinctive descriptors are required for the tracker initialization. Scale-invariant features can be identified by looking for local optima of pyramidal difference-of-Gaussian functions in scale-space [Lowe99]. We took this approach for feature detection and performed a simple test comparing the robustness of different local descriptors in terms of "precision" and "recall" at the end of which we chose the most appropriate one for feature characterization in our initialization procedure. The test consists in extracting scale-invariant points from two images, - whereat the second image was synthetically generated from the first, applying different Euclidian transformations and adding Gaussian noise of standard deviation 10 - characterizing there local point neighborhoods by the different descriptors, finding matches based on simple Euclidean vector distance in combination with a threshold and obtaining precision-recall-curves by varying this threshold.

Precision is defined as the rate between the number of correct matches and the number of returned matches. It is the opposite of the outlier rate. Recall

is defined as the number of correct matches against the number of possible matches.

We tested three different descriptors. The first one characterizes interesting points by applying the Fourier transformation to their local fixed-sized point neighborhoods and choosing few Fourier coefficients (FFT Koeff) [Spie00]. The second one describes them as coordinates within a low dimensional coordinate system (Eigen space), spanned by the Eigen vectors of the covariance matrix of a training set [Turk91]. Lepetit [Lepeti03] uses this approach for tracker initialization, whereat the Eigen-space is computed offline for the set of reference images. The last descriptor uses an orientation histogram of the image patch centering the point for characterization (SIFT). Furthermore, the size of the local neighborhood is adapted to the pyramid level, in which the point has been detected, and the histogram is given relative to the major gradient orientation. The resulting plots (see Fig. 1) show the overall Euclidian invariance of the SIFT keys and made our decision to choose SIFT for initial feature matching. Unlike Eigen features a further advantage of SIFT is its independency from a training set. SIFT features are computed autonomously and therefore support dynamical generation of reference data, if the camera pose is highly correct. For key generation we use as described in Lowe's paper, a 4x4 subsampling of the local point neighborhood and consider 8 discrete gradient orientations. So the resulting vector contains 128 elements and can be stored and matched efficiently without the related image.

Feature matching follows a simple criterion. A 2D point  $\mathbf{a}$  of the reference set matches a 2D point  $\mathbf{b}_i$  of the incoming frame, if the Euclidian distance between the related descriptors  $\mathbf{D}_a$  and  $\mathbf{D}_{b_i}$  is minimal and an additional rule is fulfilled. Let  $\mathbf{D}_{b_i}$  be the

descriptor related to point  $\mathbf{b}_j$  with the second nearest Euclidian distance to  $\mathbf{D}_a$ :

$$\| \mathbf{D}_a - \mathbf{D}_{b_i} \| < t \cdot \| \mathbf{D}_a - \mathbf{D}_{b_j} \| \quad (1)$$

$$t \in [0 \dots 1]$$

The threshold (e.g.  $t = 0.6$ ) sees that badly defined matches are discarded immediately. This includes interesting points on periodic image textures. If multiple matches occur, we choose the one with the smallest Euclidian distance and discard the others. Now we have a set of 2D/2D correspondences between the incoming frame and the reference set. Since the latter also contains 3D coordinates for every interesting 2D feature, we easily obtain 2D/3D correspondences by connecting every 2D point of the incoming frame with the 3D coordinate related to its matching reference point.

### Initial Pose Estimation

To obtain a well-conditioned set of 2D/3D correspondent points for pose estimation, we filter the correspondences retrieved from the matching by applying geometric constraints. We use the RANSAC algorithm by employing the projection matrix as geometric model [Hart00]. The projection matrix is calculated linearly from four sample correspondences by either using the original POSIT algorithm or the extension for coplanar model points according to the configuration of the sample set [DeMe92]. We obtain good results and not more than 50 RANSAC iterations were sufficient. Having robustly removed all outliers, the current pose is linearly estimated from all remaining 2D/3D correspondent points  $\mathbf{m}_i \leftrightarrow \mathbf{M}_i$ . To obtain the final pose, we optimize the reprojection errors over camera rotation  $\mathbf{R}$  and translation  $\mathbf{T}$  using the linear estimation as initial guess:

$$\min_{\mathbf{R}, \mathbf{T}} \sum_i \| \mathbf{m}_i - \mathbf{m}_i' \|^2 \quad (2)$$

$\mathbf{m}_i'$  is the projection of  $\mathbf{M}_i$  from the current pose and  $\mathbf{R}$  is parameterized by a rotation axis and an angle.

### Results

Figure 2 shows an example of our autonomous initialization method. The markers fixed to the target object have been occluded during initialization and have no influence on the procedure. Tracker initialization succeeded for the first incoming frame, although the initial pose was relatively far away from that related to the reference frame especially concerning the distance from the target object. Moreover the lighting conditions are quite different.

## 4. TRACKING

Abrupt motions and drastic changes of lighting conditions are typical for AR applications and do not only make initialization but also iterative tracking difficult. The paradigm of temporal coherence, which is the underlying principle for most traditional feature tracking methods basing on correlation, is often not fulfilled. For that reason we decided to treat the problem of iterative tracking similar to that of tracker initialization, i.e. as a matching problem with quadratic costs. To speed up the tracker we do not process the whole incoming frame in terms of feature extraction, but only a fixed-sized image region, which follows the projection of the target object within the image. Because of the latter, most features we loose by regarding not the whole image do not lie on the model surface anyway and would be useless for pose estimation. So if the size of the tracking region isn't chosen too small, the tracking procedure doesn't suffer from this technique but works far more efficiently.

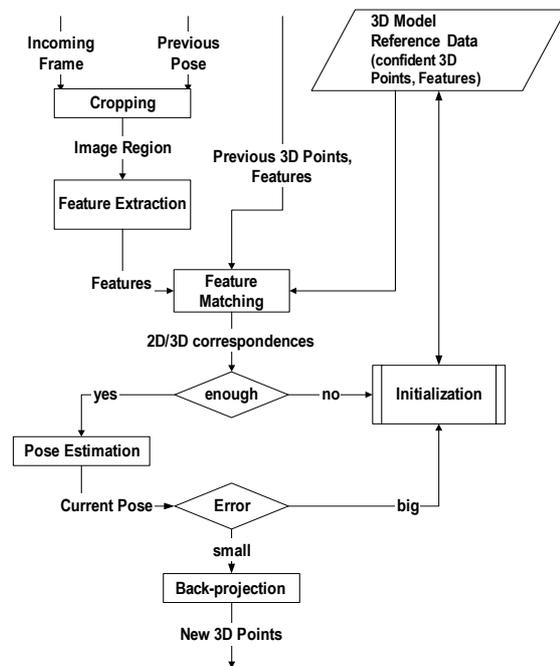


Figure 3: The principles of our tracking procedure are: processing only a relevant region of the incoming frame in terms of feature extraction, matching its features not only against those of the previous frame but also against reference features with confident 3D points to avoid drift, back-projecting new features after robust pose estimation to obtain corresponding 3D coordinates for the next iteration and automatically invoking the initialization procedure in case of tracking failure.

Figure 3 shows a global overview of the tracking procedure. It always focuses two successive frames, consisting in the current frame and the previous one with its calculation results. It has additional access to the 3D model and reference data similar to features with confident 3D coordinates, which come from initialization. At every switch from initialization to tracking, the frame, for which the initial pose could be calculated adequately (consecutively called “initial frame”), is given to the tracking module as previous frame and is taken for calculating the 3D position of the tracking region. If the tracking fails because of few matches or a badly defined pose concerning residual error during pose estimation, we invoke the initialization procedure for automatically re-initializing the tracker. To deal with appearing and disappearing points, our approach bases on constant back-projection onto the CAD model. A successfully calculated pose is always employed to back-project new features for obtaining 3D coordinates, thus handling changes concerning occlusions or light differences.

### The Tracking Region

During initialization we have no guess, where the projection of the target object is located within the initial frame. Although all matches, which do not lie on the target object surface, will have to be discarded again we have to process the whole image concerning feature extraction. Throughout iterative tracking we can make the assumption that the location of the target object does not change significantly between successive frames. As we know the previous pose as well as the 3D model, we get the previous location simply by projecting the model from that pose. We do not consider something like a convex hull of the projection but only a fixed-sized image region which mainly includes the target object. Furthermore we take into account that the target object is not necessarily textured equally well in all parts. We make the image region to contain the richly textured parts that deliver many features. The realization of these ideas is simple. The center of the tracking region is related to a 3D coordinate on the target object surface, which is projected into the current frame from the previous pose. The fixed-size rectangular tracking region is then simply cropped from the incoming frame centering this projection and in that connection follows the projection of the object. We automatized the process of dynamically finding a good 3D center of the tracking region after every (re-)initialization. During the first iteration of the tracking procedure we know all features and corresponding 3D coordinates of the initial frame. We simply choose the 3D center of the tracking region as one of those 3D coordinates

optimizing the overall number of features, which are included in the resulting image region. So if the size of the tracking region is chosen smaller than the current projection of target object, we get the region with most features inside even though.

### Feature Matching

We match the SIFT features of the incoming frame not only with those of the previous one but also with those of the reference data set. This provides confident 3D coordinates and significantly increases the stability and precision of the tracker by avoiding drift. Let the interesting points and related features and 3D coordinates at time  $t-1$  be:

$$\begin{aligned} \mathbf{m}^{t-1} &= \{\mathbf{m}_0^{t-1}, \dots, \mathbf{m}_n^{t-1}\} \\ \mathbf{D}^{t-1} &= \{\mathbf{D}_0^{t-1}, \dots, \mathbf{D}_n^{t-1}\} \\ \mathbf{M}^{t-1} &= \{\mathbf{M}_0^{t-1}, \dots, \mathbf{M}_n^{t-1}\} \end{aligned}$$

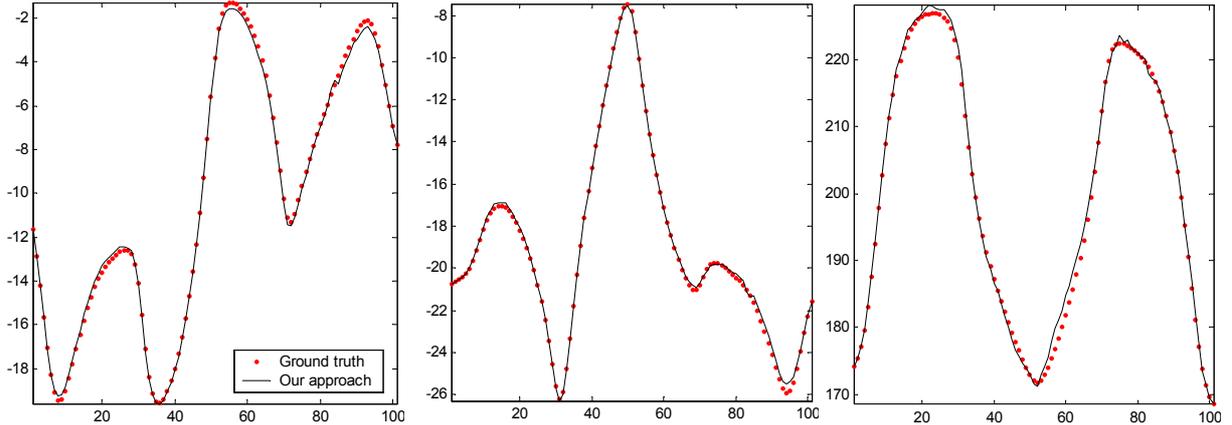
The reference data set ( $\mathbf{m}^{\text{ref}}, \mathbf{D}^{\text{ref}}, \mathbf{M}^{\text{ref}}$ ) is defined in the same way. For each descriptor in the current frame  $\mathbf{D}^t$  we choose the one in either set  $\mathbf{D}^{t-1}$  or  $\mathbf{D}^{\text{ref}}$  that minimizes the Euclidian distance and fulfils equation (1). Now we have some matches between the corresponding 2D points:

$$\mathbf{m}^{t_i} \leftrightarrow \mathbf{m}^{t-1_j} \text{ or } \mathbf{m}^{t_i} \leftrightarrow \mathbf{m}^{\text{ref}_j}$$

As correctly matched image points are different projections of the same 3D point we associate  $\mathbf{M}^t$  with the known 3D coordinate of its match  $\mathbf{M}^{t-1_j}$  or  $\mathbf{M}^{\text{ref}_j}$ . Obviously it is desirable to have as much matches as possible with reference features.

### Pose Estimation

Similar to initial pose estimation we obtain the current pose by first filtering the 2D/3D correspondences with the RANSAC algorithm and afterwards optimizing the reprojection errors of the remaining correspondences over the current camera rotation  $\mathbf{R}_t$  and translation  $\mathbf{T}_t$ . During iterative tracking the 2D/3D correspondences from matching are better conditioned and therefore RANSAC mainly converges after few iterations. Concerning nonlinear optimization we made two little changes in comparison with the initial pose estimation. Firstly, we take the pose of the previous frame as initial guess. Second, we use the robust TUKEY estimator [Rous87] for optimization. This estimator assigns a special weight [0...1] to each correspondence and thereby varies its influence on pose estimation. Distant outliers are weighted by zero and therefore have no influence.



**Figure 4:** These plots show the precision of our method on a synthetic image sequence concerning the three coordinates of camera translation. The dots represent ground truth. Note that the reconstructed camera trajectory neither suffers from drift nor from jitter.

So equation (2) becomes

$$\min_{R_i, T_i} \sum_i \rho_{\text{TUKEY}} \left( \left\| \mathbf{m}_i - \mathbf{m}_i' \right\|^2 \right)$$

where  $\rho_{\text{TUKEY}}$  is the weighting function. Due to the former RANSAC filtering, TUKEY weights mainly stay within the upper quarter of the legal interval. But the employment of this estimator provides a more continuous camera trajectory, as the weight calculation partly depends on the previous pose.

## 5. EVALUTATION

We applied our method to synthetic and real data using different types of target objects like simple and more complex, planar and non-planar as well as highly and poorly textured ones. Following validation criteria are established: precision, robustness and speed. For optical verification of the current pose, the incoming frame is augmented with the axis of the world coordinate system as well as the tracked feature points. Synthetic image sequences, generated from a CAD model, provide ground truth for exact verification in terms of precision.

### Synthetic Images

From a textured CAD model (see Figure 2 (c)) we rendered a 100 frames sequence with quick camera movements knowing extrinsic camera parameters for each frame except for a digitalization error. Applying our method to the synthetic images, which do not suffer from noise or radial distortion, and knowing intrinsic camera parameters exactly, we obtained very good results concerning precision (see Figure 4). The reconstructed camera translation exhibits an average Euclidian error of not more than 0.87 cm from the correct position without showing any drift or jitter. The maximum distance

has been measured with 3.2 cm mainly resulting from the z coordinate. We also measured the average reprojection error over four vertices of the CAD model with 1.24 pixels, whereat 2.83 was the maximum value.

### Off-line Video Sequences

With a low-end USB web cam we captured a real image sequence wearing a HMD with the camera being fixed at it and thus simulating industrial practice. Results are shown in Figure 5 (a–f) and argue for the robustness of our method against partial occlusions (a, f) and changes of lighting conditions (b) meanwhile providing a large field of activity (c–e). We also tested our method using a package box as mainly planar target object (see Figure 5 (g–i)) and a BMW armrest as poorly textured one (see Figure 5 (j–l)).

### Live Video

We applied our method to 320x240 images from both a USB web cam and a Firewire camera. Although the latter yields higher quality images in terms of noise and radial distortion, we obtained comparably stable results. With the different parameters (SIFT, size of tracking region, RANSAC iterations) being optimized for stability the system runs at real-time (19 frames/sec).

## 6. CONCLUSION

We presented a real-time 6 DOF camera tracking system, which includes an autonomous initialization procedure. It is designed for small environment tracking and works with different types of target objects the only restriction being, that a CAD model is available. We use 3D model information for constant back-projection of new features to be independent of partial occlusions. We perform fea-

ture matching against both the previous frame and confident reference data, thus increasing the stability of the tracker and avoiding jitter and drift. The system proves to be robust against sharp camera movements and changes of lighting conditions, which are typical for AR.

Future work includes the enhancement of pose estimation by also taking epipolar constraints between 2D/2D correspondences into account. Furthermore we want to put some effort on the improvement of feature extraction, especially by working on affine invariant representations of the image patches [Baum00, Miko02] and by searching for more efficient descriptors [Ke04].

## 7. REFERENCES

- [Azum95] Azuma, R. A Survey of Augmented Reality. In *Computer Graphics SIGGRAPH Proc.*, pp. 1-38, 1995
- [Baum00] Baumberg, A. Reliable feature matching across widely separated views. In *Proc. CVPR*, pages 774-781, 2000
- [Bock03] Bockholt, U., Bisler, A., Becker, M., Müller-Wittig, W.K. and Voss, G. Augmented Reality for Enhancement of Endoscopic Interventions. In *Proc. IEEE Virtual Reality Conference*, pp. 97-101, 2003
- [Comp03] Comport, A.I., Marchand, E., and Chaumette, F. A real-time tracker for markerless augmented reality. In *Proc. Second IEEE and ACM International Symposium on Mixed and Augmented Reality*, Tokyo, Japan, pp. 36-45, 2003
- [DeMe92] DeMenthon, D. and Davis, L.S. Model-based object pose in 25 lines of code. In *European Conference on Computer Vision*, pp. 335-343, 1992
- [Genc02] Genc, Y., Riedel, S., Souvannavong, F. and Navab, N. Markerless tracking for augmented reality: A learning-based approach. In *Proc. International Symposium on Mixed and Augmented Reality*, 2002
- [Harr88] Harris, C. and Stephens, M.J. A combined corner and edge detector. In *Proc. Fourth Alvey Vision Conference*, Manchester, pp. 147-151, 1988
- [Hart00] Hartley, R. and Zisserman, A. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000
- [Ke04] Ke, Y. and Sukthankar, R. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. In *CVPR*, 2004
- [Lau00] McLauchlan, P. A Batch/Recursive Algorithm for 3D Scene Reconstruction. In *Proc. CVPR*, 2000
- [Lepe03] Lepetit, V., Vacchetti, L., Thalmann, D. and Fua, P. Fully Automated and Stable Registration for Augmented Reality Applications. In *Proc. Second IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2003
- [Lowe99] Lowe, D.G. Object recognition from local scale-invariant features. In *Proc. of the International Conference on Computer Vision (ICCV)*, Corfu, pp.1150-1157, 1999
- [Miko02] Mikolajczyk, K. and Schmid, C. An affine invariant interest point detector. In *ECCV*, 2002
- [Poll99] Pollefeys, M., Koch, R. and Van Gool, L. Self-Calibration and Metric Reconstruction in spite of Varying and Unknown Internal Camera Parameters. In *International Journal of Computer Vision*, pp. 7-25, 1999
- [Rous87] Rousseeuw, P. and Leroy, A. *Robust Regression and Outlier Detection*. Wiley, 1987
- [Shi94] Shi, J. and Tomasi, C. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, Washington, pp. 593-600, 1994
- [Spie00] Spiess, H. and Ricketts, I. Face Recognition in Fourier Space. In *Vision Interface*, Montreal, pp. 38-44, 2000
- [Stri01] Stricker, D. Tracking with Reference Images: A Real-Time and Markerless Tracking Solution for Out-Door Augmented Reality Applications. In *Proc. of VAST*, 2001
- [Thom97] Thomas, G.A., Jin, J., Niblett, T. and Urquhart, A. A versatile camera position measurement system for virtual reality TV-production. *International Broadcasting Convention*, IEEE Conference Publication, pp.284-289, 1997
- [Turk91] Turk, M. and Pentland, A. Eigenfaces for Recognition. In *Journal of Cognitive Neuroscience*, Vol. 3, No. 1, pp. 71-86, 1991
- [Vass02] Vlahakis, V., Ioannidis, N., Karigiannis, J., Tstros, M., Gounaris, M., Stricker, D., Gleue, T., Dähne, P. and Almeida, L. *Archeoguide: An Augmented Reality Guide for Archaeological Sites*. In: *IEEE Computer Graphics and Applications* Vol. 22, No. 5, pp. 52-60, 2002
- [Wesa04] Wesarg, S., Firl, E., Schwald, B., Seibert, H., Zogal, P. and Roeddiger, S. Accuracy of Needle Implantation in Brachytherapy Using a Medical AR System - a Phantom Study. In *SPIE Medical Imaging Symposium*, pp. 341-352, 2004

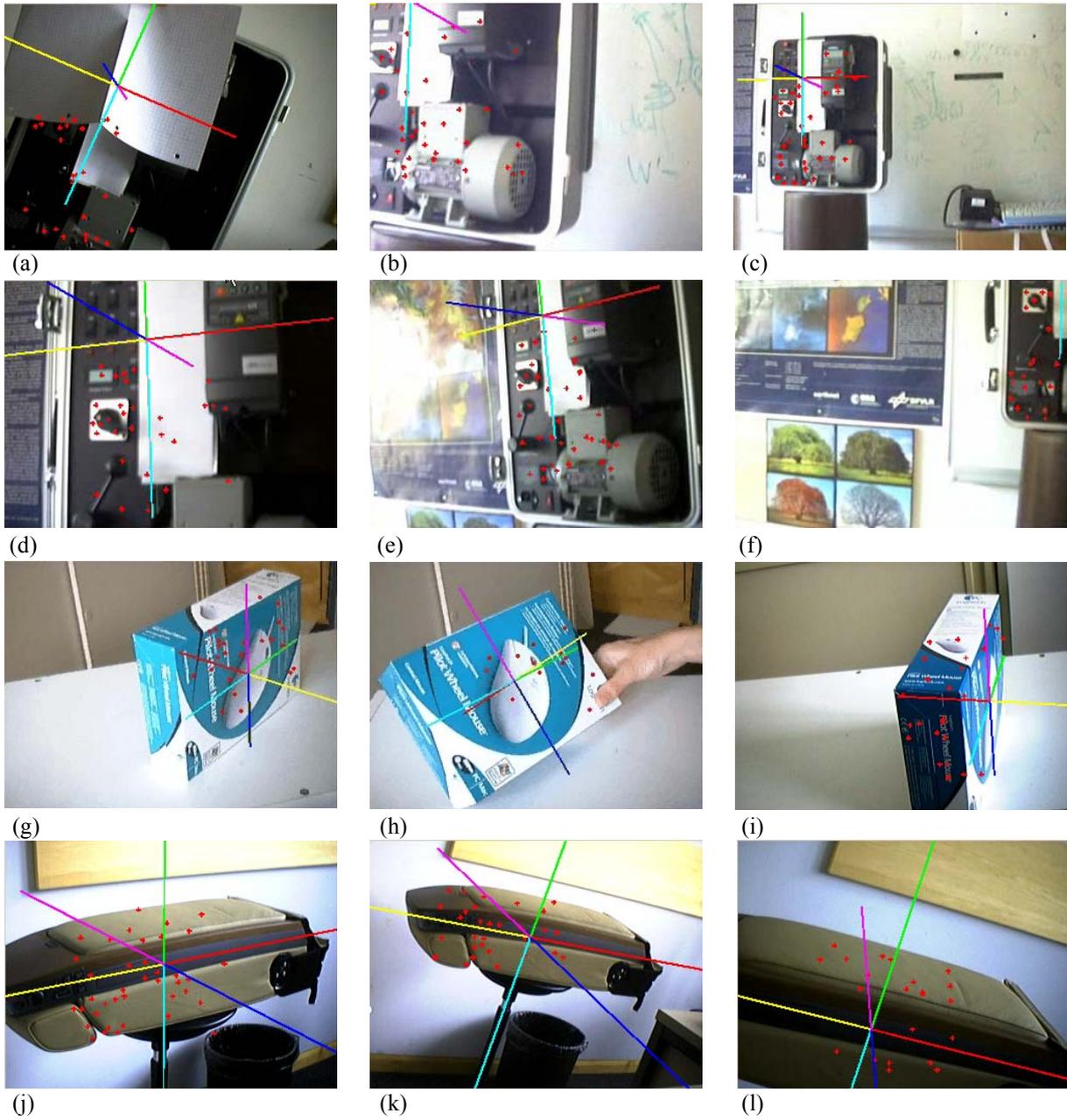


Figure 5: Results of our tracking procedure using different types of target objects (planar/non-planar, highly/poorly textured, simple/complex).

# Detection of Facial Landmarks from Neutral, Happy, and Disgust Facial Images

Ioulia Guizatdinova and Veikko Surakka

Research Group for Emotions, Sociality, and Computing  
Tampere Unit for Computer-Human Interaction  
Department of Computer Sciences  
University of Tampere  
FIN-33014 Tampere, Finland

ig74400@cs.uta.fi

Veikko.Surakka@uta.fi

## ABSTRACT

Automated analysis of faces showing different expressions has been recently studied to improve the quality of human-computer interaction. In this framework, the expression-invariant face segmentation is a crucial step for any vision-based interaction scheme. A method for detecting facial landmarks from neutral and expressive facial images was proposed. In present study, a particular emphasis was given to handling expressions of happiness and disgust. The impact of these expressions on the developed method was tested using dataset including neutral, happiness and disgust images. The results demonstrated a high accuracy in detecting landmarks from neutral images. However, the expressions of happiness and disgust had a deteriorating effect on the landmark detection.

## Keywords

Image processing, face segmentation, detection of local oriented edges, Gaussian, facial landmarks, human-computer interaction.

## 1. INTRODUCTION

In the past decades there has been a considerable interest in improving all aspects of human-computer interaction (HCI). One way to achieve intelligent HCI is making computers to interact with user in the same manner as it takes place in human-human interaction.

Humans naturally interact with each other through verbal (i.e. speech) and nonverbal (i.e. facial expressions, gesture, vocal tones, etc.) sign systems. It is argued that during human-human interaction only a small part of the conveyed messages is verbally communicated, and the greatest part is nonverbally coded. Considering nonverbal communication, it is possible to say that facial expressions occupy about a half of the transmitted signals. In the context of user-

friendly HCI, a face is an important source of information about the user to be analyzed by the computer.

Automated analysis of a computer user's face has recently become an active research field in the computer vision community. Different vision-based schemes for intelligent HCI are currently being developed. The ability of a computer to detect, analyse and, finally, recognize a user's face has many applications in the domain of HCI.

The analysis and recognition of facial expressions in the context of HCI are elements of interaction design called affective computing [Jen98]. The main idea of the affective computing is that the computer detects the user's affective state and takes an appropriate action, for example, offers assistance for the user or adapts to the user's needs. Proper detection of the changes in the user's facial cues is a precondition for the computer to take any emotionally or otherwise intelligent socially interactive actions towards the user.

The Facial Action Coding System (FACS) [Ekm78] is widely used to analyse visually observable facial expressions. FACS has been developed for objective analysis of any changes in the facial appearances.

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 2005 conference proceedings, ISBN 80-903100-7-9  
*WSCG'2005, January 31-February 4, 2005*  
*Plzen, Czech Republic.*  
Copyright UNION Agency – Science Press

According to the FACS, a muscular activity producing changes in facial appearance is coded in the terms of action units (AU). Certain specific combinations of AUs have been frequently suggested to represent seven prototypical facial displays: neutral, happiness, sadness, fear, anger, surprise, and disgust.

It is known that reliable person identification and verification are important cornerstones for improving security in various contexts of information society. A natural means of identifying person that gives a close resemblance to the way how humans recognize persons is analysing a person's face.

Face identification has two important advantages. First, it requires a minimal interaction with a person, for example, compared with such biometrics as prompted speech or fingerprints. Second, it is impossible to lose or forget a face as it might happen with passwords or key-cards.

In this framework, automated detection of a face and its features is considered to be an essential requirement for any vision-based HCI scheme [Don99, Wis97]. However, due to such factors as illumination, head pose, expression and scale, facial features vary greatly in their appearance. It is shown that facial expressions are particularly important factors affecting the automated detection of facial features [Yac95]. Nowadays the problem of effective and expression-invariant face detection and segmentation still remains unsolved.

In our previous study we have proposed a method for detecting facial landmarks from neutral and expressive facial images [GuiS]. The developed approach has combined a feature-based method for face segmentation [Sha02] and a profound knowledge on how different facial muscle activations modify the appearance of a face during emotional and social reactions [Par04, Sur98].

Experimented findings have revealed that detection of landmarks from the lower part of a face was especially affected by expressions of happiness and disgust. In particular, detection of the nose and mouth produced the greatest number of detection errors. We assumed that these expressions modify the lower face so that it becomes difficult to differentiate lower face landmarks like nose and mouth. For this reason the present aim was to analyse an accuracy of landmark detection from images of happiness and disgust to corroborate the previous findings.

## 2. FACIAL LANDMARK DETECTION

The method for detection of facial landmarks consisted of three stages: image preprocessing, image

map constructing and orientation matching [GuiS]. These stages are described below.

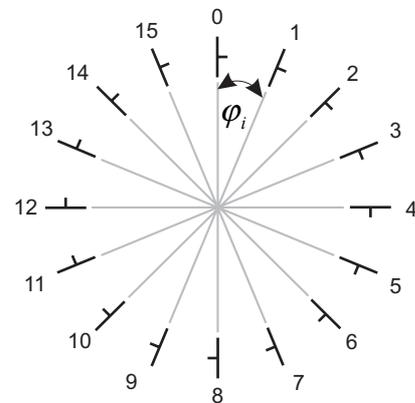
### 2.1. Image preprocessing

First, an image was transformed into the 256-grey-level-scale format. Then, a recursive Gaussian transformation was used to smooth the grey-level image [Gol00]. Image smoothing reduced a search space for detecting facial features (i.e. eliminated noise edges and removed small details) [Can86].

In the following stages of the landmark detection, the smoothed grey-level images were used to detect candidates for facial landmarks. The non-smoothed grey-level images allowed us to analyse the detected candidates in details. In that way, the amount of information to be processed was significantly reduced.

### 2.2. Image map constructing

The local high-contrast oriented edges were used as basic features for constructing edge maps of the image [Ryb98]. Apart from previous studies [Sha02], we decreased a number of edge orientations to construct edge maps of the image. In particular, we used  $2 \div 6$  and  $10 \div 14$  edge orientations (see Fig.1). Decreasing a number of edge orientations allowed us to reduce sufficiently the computational complexity of the method.



**Figure 1. Orientation template,**  
 $\phi_i = i \cdot 22.5^\circ, i = 0 \div 15.$

The oriented edges were extracted by convolving the smoothed image with a set of ten convolution kernels. Each kernel was sensitive to one out of ten chosen edge orientations. For each pixel, the contrast magnitude of a local edge was estimated with maximum response of ten kernels at this pixel location. The orientation of a local edge was estimated with orientation of a kernel that gave the maximum response. The whole set of ten kernels resulted from differences between two oriented Gaussians with shifted kernels.

After the local oriented edges had been extracted, they were filtered by a contrast. The threshold for contrast filtering was determined as an average contrast of the whole smoothed image.

Then, the extracted oriented edges were grouped into edge regions presumed to contain facial landmarks. Edge grouping was based on neighbourhood distances between edges and was limited by a number of possible neighbours for each oriented edge. The optimal thresholds for edge grouping were determined using a small set of expressive images of the same person. The optimal thresholds represented landmark candidates as regions of connected edges that were well separated from the rest of edges.

Once the limits of edge regions had been detected, these regions were analysed more precisely. The procedures of edge extracting, contrast thresholding and edge grouping were applied to the non-smoothed image within the limits of the extracted edge regions. The threshold for contrast filtering was determined as a double average contrast of the non-smoothed image.

In the end, the primary image map consisted of edge regions representing candidates for facial landmarks. The centres of mass determined the locations of the landmark candidates. In the next stage, the landmark candidates were analysed according to their orientation description and matched with an orientation model.

### 2.3. Orientation matching

The orientation portraits of the landmark candidates were constructed on the basis of their local orientation description. The analysis of the orientation portraits revealed four important findings.

First, local oriented edges extracted within regions of eyebrows, eyes, nose and mouth had a characteristic density distribution. Thus, the orientation portraits of these landmarks had two dominant horizontal orientations. The results of the present study corroborated our previous findings [Sha02].

Second, we found that prototypical facial expressions did not affect the distribution of the oriented edges in the regions of facial landmarks [GuiS]. The orientation portraits of facial landmarks still had the same structure including two dominants corresponding to horizontal orientations (see Appendix 1a).

Moreover, for the regions of eyes and mouth the number of edges corresponding to horizontal orientations was more than 50% larger when compared to a number of edges corresponding to other orientations. All edge orientations were represented by non-zero number of the edges.

Third, the average orientation portraits of facial landmarks revealed the same structure including two horizontal dominants (see Fig.2, Appendix 2) [GuiS].

Fourth, noise regions extracted from the expressive images had an arbitrary distribution of the oriented

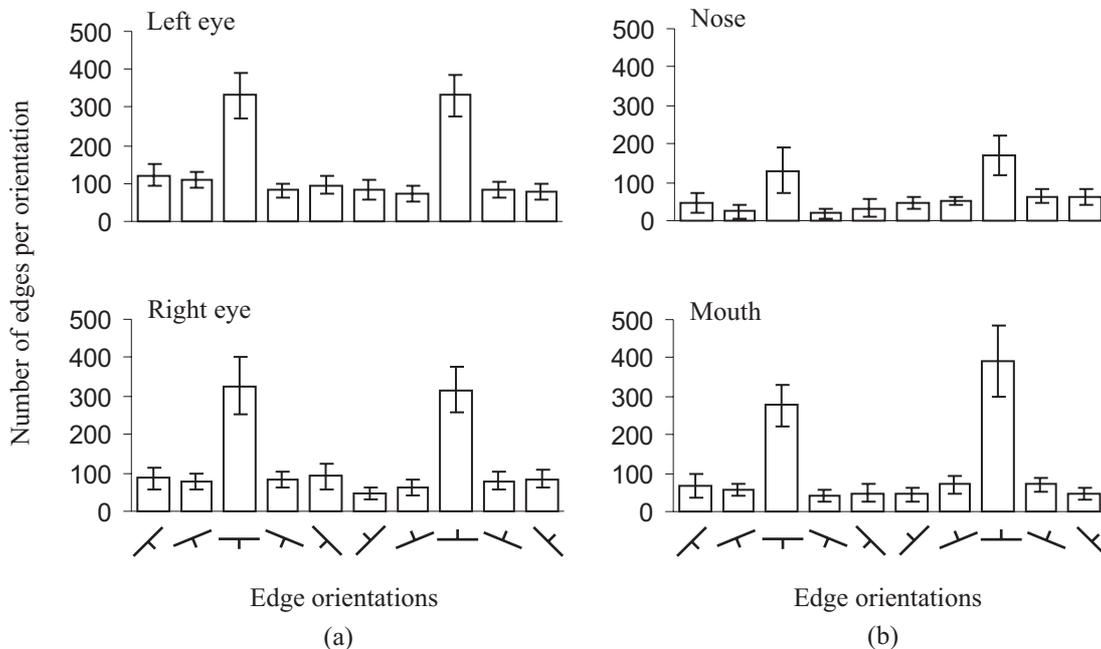


Figure 2. Orientation portraits of facial landmarks averaged over prototypical facial displays.

edges and often had orientations represented by zero number of edges (see Appendix 1b).

The knowledge on clear-cut distinction between orientation portraits of facial landmarks and noise regions allowed us to verify the existence of a landmark on the image. To do that, the orientation portraits of facial candidates were matched with an orientation model of facial landmarks.

### 2.3.1. Orientation model

The characteristic orientation model for detecting facial landmarks consisted of ten possible edge orientations, namely, edge orientations ranging from  $45^\circ$  to  $135^\circ$  and  $225^\circ$  to  $315^\circ$  in step of  $22.5^\circ$ .

The following rules defined the structure of the orientation model: (a) horizontal orientations are represented by the biggest number of edge points; (b) a number of edges corresponding to each of the horizontal orientations is more than 50% bigger than a number of edges corresponding to other orientations taken separately; and (c) orientations can not be represented by zero-number of edge points.

The candidates that did not correspond to the orientation model were removed from the final image map. In such a way, the procedure of orientation matching filtered the regions containing landmarks from the noise.

The detected candidates for facial landmarks were further classified manually into one of the following groups: noise or facial landmark (i.e. eye-eyebrow, nose and mouth).

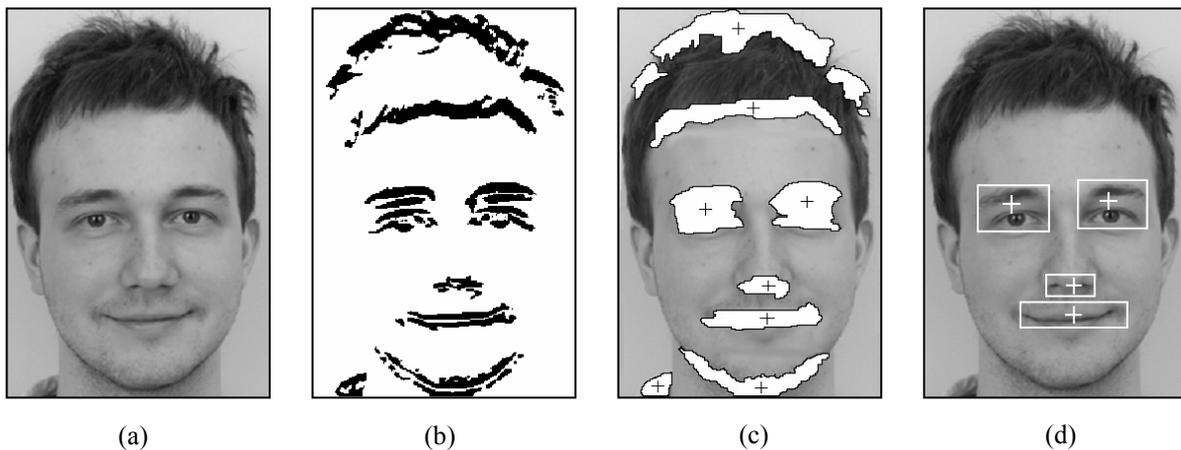
## 3. DATABASES

To evaluate the accuracy of the proposed method we used the Pictures of Facial Affect (PFA) database [Ekm76] and the Cohn-Kanade Face (CKF) database [Kan00].

The PFA database consisted of 110 frontal-view images of 14 individuals (i.e. 6 males and 8 females) representing neutral and six prototypical facial expressions of emotions: happiness, sadness, fear, anger, surprise and disgust. On average, there were about sixteen pictures per expression. The size of the images was preset into 250 by 300 pixels.

The CKF images were originally coded using single AUs and their combinations. In according to translation rules defined in the Investigator's Guide to the FACS manual [Ekm00], the images were relabelled into the emotional prototypes. The images corresponding to the prototypes of happiness and disgust were selected. Thus, there were 172 images: 65 neutral images, 65 images of happiness and 42 images of disgust expression. All the images were normalized to contain only a facial part of the original image. Either of the datasets included faces with facial hair and glasses. All the images were resized into 250 by 480 pixel arrays.

The PFA database was used to select the optimal thresholds for edge grouping and to construct the landmark orientation model [GuiS]. In present study, the CKF database was used to test the accuracy of the method in detection of facial landmarks specifically from the images showing happiness and disgust.



**Figure 3. (a) original facial image; (b) extracted local oriented edges (black dots); (c) primal edge map represents candidates for facial landmarks (white regions) and their mass centers (crosses); (d) final edge map represents the detected facial landmarks.**

## 4. RESULTS

Figure 3 gives an example of edge map composed of the local oriented edges extracted from the expressive facial images. Thus, local edges of  $45^\circ \div 135^\circ$  and  $225^\circ \div 315^\circ$  defined in step of  $22.5^\circ$  constituted the edge map of the happy image shown on Figure 3b. Figure 3c demonstrates the edge map after contrast thresholding and grouping extracted edge points into the candidates for facial landmarks. Figure 3d illustrates the final image map that included only the candidates having orientation portraits well matched with the orientation model.

The average number of the candidates per image of the primary edge map was 7.46. The results revealed that variations in facial expressions did not affect significantly the average number of the candidates per image. The average number of candidates per image was reduced to 3.71 for the final edge map. Such a fact allows us to claim that the procedure of orientation matching reduced the number of landmark candidates by 50%. Figure 4 illustrates the decrease in the number of candidates per image averaged over neutral, happy, and disgust images.

The accuracy of the proposed method was calculated as a ratio of the number of detected landmarks to the number of images used in testing. As it can be seen from Table 1, the developed method achieved a sufficiently high accuracy of 95% in detecting all four facial landmarks from the neutral images. As it can be seen from the table, both eyes are represented as a single column since these landmarks had equal detection accuracy.

However, the results showed that the expressions of happiness and disgust had a marked deteriorating effect on detecting facial landmarks. It is noteworthy that the detection of nose and mouth was more affected by facial expressions than the detection of eyes.

Three types of detection errors caused the decrease in detection accuracy. Figure 5 gives examples of such errors. The undetected facial landmarks were considered to be the errors of the first type. Such

|                                 | Eye | Nose | Mouth | Average |
|---------------------------------|-----|------|-------|---------|
| <b>Neutral</b>                  | 98  | 92   | 92    | 95      |
| <b>Happiness</b>                | 100 | 50   | 50    | 75      |
| <b>Disgust</b>                  | 67  | 57   | 59    | 62      |
| <b>Neutral &amp; Expressive</b> | 88  | 66   | 67    | 78      |

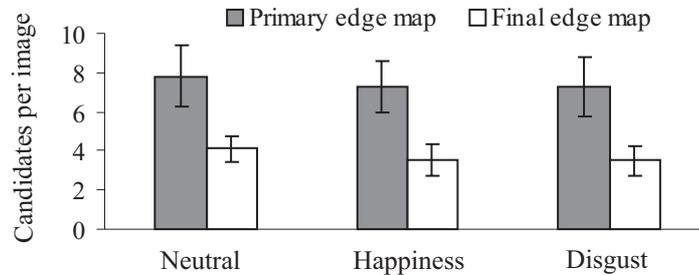
**Table 1. Average accuracy (%) of the landmark detection**

errors occurred when a facial landmark was rejected as a noise region after orientation matching. In particular, the nose was the most undetectable facial landmark (see Fig. 5a). The incorrectly grouped landmarks were regarded as the errors of the second type. The most common error of the second type was grouping regions of nose and mouth into one region (see Fig. 5b). The errors of the third type were the misdetections that occurred when the noise regions were accepted as the facial landmarks (see Fig. 5c).

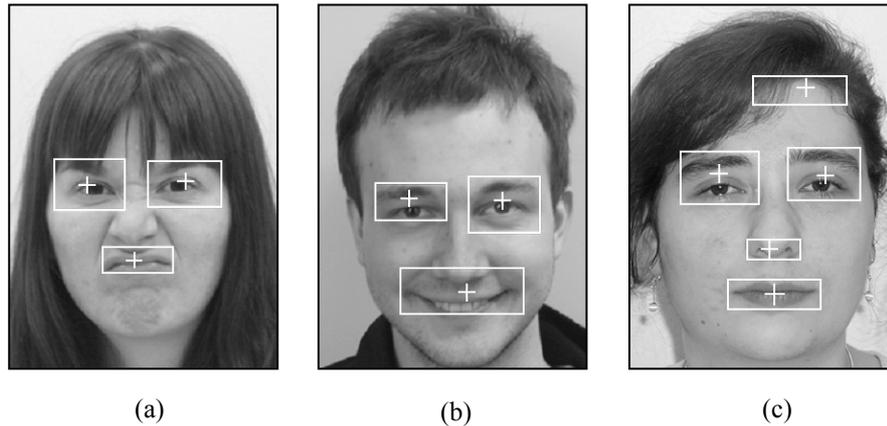
## 5. CONCLUSIONS

The method for detecting facial landmarks from both neutral and expressive facial images was presented and described. The method revealed an average accuracy of 95% in detecting four facial landmarks from neutral facial images.

However, the detection of facial landmarks from happy and disgust facial images produced a large number of detection errors. Thus, the expressions of happiness and disgust attenuated the average (i.e. over all regions) detection accuracy to 75% and of 62%, respectively. Especially the detection of nose and mouth were affected by both expressions of disgust and happiness. These expressions deteriorated the detection of nose and mouth to 50% for happiness. For the disgust expression the detection of nose and mouth deteriorated to 57 and 59, respectively. The present results corroborated our earlier findings that facial expressions have a marked deteriorating effect on the landmark detection



**Figure 4. Average number of candidates per image before and after orientation matching.**



**Figure 5. Examples of the detection errors: (a) undetected nose; (b) incorrectly grouped nose and mouth; (c) detected noise region.**

algorithms.

In summary, the accuracy of the landmark detection from neutral images was comparable with a detection accuracy of the known feature-based and colour-based methods though it is lower than neural network-based methods. The algorithms developed for landmark detection were simple and fast enough to be implemented as a part of systems for face and/or facial expression recognition.

The detection of facial landmarks from expressive images, especially from happy and disgust images needs to be improved. This is especially important in order to make a computer differentiate between positive expressions of emotions, for example, smiling and some negative expressions like disgust. To detect and differentiate between positive and negative user emotions, it is the very minimum prerequisite for affective HCI. This kind of an improvement of the method is also a precondition for recognizing facial identity of a user as well.

## 6. ACKNOWLEDGMENTS

This work was financially supported by the Finnish Academy (project number 177857), the Finnish Centre for International Mobility (CIMO), the University of Tampere and the University of Tampere Foundation.

## 7. REFERENCES

- [Can86] Canny, J. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligent* 8, No.6, pp.679–98, 1986.
- [Don99] Donato, G., Bartlett, M., Hager, J., Ekman, P., and Sejnowski, T. Classifying facial actions. *IEEE Trans. on Pattern Analysis and Machine Intelligent* 21, No.10, pp.974–989, 1999.

- [Ekm76] Ekman, P., and Friesen, W. *Pictures of facial affect*. Consulting Psychologists Press, Palo Alto, California, 1976.
- [Ekm78] Ekman, P., and Friesen, W. V. *Facial Action Coding System (FACS): A technique for the measurement of facial action*. Consulting Psychologists Press, Palo Alto, California, 1978.
- [Ekm00] Ekman, P., Friesen, W., and Hager, J. *Facial Action Coding System (FACS). UTAH: A Human Face*, Salt Lake City, 2002.
- [Gol00] Golovan, A. Neurobionic algorithms of low-level image processing. in *Second All-Russia Scientific Conference Neuroinformatics-2000 conf.proc.*, vol. 1, pp.166-173, 2000.
- [GuiS] Guizatdinova, I., and Surakka, V. Detection of facial landmarks from emotionally expressive and neutral facial images. *IEEE Trans. on Pattern Analysis and Machine Intelligent*, submitted.
- [Jen98] Jennifer, H., and Picard, J. Digital processing of affective signals. in *IEEE ICASSP'98 conf.proc.*, Seattle, 1998.
- [Kan00] Kanade, T., Cohn, J.F., and Tian, Y. Comprehensive database for facial expression analysis. in *AFGR'00 conf.proc.*, Grenoble, p.46, 2000.
- [Par04] Partala, T., and Surakka, V. The effects of affective interventions in human-computer interaction. *Interacting with Computers*, 16, pp.295-309, 2004.
- [Ryb98] Rybak, I. A model of attention-guided invariant visual recognition. *Vision Research* 38, No.15/16, pp.2387-2400, 1998.
- [Sha02] Shaposhnikov, D., Golovan, A., Podladchikova, L., Shevtsova, N., Gao, X., Guskova, V., and Gizatdinova, Y. Application of the behavioural model of vision for invariant recognition of facial and traffic sign images. *Neurocomputers: Design and Application* 7, No.8, pp.21-33, 2002.

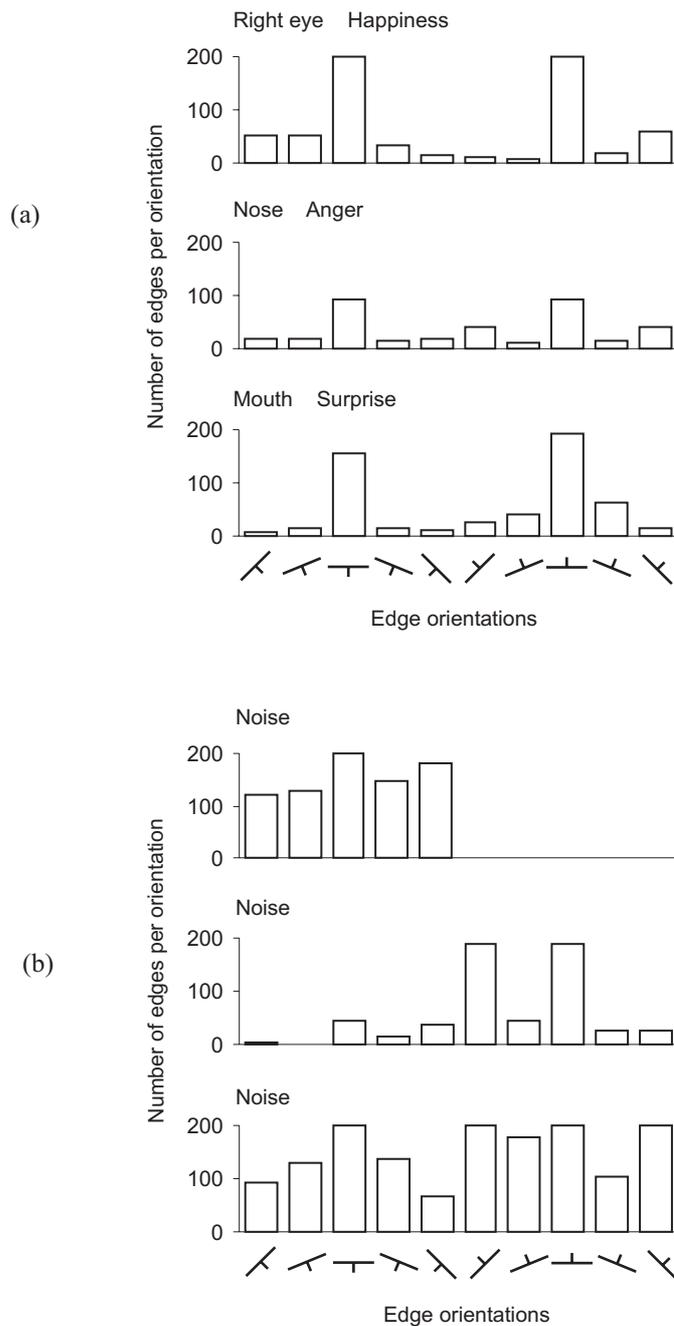
[Sur98] Surakka, V., and Hietanen, J. Facial and emotional reactions to Duchenne and non-Duchenne smiles. *International Journal of Psychophysiology* 29, pp.23-33, 1998.

[Wis97] Wiskott, L., Fellous, J-M., Kruger, N., and von der Malsburg, C. Face recognition by elastic bunch graph matching. *IEEE Trans. on Pattern*

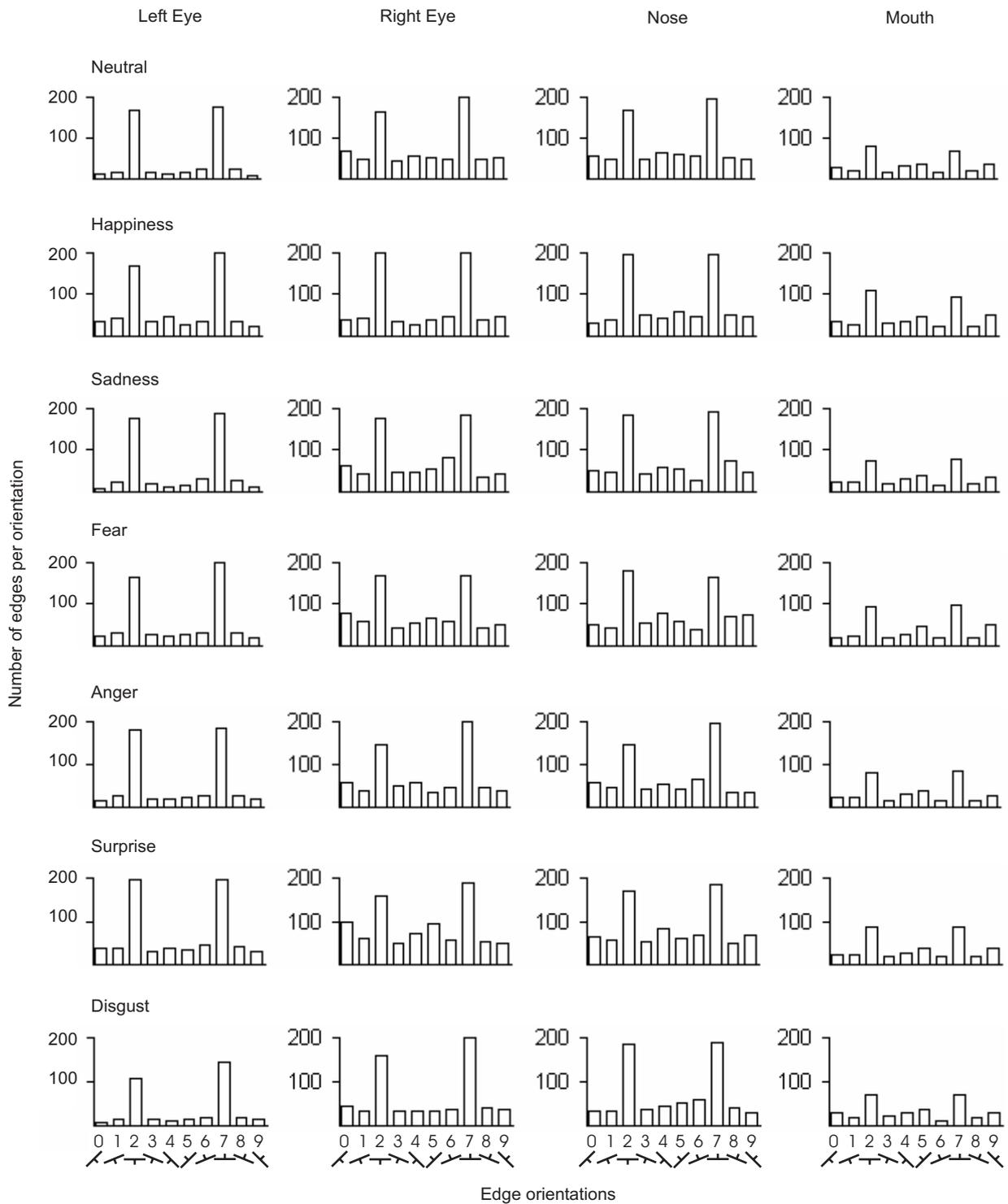
*Analysis and Machine Intelligent* 19, No.7, pp.775-779, 1997.

[Yac95] Yacoob, Y., Lam, H-M., and Davis, L. Recognizing faces showing expressions. in *IWAFGR'95 conf.proc.*, Zurich, pp.278-283, 1995.

**Appendix 1. Orientation portraits of (a) landmarks with characteristic edge distribution, and (b) noise regions with arbitrary edge distribution.**



**Appendix 2. Average orientation portraits for facial landmarks. The columns represent four facial landmarks and rows represent seven prototypical facial displays.**



# Multi-mesh caching and hardware sampling for progressive and interactive rendering

Gabriel Fournier

Bernard Péroche

L.I.R.I.S : Lyon Research Center for Images and Intelligent Information Systems  
CNRS / INSA de Lyon / Université Lyon 1 / Université Lyon 2 / Ecole Centrale de Lyon  
Bâtiment Nautibus, 8 boulevard Niels Bohr  
69622 Villeurbanne Cedex, FRANCE

gabriel.fournier@liris.cnrs.fr

bernard.peroche@liris.cnrs.fr

## ABSTRACT

We present a framework for progressive and interactive rendering with soft shadows and indirect illumination of a triangulated scene. Our method is a multi-pass algorithm that separates the rendering of each main component of radiance in order to update the image as fast as new samples are computed. Those radiance samples are computed at the vertices of multiple recursively subdivided meshes, allowing fast hardware interpolation between the samples. These radiance samples are computed using irradiance values cached in multiple meshes. These meshes separate the direct irradiance from each light source and the indirect one. Using multiple meshes gives us the ability to better reuse samples and to better adapt the sampling density than if a unique mesh was used. We also propose to quickly compute accurate soft shadows and indirect irradiance using the graphics hardware for visibility determination.

## Keywords

global illumination, irradiance caching, progressive rendering, interactive rendering, graphics hardware, area light source

## 1 INTRODUCTION

Real time realistic rendering on a standard PC, with area light sources and indirect illumination, is still a major challenge in computer graphics. In this paper, we suggest a framework and a few tricks that should bring us closer to this goal. Our approach allows progressive and interactive realistic rendering on a single office PC of a triangulated scene lit by area light sources. We chose to favor interactivity over image quality, but our progressive rendering algorithm makes the image tends towards full quality when the user lingers in the same area. Our method does not need a long preprocessing. It can provide fair quality images in a few seconds, hence it can be useful for image preview while designing a scene.

Our approach is a multi-pass method. We separate the rendering of the main radiance components: direct diffuse, direct specular and indirect diffuse. These ra-

diances are computed at the vertices of multiple progressively refined meshes, using a different mesh for each part of the radiance. To compute these radiances, we use an irradiance sample cache. This cache is also made of multiple progressively refined meshes. We use different meshes to store the direct irradiance of each light source, and the indirect irradiance. This new approach allows us to limit the number of computed samples by reusing already computed ones. Irradiance samples are computed using the hardware for visibility determination.

After a few words on the ideas that led us to our solution (Section 2), we will give a quick overview of our method (Section 3). Then we will describe with more details our multi-mesh caching framework (Section 4), how we propose to sample irradiance to fill our cache (Section 5) and how this cache is rendered through multiple passes (Section 6). We will give some results (Section 7) that we will discuss (Section 8).

## 2 PREVIOUS WORK

Our approach rests on well-known techniques: irradiance caching, progressive refinement and use of a triangular mesh for hardware interpolations, uncoupling of rendering and lighting computations, storage of illumination samples in an object-space hardware-rendered mesh, and hardware irradiance sampling. All

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 80-903100-7-9*  
*WSCG'2005, January 31-February 4, 2005*  
*Plzen, Czech Republic.*  
Copyright UNION Agency - Science Press

these techniques will be developed in the next subsections.

### **Irradiance and radiance caching**

Computing high quality indirect irradiance samples for each pixel of an image is very costly. Fortunately, indirect irradiance changes very slowly over a surface. Irradiance caching, introduced by Ward et al. [War88], takes advantage of this property by interpolating indirect irradiance between a few fully computed and cached samples.

Zaninetti et al. extended this method proposing *light vectors* [Zan98]. Instead of caching irradiances, they took into account the BRDF of the objects and stored radiances, allowing glossy objects to be rendered. Noticing that direct, indirect and caustic components of radiance have different frequencies, they sampled and cached each one separately. In their method, rendering is a two pass process. First a seed of samples is generated, then the scene is rendered using those already computed samples to interpolate radiance in between. With this technique, samples are cached in a kd-tree and interpolations are computed using a various number of samples. Those interpolations eventually lead to noisy radiance values. The major problem of this method when trying to implement an interactive renderer, is that radiance samples are view dependent, hence they cannot be re-used from frame to frame.

To overcome this difficulty, Crespín and Péroche [Cre04] extended the *light vectors* cache and used a five dimensional cache where light vectors are computed and cached for many viewing directions. While rendering, light vectors are interpolated according to the viewer's position. However, none of these caching methods is able to provide interactive rendering, and they all require a costly first pass.

### **Progressive rendering**

Progressive rendering algorithms make it possible to bypass a long preprocessing. This comes at a cost: the first rendered images are only coarse approximations, but they may provide useful information to the user who is no more required to wait. The *light vectors* method caches samples in an object space structure. Another solution is to work in the image space. This way, Painter and Sloan [Pai89] proposed to partition the image as a 2D-tree whose leaves are recursively subdivided according to the number of pixels they cover and the variance of the samples they contain.

The idea of working in the image space was taken up by Pighin et al. [Pig97]. They create a triangulation of

the image from a set of samples taken around the discontinuities so that these discontinuities can correctly be rendered. The generated triangles are quickly rendered using the graphics hardware that interpolates between the samples. This technique is not interactive but it allows a quick preview of a scene.

The *Tapestry* method proposed by Simmons and Séquin [Sim00] provides interactivity through the use of a 2D and a half mesh. This mesh is projected onto a sphere around the observer: this way, full recomputation of the image can be avoided when the observer moves only a little. Nevertheless, geometric and lighting discontinuities remain fuzzy at the beginning of the rendering process.

### **Rendering and sampling uncoupling**

To reach interactivity, Walter et al. [Wal99] proposed to separate illumination sampling from rendering. The sampling process computes light samples and stores them in a *render cache*, while the rendering process uses those samples to generate the current one. The *render cache* keeps old samples from the previous frame that are reprojected in the current frame by the rendering process. Nonetheless, this image based method suffers from artefacts when the user views some part of the scene that has never been rendered before.

To overcome the *render cache* artefacts, Tole et al. [Tol02] proposed to compute and store shading values in the object space, more precisely at the vertices of a progressively refined mesh. Their *shading cache* is gradually filled while a process renders the image using the graphics hardware. This method does not suffer from reprojection artefacts. Even if the illumination is coarsely computed, the scene geometry and textures are rendered at full quality providing the user much more pleasant images than the *render cache*. This method uses a unique mesh to approximate radiance from different sources, leading to unnecessary sampling and limited re-use of already computed samples. Our method is greatly inspired by this last one, but as explained in the upcoming sections, we use more than one mesh to overcome the *shading cache* limitations.

Dmitriev et al. [Dmi02] also reach interactivity by separating the rendering of hardware computed direct lighting from the rendering of indirect lighting computed by path tracing. Photons are stored at the vertices of a dense mesh of the scene. Photons are packed around a *pilot photon* that has a path close to theirs. When the scene is modified, *pilot photons* are traced again through the scene to detect changed areas that need resampling.

## Hardware sampling

Graphics hardware is getting more and more programmable at each new generation, allowing new usages of GPUs. Purcell et al. [Pur02] showed that it is possible to interactively ray trace images using the GPU to compute ray triangles intersections. Global illumination can also be computed with the *photon map* algorithm [Jen96] using the GPU as proposed by Purcell et al. [Pur03]. Those methods use *fragment programs* and *render to texture* functionality of current hardware but are not really faster than the same algorithms implemented on a CPU.

Larsen and Christensen [Lar04] make a more useful usage of both the CPU and the GPU, giving each one some work. Their method separates the rendering of direct and indirect illumination. Direct illumination is computed by the hardware. Indirect illumination is computed using a *photon map*. Photons are traced on the CPU whereas the final gathering step is made on the GPU. This method reaches interactivity but according to the authors results, indirect illumination is very sparsely sampled.

## 3 OVERVIEW OF OUR METHOD

The framework we propose makes use of the irradiance cache idea. Like Zaninetti and al. [Zan98], we store each irradiance component separately. Like Tole and al. [Tol02], our sample cache is built on the geometrical mesh. This irradiance cache is filled with hardware sampled direct and indirect irradiances. The cached irradiance values are used to progressively and adaptively build and refine a radiance mesh. Final images, displayed to the user, are rendered through multiples passes, mixing radiances with the object colors. (Fig. 1) gives an overview of our framework that will be explained with more details in the next sections.

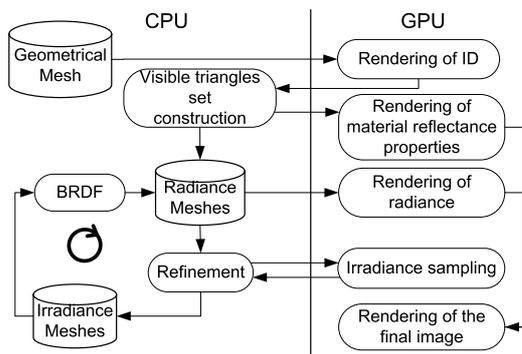


Figure 1: Overview of our framework

## 4 MULTI-MESH CACHING

Our strategy to reach interactivity is to compute as few radiance samples as possible. With this aim in view,

we need to re-use samples from frame to frame and interpolate between samples in the same frame. More than that, we want to re-use part of already computed samples. While computing indirect irradiance, the direct one is needed. Hence, we chose to split direct and indirect irradiance computations and storage in order to be able to use the direct irradiance to compute the indirect one.

## Splitting irradiance and radiance

We want to progressively compute and render samples. A recursively subdivided triangular mesh allows us to easily refine the sampled radiance field and to quickly update the image using the graphics hardware. The traditional object space cache approach is a unique irradiance cache that mixes direct and indirect irradiance, limiting their re-use and leading to unnecessary sampling. To overcome those limits, we chose to use more than one cache mesh. Moreover, we want to distinguish irradiance (energy incoming from any directions) from radiance (energy emitted in a particular direction). Irradiance is long to compute but can be re-used as long as the scene does not change, while radiance is fast to compute using cached irradiance values (see Fig. 2). We currently use:

- one direct irradiance mesh for each light source
- one indirect diffuse irradiance mesh
- one direct diffuse radiance mesh
- one direct specular radiance mesh

All those meshes are built over the same geometrical mesh. When a triangle is subdivided, it is always split in four, its edges being split in half.

We store in separate meshes the direct irradiance of each light source. This allows us to save computations when the direct irradiances of different light sources do not have the same discontinuities. In a given area, only the meshes whose irradiance contains discontinuities are refined. We use another mesh for indirect irradiance that has far less sharp discontinuities than the direct ones. The irradiance meshes can be seen as illumination maps. The direct ones contain direct shadows, while the indirect one contains color bleedings and indirect shadows. The mesh representation is more suited to store soft shadows as sharp discontinuities lead to deeper subdivision, but it is nevertheless able to handle sharp shadows of point light sources.

What we need to render is the radiance emitted by the seen objects that reaches the eye of the observer. Radiance can be divided in two parts: direct radiance and indirect one. Using a separable BRDF model, direct radiance can also be split in two parts: the diffuse one and the specular one. The diffuse part of direct radiance does not change when the user moves. Thus it can be computed once, cached in the vertices of a

mesh and re-used for many different frames. We use a mesh to progressively compute the specular part of direct radiance for each frame, starting from scratch when the observer moves. Indirect radiance is very long and difficult to compute if all kinds of light paths are taken into account. We currently only take into account indirect diffuse radiance that can be directly computed, on the fly while rendering, from indirect diffuse irradiance, so we don't use another mesh for indirect irradiance.

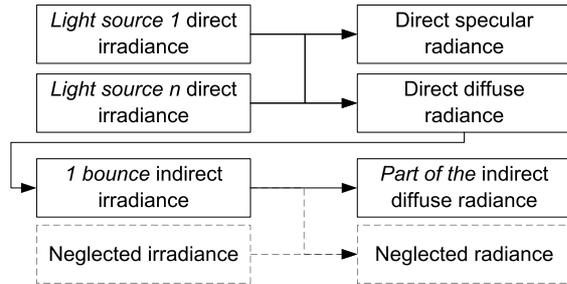


Figure 2: From irradiance to radiance

## From irradiance to radiance

We chose the modified Phong BRDF [Laf94] for its simplicity and its energy conservation property, but other separable BRDFs could be used. The radiance at point  $x$  incoming from direction  $\vec{\omega}_r$  we need to compute and display is

$$L_d(x, \vec{\omega}_r) = \int_{\Omega} f_d(x, \vec{\omega}_r, \vec{\omega}_i) L(x', \vec{\omega}_i) \cos\theta \, d\omega \quad (1)$$

$$L_s(x, \vec{\omega}_r) = \int_{\Omega} f_s(x, \vec{\omega}_r, \vec{\omega}_i) L(x', \vec{\omega}_i) \cos\theta \, d\omega \quad (2)$$

where  $f_s$  and  $f_d$  are the diffuse and specular components of each object BRDF.

In the direct irradiance mesh, we store for each light source  $ls$  of area  $S$  the irradiance

$$E_d(x, ls) = \int_{ls} L(x', \vec{\omega}_i) \cos\theta \cos\theta' \frac{dS}{r^2} \quad (3)$$

The visibility of the light source is taken into account in the computation of  $E_d(x, ls)$ . Using the modified Phong BRDF, diffuse direct radiance can easily be computed from the irradiance cached for each light in the irradiance mesh:

$$L_{dd}(x, \vec{\omega}_r) = \frac{1}{\pi} \sum_{lights} E_d(x, ls) \quad (4)$$

To compute the specular part of the direct radiance we should integrate:

$$\frac{n+2}{2\pi} \cos^n \alpha L(x', \vec{\omega}_i) \cos\theta \quad (5)$$

over the solid angle sustained by each light source, where  $n$  is the specular exponent and  $\alpha$  the angle between the perfect specular reflection of the light source center and the outgoing direction  $\vec{\omega}_r$ . This would be too costly for interactive rendering. We want to re-use already computed samples, so we use the irradiance stored in the irradiance mesh for each light source and multiply it by the specular part of the BRDF evaluated at the center of the light source:

$$L_{ds}(x, \vec{\omega}_r) = \sum_{lights} \left( \frac{n+2}{2\pi} \cos^n \alpha E_d(x, ls) \right) \quad (6)$$

This simplification comes down to replacing area light sources with point light sources for specular radiance estimation. The resulting errors can be misplaced and wrong shaped specular highlights.

Sampling all the components of indirect irradiance is currently too costly, we only sample a part of diffuse indirect irradiance: light paths that diffusely bounce only once between the light source and the point of interest. Paths with two bounces could easily be added using the one-bounce indirect irradiance being computed; longer paths can often be neglected since in a directly lit scene, their contributions are very small. This method is biased but provides visually satisfying images. Diffuse indirect irradiance is approximated using the diffuse direct radiance:

$$E_i(x) = \int_{\Omega} L_{dd}(x', \vec{\omega}_i) \cos\theta \, d\omega \quad (7)$$

To compute the diffuse indirect radiance, we just compute:

$$L_{id}(x) = \frac{1}{\pi} E_i(x) \quad (8)$$

## Mesh subdivision

The first image displayed is rendered using radiance and irradiance that are only computed at the vertices of the radiance meshes roots. These roots are in fact the geometrical mesh elements of the scene. The radiance meshes are then progressively refined to take into account radiance discontinuities. Each time a radiance is computed, irradiance values are fetched, for each light source, in the corresponding irradiance meshes. When those values are not available or cannot be confidently interpolated, the irradiance meshes are subdivided. The radiance meshes refinement guides the direct irradiance meshes ones. We will explain in the next subsections how our meshes are subdivided.

### 4.3.1 Radiance mesh subdivision criteria

Each mesh element is a triangular patch. A radiance patch may be subdivided only if it lies over more than

one pixel. To decide if a mesh element should be subdivided, each of its edges is split into two equal parts. Two radiances are evaluated at its middle: one is computed and the other is interpolated. If the computation and the interpolation give a close result, the edge will not be subdivided any more. When at least one edge of a triangle has been subdivided because it contains a discontinuity, the triangle is split into four triangles. This avoids visible T-vertices, since triangles on both sides of the subdivided edge will be split. A size criterion is required in order not to miss small radiance discontinuities: triangles are subdivided until they cover a small number of pixels. Once triangles have been subdivided enough to meet the size criterion without finding any discontinuities, the unnecessary subdivisions are undone to save memory space.

#### 4.3.2 Noticeable color differences

To compare interpolated and sampled radiances at the middle of an edge, we need a criterion that takes into account the user perception. The maximum unnoticeable difference between the two values depends on the radiance of the rendered pixel on screen, the user visual system, the monitor settings and its environment. Modeling this whole chain is by itself a research area; we wanted something simple. The problem we ran on was that a lot of useless mesh subdivisions occurred in dark areas of the scene. On our monitor, we cannot distinguish a black (`0x000000`) patch from a lighter black (`0x0A0A0A`) one. Our algorithm has to take this fact into account. Supposing the user, the environment and the monitor do not change, we experimentally generate a map that associates a maximum unnoticeable difference value for a set of radiances. To compare interpolated and sampled radiance, we first process these values with the tone mapping algorithm to get an idea of the color values that will be rendered on the screen. Then we compute the difference and compare it with the corresponding value in our map. A real visual model would give far more accurate values but at the expense of a high computation cost. Our solution gives acceptable results at almost no cost.

#### 4.3.3 Priorities

To provide the best quality images in the shortest time, the radiance mesh elements are given subdivision priorities. These priorities depend on the radiance difference between the element vertices and on the element visible size: a high contrast over an element is a good hint for radiance discontinuities and the bigger an element is, the more chance it has to contain discontinuities. We use the visible size of the triangle because we don't want to subdivide hidden triangles. The mesh can be seen as a set of quad-trees, each quad-tree being built over a triangle of the geometrical mesh. The

priorities are computed at the leaves of the quad-trees and are spread to their root. The priority of a node is the maximum of its sons priority. This way, given a geometrical mesh triangle, we can quickly find its radiance element leaf that requires to be subdivided first.

To decide which geometrical triangle will be subdivided, we could sort the elements or use a hit and test method as in [Tol02]. Sorting is too slow and the hit and test method requires many tests. Instead, we developed the following algorithm. We start the subdivision process with a big triangle which has a quite high priority. Then we randomly pick a triangle; if the picked triangle has a higher priority than the last subdivided one, we switch to the picked one, otherwise we keep refining the same triangle until its priority falls under the priority of the next randomly picked triangle.

#### 4.3.4 Irradiance mesh subdivision

The subdivision of the direct irradiance meshes is guided by the subdivision of the radiance mesh described before. Each time a direct radiance value needs to be computed, the irradiance of each light source is fetched in the irradiance mesh. If the direct irradiance mesh is not subdivided enough to provide the requested value, it is subdivided at this time. The irradiance mesh subdivision is limited by the subdivision of the radiance one, thus an element will not be indefinitely subdivided.

Diffuse indirect radiance and diffuse indirect irradiance are directly linked (equation 8). As the radiance is computed to be displayed, we use the same view dependent criteria that was used to subdivide the radiance meshes.

## 5 SAMPLING

Our mesh-based progressive rendering approach is very sensitive to noise. Noisy samples may lead to needless subdivisions of the meshes. To quickly render high quality images, we need to compute high quality irradiance values to fill our irradiance caches. As told earlier, we sample direct irradiance from each light source and indirect irradiance separately. Traditionally, ray tracing was used to collect irradiance. Area light sources were sampled using hundreds of rays to determine their visibility from the point to shade. Indirect irradiance was collected by sampling an hemisphere built over the point to shade. These methods provide good results but are often too slow to be useful in interactive rendering on a single CPU. Furthermore those methods monopolize the CPU and make no use of the GPU. Using the GPU as a SIMD coprocessor to compute ray object intersections is feasible but at a high cost: ray packing and asynchronous

results handling. Our idea is to use the GPU in a more regular way.

### Area light sources sampling

We propose to take advantage of the efficient visibility determination capacity of the graphics hardware to obtain a high quality estimation of the irradiance of an area light source at a given point. We render the scene observed from the point to shade, clipping the viewing frustum to a small frustum that fully includes the area light source.

Assuming that the light sources are isotropic, their radiance is the same all over their surface. We need to compute equation (3). Our method uses a fragment program to compute accurate values:  $\cos\theta$  and the solid angle of the pixel are evaluated for each pixel. To sum the fragment program output values represented and stored as floats in a pixel buffer, we use two pixel buffers to progressively reduce the image until all the values are accumulated in a small enough image (16x16) that can be quickly read back to the CPU.

When the scene is rendered from the point to shade, the whole scene does not need to be sent to the GPU, since only a few triangles in the viewing frustum can occlude the light source. We use a grid to store the scene triangles. Computing the exact intersection of the frustum and the grid could be quite time consuming, so we chose to approximate the frustum with a carefully chosen bounding cone and each grid cell with a bounding sphere. To know if the cone intersects a grid cell, only one test is required: does the grid cell center belongs to the bounding cone? (see Fig. 3). This method is conservative, a few grid cells are being falsely detected as crossed by the frustum, but none are forgotten.

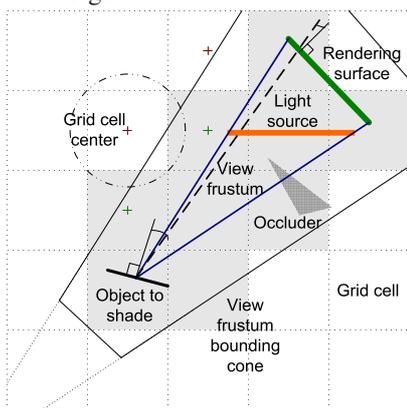


Figure 3: Area light source sampling

### Indirect irradiance sampling

Sampling indirect irradiance through an hemisphere using a ray tracing method is far more time consuming than sampling the direct one. More rays are needed

and those rays are not as coherent as those sent when sampling direct irradiance over an area light source. This makes CPU SIMD optimizations like those proposed by Wald et al. [Wal01] less efficient. As for sampling direct irradiance, we chose to use the graphics card high visibility determination capacity.

As said earlier, we currently only take into account in our indirect irradiance computations light paths with one diffuse bounce. To gather this indirect irradiance, as Larsen and Christensen [Lar04], we chose to render the scene only once on a plane. Using a field of view of 160 degrees, we forget only 2% of the incoming radiance. Rendering the whole diffuse radiance mesh would be very costly and useless. If the scene is sampled using a 128x128 image, small triangles in the radiance mesh won't be visible. We chose to send an undersampled version of the radiance mesh to the GPU to increase speed. This coarse version of the direct diffuse radiance mesh has to be built for the whole scene before starting to sample indirect radiance since any part of the scene could indirectly contribute to the indirect radiance of a visible geometrical mesh element. The energy is collected in the image by summing it with two puffers as we did for direct irradiance. Again, the value of each pixel has to be weighted with the solid angle covered by the pixel and with the  $\cos\theta$  term of the irradiance formula. This time, the solid angle of field of view is constant so the weight of each pixel can be precomputed and stored in a texture.

## 6 MULTI-PASS RENDERING

Generating an image with our method is done through multiple rendering passes. In a first pass, we only render the geometrical mesh using an identifier (a 32 bits address, or an index in an array) for each triangle as color. The generated image is read back to the CPU. This is costly, but this allows us to compute the visible size of each triangle and to build the set of the visible triangles. The occlusion query extension could have been used to avoid the read back, but to get the exact number of visible pixels of each triangle, two rendering passes would have been needed: the first one to initialize the z-buffer, the second one to count visible pixels.

Using the visible triangles set as input, the second pass generates two images containing the material reflectance properties of each triangle. We split in two images the diffuse and specular properties. These material reflectance properties are the diffuse or specular colors of each triangle times their diffuse or specular reflectance coefficient. Those two images can be generated in a single pass using multiple output buffers or packing the two images in a single output texture.

With the number of visible pixels computed in the first pass, we update the priority of each radiance mesh element. This radiance mesh can be rendered in a third pass. Actually, two passes are needed to render the diffuse and the specular direct radiance meshes, and one more pass is needed to render the diffuse indirect irradiance one whose irradiances are transformed on the fly in radiances according to equation (8).

A final pass is needed to merge all the material properties images and the radiance ones. This last pass is very fast, it requires to render only a single quad textured with the five images computed in the preceding passes. According to the number of texture units available on the graphics card, more than one pass may be needed. The fragment program used in this last pass includes a tone mapping algorithm to convert the radiance of each pixel into a displayable color.

This image decomposition may seem costly, but it allows to progressively update the image almost as fast as new samples are computed. When the user keeps looking at the same frame, the mesh subdivision thread works at full speed. Each time a radiance mesh element is subdivided, the four triangles created are rendered on the image corresponding to the subdivided mesh (direct diffuse, direct specular or indirect diffuse). The final pass that recombines all the radiance components and the object reflectance is applied at constant time rate to update the image displayed to the user.

## 7 RESULTS

We tested our method on a P4 2.5 GHz - 768MB of RAM - Nvidia 256MB QuadroFX3000. The following results are computed on a small scene of 7000 triangles with one area light source. About fifteen seconds are required to compute the coarse direct radiance mesh. During this time a constant indirect radiance value is used while direct radiance is progressively subdivided. Then, direct and indirect radiance meshes are simultaneously refined and the image is progressively updated.

The first rendering pass that requires a read back to the CPU of the rendered triangles identifier is the costliest. Working with a 640x480 image requires 40ms. The material reflectance property and radiance rendering passes are faster: about 12ms each. The final pass that merges the luminance and material reflectance images is also fast: 10ms. Globally, 100ms are required to obtain an image using already computed samples.

On our test scene, the coarse direct radiance mesh used to compute the indirect one contains 9200 samples. To obtain the image (Fig. 4) 26000 direct and 9000 indirect samples are required. The total number of computed samples is linked to the subdivision criteria.

The direct irradiance sampling requires two phases: rendering the objects that might be in the point to area light source frustum and summing the irradiance of each pixel. The current read back limits make the second phase the costliest one (about 70% of the total time). Using a 64x64 image to sample direct irradiance requires about 0.9ms. Sampling the light source with a ray tracer with the same density requires 32ms. With a ray tracer, less rays are needed since non uniform sampling (that provides less banding artifacts) can be used. Ray tracing with 16x16 rays requires 2.6ms, which is the cost of hardware sampling using a 128x128 image. Hardware sampling is a lot faster than ray tracing and it frees the CPU that can be used for other tasks, like mesh subdivision, while samples are computed. Indirect irradiance sampling is a bit longer as all the scene is sent to the GPU. 9ms are required for each sample using a 256x256 pixels image.

## 8 DISCUSSION AND FUTURE WORK

We think our method has interesting advantages over similar ones. It does not require preprocessing as *photon map* based methods do. Sampling radiances at the vertices of a mesh avoids the costly interpolations of *light vectors* methods or the density estimations of the *photon map*. Our caching method, splitting irradiances and radiances, allows more re-use of already computed values. Our area light source sampling method, that uses different meshes to progressively compute each light source shadow, limits the number of computed samples to a minimum. We chose not to sample direct radiance using traditional interactive GPU based methods (shadow maps or shadow volumes) that might be faster, but would not allow to store the computed radiance in RAM for use in the indirect radiance computations. Our sampling method is not yet able to produce real time soft shadows nor real time indirect illumination, but it is a lot faster than ray tracing based methods. Upcoming GPU extensions might allow us to keep the direct radiance on the GPU board avoiding the GPU to CPU transfer bottleneck.

The strong link between the caches and the geometry can be criticized for its lack of freedom in the scene representation, but this is the key for interactive speed. The major problem of our solution is its memory cost. Rendering large scene containing very tessellated objects is a problem since we construct our meshes over the geometrical one. A solution we are working on is the construction of the radiance meshes on the object bounding boxes, building cubemaps to render the objects radiances. Other currently missing features we are working on include specular reflections, caustics and dynamic scene handling. We anticipate that our use of different meshes should help us to notice

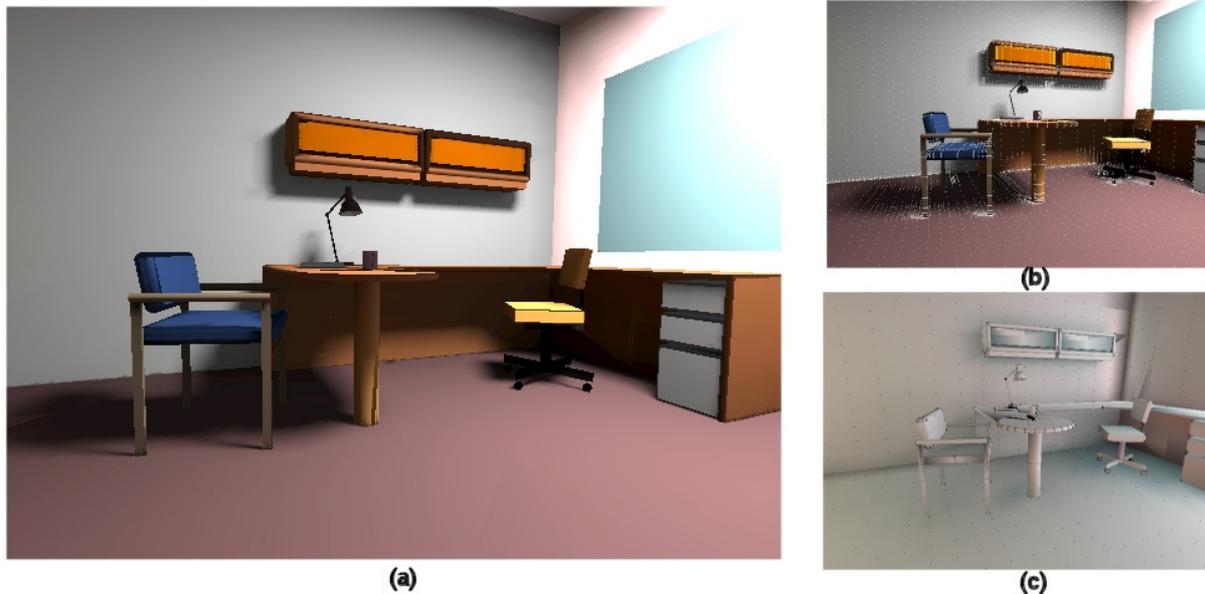


Figure 4: (a) Final image (b) Direct radiance samples (c) Indirect radiance image and samples

changes in dynamic scene irradiance, allowing us to quickly update our irradiance caches. The use of a visual model to guide our mesh subdivision is another research area we want to explore.

## 9 CONCLUSION

Real time rendering of large scenes with soft shadows, indirect illumination and caustic is not achieved yet. We think our multi-mesh method is an interesting step towards this goal. The fast increase in performance and functionality of GPUs will offer us new possibilities. In our mind, porting to the GPU existing CPU algorithms does not take full advantage of the graphics hardware. GPUs are fast for geometry rasterization and interpolations; its limited programming and memory model are quickly evolving, we have to foresee new algorithms to exploit those upcoming capacities to compute shadows and global illumination.

## References

- [Cre04] Crespín, R., and Péroche, B. Lights Vectors for a Moving Observer. In *12-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG), Plzen, Czech Republic*, pages 99–96, Plzen, Czech Republic, 2-9 february 2004. University of West Bohemia.
- [Dmi02] Dmitriev, K., Brabec, S., Myszkowski, K., and Seidel, H.P. Interactive global illumination using selective photon tracing. In *Rendering Techniques 2002: 13th Eurographics Workshop on Rendering*, pages 25–36, June 2002.
- [Jen96] Jensen, H.W. Global illumination using photon maps. In *Eurographics Rendering Workshop 1996*, pages 21–30, June 1996.
- [Laf94] Lafortune, E.P., and Willems, Y.D. Using the Modified Phong BRDF for Physically Based Rendering. Technical Report CW197, Leuven, Belgium, 1994.
- [Lar04] Larsen, B.D., and Christensen, N. Simulating photon mapping for real-time applications. In *Eurographics Symposium on Rendering*, jun 2004.
- [Pai89] Painter, J., and Sloan, K. Antialiased ray tracing by adaptive progressive refinement. In *Computer Graphics (Proceedings of SIGGRAPH 89)*, volume 23, pages 281–288, July 1989.
- [Pig97] Pighin, F.P., Lischinski, D., and Salesin, D.H. Progressive previewing of ray-traced images using image plane discontinuity meshing. In *Eurographics Rendering Workshop 1997*, pages 115–126, June 1997.
- [Pur02] Purcell, T.J., Buck, I., Mark, W.R., and Hanrahan, P. Ray tracing on programmable graphics hardware. *ACM Transactions on Graphics*, 21(3):703–712, July 2002.
- [Pur03] Purcell, T.J., Donner, C., Cammarano, M., Jensen, H.W., and Hanrahan, P. Photon mapping on programmable graphics hardware. In *Graphics Hardware 2003*, pages 41–50, July 2003.
- [Sim00] Simmons, M., and Séquin, C.H. Tapestry: A dynamic mesh-based display representation for interactive rendering. In *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, pages 329–340, June 2000.
- [Tol02] Tole, P., Pellacini, F., Walter, B., and Greenberg, D.P. Interactive global illumination in dynamic scenes. *ACM Transactions on Graphics*, 21(3):537–546, July 2002.
- [Wal01] Ingo Wald, Philipp Slusallek, Carsten Benthin, and Markus Wagner. Interactive rendering with coherent ray tracing. *Computer Graphics Forum*, 20(3):153–164, 2001.
- [Wal99] Walter, B., Drettakis, G., and Parker, S.. Interactive rendering using the render cache. In *Eurographics Rendering Workshop 1999*, pages 19–30, June 1999.
- [War88] Ward, G.J., Rubinstein, F.M., and Clear, R.D. A ray tracing solution for diffuse interreflection. In *Computer Graphics (Proceedings of SIGGRAPH 88)*, volume 22, pages 85–92, August 1988.
- [Zan98] Zaninetti, J., Serpaggi, X., and Péroche, B. A vector approach for global illumination in ray tracing. *Computer Graphics Forum*, 17(3):149–158, 1998.

# Interactive Ray Tracing of Trimmed Bicubic Bézier Surfaces without Triangulation

Markus Geimer

Oliver Abert

Institute for Computational Visualitics  
University of Koblenz-Landau  
Universitätsstraße 1  
D-56070 Koblenz, Germany

mgm@uni-koblenz.de

abert@uni-koblenz.de

## ABSTRACT

By carefully exploiting the resources of today's computer hardware, interactive ray tracing recently became reality even on a single commodity PC. In most of these implementations triangles are used as the only geometric primitive. However, direct rendering of free-form surfaces would be advantageous for a large number of applications, since robust tessellation of complex scenes into triangles is a very time-consuming process. Additionally, scenes consisting of free-form surfaces require less memory and provide a much higher precision resulting in less rendering artifacts.

In this paper, we present our implementation of an efficient and robust algorithm for rapidly finding intersections between rays and trimmed bicubic Bézier surfaces. Using SIMD instructions provided by many of today's CPUs, we perform the intersection test of a packet of four rays with a single Bézier surface in parallel. An optimized bounding volume hierarchy provides good initial guesses needed for fast convergence of the Newton iteration, which forms the core of our intersection algorithm. As a result, we demonstrate that it is feasible to render complex scenes of several thousand Bézier surfaces at video resolution with interactive frame rates on a single PC.

## Keywords

Interactive Ray Tracing, Bézier Surfaces, Trimming

## 1. INTRODUCTION

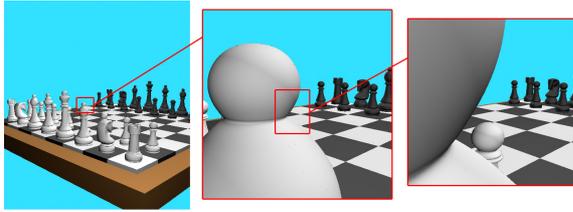
Free-form surface representations such as splines, NURBS, or subdivision surfaces provide a simple but still powerful way of describing three-dimensional geometrical objects for use in computer graphics applications. Unlike triangle meshes, which are the second commonly used way of defining 3D shapes, they are able to describe curved surfaces exactly. Therefore, free-form surfaces form the foundation of most CAD systems used in the industry today.

If the models should be displayed in an interactive setting, however, free-form surfaces are currently tessellated into triangles as well, since this is the only primitive that can be handled by today's rasterization hardware. For this kind of applications it would therefore be desirable to render free-form surfaces directly.

Direct rendering of free-form surfaces instead of triangle meshes has a number of advantages. Obviously, the time-consuming overhead of triangulating the surfaces can be avoided. Secondly, due to the smaller number of primitives the costs for additional preprocessing needed for most rendering algorithms, e.g. building up acceleration data structures, are also reduced. Moreover, representing objects as free-form surfaces requires less memory, which can be a limiting factor for complex scenes. In addition, rendering free-form surfaces directly provides a much higher precision resulting in less rendering artifacts. For example, cracks between adjacent surfaces due to different tessellation parameters can be completely avoided.

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 2005 conference proceedings ISBN 80-903100-7-9*  
*WSCG'2005, January 31-February 4, 2005*  
*Plzen, Czech Republic.*  
Copyright UNION Agency – Science Press



**Figure 1. The Chessboard from different view points. Note that due to the direct rendering of free-form surfaces, the objects remain curved, even from the shortest viewing distance.**

Finally, free-form surfaces are often used in conjunction with trimming curves that cut out parts of the surface in the parametric domain. Robust tessellation of free-form surfaces with trimming curves is a non-trivial task with a high computational cost.

In contrast to current rasterization hardware, ray tracing is capable of rendering every primitive for which the intersection between a ray and the surface can be calculated [Whi80a]. Nevertheless, it has the reputation of being a very time-consuming rendering algorithm. However, recently it has been shown that it is possible to achieve interactive frame rates even on a single commodity PC by carefully exploiting the resources of today’s CPUs [Wal01a].

In this paper, we present the details of our implementation of the intersection test between a ray and a trimmed bicubic Bézier surface in the context of an interactive ray tracing system. While we do not introduce any new intersection algorithm, we rather show that a significant speed-up can be achieved by carefully optimizing a well-known intersection technique. As a result, we demonstrate that it is feasible to render complex scenes of several thousand Bézier surfaces at video resolution with interactive frame rates on a single PC.

Although we currently restrict ourselves to bicubic Bézier surfaces, our approach may be extended to support other parametric surfaces as well. In addition, more general spline surfaces such as NURBS can be converted into Bézier patches [Roc89a].

## 2. PREVIOUS WORK

Ray tracing at interactive frame rates has been first presented by [Muu95a]. Later, [Par99a] described a full-featured interactive ray tracing system running on a shared-memory supercomputer that is able to handle arbitrary geometry, including parametric surfaces (e.g. NURBS). However, they had to use expensive high-end hardware to achieve this goal.

By contrast, [Wal01a, Wal01b] presented a highly optimized ray tracer running on a cluster of

commodity PCs, paying careful attention to data layout, coherence, and caching issues. In addition, their system extensively uses SIMD instructions provided by most of today’s CPUs to trace packets of rays in parallel, thereby achieving a speed-up of more than an order of magnitude compared to other well-known ray tracers. Meanwhile, the employed algorithms have been further improved [Wal04a]. However, this ray tracer is restricted to triangles as the only geometric primitive.

In the last 25 years, a variety of algorithms has been proposed to calculate the intersections between a ray and parametric surfaces of different kinds. One common approach is to use a multivariate Newton iteration. This method has the advantage of being general enough to handle any parametric surface, but requires a good initial value to ensure correctness and fast convergence.

For example, [Swe86a] refine the control meshes of B-spline surfaces until they closely approximate them. Then, the intersection of the ray and the control mesh is used as the initial value of the following Newton iteration. By contrast, [Mar00a] employ a hierarchy of bounding volumes enclosing disjoint regions of NURBS surfaces to yield a suitable initial value.

Another numerical approach for Bézier surfaces is called Bézier Clipping [Nis90a]. This algorithm tries to iteratively identify regions of the patch that are known not to be intersected by the ray, thereby restricting the parameter domain where intersections can occur. This approach is also used by [Wan01a], who combine Bézier Clipping with Newton iteration. Additionally, they exploited the coherence of neighboring rays to speed up the calculation. Nevertheless, all approaches mentioned above were far from interactive.

Recently, [Ben04a] presented their implementation of a subdivision method, which refines the control meshes of the surfaces on-the-fly and calculates an approximate intersection point using a triangle mesh generated from the control points. Depending on the model and the number of refinement steps, they achieve up to 5.5 fps at video resolution on a single PC. Nevertheless, the number of refinement steps has to be the same for all surfaces to avoid cracks between adjacent patches.

## 3. SYSTEM OVERVIEW

Before going into the details of our implementation of the intersection test between a ray and a bicubic Bézier surface, we present a brief overview of the underlying interactive ray tracing system.

Similar to [Wal01a], we use SIMD instructions found in many of today’s CPUs to trace packets of

four rays in parallel. This applies to both the traversal of an acceleration data structure as well as the actual intersection calculations. However, instead of targeting only at a single CPU architecture, we have implemented our ray tracing system on top of a SIMD abstraction layer that allows us to write platform-independent SIMD code (see [Gei03a] for details). Currently, Intel's SSE [Int04a] and Motorola's AltiVec [Mot99a] instruction sets are supported, as well as a special mode that uses the FPU to emulate the specified functionality.

In order to achieve a good rendering performance, it is crucial to reduce the total number of intersection calculations to a minimum. Therefore, we employ a hierarchy of axis-aligned bounding boxes that is iteratively traversed in depth-first order [Smi98a]. Although other acceleration data structures are usually considered to be faster in the context of ray tracing, we have found that this approach is well suited for a SIMD implementation and also adapts well to our intersection algorithm presented below.

Currently, our ray tracing system is restricted to static scenes, allowing only interactive walk-throughs. However, it could be easily extended to support dynamic scenes with hierarchical movements as well using the ideas presented in [Lex01a] and [Wal03a].

Because ray tracing naturally lends itself to a parallel implementation, our system is no exception. At present, we support rendering with multiple threads, which allows us to take advantage of multiprocessor PCs as well as Intel's HyperThreading technology [Int04b].

#### 4. OUR APPROACH

Instead of using complex algorithms, we rather take a "brute force" approach. While this doesn't seem to be very clever at first sight, it has been shown that a carefully optimized implementation can easily outperform more complex algorithms on current CPU architectures (e.g. [Wal01a] or [Ben04a]).

Our intersection algorithm combines many known techniques to achieve a fast computation of the intersection point of a ray and bicubic Bézier patches as well as the corresponding surface normals. Similar to [Mar00a], we employ a hierarchy of axis-aligned bounding boxes to find a suitable initial guess needed for the Newton iteration that is used to calculate the intersection point. In the following subsections, we will describe the core components of our approach.

##### Preprocessing

As a first step, the scene description file is read (currently, the IGES format is partially supported) and converted into a binary data file that can then be processed by the actual rendering application.

During this preprocessing, we convert the trimming curves originally represented as B-splines into piecewise cubic Bézier curves for use with our trimming algorithm and build up the bounding volume hierarchy that is used to speed up the intersection calculation.

Compared to other bounding volume hierarchies, on the lowest level our bounding boxes do not surround entire scene objects (i.e. Bézier surfaces) or even lists of objects. Instead, we create them for small, disjoint regions of the individual surfaces. On the one hand, this leads to tighter bounds, thereby reducing the number of unnecessary intersection calculations. On the other hand, we are able to get better initial guesses for the Newton iteration used to calculate the actual intersection point.

Therefore, we recursively subdivide each Bézier surface into a larger number of subpatches and calculate their corresponding control points using de Casteljau's algorithm. Due to the convex hull property of Bézier surfaces, we can use these control points to determine an axis-aligned bounding box for each subpatch.

At present, we alternately subdivide the surfaces resp. subpatches at the mean of the parameter domain in  $u$  and  $v$  direction, until the generated patches are reasonably flat or a predefined maximum subdivision depth has been reached. These two parameters can be easily controlled by the user. Unfortunately, they are scene dependent and must be chosen with care. For our test scenes, we have found that a maximum subdivision depth of 4-6 is already sufficient.

Since our Bézier surfaces are subdivided into small, disjoint regions by the bounding volume hierarchy, the individual subpatches can be classified during preprocessing to improve the rendering performance of trimmed surfaces. For each region that is known to be enclosed by a trimming curve (i.e. considered not to be part of the patch), we can immediately discard the corresponding subpatch as well as its associated bounding box, thereby avoiding any intersection calculation. By contrast, for regions that are known to lie completely inside the patch, the trimming test can be skipped during rendering. Otherwise, at least one trimming curve cuts out a part of the region and we have to perform the entire trimming calculation.

As we use the original Bézier surface for the actual intersection test, the control points of the subpatches are only needed for creating the bounding boxes and the aforementioned classification, so they can be discarded immediately afterwards.

Finally, the bounding volume hierarchy is created from the bounding boxes of the remaining subpatches using the top-down approach presented by [Gol87a].

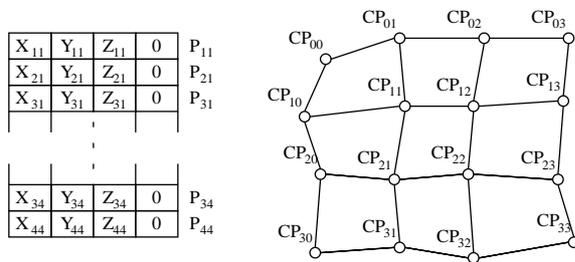
## Patch Representation and Data Layout

A bicubic Bézier surface is given by its 16 control points and can be represented in matrix form through

$$S(u, v) = [U][N][CP][N][V]^T.$$

Here, the 4x4 matrix  $[CP]$  stores the control points,  $[N]$  contains the Bernstein polynomial coefficients, and  $[U] = [u^3 \ u^2 \ u \ 1]$  resp.  $[V] = [v^3 \ v^2 \ v \ 1]$ . Obviously, the inner product  $[P] = [N][CP][N]$  can be computed in advance.

We store the 16 vector components of  $[P]$  as SIMD data in an array (see Figure 2). The first three elements of each SIMD variable store the X, Y, and Z component of the corresponding matrix element, whereas the last component remains unused. With this approach, each Bézier surface can be represented by exactly  $16 * 4 * \text{sizeof}(\text{float}) = 256$  bytes.

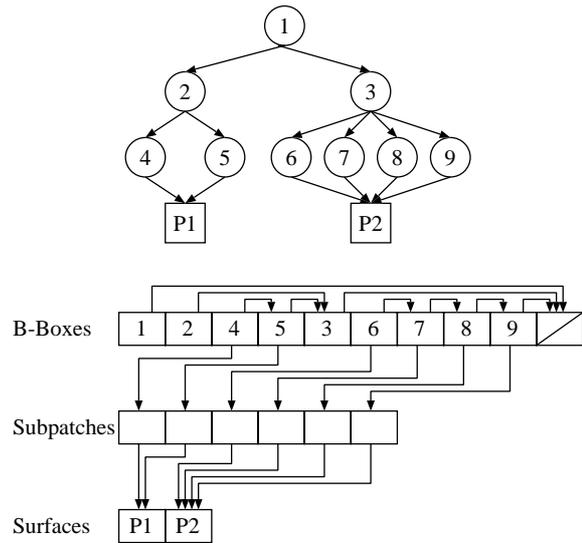


**Figure 2. Patch representation: The matrix  $[P]$  computed from the control points  $CP_{ij}$  is stored column-major in an array of SIMD variables, leaving one component of each element unused.**

Storing the patch data as a structure-of-arrays, i.e. storing the same components of four control points together in a single SIMD variable, would reduce the memory requirements by 64 bytes per surface, but during our experiments we have found that this data layout results in a significantly slower evaluation algorithm due to the necessary shuffling of data. Moreover, the fourth component of each element in our SIMD data array can be used to store additional patch information, e.g. a pointer to the associated list of trimming curves.

As already mentioned before, the subpatch data structure does not contain any control point information. Here we only store the parameter domain of the patch, the classification flag indicating whether it has to be trimmed or not, and the pointer to the original Bézier surface. For alignment reasons, we pad this data structure to a total size of 32 bytes.

For each bounding box, we store the minimum and maximum coordinate value along each axis, a pointer to the associated geometry (i.e. a subpatch), and the skip pointer used during the traversal of the hierarchy. This data structure also occupies 32 bytes.



**Figure 3. Data layout: During preprocessing, the example hierarchy shown at the top is converted into the internal representation (a set of arrays) shown at the bottom.**

## Rendering Core

As already stated in section 3, our rendering core traces packets of four rays in parallel using SIMD instructions. First, each ray packet generated by the camera iteratively traverses the bounding volume hierarchy to restrict the number of intersection candidates. Here, if any ray of the packet hits a bounding box, the entire packet has to continue the traversal of its children.

Since the leaf nodes of our hierarchy correspond to small surface regions and not to an entire Bézier patch, this approach does not only restrict the number of intersection candidates, but also the parametric domain where intersections may occur. Provided that a proper hierarchy has been created, the center of the enclosed parametric domain can then be used as an initial guess for a Newton iteration that calculates the actual intersection point between the ray and the surface. Of course, this can also be done for four rays in parallel using SIMD instructions.

### 4.3.1 Intersection Test

The core of the intersection test is similar to the approach presented by [Mar00a] who solved the problem for NURBS surfaces but without targeting interactivity.

We represent each ray by two orthogonal planes  $P_1 = (N_1, d_1)$  and  $P_2 = (N_2, d_2)$  where the  $N_i$  are orthogonal vectors of unit length, perpendicular to the ray direction  $D$ . The  $d_i$  are given by  $d_i = -N_i \circ O$ . Here,  $O$  denotes the origin of the ray. To find the intersection point between the ray

and a parametric surface  $S(u, v)$ , we have to solve for the roots of

$$R(u, v) = \begin{bmatrix} N_1 \cdot S(u, v) + d_1 \\ N_2 \cdot S(u, v) + d_2 \end{bmatrix}.$$

Several numerical methods exist that could be used to target this problem. We have chosen the classical approach of Newton iteration for several reasons: First, it converges quadratically if the initial guess is close to the actual root, which can be assured by our bounding volume hierarchy. Secondly, the surface derivatives exist and are very easy to compute. And last but not least, this algorithm is well suited for an implementation using SIMD instructions.

Basically, Newton's method is a Taylor series which is truncated after the first derivative. As we are solving a two-dimensional problem, the Newton step is defined as

$$\begin{bmatrix} u_{n+1} \\ v_{n+1} \end{bmatrix} = \begin{bmatrix} u_n \\ v_n \end{bmatrix} - J^{-1} \cdot R(u_n, v_n),$$

where  $J$  is the Jacobian matrix of  $R$  which is given by

$$J = \begin{bmatrix} N_1 \cdot S_u(u, v) & N_1 \cdot S_v(u, v) \\ N_2 \cdot S_u(u, v) & N_2 \cdot S_v(u, v) \end{bmatrix}.$$

Here,  $S_u$  and  $S_v$  denote the partial derivative in the corresponding parametric direction. The inverse of the Jacobian can be efficiently computed using the submatrices  $J_{ij}$  of  $J$  that remain when the  $i$ th row and the  $j$ th column are removed:

$$J^{-1} = \frac{1}{\det(J)} \cdot \begin{bmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11} \end{bmatrix}.$$

We continue the iteration until one of three criteria is met: An intersection between the ray and the surface is found, if and only if we are closer to the root than some user defined threshold  $\varepsilon$

$$|R(u_n, v_n)| < \varepsilon.$$

Otherwise, the iteration continues until either this threshold criterion is met, the next iteration takes us further away from the root, i.e.

$$|R(u_{n+1}, v_{n+1})| > |R(u_n, v_n)|,$$

or a maximum number of iterations has been performed.

Unfortunately, since we employ SIMD instructions to calculate four intersections at once, the iteration can be stopped only if all rays of a packet meet any

of these criteria. However, this is still more than three times faster than computing the intersections sequentially.

#### 4.3.2 Evaluation

The Newton iteration often needs to evaluate surface points as well as partial derivatives for given parameter values  $(u, v)$ . In this section we present a way how these can be computed efficiently.

Basically, we have to compute the product of three matrices. That is

$$S(u, v) = [u^3 \ u^2 \ u \ 1][P][v^3 \ v^2 \ v \ 1]^T.$$

A naive implementation would simply compute the two matrix products, which consists of 60 multiplications and 45 additions ( $[P]$  contains three-dimensional vectors!), ignoring the operations needed to compute of  $[U]$  and  $[V]$ . However, as the surface evaluation is used very often, it is necessary to optimize it as much as possible.

First of all, the equation above can be rewritten as

$$S(u, v) = \sum_{i=1}^4 [U][P_i][V_i],$$

where  $[P_i]$  denotes the  $i$ th column of  $[P]$  and  $[V_i]$  the  $i$ th row of  $[V]$ . Note that  $[V_i]$  represents only a single floating-point.

Given the parameters  $(u, v)$ , we can then evaluate a point  $S(u, v)$  on the surface using the following C-like pseudo code fragment:

```

u2 = u * u
u3 = u * u2
f = 1
s = 0
i = 15
while (i >= 0) {
    t = p[i--]
    t = t + u * p[i--]
    t = t + u2 * p[i--]
    t = t + u3 * p[i--]
    s = s + t * f
    f = f * v
}
return s

```

This code is pretty straightforward to implement using SIMD instructions, thereby calculating four surface evaluations in parallel. Moreover, we can easily take advantage of the combined multiply-add instructions provided by some CPU architectures (e.g. PowerPC). Furthermore, by performing loop-unrolling, additional unnecessary operations can be eliminated (for example the last two statements in the loop for the first iteration).

| Scene      | # Patches | # Trims | #B-Boxes | Memory consumption | Preprocessing time (min) | Average Framerate |
|------------|-----------|---------|----------|--------------------|--------------------------|-------------------|
| Teapot     | 32        | -       | 2736     | 150 kB             | 0:01                     | 6.4 fps           |
| Cessna     | 1555      | -       | 53216    | 3.2 MB             | 0:03                     | 4.2 fps           |
| Chessboard | 16182     | -       | 227794   | 15.6 MB            | 0:13                     | 6.7 fps           |
| VW Polo    | 11576     | 38556   | 448500   | 28.3 MB            | 5:26                     | 5.1 / 3.4 fps*    |

**Table 1. Statistics for our test scenes (\*=trimming disabled/enabled)**

The computation of both partial derivatives can be executed even faster. For example, the derivative in  $u$  direction is

$$S_u(u, v) = [3u^2 \ 2u \ 1 \ 0][P][V]^T.$$

If we apply the same optimizations as presented above, the calculation of both partial derivatives simplifies to 33 multiplications and 33 additions each.

#### 4.3.3 Trimming

For surface areas that have been classified to need trimming during the preprocessing step, an additional 2D point-in-curve test is performed after an intersection has been found. Currently, this is done sequentially for the (up to) four intersection points, as we do not employ any SIMD instructions for trimming yet.

In addition, we currently perform this point-in-curve test for all trimming curves associated to the original Bézier surface and not only for those parts that are relevant for the examined region (i.e. the subpatch containing the intersection point).

In our implementation, we use the point classification approach presented by [Nis90a]. However, instead of using Bézier Clipping to subdivide the trimming curves into three segments if they cannot be clearly classified, we again take a brute force approach by splitting the curves at  $t = 0.5$  and recursively test both segments.

#### 4.3.4 Shading

Finally, all valid intersection points are shaded for display. At present, we support the simple Phong shading model [Bui75a]. This is also done using SIMD instructions, calculating the color of up to four intersection points in parallel. Here, additional rays for calculating shadows, reflections or refraction may be generated.

## 5. RESULTS

In this section we present some timings of our ray tracing system for a couple of test scenes of varying complexity (see Table 1 for the numbers and Figure 4 for renderings), measured on a dual processor

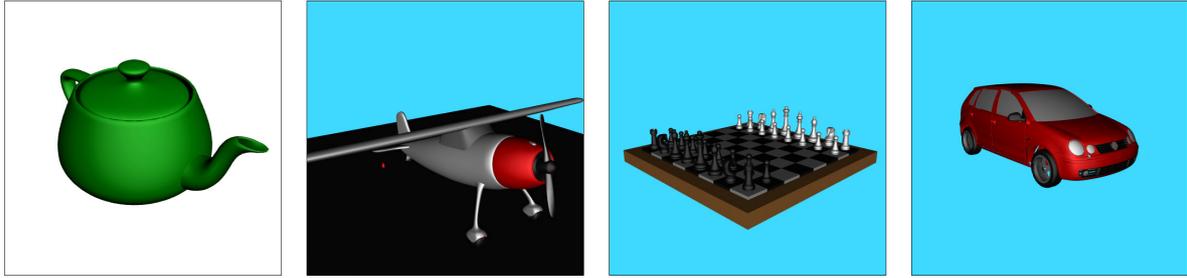
PowerMac G5 running at 2 GHz using only a single rendering thread. All images are generated at a fixed screen resolution of 512x512 pixels using simple Phong shading. Note that all scenes are lit by a single point light source, except for the Chessboard which is lit by three point lights.

Our ray tracing system based on the algorithms presented above is able to render the Utah Teapot consisting of 32 Bézier patches at an average frame rate of 6.4 fps, whereas the more complex Cessna and Chessboard scenes can be rendered at 4.2 frames/s and 6.7 frames/s respectively.

To compare our results to the approach presented by [Ben04a], we additionally rendered the Teapot model at a screen resolution of 640x480. Here, the frame rate drops to 5.5 fps due to the larger number of pixels. This frame rate is comparable to their result for a subdivision depth of 0 (i.e. directly rendering the control mesh), already taking into account that we use a different test platform that is approx. 35 percent faster. Nevertheless, as the number of refinement steps usually has to be higher in order to obtain a good rendering quality, our method easily outperforms the subdivision approach for this model. For example, when using only four refinement steps, our implementation is already twice as fast.

In contrast to the other test scenes, the VW Polo is the only model that contains trimming curves. Here, the large number of trims comes from the conversion of the original B-splines into piecewise cubic Bézier curves. It should also be noted that most of the preprocessing time is spent for reading the IGES file (~15%) and for the classification of the generated subpatches (~78%). However, the preprocessing code is fairly unoptimized at the moment.

As can be seen from Table 2, the classification of the patches works quite well for the Polo model. Trimming calculations need to be performed only for a small fraction of the generated patches during rendering, as most subpatches can be categorized in advance.



**Figure 4. Renderings of our test scenes: Utah Teapot, Cessna, Chessboard, and VW Polo (with trimming).**

With trimming disabled, the Polo can be displayed at an average frame rate of 5.1 frames per second. After trimming is enabled, the average frame rate drops to 3.4 fps. On the one hand, this can be explained by the fact that trimming is currently performed sequentially for the intersections found. On the other hand, we still perform a lot of unnecessary trimming calculations, as the point-in-curve test is executed for all trimming curves associated to the original Bézier patch and not only for those parts that are relevant to the examined subpatch.

| # Subpatches (total) | 496827 | 100.0% |
|----------------------|--------|--------|
| Without trims        | 70523  | 14.2%  |
| With trims           | 426304 | 85.8%  |
| Totally in           | 241480 | 48.6%  |
| Need trimming        | 33987  | 6.8%   |
| Discarded            | 150837 | 30.4%  |

**Table 2. Number of subpatches and their classification for the VW Polo model.**

For comparison, we also rendered a tessellated model of the VW Polo consisting of 326159 triangles using a modified version of our ray tracing system, achieving an average frame rate of 4.8 fps. Although this is still faster than ray tracing the Bézier-based model, one should keep in mind that comparable models used for design reviews in the industry today usually consist of several million triangles that easily occupy gigabytes of memory.

As memory access has been shown to be a limiting factor for ray tracing [Wal01a], direct rendering of bicubic Bézier surfaces can already be a valuable alternative to triangle-based approaches, especially for large models.

## 6. FUTURE WORK

Since our interactive ray tracing system for trimmed bicubic Bézier surfaces is still at an early stage of development, there is much room for improvements left, both in terms of rendering performance and image quality.

Currently, we are working on a distributed version of our rendering system, running on a cluster of

heterogeneous PCs. First results indicate that the performance scales almost linearly with the number of processors, as could be expected.

In addition, the trimming code can be improved in different ways. As already stated before, the trimming curves can be split up in such a way that for each subpatch only the relevant parts have to be tested. Secondly, it would be interesting to examine if the usage of SIMD calculations can also speed up the trimming calculations. And finally, efficient handling of the special case of line segments may further improve performance.

Moreover, the creation of our bounding volume hierarchy could be improved by using a more sophisticated heuristic to guide the subdivision of the Bézier patches (e.g. taking the curvature of the surface into account) instead of using the simple flatness criterion.

As already stated in the introduction, our system is currently restricted to bicubic Bézier surfaces. Extending the presented approach to Bézier patches of arbitrary degree is relatively easy to implement. Thinking even further, it would be interesting to investigate more complex surface descriptions such as B-splines or NURBS.

Finally, the next step in terms of image quality will be to add support for the typical ray tracing effects like shadows, reflections, and refractions.

## 7. CONCLUSION

Recently, it has been shown that it is possible to ray trace complex scenes at interactive frame rates even on a single commodity PC. Currently, however, almost all of these implementations use triangles as the only geometric primitive.

In this paper, we have presented the details of our ray tracing system that is capable of directly rendering trimmed bicubic Bézier surfaces. We have shown that by carefully optimizing the implementation of a well-known intersection technique using SIMD instructions provided by many of today's CPUs, it is feasible to render this kind of free-form surfaces at interactive frame rates as well.

Our results indicate that direct ray tracing of Bézier surfaces is already a valuable alternative to triangle-based approaches, especially for complex models. Moreover, we also suggested a number of possible improvements to further increase the achievable frame rate, which will make this method even more competitive.

## 8. ACKNOWLEDGEMENTS

We would like to thank all the people that have contributed to this paper, in particular Matthias Biedermann and Thorsten Grosch for many helpful discussions and their comments on preliminary versions, as well as Arne Claus for modeling the Cessna and Chessboard scenes. In addition, we would like to thank the reviewers for their suggestions to improve this paper. Special thanks also go to the Volkswagen AG for providing the data of the Polo model and granting permission to use it in this publication.

## 9. REFERENCES

- [Ben04a] Benthin, C., Wald, I., and Slusallek, P. Interactive Ray Tracing of Free-Form Surfaces. ACM Afrigraph, pp.99-106, 2004.
- [Bui75a] Bui-Tuong, P. Illumination for Computer Generated Pictures. Com. of ACM 18, No.6, pp.311-317, 1975.
- [Gei03a] Geimer, M., and Müller, S. A Cross-Platform Framework for Interactive Ray Tracing, Proc. of GI Graphiktag, pp.25-34, 2003.
- [Gol87a] Goldsmith, J., and Salmon, J. Automatic Creation of Object Hierarchies for Ray Tracing. IEEE CG&A 7, No.5, pp.14-20, 1987.
- [Int04a] Intel Corp. IA-32 Architecture Software Developer's Manual. 2004.
- [Int04b] Intel Corp. Hyper-Threading Technology. <http://www.intel.com/technology/hyperthread/>
- [Lex01a] Lext, J., and Akenine-Möller, T. Towards Rapid Reconstruction for Animated Ray Tracing. EUROGRAPHICS Short Presentations, pp.311-318, 2001.
- [Mar00a] Martin, W., Cohen, E., Fish, R., and Shirley, P. Practical Ray Tracing of Trimmed NURBS Surfaces. JGT 5, No.1, pp.27-52, 2000.
- [Mot99a] Motorola, Inc. AltiVec Technology Programming Interface Manual. 1999
- [Muu95a] Muuss, M. J. Towards Real-Time Ray-Tracing of Combinatorial Solid Geometric Models. Proc. BRL-CAD Symposium, 1995.
- [Nis90a] Nishita, T., Sederberg, T.W., and Kakimoto, M. Ray Tracing Trimmed Rational Surface Patches. Computer Graphics 24, No.4, pp.337-345, 1990.
- [Par99a] Parker, S., Martin, W., Sloan, P.-P. J., Shirley, P., Smits, B., and Hansen, C. Interactive Ray Tracing. Sym. Interactive 3D Graphics, pp.119-126, 1999.
- [Roc89a] Rockwood, A., Heaton, K., and Davis, T. Real-Time Rendering of Trimmed Surfaces. Computer Graphics 23, No.3, pp.107-116, 1989.
- [Smi98a] Smits, B. Efficiency Issues for Ray Tracing. JGT 3, No. 2, pp.1-14, 1998.
- [Swe86a] Sweeney, M., and Bartels, R. Ray Tracing Free-Form B-Spline Surfaces. IEEE CG&A 6, No.3, pp.41-49, 1986.
- [Wal01a] Wald, I., Slusallek, P., Benthin, C., and Wagner, M. Interactive Rendering with Coherent Ray Tracing. Computer Graphics Forum 20, No. 3, pp.153-164, 2001.
- [Wal01b] Wald, I., Slusallek, P., and Benthin, C. Interactive Distributed Ray Tracing of Highly Complex Models. Rendering Techniques 2001, pp. 274-285, Springer, 2001.
- [Wal03a] Wald, I., Benthin, C., and Slusallek, P. Distributed Interactive Ray Tracing of Dynamic Scenes. IEEE Sym. on Parallel and Large-Data Visualization and Graphics, pp.77-86, 2003.
- [Wal04a] Wald, I. Realtime Ray Tracing and Interactive Global Illumination. PhD thesis, Saarland University, Saarbrücken, Germany, 2004.
- [Wan01a] Wang, S., Shih, Z, and Chang, R. An Efficient and Stable Ray Tracing Algorithm for Parametric Surfaces. Journal of Information Science and Engineering 18, pp.541-561, 2001.
- [Whi80a] Whitted, T. An Improved Illumination Model for Shaded Display. Com.of ACM 23, No.6, pp.343-349, 1980

# Realistic Solar Disc Rendering

Andrei Lințu

Jörg Haber

Marcus Magnor

MPI Informatik  
Stuhlsatzenhausweg 85  
66123 Saarbrücken, Germany

lintu@mpi-sb.mpg.de

haberj@mpi-sb.mpg.de

magnor@mpi-sb.mpg.de

## ABSTRACT

This paper concentrates on rendering the solar disc considering Rayleigh scattering, Mie scattering, absorption, and refraction. The atmosphere is modeled in layers, each layer having a set of individual optical properties. Based on different atmospheric temperature profiles and climates, the solar disc is rendered in realistic shape and color. In particular, we replicate optical phenomena such as the red and the green flash, limb darkening, and refractive distortions of the solar disc.

## Keywords

Photo-realistic rendering, scattering, solar disc, atmosphere, ray-tracing

## 1. INTRODUCTION

Computing a physically and visually correct reproduction of the colors of the sky dome around the observer is an essential task for outdoor scene renderings. Although some work has been done on the simulation of the sky colors during daytime [PSS99; NDKY96], nighttime [WJDS<sup>+</sup>01], and twilight periods [HMS], realistic rendering of the solar disc has received little attention in the literature so far. In order to achieve realistic renderings, the optical phenomena occurring in the atmosphere need to be considered. Based on the physical structure of the Earth's atmosphere, this paper reproduces solar disc appearances at sunrise/sunset in correct form and color. We take into account Rayleigh scattering due to air molecules as well as Mie scattering due to aerosols present in the atmosphere, thus obtaining a realistic color of the solar disc. To obtain the correct shape of



Figure 1: A sunset scenario in a polluted maritime climate, few minutes before the Sun is setting.

the solar disc, we also model refraction, allowing us to trace rays correctly through the atmosphere.

Based on the actual structure of a given input atmosphere model, i.e. height-dependent temperature profile and distribution of aerosols, we observe different colors and shapes of the solar disc. The color of the disc varies with the climate, its shape differs due to mirage phenomena. By rendering the Sun taking different wavelengths into account, chromatic aberration phenomena such as the green flash or the red flash can be reproduced.

In the following Section we give a short overview of previous work in the field of physically based rendering methods of the solar disc and the at-

Permission to make digital or hard copies of all or part of his 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 80-903100-7-9*

*WSCG'2005, January 31-February 4, 2005*

*Plzen, Czech Republic.*

Copyright UNION Agency - Science Press

mosphere. After an introductory review of the physics in Section 3, we give a system overview in Section 4. We describe our approach on solar disc rendering in Section 5. Results are presented in Section 6, before we conclude and point to future work in Section 7.

## 2. RELATED WORK

Realistic renderings of atmospheric phenomena has been a well researched topic in the computer graphics community. We review papers focusing on solar disc rendering, as well as work on rendering the sky.

Some publications focus on physically correct and realistic atmosphere simulations, for daytime [PSS99], [NDKY96] as well as for nighttime [WJDS<sup>+</sup>01]. Also, a system for rendering the atmosphere from a viewpoint situated in space is presented by Nishita *et al.* [NSTN93].

These approaches concentrate on fast rendering of the atmosphere, approximating the physics of atmospheric light transport. A work focusing on a physically correct simulation of the atmosphere during twilight phenomena is presented in [HMS].

A different method focusing on the rendering of the solar disc is presented by Bruton [Bru96]. In his thesis, ray-tracing through the atmosphere is performed using Lehn’s model [Leh85], and solar disc appearance is simulated from diverse input temperature profiles. However, this work considers only Rayleigh scattering due to air molecules. Thus, different types of sunsets depending on current aerosol distribution in the atmosphere cannot be simulated.

One possible approach is to simulate the green flash using an approach based on photon mapping [GSAM04; SGGC04], this work implements a “Curved Photon Mapping” algorithm. Although the obtained result is highly realistic, this approach is slow (due to the photon mapping algorithm), and it lacks any dependence on climate conditions.

In contrast, our approach concentrates on combining the simplified parabolic model for ray-tracing in the atmosphere presented by Lehn [Leh85] and the climate dependent stratified atmosphere model presented in [HMS] in order to create a ray-tracing system which realistically reproduces several possible sequences of the solar disc at sunset or sunrise.

## 3. SUNSET SCIENCE

Our approach for rendering the solar disc is able to faithfully reproduce a variety of optical phenom-

ena such as mirages and chromatic aberrations. In the following, we give physical descriptions of these phenomena.

### 3.1 Mirages

Mirages are caused by strong ray-bending due to steep temperature gradients in the atmosphere. According to the position of the mirrored images relative to the original object, mirages can be classified into two main categories:

- inferior mirages: mirrored image below object
- superior mirages: mirrored image above object

The *inferior mirage* occurs if a layer of hot air is close to the ground, bending the grazing rays upwards. It can be observed in deserts or above asphalt pavings on sunny days. An example of this mirage is presented in Figure 4, the so called *Omega sunset*.

For the *superior mirage* to take place, the observer has to be situated *inside* a layer of air with a thermal inversion, i.e. there is a sudden increase in temperature above the observer. The rays in this duct intersect after traversing kilometers through the atmosphere, creating the inverted image of a distant object. In the case of the superior mirage, the intersecting rays remain inside the inversion layer. Thus superior mirages do not occur of astronomical objects that are situated outside the Earth’s atmosphere.

If the observer is above a thermal inversion layer the just recently understood *mock mirage* can occur [You04]. In this case the intersecting rays can escape the Earth’s atmosphere, the intersection points being far away from the observer.

A large variety of mirages can occur. They are highly dependent on the altitude of the observer. In order to be able to see a solar disc mirage, at least one temperature inversion layer has to be present in the atmosphere at the time of observation. This inversion layer produces a sudden change in the refraction index of the air, thus creating a mirage.

### 3.2 The Green Flash

This peculiar optical phenomenon consists of a short green flash (lasting only a few seconds) that can appear on top of the solar disc when the Sun sets. It is due to the large variation of refraction and induced dispersion close to the horizon.

There are several types of green flashes, each one being associated with a mirage phenomenon. The green color is mainly due to *atmospheric dispersion* which makes the red component of the

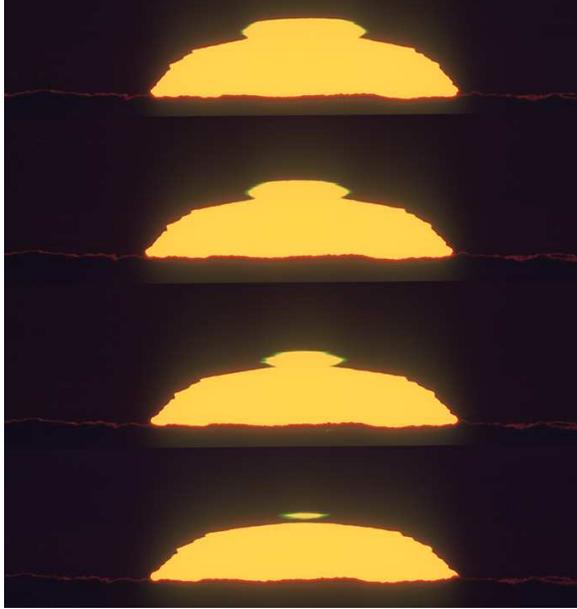


Figure 2: Photograph of a Green Flash Sequence  
Photograph by Mario Cogo, [www.intersoft.it/galaxlux](http://www.intersoft.it/galaxlux), used with permission

light spectra disappear first, followed by green, blue, and violet during sunset. Another effect contributing to the green flash is *atmospheric extinction*, which mainly consists of *atmospheric scattering* due to air molecules. In this case, the shortest wavelengths are almost completely removed. Therefore, it is sometimes possible to observe only the *green* component of the light spectra for a few seconds during sunset.

The green flash phenomenon depends also on the adaptation of the human visual system during observation. The high intensity of the solar disc bleaches the red-sensitive photo-pigments on the retina, thus allowing also yellow flashes to be perceived as green.

In Figure 2 a sequence of photographs of this phenomenon are presented. Figure 13 illustrates a simulation generated using our system. An exhaustive explanation and bibliography of this phenomenon can be found in [You04].

### 3.3 The Red Flash

Another type of mirage, which is harder to observe than the green flash, is the red flash. The red flash can occur due to a *mock mirage*, where it consists of a round red “droplet” below the “cropped” solar disc (see Figure 3), or as a consequence of an *inferior mirage*, where it is visible as a red middle region of the *Omega sunset* (see Figure 11).



Figure 3: Photograph of a Red Flash  
Photograph by Mario Cogo, [www.intersoft.it/galaxlux](http://www.intersoft.it/galaxlux), used with permission



Figure 4: Photograph of an Omega Sun  
Photograph by Mario Cogo, [www.intersoft.it/galaxlux](http://www.intersoft.it/galaxlux), used with permission

### 3.4 Limb Darkening

This phenomenon consists of the darkened outer rim of the solar disc. It is due to the fact that light from the center of the Sun traverses less gas of the Sun’s photosphere where it is partially absorbed.

Limb darkening can be phenomenologically described by

$$I'(\lambda) = I(\lambda) \cdot \left(1 - u \cdot \left(1 - \sqrt{1 - \frac{d^2}{r^2}}\right)\right),$$

where  $I$  is the solar irradiance,  $d$  is the distance from the center of the Sun,  $r$  is the radius of the solar disc, and  $u$  is the limb darkening coefficient for the Sun, which is approximately 0.6 for visible sunlight [Bru96].

We incorporate limb darkening in our system in order to obtain realistic renderings of the solar disc.

## 4. SYSTEM OVERVIEW

Our solar disc rendering system combines tracing of parabolic rays through the atmosphere [Leh85; Bru96] with an atmosphere model incorporating different climate-dependent characteristics [HMS] to reproduce both the correct *shape* of the solar

disc and its corresponding correct *color*. The used atmosphere model incorporates the scattering and absorption coefficients for Rayleigh scattering and Mie scattering. The effects of different climate types, air humidities, and atmosphere temperature profiles on the appearance of the solar disc at sunset or sunrise are simulated.

Ray-tracing through the atmosphere is computed using Lehn’s parabolic model for a light ray traveling through an atmosphere layer [Leh85]. This model reduces the amount of computation time otherwise required to solve the Eikonal PDE for light rays traveling through the atmosphere. To determine the distribution of the aerosols in the atmosphere we use the OPAC software package [HKS98; Hes98].

As input to our simulations, in addition to the required climate type and temperature profile, we also specify *observer height*.

We now give a short, step by step description of the used method:

- specify input: *temperature profile, climate, number of layers* and *observer height*;
- precompute the *radius of curvature* for the parabolic ray approximations;
- compute *solar disc shape* – ray-trace through the atmosphere model and compute the length the rays travel through each layer;
- compute *solar disc color* – multiply the initial intensity with the extinction factor;

## 5. SOLAR DISC RENDERING

In this section we describe the atmosphere model and the ray-tracing mechanism used in our simulations.

### 5.1 Extinction Coefficients

To determine the optical properties of the *aerosols* present in the atmosphere, we use the publicly available OPAC software package [HKS98; Hes98]. Using this package we compute wavelength-dependent aerosol absorption coefficients  $\sigma_a^{\text{aerosol}}(\lambda)$ , scattering coefficients  $\sigma_s^{\text{aerosol}}(\lambda)$ , and anisotropy factors  $g(\lambda)$  for the given input climate type of an arbitrary aerosol composition and humidity.

Values for the wavelength-dependent scattering coefficient  $\sigma_s^{\text{air}}(\lambda)$  of *air* molecules are taken from Nagel *et al.* [NQKW78]. Pure air does not significantly absorb visible light. Thus, the extinction coefficient of air  $\sigma_e^{\text{air}}$  is assumed to be equal to the scattering coefficient,  $\sigma_e^{\text{air}}(\lambda) = \sigma_s^{\text{air}}(\lambda)$ .

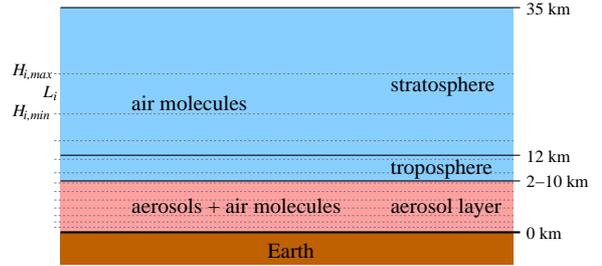


Figure 5: Cross section of the atmosphere model. It is composed of an aerosol-containing region, clear troposphere and stratosphere [HKS98]. For the aerosol region the height is climate dependent.

### 5.2 Atmosphere Model

The atmosphere model used in our system is stratified, consisting of atmosphere layers located geocentrically around the surface of the Earth.

The height of individual layers is chosen such that approximately the same amount of molecules is contained in each one of them. Our atmosphere model reaches up to a height of  $H_{\text{max}} = 35$  km. The number of molecules above this height can be considered negligible. A schematic description of the used model is depicted in Figure 5.

We discretize the atmosphere into a set of geocentric atmosphere layers  $L_i$ , ( $i = 1, \dots, N$ ). Each layer  $L_i$  has an individual upper and lower boundary at height  $H_{i,\text{max}}$  and  $H_{i,\text{min}}$ , respectively. To each layer  $L_i$  we assign the relative humidity  $w_i$  and the following optical parameters:

- aerosol scattering coefficient  $\sigma_{s,i}^{\text{aerosol}}(\lambda)$  and extinction coefficient  $\sigma_{e,i}^{\text{aerosol}}(\lambda) = \sigma_{s,i}^{\text{aerosol}}(\lambda) + \sigma_{a,i}^{\text{aerosol}}(\lambda)$ ;
- Henyey-Greenstein scattering anisotropy coefficient  $g_i(\lambda)$ ;
- isotropic scattering coefficient of air  $\sigma_{s,i}^{\text{air}}(\lambda)$ ;
- mean index of refraction  $\eta_i(\lambda)$  and indices of refraction  $\eta_{i,\text{min}}(\lambda)$  and  $\eta_{i,\text{max}}(\lambda)$  corresponding to  $H_{i,\text{min}}$  and  $H_{i,\text{max}}$ , respectively.

All these parameters are functions of the wavelength  $\lambda$  and are evaluated for a discrete number of wavelengths. A more detailed description of the used layered atmosphere model can be found in [HMS].

The simulations presented in this paper are typically computed using 300–30000 layers, depending on the input temperature profile. The number of input layers is dependent on the height of the inversion layers. For a thermal inversion layer close to the surface of the Earth, we need a fine sampling of our atmosphere model in order to be

able to correctly simulate the corresponding mirage, due to the exponential distribution of the atmosphere layers.

The solar irradiance  $I_0(\lambda)$  outside the atmosphere, i.e. before the sun light reaches the ozone layer, is computed from the solar spectrum data measured by Kurucz [KFBT84] for wavelengths from 200 nm to 1000 nm. We filter the solar irradiance using the approach presented in [HMS], accounting for wavelength-dependent absorption in the ozone layer.

### 5.3 Temperature Profiles

Our simulations are based on temperature profiles of the atmosphere, the starting point being the U.S. Standard Atmosphere [Bru96; You04]. In order to simulate different mirage phenomena, several input height-temperature profiles differing from the U.S. Standard Atmosphere are used. The profile of the atmosphere is specified as the temperature gradient at discretized heights.

### 5.4 Atmospheric Refraction

A model based on exact computation of refraction taking place in the atmosphere requires numerically solving a PDE and is computationally too expensive to be practically useful. The simplest solution is to assume all rays traveling linearly through each layer. We chose to implement a quadratic-error model developed by Lehn [Leh85]. This model assumes that the circular arcs representing the Earth's surface, the light rays, and the layer boundaries of the atmosphere model can be locally approximated by parabolas.

Taking into consideration the specified atmosphere temperature profile, climate and humidity for the current rendering, parameters regarding to the used model can be computed before the actual ray-tracing process. For each layer  $L_i$  we precompute the wavelength-dependent radius of curvature  $\kappa_i(\lambda)$  using

$$\kappa_i(\lambda) = -K \cdot \eta_i(\lambda) \cdot \frac{(H_{i,\max} - H_{i,\min})}{(\eta_{i,\max}(\lambda) - \eta_{i,\min}(\lambda))},$$

where  $\eta_i(\lambda)$  is the mean index of refraction of air in layer  $L_i$ , and  $\eta_{i,\max}(\lambda)$ ,  $\eta_{i,\min}(\lambda)$  are the refractive indices of air corresponding to the upper and lower heights  $H_{i,\max}$  and  $H_{i,\min}$ . The value for  $K$  depends on the initial parameters in the ray-tracing process and can be found in Bruton's thesis [Bru96]. The rays are traced through the atmosphere starting from the layer containing the observer.

In Figure 6, the two possible cases of the intersection between the currently traced ray and the

boundaries of an atmosphere layer are depicted. The expected trajectory of the ray is denoted as Path 1. However, if a thermal inversion is present, the ray can bend downwards following Path 2.

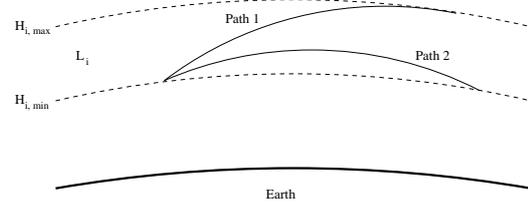


Figure 6: Two possible ray paths through the atmosphere. Path 1 depicts the normal trajectory of a ray traveling through layer  $L_i$ . For a thermal inversion present in  $L_i$ , the ray trajectory may follow the path indicated as Path 2.

The distance  $\Delta\gamma_i(\lambda)$  that a light ray of wavelength  $\lambda$  travels through layer  $L_i$  is numerically determined by

$$\Delta\gamma_i(\lambda) = 2 \cdot \kappa_i(\lambda) \cdot \arcsin\left(\frac{d}{2 \cdot \kappa_i(\lambda)}\right) \quad (1)$$

with

$$d = \sqrt{\Delta x^2 + \left(\Delta z - \frac{\Delta x^2}{2R}\right)^2}.$$

The values for  $\Delta x$  and  $\Delta z$  are computed during the ray-tracing process through the atmosphere layers as described by Lehn [Leh85] and Bruton [Bru96]. Based on the current optical parameters during ray-tracing, for each layer  $L_i$  a decision is taken [Leh85; Bru96] whether Path 1 or Path 2 is chosen for the current ray (see Figure 6).

In order to accurately compute the indices of refraction for each atmosphere layer based on the input temperature profiles, we employ the formulas proposed by Ciddor [Cid96]:

$$\eta_{h,w}(\lambda) = 1 + \frac{\rho_a}{\rho_d} \cdot (\tilde{\eta}_d - 1) + \frac{\rho_w}{\rho_w} \cdot (\tilde{\eta}_w - 1),$$

where  $\rho_d$  is the density of dry air at 15 °C and 101325 Pa,  $\rho_w$  is the density of pure water vapor at 20 °C and 1333 Pa, and  $\rho_a$  and  $\rho_v$  are the densities of the dry air component and water vapor component for the current conditions. The equations needed to calculate the air densities in the above formula are given by Ciddor [Cid96]. To compute the values for  $\tilde{\eta}_d$  and  $\tilde{\eta}_w$ , the following equations are used [Cid96]:

$$\tilde{\eta}_d(\lambda) = 1 + \left( \frac{5792105.0}{238.0185 - \lambda^{-2}} + \frac{167917.0}{57.362 - \lambda^{-2}} \right) \cdot 10^{-8}$$

$$\tilde{\eta}_w(\lambda) = 1 + (295.235 + 2.6422 \lambda^{-2} - 0.03238 \lambda^{-4} + 0.004028 \lambda^{-6}) \cdot 1.022 \cdot 10^{-8}.$$

## 5.5 Atmospheric Extinction

To accurately determine the color of each pixel in the rendered images, we first compute the intensity  $I(\lambda)$  of sun rays that reach the observer *after extinction* in the atmosphere according to:

$$I(\lambda) = I_0(\lambda) \cdot \xi_\gamma(\lambda), \quad (2)$$

where  $I_0(\lambda)$  is the solar irradiance filtered by absorption in the ozone layer and the extinction factor  $\xi_\gamma(\lambda)$  is given as:

$$\xi_\gamma(\lambda) = \exp\left(-\int_\gamma (\sigma_e^{\text{aerosol}}(\lambda) + \sigma_e^{\text{air}}(\lambda)) ds\right).$$

The value for the extinction factor  $\xi_\gamma(\lambda)$  is numerically determined by

$$\xi_\gamma(\lambda) = \exp\left(-\sum_{i=1}^N (\sigma_{e,i}^{\text{aerosol}}(\lambda) + \sigma_{e,i}^{\text{air}}(\lambda)) \cdot \Delta\gamma_i(\lambda)\right), \quad (3)$$

where  $\Delta\gamma_i(\lambda)$  denotes the path length through layer  $L_i$ , see Equation (1).

We take into account both multiple Rayleigh scattering (by air molecules) and Mie scattering (by aerosols). Mie scattering is modeled using the well-known Henyey-Greenstein approximation [HG41]. For rendering the solar disc, the phase angle (i.e. the angle between incident light and scattering direction) in this approximation can be considered equal to zero, since the diameter of the solar disc is merely  $0.5^\circ$ . As a consequence, the extinction factor  $\xi_\gamma(\lambda)$  from Equation (3) is modified as follows:

$$\xi'_\gamma(\lambda) = \exp\left(-\sum_{i=1}^N \sigma_{e,i}^{\text{mult}}(\lambda) \cdot \Delta\gamma_i(\lambda)\right), \quad (3')$$

$$\sigma_{e,i}^{\text{mult}}(\lambda) = (1 - g(\lambda)) \cdot \sigma_{e,i}^{\text{aerosol}}(\lambda) + \sigma_{e,i}^{\text{air}}(\lambda).$$

Substituting  $\xi'_\gamma(\lambda)$  for  $\xi_\gamma(\lambda)$  in Equation (2) yields the final formula to compute the intensity  $I(\lambda)$ .

## 5.6 Gamma Correction

For the final rendering, we convert the sampled spectral distribution into its corresponding color in XYZ color space by convolution with the CIE (1964)  $10^\circ$  color matching functions. Due to the high dynamic range of intensities, we have to apply gamma correction to faithfully display our results. We thus transform from XYZ color space to xyY color space and perform gamma-correction on the Y-value:

$$Y' = Y^{1/\bar{\gamma}},$$



Figure 7: Sunset in a polluted urban climate with 80% humidity.

where we use  $\bar{\gamma} = 2.5$  for the correction coefficient. Finally, we convert back to XYZ and from XYZ to RGB color space using the sRGB primaries from CIE Rec. 709 and a D<sub>65</sub> whitepoint. For details of these spectral conversions see the textbooks by Wyszecki *et al.* [WS82] or Hall [Hal89].

## 6. RESULTS

We have rendered a variety of sunset sequences for different meteorological conditions using the approach presented in this paper. To obtain a large diversity of solar disc renderings, various aerosol distributions, air humidity values, and temperature profiles of the atmosphere have been used.

Figure 1 shows a sunset scenario for a polluted maritime climate with 80% humidity. A polluted urban climate with 80% humidity has been used to simulate the sunset depicted in Figure 7. The sunset reproduced in Figure 8 has been simulated for a continental climate with 70% humidity. For all of the above mentioned images, the colors of the sky have been computed and rendered using the approach presented in [HMS].

All images presented in this paper have been generated using a discretization of  $N_\lambda = 8$  wavelengths in the range from 380 nm to 720 nm. Depending on the number  $N$  of atmosphere layers used for the simulations and the resolution of the generated images, computation times on a 2.4 GHz Pentium4 PC are in the range of a few minutes for our unoptimized implementation.

Figures 9 and 10 show the effect of varying humidity on the appearance of the solar disc at sunset or sunrise. With increasing air humidity the solar disc becomes noticeably darker.

The consequences of a temperature inversion layer close to the ground is depicted in Figure 11. This is the so called *Omega Sun*, the form of the solar disc being reminiscent of the Greek letter  $\Omega$ .



Figure 8: Sunset for a continental climate with 70% humidity.

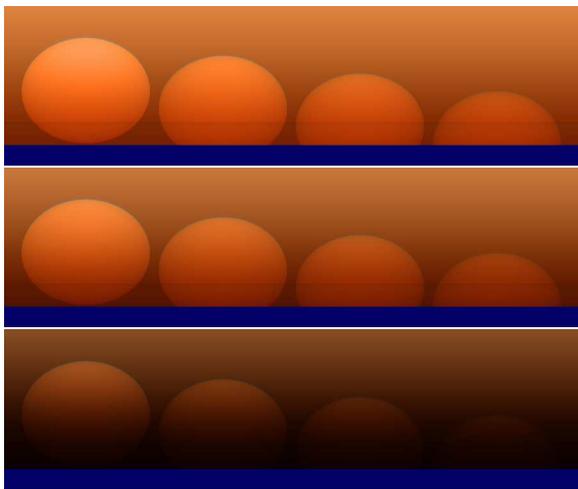


Figure 9: Sunset in continental climate for different humidities. Top to bottom: 50%, 80%, and 95% humidity.

The red flash is simulated in Figure 12. An atmosphere containing a weak inversion layer is used here to simulate a mock mirage for an observer situated at an altitude of 45 m above sea level.

In Figure 13, a green flash is replicated. The atmosphere is identical to the one used in Figure 12. Here, however, the Sun is at a lower altitude. The difference in altitude between successive renderings is below one arc-minute.

## 7. CONCLUSIONS

A system for realistic rendering of the solar disc has been presented. Atmospheric optical phenomena such as mirages, red and green flash, and limb darkening are simulated based on physical laws and meteorological conditions. In order to validate the obtained results, one future research di-

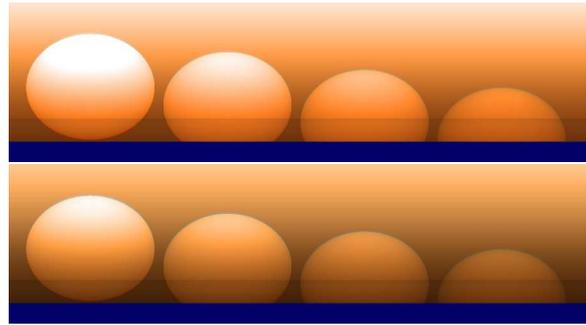


Figure 10: Sunset in tropical maritime climate for 80% (top) and 90% (bottom) humidity.

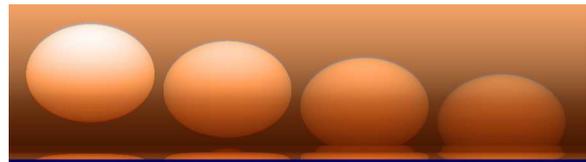


Figure 11: The omega sun frequently occurs in desert climates. A strong inversion layer right above the ground causes this inferior mirage.

rection is to calibrate the rendered images with real life photographs of sunsets. Furthermore, due to the large variation in extinction of various climates, which are reflected in the high dynamic range of our obtained results, an efficient tone mapping operator should be developed.

## Acknowledgements

The authors would like to thank Mario Cogo for his kind permission to use his photographs of the sun [Cog04]. The discussions had with Ivo Ihrke are also highly appreciated. Thanks to Andrew T. Young for sharing his comprehensive knowledge about the Green Flash on his website [You04].

## 8. REFERENCES

- [Bru96] Dan Bruton. *Optical Determination of Atmospheric Temperature Profiles*. PhD thesis, Texas A&M University, August 1996.
- [Cid96] Philip E. Ciddor. Refractive index of air: new equations for the visible and near infrared. *Applied Optics*, 35(9):1566–1573, March 1996.
- [Cog04] Mario Cogo. Astrophotography. available from <http://www.intersoft.it/galaxlux>, 1996 - 2004.
- [GSAM04] D. Gutierrez, F. Seron, O. Anson, and A. Munoz. Chasing the Green Flash: a Global Illumination Solution for Inhomogeneous Media. In

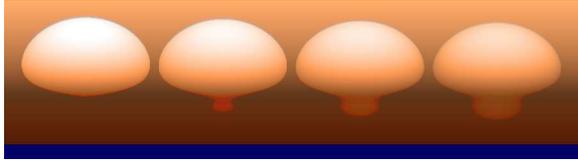


Figure 12: The red flash—a frequent mirage, hardly observed because of the small hue difference between the color of the solar disc and that of the flash.

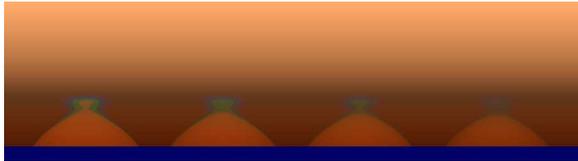


Figure 13: The green flash resulting from a mock mirage.

*Spring Conference On Computer Graphics 2004*, pages 95–103, 2004.

[Hal89] Roy Hall. *Illumination and Color in Computer Generated Imagery*. Springer, New York, 1989.

[Hes98] M. Hess. OPAC (Optical Properties of Aerosols and Clouds). available from <ftp://ftp.lrz-muenchen.de/pub/science/meteorology/aerosol/opac/>, 1998.

[HG41] Louis G. Henyey and Jesse L. Greenstein. Diffuse Radiation in the Galaxy. *Astrophysical Journal*, 93:70–83, 1941.

[HKS98] M. Hess, P. Koepke, and I. Schult. Optical Properties of Aerosols and Clouds: The Software Package OPAC. *Bulletin of the American Meteorological Society*, 79(5):831–844, May 1998.

[HMS] Jörg Haber, Marcus Magnor, and Hans-Peter Seidel. From Dust to Dawn — Physically based Simulation of Twilight Phenomena. *ACM Transactions on Graphics*. to appear.

[KFBT84] R. L. Kurucz, I. Furenlid, J. Brault, and L. Testerman. Solar Flux Atlas from 296 to 1300 nm. Technical report, NOAO, Sunspot, NM, 1984. available from <http://kurucz.harvard.edu/sun/fluxatlas/>.

[Leh85] W. H. Lehn. A simple parabolic model for optics of the atmospheric surface layer. *Applied Mathematical Modelling*, 9:447–453, December 1985.

[NDKY96] Tomoyuki Nishita, Yoshinori Dobashi, Kazufumi Kaneda, and Hideo Yamashita. Display Method of the Sky Color Taking into Account Multiple Scattering. In *Proc. Pacific Graphics '96*, pages 117–132. IEEE, 1996.

[NQKW78] M. R. Nagel, H. Quenzel, W. Kwet, and R. Wendling. *Daylight Illumination — Color-Contrast Tables for Full-Form Objects*. Academic Press, New York, 1978.

[NSTN93] Tomoyuki Nishita, Takao Sirai, Katsumi Tadamura, and Eihachiro Nakamae. Display of The Earth Taking into account Atmospheric Scattering. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Conf. Proc.)*, pages 175–182. ACM SIGGRAPH, August 1993.

[PSS99] Arcot J. Preetham, Peter Shirley, and Brian Smits. A Practical Analytic Model for Daylight. In *Computer Graphics (SIGGRAPH '99 Conf. Proc.)*, pages 91–100. ACM SIGGRAPH, August 1999.

[SGGC04] F. Seron, D. Gutierrez, G. Gutierrez, and E. Cerezo. Visualizing Sunsets through Inhomogeneous Atmospheres. In *Proceedings of Computer Graphics International 2004 (CGI 2004)*, pages 349–356. IEEE Computer Society Press, 2004.

[WJDS<sup>+</sup>01] Henrik Wann Jensen, Frédo Durand, Michael M. Stark, Simon Premoze, Julie Dorsey, and Peter Shirley. A Physically-Based Night Sky Model. In *Computer Graphics (SIGGRAPH 2001 Conf. Proc.)*, pages 399–408. ACM SIGGRAPH, August 2001.

[WS82] Günter. Wyszecki and Walter S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. John Wiley & Sons, New York, 2nd edition, 1982.

[You04] Andrew T. Young. A Green Flash Page. available from <http://mintaka.sdsu.edu/GF/index.html>, 1999 - 2004.

# Coherent and Exact Polygon-to-Polygon Visibility

F. Mora, L. Aveneau, M. Mériaux  
SIC, CNRS FRE 2731, SP2MI  
Bd Marie et Pierre Curie, BP 30179  
80962 Futuroscope Chasseneuil Cedex – France  
{mora,aveneau,meriaux}@sic.univ-poitiers.fr

## ABSTRACT

Visibility computation is a classical problem in computer graphics. A wide variety of algorithms provides solutions with a different accuracy. However, the four dimensional nature of the 3D visibility has prevented for a long time from leading to exact from-polygon visibility algorithms. Recently, the two first tractable solutions were presented by Nirenstein, then Bittner. Their works give the opportunity to design exact visibility tools for applications that require a high level of accuracy. This paper presents an approach that takes advantage of both Nirenstein and Bittner methods. On the one hand, it relies on an optimisation of Nirenstein's algorithm that increases the visibility information coherence and the computation robustness. On the other hand, it provides an exact visibility data structure as Bittner does, but also suited for non-oriented polygon-to-polygon visibility queries.

## Keywords

Exact visibility, CSG, Plücker space

## 1. INTRODUCTION

Visibility computation is a recurring problem in computer graphics applications. A wide variety of algorithms exists in the literature but they provide a different accuracy. This has led to a general algorithm classification:

- Aggressive : the visibility is underestimated.
- Conservative : the visibility is overestimated.
- Approximate : both aggressive and conservative.
- Exact : the visibility is exactly computed.

Solutions for these first three categories are usually fast. Most of them are designed in a context of visibility culling [Coh03a]. In contrast, exact algorithms require a significant computational effort, especially for from-polygon or from-region visibility. This problem has been considered for a long time as intractable due to the four dimensional nature of the visibility in 3D environment.

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

Conference proceedings ISBN 80-903100-7-9  
WSCG'2005, January 31-February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency - Science Press.

Previous works attempt to compute a global and exact visibility information, as Pellegrini [Pel93a] or Durand [Dur02a] with the 3D visibility complex. But these solutions are not practicable.

The first tractable algorithm was recently published by Nirenstein [Nir02a]. It allows an exact computation of the visibility from a polygon. At the same time, Bittner[Bit02c] proposed another solution.

The exact visibility is not necessary for most application. However, it has potential to improve high quality rendering of complex scenes or realistic lighting effects. The works of Nirenstein and Bittner give the opportunity to design efficient visibility tools encoding an exact information.

This paper presents an exact visibility algorithm that takes advantage of both Nirenstein and Bittner methods. It relies on Nirenstein algorithm but provides in output a structured visibility information as Bittner does. Moreover, it presents an optimisation of Nirenstein algorithm that improves the visibility information coherence. As a consequence, it gets a noticeable property : By reducing the visibility result complexity, the number of performed operations decreases, improving the robustness.

The second section explains the mathematical underlying and the general approach used by Nirenstein and Bittner for exact visibility computation. The third one gives an overview of the two existing algorithms and underlines their differences. From this short study, the section four presents our approach emphasising our

optimisation that provides a coherent visibility information and improves robustness. At last, results are given in the section five.

## 2. BACKGROUND

The two solutions proposed by Nirenstein and Bittner both rely on the same approach. They solve the visibility problem between polygons by performing CSG operations on polytopes (convex “volume”) in the Plücker space. This section begins with a presentation of the Plücker space where operates the solution. Next, it gives an overview of the approach allowing exact visibility computation from 3D polygons.

### Plücker Space

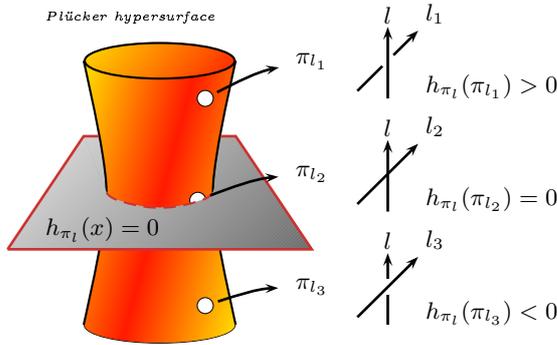
The Plücker space [Som59a] is a five dimensional projective space  $\mathbb{P}^5$ . It provides an elegant parametrisation for dealing with directed lines in  $\mathbb{R}^3$ . Each line  $l$  passing through the point  $(p_x, p_y, p_z)$  and next through  $(q_x, q_y, q_z)$  is defined in  $\mathbb{P}^5$  by  $\pi_i = (\pi_0, \pi_1, \pi_2, \pi_3, \pi_4, \pi_5)$ , with :

$$\begin{aligned}\pi_0 &= q_x - p_x & \pi_3 &= q_z p_y - q_y p_z \\ \pi_1 &= q_y - p_y & \pi_4 &= q_x p_z - q_z p_x \\ \pi_2 &= q_z - p_z & \pi_5 &= q_y p_x - q_x p_y\end{aligned}$$

Notice that  $(\pi_0, \pi_1, \pi_2)$  is the direction of  $l$  and  $(\pi_3, \pi_4, \pi_5)$  encodes its location.

Next, let us consider the dual mapping within  $\mathbb{P}^5$  : Each  $\pi \in \mathbb{P}^5$  can be associated with a dual hyperplane  $h_\pi$  defined by :

$$h_\pi = \{x \in \mathbb{P}^5 \mid \pi_3 x_0 + \pi_4 x_1 + \pi_5 x_2 + \pi_0 x_3 + \pi_1 x_4 + \pi_2 x_5 = 0\}$$



**Figure 1: Line orientation in the Plücker space :** There are three different cases for an oriented line to pass another :  $l_1$  passes on the left of  $l_0$ ,  $l_2$  is incident on  $l_0$  and  $l_3$  passes  $l_0$  on the right. The Plücker mapping of  $l_1$ ,  $l_2$ ,  $l_3$  will respectively lie above, on and below the dual hyperplane of  $l_0$ .

Given two lines  $l_1$  and  $l_2$  and their Plücker mapping  $\pi_{l_1}$  and  $\pi_{l_2}$ , a crucial property is :  $l_1$  and  $l_2$  are incident

if and only if  $\pi_{l_1}$  lies on the dual hyperplane of  $\pi_{l_2}$  (and vice versa). If  $h_{\pi_{l_1}}(\pi_{l_2}) \neq 0$ , the sign of  $h_{\pi_{l_1}}(\pi_{l_2})$  determines the relative orientation of  $l_1$  and  $l_2$  as illustrated on figure 1.

At last, each line in  $\mathbb{R}^3$  maps to a point in  $\mathbb{P}^5$  but each point in  $\mathbb{P}^5$  does not map to a line in  $\mathbb{R}^3$ . The mapping of all real lines in  $\mathbb{P}^5$  forms a four-dimensional quadric surface called the *Plücker hypersurface*.

### Exact From Polygon Visibility Principle

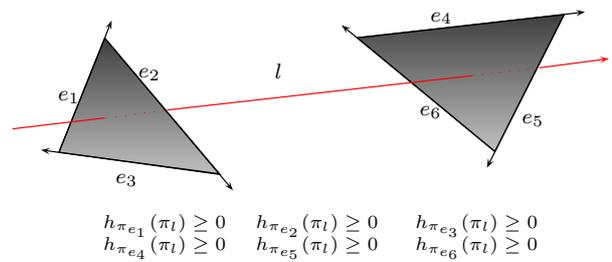
#### 2.2.1 Lines stabbing polygons

Previous definitions are useful to characterise the set of lines stabbing convex polygons. In the Plücker space, these lines are a connected subset of points on the hypersurface. For computational convenience, it is easier to deal with a polyhedral representation of this subset by using the dual hyperplane mapping of each polygon edges. The intersection of this polyhedral structure with the Plücker hypersurface gives exactly the set of lines stabbing each polygons. Such an approach was already used by Teller [Tel92a] for computing the anti-penumbra of an area light source through a sequence of polygons.

Figure 2 illustrates a two triangles case since we are in a context of polygon to polygon visibility. More generally, if  $A$  and  $B$  are two polygons with  $n$  and  $m$  edges  $e_1, \dots, e_{n+m}$  consistently oriented, all the lines  $l$  passing through  $A$  then  $B$  satisfy :

$$\forall i \in [1..n+m], h_{\pi_{e_i}}(\pi_l) \geq 0$$

This system of inequations is the hyperplane repre-

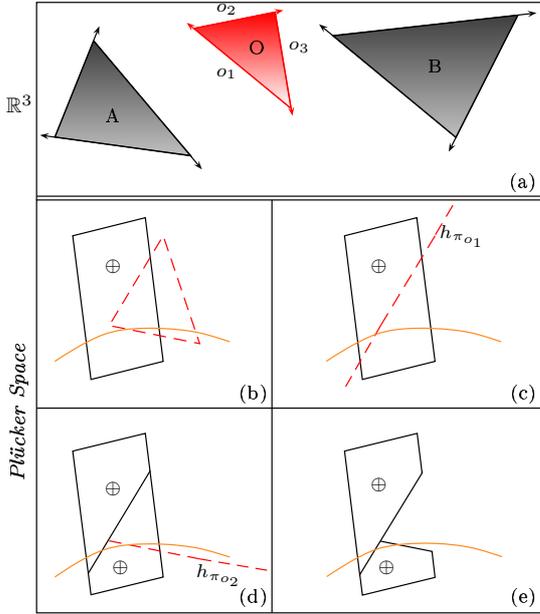


**Figure 2: Lines stabbing two polygons :** The Plücker mapping of the polygons edges induces the hyperplane representation of a polytope. Its intersection with the Plücker hypersurface is the set of all lines stabbing the two polygons.

sentation of an unbounded polyhedron in the Plücker Space. Both Nirenstein and Bittner add constraints to obtain a closed polyhedron: a polytope. Of course these additional constraints do not affect the intersection of the polyhedron with the Plücker hypersurface. The polytope representation allows to limit computations to the zone of the Plücker hypersurface.

### 2.2.2 Occluders removal

Let  $P_{AB}$  be the polytope that represents the set of lines stabbing  $A$  and  $B$ . Figure 3 gives a 2D illustration of the process that removes from  $P_{AB}$  the set of lines blocked by an occluder. This has to be applied to each occluder. The remaining parts of  $P_{AB}$  intersecting the hypersurface are exactly the set of lines that stabs  $A$  and  $B$  without stabbing any occluders. If no such a part remains,  $A$  and  $B$  are not visible.



**Figure 3:** (a) An occluder  $O$  blocks some visibilities between two polygons  $A$  and  $B$ . (b) The Plücker representation of lines stabbing  $A$  and  $B$  and lines stabbing  $O$ . (c) (d) : To remove the subset of blocked lines,  $P_{AB}$  is successively split in sub-polytopes using the hyperplanes associated to the occluder edges. (e) The sub-polytope corresponding to blocked lines is removed.

## 3. EXISTING ALGORITHMS

Nirenstein and Bittner methods both rely on the approach explained in the previous section. However these two algorithms were not designed for providing the same visibility information. Moreover, they have noticeable differences between their CSG operations on polytopes.

Exact visibility requires a consequent computational effort, using non trivial n-dimensional geometric algorithms. An effective implementation of the process has to be carefully considered.

In this section, a short overview of each algorithm is given, including some comparisons on their processes. Then, we justify our choices from this study.

## Algorithms overview

### 3.1.1 Nirenstein's algorithm

Nirenstein designed his algorithm to query if two polygons were visible or not. First, an initial polytope representing their stabbing lines is built. Next CSG operations are computed in the Plücker space to remove the lines blocked by each occluder. This is the same process as depicted by figure 3. Only the visible parts of the initial polytope are preserved during the process. However, this information is not organised and is only maintained as a set of sub-polytopes. As soon as the visibility or the invisibility is established, they are all dropped.

Exact visibility computation is sensitive to the number of occluders that have to be removed. Nirenstein makes a selection of the most effective occluders to be removed first. The occluded lines set can be removed using less occluders. This method minimises the number of intersection computation to perform. As a consequence, the algorithm termination is accelerated.

Nirenstein uses his algorithm to compute Potentially Visible Sets (PVS) for viewcells in 3D environment. In this context, he develops a framework including several optimisations that aim either to quickly find simple visibilities/invisibilities, or to choose an effective order for removing occluders. On the one hand, ray sampling handles trivial visibilities and finds effective occluders. On the other hand, a hierarchical subdivision of the scenes is used. Visibility queries are first applied to the cells of this hierarchy. Only visibility with polygons inside visible cells have to be computed, whereas invisible cells can be used as virtual occluders.

### 3.1.2 Bittner's algorithm

The purpose of Bittner's algorithm is different from Nirenstein's one. It was first developed in 2D [Bit01b] and then extended to three dimensional environments. It aims to encode all the visibilities with a scene from a source polygon. This information is encoded and structured by an occlusion tree [Bit98a]. Each leaf represents either a visibility or an invisibility set (when nothing can be seen). In particular, each in-leaf represents a set of lines that first stabs the same visible polygon.

The occlusion tree construction implies to treat occluders in a front to back order. This assumes the scene pre-processing to avoid overlapping occluders. For each of them, the associated polytope is inserted into the occlusion tree, from the root to the leaves, and is tested against each node met. If the hyperplane stored in a node splits the polytope, the algorithm continues in both subtree with the two relevant fragments. If an out-leaf is reached, the occluder is visible and the out-leaf is replaced by its fragment elementary occlusion

tree. If an in-leaves is reached, the fragment elementary occlusion tree is merged to update the visibility information.

Bittner also uses a hierarchical subdivision of the scene to enhance the occlusion tree construction. At the end of the process, the occlusion tree provides each part of the geometry that can be seen from the source polygon. Like Nirenstein, Bittner uses this information to compute PVS for viewcells. He also gives an example of virtual occluders extraction, valid from any viewpoint on the source polygon.

### Algorithms Implementation

As explained, computing exact visibility implies an important computational effort. This can not be trivially implemented. Some computational differences can be noticed between Nirenstein and Bittner implementations :

#### *Polytope construction*

The hyperplane representation of a polytope can be easily obtained from the Plücker mapping of polygons edges. However the intersection tests require the vertex representation. In any case, this can be achieved using an enumeration algorithm as in [Avi96a]. Such an algorithm is used by Bittner. For two given polygons, Nirenstein proposes in his thesis [Nir03a] a more efficient solution, including explicitly the additional constraints to cap the unbounded polyhedron. In contrast, the capping of the polyhedron in Bittner's algorithm requires more computation tests.

#### *Intersection tests*

Before splitting a polytope, intersection tests are made with hyperplanes. A common test to both methods is to compute whether polytope vertices fall in both half spaces induced by a hyperplane. In addition, Nirenstein implementation first makes a conservative test using the bounding sphere of a polytope. Then, a rejection test is applied using the other hyperplanes of the same occluder, as detailed in the next section. Contrary to Bittner's algorithm, this allows to limit the intersection computation to the zone of occluded lines.

#### *Polytope splitting*

The key for occluders removal is to compute the intersection of a polytope with a hyperplane. The implementation of Bittner computes implicit intersections. This means that a splitting hyperplane is added to the hyperplanes representation of a polytope, and all the vertices are enumerated again.

Nirenstein computes explicitly intersections using an algorithm similar to Bajaj and Pascucci's one [Baj96a]. As a requirement to this algorithm, the full face lattice of the polytope has to be computed. Nirenstein uses a combinatorial face enumeration as in [Fuk94a]. This is potentially more efficient since only the new ver-

tices are computed, whereas Bittner enumerates all the vertices again.

### Algorithms discussion

Our purpose is to compute a coherent and exact visibility information between two polygons. This information has to be structured to be available from the two queried polygons. The more coherent the visibility information will be, the more efficient its use will be.

Bittner's algorithm is interesting because it provides a structured visibility information. However this information is oriented since it is significant from the source polygon. As a consequence, it is not suited for non-oriented polygon-to-polygon visibility queries. We can notice that the occlusion tree construction can be restricted between two polygons. But it would still encodes the occluder fragments only visible from one polygon.

The computational part of the Nirenstein algorithm seems to be more efficient. In particular, the different tests made to limit the computation to the zone of an occluded lines set are interesting for our purpose. This should improve the coherence of the visibility information computed between two polygons. Besides, his algorithm is more flexible than Bittner algorithm with its fixed "front to back" subtraction order. As an example, the optimisation presented in this paper could not be applied to his algorithm without corrupting the output. This will be explained in the next section. Nirenstein algorithm can provide a set of polytopes representing all the visibility between two polygons. This information is non-oriented. However it is not structured like Bittner.

From this study, we choose to take advantage of the Nirenstein algorithm for CSG computation on polytopes. But the algorithm output is modified to organise the visibility information, like Bittner does. Our structure is suited for non-oriented polygon-to-polygon visibility. The next section presents our approach and an optimisation of the Nirenstein algorithm. In particular, this optimisation minimises the fragmentation of the visibility information, and improves robustness.

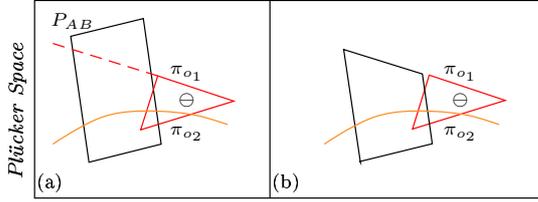
## 4. PROPOSED APPROACH

Firstly, this section explains how useless polytope fragmentation can occur. Next an overview of our approach is given and illustrates how the visibility information is encoded. At last, we presents the "back splitting" optimisation that allows to minimise the polytope fragmentation and to improve the robustness.

### Unnecessary Splitting

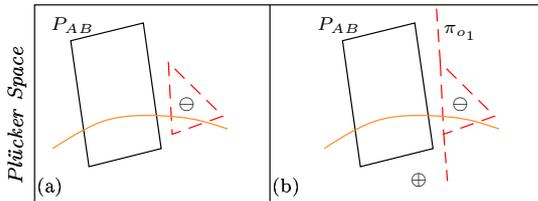
Two configurations exist where unnecessary splitting are computed. The first one appears when the splitting of a polytope  $P$  results in a first polytope having the

same intersection as  $P$  with the Plücker hypersurface, and a second one having no intersection with the hypersurface. Since only the Plücker surface intersection is of interest, it becomes clear that such a splitting is useless. However, we will see how to take advantage of this problem.



**Figure 4:** (a) A polytope  $P_{AB}$  that does not need to be intersected by each hyperplanes of an occluder. (b) As an example, the intersection with  $\pi_{o1}$  does not modify the intersection of  $P_{AB}$  with the hypersurface. This would be the same with  $\pi_{o2}$ .

The second configuration is within the rejection test of Nirenstein. It checks if a polytope is rejected by at least one hyperplane from a given occluder. This test, depicted in figure 5, can not always prevent unnecessary splitting operations. Figure 6 illustrates such a case. This explains why useless polytopes fragmentation still happens.

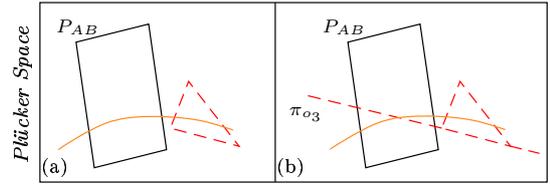


**Figure 5:** (a) A polytope  $P_{AB}$  that does not need to be intersected by any hyperplanes  $\pi_{o1}$ ,  $\pi_{o2}$ ,  $\pi_{o3}$  of an occluder. (b) Because all the vertices of  $P_{AB}$  lie in the positive half space of  $\pi_{o1}$ ,  $P_{AB}$  will be rejected and unnecessary splitting will be avoided.

Useless splitting generates two main problems. Firstly, it seems obvious that the probability to face numerical instability grows with the number of successive splitting operations performed. Next it leads to an unnecessary polytopes fragmentation, and so to a fragmentation of the visibility information. These two problems are obviously correlated. As a consequence, reducing the polytopes fragmentation must be a solution to both of them.

## Overview

Our approach uses a similar algorithm to Nirenstein's one. In this paper, notice we are not in a specific context. As a consequence, we do not use a framework as



**Figure 6:** (a) Another configuration where  $P_{AB}$  does not have to be split. (b) Because each hyperplane  $\pi_{o1}$ ,  $\pi_{o2}$ ,  $\pi_{o3}$  intersects  $P_{AB}$ , it will not be rejected. As a consequence, unnecessary splits will be performed. In particular the split by  $\pi_{o3}$  will generate useless fragmentation of the visibility information.

developed by Nirenstein for PVS computation. Moreover some parts of this framework are out of matter. For example, ray sampling can not be used since we are interested in the whole visibility information computation.

We consider two polygons and obtain from their edges the hyperplanes representation for the associated polytope in the Plücker space. Our implementation takes advantage of the Nirenstein solution [Nir03a] for computing its vertices representation. Its full face lattice is obtained with [Kai02a] that provides a better complexity than [Fuk94a]. At last, explicit splitting operations are achieved using the approach of [Baj96a].

To store the visibility information, we build a “history tree”. It has some similarities with an occlusion tree. It is a binary tree whose inner nodes are associated with splitting hyperplanes. A leaf represents either a set of blocked lines, or a set of visibilities between two polygons. In the later case, the polytope for this set is associated to the leaf.

But the history tree construction is different. It does not require a traversal from the root to the leaves. Intersection tests are made on visible leaves. When a leaf is split, it becomes an inner node associated with the splitting hyperplane. Its children represent each part of the intersected polytope. Initially, the root node is set with the polytope associated to the two queried polygons. A history tree can be understood as the history of the successive splitting operations.

At the end of the process, it encodes and represents all the visibility information between two polygons. The history tree is easy to build and does not require excessive computation time. This structure is suited to be used as a visibility tool.

Moreover, it gives the opportunity to minimise unnecessary splits. Since this induces an useless polygon fragmentation, we propose to detect and to cancel them using the history tree. This is the purpose of the “back splitting” algorithm. It also improves the visibility information coherence.

## Back Splitting Algorithm

The back splitting optimisation takes advantage of the history tree to cancel useless operations. This implies the following modifications: Before splitting a polytope, its copy is left in its associated node of the history tree. An inner node is then associated with a polytope and a splitting hyperplane. Back splitting affects the construction of the history tree, with the combination of two rules.

The first one aims to reduce the number of splits to improve the robustness. It is applied during an occluder removal, after a splitting operation. If the intersection with the hypersurface has not been modified, this means the splitting was useless. However, to keep the operation benefit, the smaller polytope representing the same set of lines replaces the copy of the initial polytope. This may seem a contradiction to the robustness improvement. Our motivation is to work with polytopes as close as possible to the hypersurface, to improve further rejection tests, and so to decrease the number of splitting operations. The robustness is related to this number. Our tests have shown that keeping the smaller polytope gives finally a smaller splitting number than keeping the initial polytope.

The second rule tries to minimise the polytopes fragmentation. It has to be applied after each occluder removal. It relies on a quick analysis of each pair of leaves sharing the same father. This rule is applied each time one of the two following configuration occurs:

- If both leaves are visible, this means that the polytope set in the father node was unnecessary split, as shown in Figure 6(b). In this case, this operation is cancelled. Both children are removed and the father node restored as a visible leaf.
- If both leaves are invisible, children are removed and the father node is replaced by a leaf marked invisible. A similar implication was proposed by Bittner for its occlusion tree. However, due to the back to front order constraint, he could not apply the previous configuration.

This optimisation helps to minimise the polytopes fragmentation and the number of splitting operations. This may seem a contradiction since less fragmentation implies bigger polytopes and bigger polytopes is a contradiction to the first rule. However the justification is that a polytope can be “big” as long as it remains close to the Plücker hypersurface.

Moreover, this allows a smaller history tree that describes the same visibility information. As a consequence, we can expect a more efficient use of this information and to spare memory. Notice that all the polytopes associated with nodes can be removed after the construction of the history tree. The tree with the splitting hyperplanes in inner nodes is then sufficient.

In the next section, we present some experimental results emphasising the back splitting improvement. We test different configurations depending on the visual complexity.

## 5. RESULTS

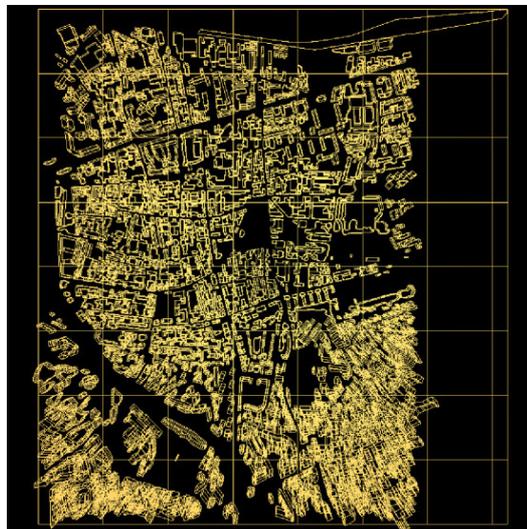


Figure 7: Test scene

To test the back splitting optimisation we use an urban environment composed by 38834 polygons as depicted in figure 7. This scene was chosen for the various configurations proposed in terms of occlusion and visual complexity. The hardware used is an Athlon XP1800+ (1.5 GHz) with 512Mo RAM.

From the test scene, we choose three sets of buildings, each one representing a different configuration for occlusion and visual complexity.

- Set 1 The first set is composed of the ten smallest buildings in the scene. Here, the occlusion is strong, and the visual complexity is low.
- Set 2 In opposition to the first set, the second set is composed of the ten highest buildings in the scene. This implies many visibilities and few occlusion.
- Set 3 The third set contains ten buildings with an average height. It combines both visual complexity and depth visibility. This means that occlusion can not be defined by a small subset of occluders.

From each set, the exact visibility between each building wall and the other scene polygons are computed. Table 1 shows results for the three sets.

For the first set, the back splitting has no contribution. We can notice a small time over cost for a query using back splitting. The explanation lies in the fact that the

| Set 1      | AVG Occluders | AVG Polytopes | AVG Splitting | Time/query (ms) |
|------------|---------------|---------------|---------------|-----------------|
| Without BS | 66.36         | 1.67          | 1.02          | 48.63           |
| With BS    | 66.36         | 1.67          | 1.02          | 49.74           |
| Set 2      | AVG Occluders | AVG Polytopes | AVG Splitting | Time/query (ms) |
| Without BS | 136.7         | 40.5          | 62.26         | 203.11          |
| With BS    | 136.7         | 20.44         | 37.25         | 132.09          |
| Set 3      | AVG Occluders | AVG Polytopes | AVG Splitting | Time/query (ms) |
| Without BS | 157.3         | 17.39         | 39.87         | 357.55          |
| With BS    | 157.3         | 7.55          | 27.36         | 244.37          |

**Table 1: Results without and with back splitting (BS). AVG Occluders is the average number of occluders per query. AVG Polytopes is the average number of polytopes representing the visibility information between two visible polygons. AVG Splitting is on average the number of splitting operation for each query.**

(in)visibilities are quickly determined, before the two rules could have an impact on the computation.

For the second set, a significant contribution appears : 40% of the splitting operations are avoided. This leads to an improvement (35%) of the time computation per query. However, the most interesting result is the number of polytopes reduced from 40.5 to 20.44. As the size of the history tree is connected to the fragmentation of the polytopes, this implies a better coherence of the information and a reduction of the memory requirement. In spite of an important number of occluders and a significant visual complexity, back splitting remains efficient on the third set, where 31% of the splitting operation are avoided. We note the same reduction for the computation time per query. Once again, the main result is the minimisation of the fragmentation of the polytopes. Back splitting reduces fragmentation from more than 56%.

This result illustrates the back splitting efficiency for reducing the fragmentation of the polytopes. Moreover, since less splitting operations are performed, this increases the robustness of the visibility computation. As a secondary result, the time per query is improved.

## 6. CONCLUSION

This paper has presented a solution to compute a coherent and exact visibility information between two polygons. The fundamental principles to achieve exact visibility computation has been recalled. From the first two tractable solutions study, we have proposed an unified approach taking advantage of both of them. It modifies the Nirenstein algorithm to provide a history tree. This allows to organise the visibility information similarly to Bittner, but suited for non-oriented polygon-to-polygon queries. Moreover, we have presented the back splitting optimisation that improves the visibility coherence and the robustness. As the information is described using a smaller set of polytopes, memory can also be spared.

Results show that our approach is mainly efficient with

a consequent visibility complexity, which is the most challenging configuration in graphic applications such as realistic image synthesis.

As a future work, we plane to enhance such applications by taking advantage of the history tree as a visibility tool. Moreover, a collaboration with a telecommunication department is already in progress for an accurate and fast visualisation of electromagnetic waves.

## 7. REFERENCES

- [Avis96a] David Avis and Komei Fukuda. *Reverse search for enumeration*. Discrete Appl. Math., vol 65, 21-46, 0166-218X, Elsevier Science Publishers B. V.
- [Baj96a] C. L. Bajaj and V. Pascucci. *Splitting a Complex of Convex Polytopes in any Dimension*. In Proceedings of the 12th Annual Symposium on Computational Geometry, ACM, 88-97, May 1996
- [Bit98a] J. Bittner and V. Havran and P. Slavik. *Hierarchical Visibility Culling with Occlusion Trees*. Proceedings of Computer Graphics International '98 (CGI'98), 207-219, 1998.
- [Bit01b] Bittner and Jan Prikryl. *Exact Regional Visibility using Line Space Partitioning*. TR-186-2-01-06, 1-13, 2001
- [Bit02c] J. Bittner. *Phd dissertation : Hierarchical Techniques for Visibility Computations*. Czech Technical University in Prague, October 2002.
- [Coh03a] Cohen-Or, Chrysanthou, Silva, Durand. *A survey of visibility for walkthrough applications*. IEEE TVCG, pages 412-431, september 2003
- [Dur02a] F. Durand, G. Drettakis, C. Puech. *The 3D visibility complex*. ACM Trans. Graph., vol. 21, 2, pages 176-206, ACM Press.
- [Fuk94a] K. Fukuda and V. Rosta. *Combinatorial face enumeration in convex polytopes*. Computational Geometry: Theory and Application, vol. 4, 4, pages 191-198, 1994
- [Kai02a] V. Kaibel and M. E. Pfetsch. *Computing*

*the face lattice of a polytope from its vertex-facet incidences.* Computational Geometry: Theory and Applications, vol. 23, 3, pages 281-290, November 2002

[Nir02a] S. Nirenstein and E. Blake and J. Gain. *Exact from-region visibility culling.* Proceedings of the 13th Eurographics workshop on Rendering, pages 192-202, 2002.

[Nir03a] S. Nirenstein. *Fast and accurate visibility preprocessing.* University of Cape Town, South

Africa, October 2003.

[Pel93a] M. Pellegrini. *Ray Shooting on Triangles in 3-Space.* Algorithmica, vol. 9, 5, pages 471-494, 1993

[Tel92a] Seth J. Teller. *Computing the antipenumbra of an area light source.* Computer Graphics, (Proc. Siggraph '92), 26:139-148, July 1992.

[Som59a] Sommerville. *Analytical Geometry in Three Dimension.* Cambridge University Press, 1959.

# New spectral decomposition for 3D polygonal meshes and its application to watermarking

Kohei MUROTANI and Kokichi SUGIHARA

Department of Mathematical Informatics Graduate School of Information Science and Technology University of Tokyo  
7-3-1, Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan  
muro@simplex.t.u-tokyo.ac.jp and sugihara@mist.i.u-tokyo.ac.jp

## Abstract

This paper presents a generalization of a data analysis technique called a singular spectrum analysis (SSA). The original SSA is a tool for analyzing one-dimensional data such as time series, whereas our generalization is suitable for multi-dimensional data such as 3D polygonal meshes. One of applications of the proposed generalization are also shown. The application of the generalized SSA is a new robust watermarking method that adds a watermark to a 3D polygonal mesh. Watermarks embedded by our method are resistant to similarity transformations and random noises. Our method has the advantage in that it requires smaller calculation cost than other methods with nearly equal performance.

**Keywords:** singular spectrum analysis (SSA), watermarking, 3D polygonal meshes, spectral decomposition

## 1 Introduction

Techniques of the spectrum decomposition have been developed in various fields, such as signal processing and financial data analysis. The Fourier analysis and the wavelet analysis are the most frequently used techniques of the spectrum decomposition. However, since they are the decompositions by a certain basis functions, the data should be represented in a parameterized form. They are applied to one-dimensional series or the tensor product of one-dimensional series naturally, while they cannot be applied to non-parameterized data. For example, we cannot represent an unbounded two-manifold having the same topology as a sphere by two parameters. In this case, for example, we can perform the spectrum decomposition using the spherical harmonics. But, if this spectrum decomposition is performed for a piecewise linear function, such as a 3D polygonal mesh, the numerous terms are required. Since spectrum decompositions by hitherto known functions have limitations

like this, there is a real need for new spectrum decomposition methods.

We generalize the singular spectrum analysis (SSA) in such a way that it is applicable to the 3D polygonal mesh, and apply it to engineering problems [6, 8, 9, 10]. In this paper, the generalized SSA is applied to a robust watermarking method that adds a watermark to a mesh.

## 2 Generalization of singular spectrum analysis

### 2.1 From basic SSA to generalized SSA

Since the basic singular spectrum analysis (SSA) [2, 3, 17] is designed for the analysis one dimensional sequences such as time series, it is not appropriate for the 3D polygonal mesh. In this section, we generalize the basic SSA in such a way that it can be applied to the analysis of multi-dimensional data such as polygonal meshes [9].

### 2.2 Generalized SSA

Let the elements of the series  $F$  be the values given to the vertices of the mesh. In the case of the 3D polygonal mesh, each element of the series  $F$  consists of the coordinates of a vertex in the mesh, i.e. the position vector. For simplicity, we consider the mesh specified by the heights of vertices of a graph on the plane as shown in Figure 1. Let the elements of the series  $F$  be the heights given to the vertices of the graph.

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 80-903100-7-9  
WSCG '2005, January 31-February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency - Science Press*

Let  $N$  be a positive (usually large) integer. Consider a real-value series  $F = (f_0, f_1, \dots, f_{N-1})$  of length  $N$ . Assume that  $F$  is a nonzero series, that is, there exist at least one  $i$  such that  $f_i > 0$ . The generalized SSA consists of two stages, the decomposition stage and the reconstruction stage, which are shown in the following two subsections.

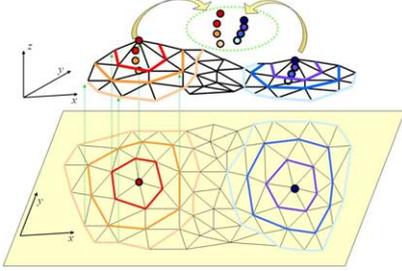


Figure 1. The mesh where height values are given to vertices of a graph on the plane.

### 2.2.1 Decomposition stage

The decomposition stage consists of the next two steps.

**1st step: Embedding** In the first step, let us define the linear operator  $\mathbf{A}$  which maps  $F$  to matrix  $\mathbf{X} = \mathbf{A}(F)$  as

$$\mathbf{A}(F) = \begin{pmatrix} FA_{0,0} & FA_{0,1} & \cdots & FA_{0,K-1} \\ FA_{1,0} & FA_{1,1} & \cdots & FA_{1,K-1} \\ \vdots & \vdots & \ddots & \vdots \\ FA_{L-1,0} & FA_{L-1,1} & \cdots & FA_{L-1,K-1} \end{pmatrix} \quad (1)$$

where

$$A_{l,k} = (a_{l,k,0}, a_{l,k,1}, \dots, a_{l,k,N-1})^T \quad (2)$$

and the elements of  $\mathbf{A}(F)$  are

$$FA_{l,k} = \sum_{n=0}^{N-1} a_{l,k,n} f_n. \quad (3)$$

We call the matrix (1) the trajectory matrix or the generalized trajectory matrix.

**2nd step: Singular value decomposition** In the second step, the singular value decomposition is applied to the trajectory matrix  $\mathbf{X}$ . Let  $\mathbf{S} = \mathbf{X}\mathbf{X}^T$ . Denote by  $\lambda_1, \dots, \lambda_L$  the eigenvalues of  $\mathbf{S}$  taken in the decreasing order of magnitude ( $\lambda_1 \geq \dots \geq \lambda_L \geq 0$ ), and by  $U_1, \dots, U_L$  the orthonormal system of the eigenvectors of the matrix  $\mathbf{S}$  corresponding to these eigenvalues. Let  $d = \max\{i \mid \lambda_i > 0\}$ . We define  $V_i = \mathbf{X}^T U_i / \sqrt{\lambda_i}$  and  $\mathbf{X}^{(i)T} = \sqrt{\lambda_i} U_i V_i^T$  ( $i = 1, \dots, d$ ). Then the singular value decomposition of the trajectory matrix  $\mathbf{X}$  can be written as

$$\mathbf{X} = \mathbf{X}^{(1)} + \mathbf{X}^{(2)} + \dots + \mathbf{X}^{(d)}. \quad (4)$$

The matrix  $\mathbf{X}^{(i)}$  has rank 1. Therefore they are elementary matrices. The collection  $(\lambda_i, U_i, V_i)$  is called  $i$ -th eigentriple of singular value decomposition (4).

### 2.2.2 Reconstruction stage

**3rd step: Reconstruction of the original series** In the last step, each matrix in the decomposition (4) is transformed into a new series of length  $N$ . This step is called the reconstruction of the series.

The series  $F^{(i)} = (f_0^{(i)}, f_1^{(i)}, \dots, f_{N-1}^{(i)})$  is defined as the solution of the next optimization problem:

$$\begin{aligned} \min \sum_{i=1}^d \|\mathbf{X}^{(i)} - \mathbf{A}(F^{(i)})\|^2 \\ = \min \sum_{i=1}^d \sum_{l,k} (x_{l,k}^{(i)} - F^{(i)} A_{l,k})^2 \end{aligned} \quad (5)$$

$$\text{s.t. } F = \sum_{i=1}^d F^{(i)}, \quad (6)$$

where the norm of the matrix is the Frobenius norm. If  $A_{l,k}$  ( $0 \leq l \leq L-1, 0 \leq k \leq K-1$ ) span  $N$  dimensional spaces, then the matrix  $\sum_{l,k} A_{l,k} A_{l,k}^T$  is regular, and consequently the solution of the expression (5) is given by

$$F^{(i)} = \left( \sum_{l,k} x_{l,k}^{(i)} A_{l,k}^T \right) \left( \sum_{l,k} A_{l,k} A_{l,k}^T \right)^{-1}. \quad (7)$$

Since this  $F^{(i)}$  satisfies

$$\begin{aligned} \sum_{i=1}^d F^{(i)} \sum_{l,k} A_{l,k} A_{l,k}^T &= \sum_{i=1}^d \sum_{l,k} x_{l,k}^{(i)} A_{l,k}^T \\ &= \sum_{l,k} \left( \sum_{i=1}^d x_{l,k}^{(i)} \right) A_{l,k}^T \\ &= \sum_{l,k} \left( FA_{l,k} \right) A_{l,k}^T \\ &= F \sum_{l,k} A_{l,k} A_{l,k}^T, \end{aligned}$$

the constraint (6) is satisfied automatically. The solution  $F^{(i)}$  of the optimization problem (5) and (6) is obtained by the expression (7).

Finally, from the expression (7), the original series  $F$  is reconstructed as  $F = \sum_{i=1}^d F^{(i)}$ . These are the basic ideas of the generalized SSA.

## 2.3 Linear operator $\mathbf{A}$

In this subsection, we give a particular example of the linear operator  $\mathbf{A}$  in the expression (1) that reflects the connectivity structure of the mesh.

Let  $P$  be a 3D polygonal mesh, and let  $v_k, k = 0, 1, \dots, N-1$  be the vertices of the mesh. Suppose that some scalar value  $f_k$  is assigned to each vertex

$v_k$ , and let  $F$  be the series  $F = (f_0, f_1, \dots, f_{N-1})$ . We define the distances  $D_{v_k}(v_j)$  from  $v_k$  to  $v_j$  as the number of edges in the shortest path in the graph where the length 1 is given to the all edges, so-called the Dijkstra distance. Figure 2 shows an example of a part of the graph structure associated with the polygonal mesh. Let  $v_k$  be the vertex represented by the black dot in Figure 2. Then, the vertices  $v_j$ 's with  $D_{v_k}(v_j) = 1$  are as shown by empty circles, and the vertices with  $D_{v_k}(v_j) = 2$  are as shown empty squares.

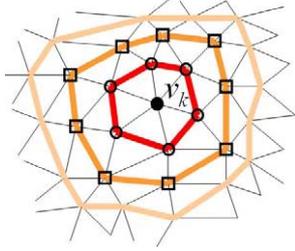


Figure 2. A vertex of the mesh and the set of vertices with the same Dijkstra distances.

Let the number of the rows of the trajectory matrix  $\mathbf{A}(F)$  be  $K := N$ . Let the elements on the first row and the  $k$ -th column ( $k = 0, 1, \dots, N - 1$ ) of the trajectory matrix  $\mathbf{A}(F)$  be the value  $f_k$  given to the vertex  $v_k$  and let the elements on the  $l$ -th ( $1 \leq l \leq L - 1$ ) row and the  $k$ -th column be the average value of the values on the vertices whose Dijkstra distances from  $v_k$  are  $l$ . Therefore the  $k$ -th column of the matrix  $\mathbf{A}(F)$  corresponds to the vertex  $v_k$  of the mesh. For example; for the vertex  $v_k$  in Figure 2.  $(1, k)$  element of  $\mathbf{A}(F)$  is  $f_k$ ,  $(2, k)$  element of  $\mathbf{A}(F)$  is the average of the value  $f_j$  over the vertices represented by the empty circles,  $(3, k)$  element of  $\mathbf{A}(F)$  is the average of the value  $f_j$  over the vertices represented by the empty squares. Thus, let  $FA_{l,k}$  be

$$FA_{l,k} = \frac{\sum_{D_{v_k}(v_j)=k} f_j}{\#\{D_{v_k}(v_j) = k\}} \quad (8)$$

where  $\#\{D_{v_k}(v_j) = l\}$  is the number of the vertices whose Dijkstra distances from  $v_k$  are  $l$ . The linear operator  $\mathbf{A}$  in the expression (8) is represented as

$$a_{l,k,n} = \begin{cases} \frac{1}{\#\{D_{v_k}(v_n)=l\}} & (D_{v_k}(v_n) = l), \\ 0 & (D_{v_k}(v_n) \neq l) \end{cases} \quad (9)$$

Note that, since the  $A_{l,k}$  ( $0 \leq l \leq L - 1, 0 \leq k \leq K - 1$ ) constructed as stated above span an  $N$ -dimensional space, the matrix  $\sum_{l,k} A_{l,k} A_{l,k}^T$  is usually regular. This is because  $A_{1,k}$  ( $0 \leq k \leq N - 1$ ) is the vector where the  $k$ -th element is 1 and the other elements are 0.

The rows of  $\mathbf{A}$  correspond to the vertices of the mesh, and the columns of  $\mathbf{A}$  correspond to the set of vertices with the same Dijkstra distances. Here the linear operator reflects the connectivity structure of the mesh.

## 2.4 Laplacian trajectory matrix

We may not obtain sufficient amount of eigen-triples for  $\mathbf{A}(F)$  by our method in subsection 2.3. If the amount of eigen-triples is small, some problems occur in engineering applications. For example, in the case of watermarking in section 4, we can not embed a lot of data. In order to solve this problem, we present a new Laplacian trajectory matrix in this subsection.

In subsection 2.3,  $f_h$  is assigned to each vertex  $v_h$ . In this subsection, in order to consider a new another trajectory matrix, let  $f'_h$  be Laplacian for a 3D polygonal mesh and  $f'_h$  is assigned to each vertex  $v_h$ .  $f'_h$  is defined as

$$f'_h = C\Delta f_h = C \left( \frac{\sum_{D_{v_h}(v_j)=1} f_j}{\#\{D_{v_h}(v_j) = 1\}} - f_h \right) \quad (10)$$

where  $C$  is a constant number. Figure 3 shows a mesh whose vertices  $v_h$  are transposed to Laplacian  $f'_h$ . Let  $F'$  be a series  $F' = (f'_0, f'_1, \dots, f'_{N-1})$ . Since  $f'_h$  is a linear combination of elements of  $F$ ,  $A'_{i,j}$  exists such that  $F'A_{i,j} = FA'_{i,j}$  and  $\mathbf{A}(F')$  can be transposed to  $\mathbf{A}'(F)$  as the following equation:

$$\begin{aligned} \mathbf{A}(F') &= \begin{pmatrix} F'A_{0,0} & \cdots & F'A_{0,K-1} \\ F'A_{1,0} & \cdots & F'A_{1,K-1} \\ \vdots & \ddots & \vdots \\ F'A_{L-1,0} & \cdots & F'A_{L-1,K-1} \end{pmatrix} \\ &= \begin{pmatrix} FA'_{0,0} & \cdots & FA'_{0,K-1} \\ FA'_{1,0} & \cdots & FA'_{1,K-1} \\ \vdots & \ddots & \vdots \\ FA'_{L-1,0} & \cdots & FA'_{L-1,K-1} \end{pmatrix} \\ &= \mathbf{A}'(F). \end{aligned} \quad (11)$$

A combined trajectory matrix  $\mathbf{B}(F)$  is defined as

$$\mathbf{B}(F) = \begin{pmatrix} \mathbf{A}(F) \\ \mathbf{A}'(F) \end{pmatrix}. \quad (12)$$

If we use  $\mathbf{B}(F)$ , we can get the double sets of eigen-triples for  $\mathbf{A}(F)$ .

We can expand  $\mathbf{B}(F)$  likewise. Let  $\mathbf{A}^0(F) = \mathbf{A}(F)$ ,  $\mathbf{A}^1(F) = \mathbf{A}'(F)$  and  $\mathbf{A}^i(F)$  be the duplicated Laplacian trajectory matrices by  $f_h^i$ , where  $f_h^i$  is defined as

$$f_h^i = C_i \Delta^i f_h. \quad (13)$$

$\mathbf{B}^i(F)$  is defined de by  $\mathbf{A}^0(F), \dots, \mathbf{A}^i(F)$  as

$$\mathbf{B}^i(F) = \begin{pmatrix} \frac{\mathbf{A}^0(F)}{\mathbf{A}^1(F)} \\ \vdots \\ \frac{\mathbf{A}^i(F)}{\mathbf{A}^i(F)} \end{pmatrix}. \quad (14)$$

If we use  $\mathbf{B}^i(F)$ , we can get the  $i$ -fold sets of eigen-triples for  $\mathbf{A}(F)$ .

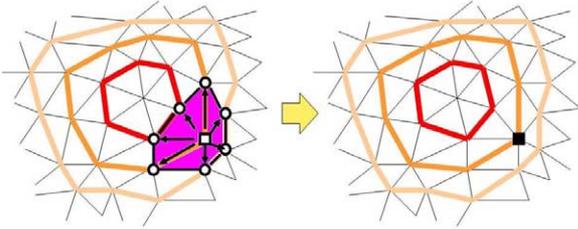


Figure 3. A mesh whose vertices are transposed to laplacian.

## 2.5 Comparisons of the decompositions

In this subsection, we compare the decomposition given by the generalized SSA, the basic SSA [6, 7, 8] and Laplacian matrix [15]. Table 1 shows their characteristics.

The 3D polygonal mesh represents the boundary of the three-dimensional region, and this boundary is a two-dimensional manifold. However, it is not easy to globally parameterize any two-dimensional manifold. Note that the three methods compared in this subsection do not require any parameterization of the boundary. On the other hand, the traditional multidimensional spectral decompositions such as the multidimensional Fourier transformation and the multidimensional wavelet transformation require the parameterization of the boundary, and hence cannot be used for general 3D polygonal meshes unless the mesh is partitioned into sub-meshes homeomorphic to disk. In this sense, these three methods are typical tools for the analysis of polygonal meshes.

Since the spectral decomposition of a mesh using the Laplacian matrix requires the eigenvalue decomposition of a matrix whose rank is the number of the vertices, the calculation cost is large and the method cannot be applied to huge meshes. On the other hand, our generalized SSA and the basic SSA require the eigenvalue decomposition of the matrix whose rank is determined by the linear operator e.g., the size of lag  $L$ . Moreover, we can choose a relative small value of  $L$ . Therefore, the calculation cost can be small, and consequently our generalized SSA and the basic SSA method can be applied to huge meshes.

The spectrum decomposition for 3D polygonal meshes is desired to be independent for the change of the vertex number. In other words, the basis of the spectrum decomposition is desired to be invariant from the change of the vertex number, because the shape of the mesh is invariant for the change of the vertex number. In the three methods, our generalized SSA and the method using the Laplacian matrix satisfy this requirement.

As stated above, our generalized SSA overcomes the respective demerits of the basic SSA and the method using the Laplacian matrix and can be a powerful new tool for the analysis of 3D polygonal meshes.

## 3 Spectral decomposition of 3D polygonal meshes using the generalized SSA

### 3.1 Spectral decomposition using the generalized SSA

In this subsection, we perform spectral decomposition of the 3D polygonal meshes using the generalized SSA.

Though we have been considering a scalar-value series  $F = (f_0, f_1, \dots, f_{N-1})$ , we hereafter consider tri-value series  $\mathbf{F} = (F_0, \dots, F_{N-1})$  where  $F_n = (f_{n,x}, f_{n,y}, f_{n,z})$  are the coordinates of the vertex  $v_n$ . Consequently, the trajectory matrix (1) is an  $L \times 3K$  matrix

$$\mathbf{X} = \mathbf{A}(\mathbf{F}) = \begin{pmatrix} \mathbf{F}A_{1,1} & \cdots & \mathbf{F}A_{1,K} \\ \mathbf{F}A_{2,1} & \cdots & \mathbf{F}A_{2,K} \\ \vdots & \ddots & \vdots \\ \mathbf{F}A_{L,1} & \cdots & \mathbf{F}A_{L,K} \end{pmatrix}, \quad (15)$$

where

$$\mathbf{F}A_{l,k} = \left( \sum_{n=0}^{N-1} a_{l,k,n} f_{n,x}, \sum_{n=0}^{N-1} a_{l,k,n} f_{n,y}, \sum_{n=0}^{N-1} a_{l,k,n} f_{n,z} \right). \quad (16)$$

We perform singular value decomposition (SVD) for this trajectory matrices.

In our experiments, we used two popular mesh models, the bunny model (1494 vertices, 2915 faces) shown in Figure 4 (a). Figure 4 shows that the original bunny model mesh is decomposed using trajectory matrix with  $L = 21$  and high frequency components are added gradually. Figure 4 (a) shows the original meshes. (b) shows the mesh constructed using the sum of the lowest frequency components. Figure 4 (c) or (d) shows the sum of 6 or 15 lower frequency components, respectively. Figure 5 is the decomposed mesh with  $L = 10$  and Figure 6 is the decomposed mesh with  $L = 5$ . Figure 7 is the decomposed mesh of  $L = 5$  using the

Table 1. Comparisons of three spectrum decomposition methods.

|   | decomposition using Laplacian matrix                                | decomposition using the basic SSA  | decomposition using the generalized SSA                          |
|---|---|--|--|
| decomposition algorithm                                 | eigenvalue decomposition of Laplacian matrix                        | the basic SSA  | the generalized SSA  |
| meaning of singular value or eigenvalue                 | frequency   | power spectrum   | power spectrum   |
| parameterizations on 3D polygonal meshes                | no parameterizations (merit)  | no parameterizations (merit)   | no parameterizations (merit)                                     |
| spectrum decomposition and changes of the vertex number | independence (merit)  | dependence (demerit)   | independence (merit)   |
| rank of the decomposed matrix (calculation cost)        | order of the vertices of the mesh (large calculation cost: demerit) | order of the lag $L$ , ( $L < \frac{N}{2}$ ) (small calculation cost: merit) | order of the number of the rings (small calculation cost: merit) |

mesh whose vertices are transposed to laplacian in subsection 2.4.

From Figures 4, 5, 6 and 7, we can confirm the following empirical fact. “Approximately, large singular values correspond to lower spatial frequencies, and small singular values correspond to higher spatial frequencies. Elementary matrices associated with higher singular values represent global shape features, while elementary matrices associated with lower singular values represent local or detail shape features. We made computational experiments in order to evaluate the performance of the proposed algorithms”.

The computer used in this experiment is Precision 330 of Dell with Intel Pentium 4 2.8G Hz processor and 1GB memory. Programming language is Mathematica 4.0. Calculation times were 11 minute in case of bunny mode with  $L = 10$ .

## 4 Application — Watermarking 3D polygonal meshes

In this section, we propose a new method of watermarking for the 3D polygonal meshes.

### 4.1 What is watermarking ?

Digital watermarking is a technique for adding secret information called a watermark to various target objects data. A lot of papers on watermarking have been published [5]. However most of the previous researches have been concentrating on watermarking “classical” object data types, such as texts, 2D still images, 2D movies, and audio data. Recently, on the other hand, 3D objects data, such as 3D polygonal meshes and various 3D geometric CAD data, become more and more popular and important, and hence techniques to watermark 3D models also become more important [1, 4, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20].

In the field of image watermarking, a majority of the watermarking algorithms published depends on some form of transformations, e.g., wavelet or Fourier transformations. This is because transformed domain techniques offer various advantages. For example, by modifying the spatial fre-

quency band which human are not very sensitive to, we can make a watermark embedded in an image less visible. Moreover, the transformed domain is a suitable place to hide the secret data (watermarks). Therefore, in those techniques of watermarking, some kind of spectrum decomposition is required.

This section presents experiments and results of an algorithm that embeds watermarks into 3D polygonal meshes. The proposed method is based on a new kind of spectrum decomposition, and can be used for any mesh structures, for details refer to [6, 8, 9, 10]. We propose a new algorithm for embedding watermarks into 3D polygonal meshes based on the generalized SSA. The spectra of the 3D polygonal mesh are computed by the singular decomposition of the trajectory matrix, and the watermarks are embedded into the singular values. The watermark embedded by the algorithm is robust against similarity transformation (i.e., rotation, translation, and uniform scaling). It is also resistant against random noises added to vertex coordinates. Figure 8 is the outline of embedding a watermark.

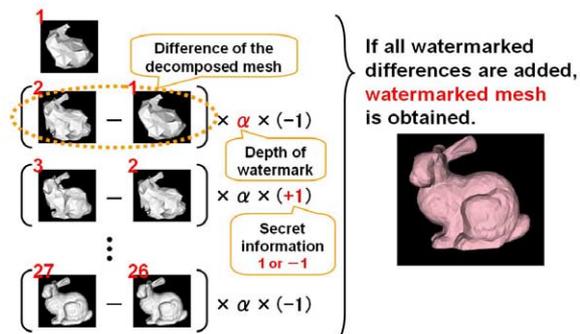


Figure 8. Outline of embedding watermark.

## 4.2 Experiments and results

### 4.2.1 Method

In our experiments, we used a popular mesh model, the bunny model (1494 vertices, 2915 faces) shown in Figure 4 (a).

We compare five watermarking methods. The first two watermarking methods are based on the

basic SSA using two kinds of the vertex series in [6, 8, 9, 10], which we call the “Euclidean norm method” and the “random order method”. The third watermarking method is the Laplacian matrix method proposed by Ohbuchi et al. [15]. We call this watermarking method the “Ohbuchi’s method”. The fourth watermarking method is based on our generalized SSA. We call this watermarking method the “generalized SSA ring 21” in the case of  $L = 21$ , and so on. The fifth watermarking method is based on our generalized SSA using Laplacian trajectory matrix in subsection 2.4. We call this watermarking method the “generalized SSA ring 5 Laplacian” in the case of  $L = 5$ . This method have 30 sets of eigentriples (i.e.,  $\times 3$  by  $xyz$ -coordinates and  $\times 2$  by Laplacian).

In the “generalized SSA” method, we embedded 15 bits data, and each bit was embedded only once (i.e., chip rate is 1). In the other method, we embedded 15 bits data 20 times (i.e., chip rate is 20). If a mesh is fixed, a higher chip rate means a lower data capacity and higher robustness.

The watermark embedding amplitudes  $\alpha$  is defined as  $\alpha = \beta \times l$  where  $l$  is the largest length of the edges of the axis-aligned bounding box of the target mesh and  $\beta$  is defined as a ratio of the amplitude. In this experiments,  $l = 156$  model was set. In Figure 9, the appearances for  $\beta = 0.1, 1$  are presented. If  $\alpha$  is larger, the watermark withstands against more disturbances, (for example, adding random noises and mesh smoothing) but the shape itself is distorted.

#### 4.2.2 Appearances of watermarked meshes

Figure 9 show appearances of the watermarked meshes generated by the generalized SSA ring 21, while (a) and (b) show the watermarked meshes for  $\beta = 0.1$  and 1, respectively. The appearances of (a) can hardly be distinguished from the appearances of the original mesh. Thus they are watermarked successfully. On the other hand, the appearances of the original meshes are not preserved in (b). Thus the watermarks are too large in those cases.

Table 2 shows RMS of the differences between the original meshes and the watermarked meshes divided by  $l$ . RMS (root mean square) is the mean of 2-norm between the vertices of the original mesh and the corresponding vertices of the watermarked mesh. In the appearances of the watermarked meshes, we cannot see much difference among the basic SSA methods and the generalized SSA method. In these experiments, we set  $\beta = 0.1$  in the basic SSA and the generalized SSA, and  $\beta = 0.0035$  in the Ohbuchi’s method.

#### 4.2.3 Robustness

We experimentally evaluated the robustness of our watermarks against the uniform random noises.

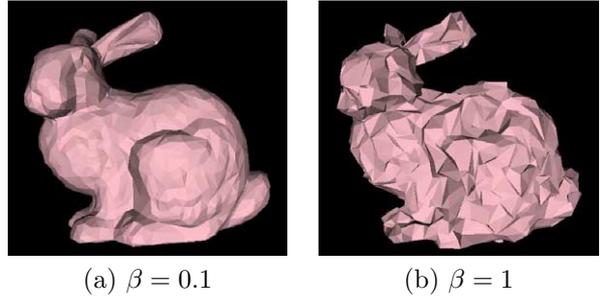


Figure 9. Watermarked bunny meshes.

Table 2.  $\frac{\text{RMS}}{l}$  of original meshes and watermarked meshes.

|  |        |
|--|--------|
| Euclidean norm ( $\beta = 0.1$ )                   | 0.0239 |
| random order ( $\beta = 0.1$ )                     | 0.0211 |
| Ohbuchi’s method ( $\beta = 0.0035$ )              | 0.0211 |
| generalized SSA ring 21 ( $\beta = 0.1$ )          | 0.0228 |
| generalized SSA ring 10 ( $\beta = 0.1$ )          | 0.0201 |
| generalized SSA ring 5 Laplacian ( $\beta = 0.1$ ) | 0.0134 |

**Uniform random noises** Figure 10 shows the appearances of the watermarked mesh whose vertex coordinates were disturbed with uniform random noises with amplitude  $\alpha \times \gamma$  ( $\beta = 0.1$ ). Figure 10 (a) are the meshes with uniform random noises with  $\gamma = 0.01$  and (b) are the meshes with uniform random noises with  $\gamma = 0.1$ . From Figure 10, we can see that the noises of  $\gamma = 0.1$  deformed the appearances of the original meshes to a certain extent.

We counted the number of the bits reconstructed correctly; we repeated the experiment 100 times. The result is shown in Table 3. From this experiment, we can see that the watermark can withstand against uniform noises for  $\gamma \leq 0.01$ . Moreover, we cannot see much difference among the five methods.

In the Euclidean norm method, the random order method and Ohbuchi’s method, the same bit was embedded many times (20 times, for example) because each bit is very fragile. On the other hand, in the proposed method, each bit is embedded only once, but still the watermark can be reconstructed almost in the same accuracy as the other methods, as shown in Table 3. In this sense, the proposed watermark method is very robust against random noises.

This robustness is due to the characteristic of the linear operator  $\mathbf{A}$ . Since the elements of the generalized trajectory matrix are represented as the linear combinations of the vertices of the mesh, these linear combinations counteract the uniform random noises in this step. Therefore, since the generalized SSA counteracts the uniform random noises before spectrum decomposition, while the other methods counteract the uniform random

noises after spectrum decomposition; we can see almost the same robustness against random noises among these methods.

Table 3. Ratios of the correctly recovered watermarks under random noises.

|                                  | $\gamma = 0.1$ | $\gamma = 0.01$ |
|----------------------------------|----------------|-----------------|
| Euclidean norm                   | 92%            | 100%            |
| random order                     | 98%            | 100%            |
| Ohbuchi's method                 | 98%            | 100%            |
| generalized SSA ring 21          | 96%            | 100%            |
| generalized SSA ring 10          | 99%            | 100%            |
| generalized SSA ring 5 Laplacian | 98%            | 100%            |

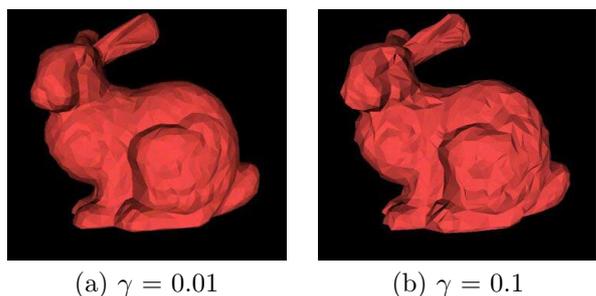


Figure 10. Bunny models to which uniform random noises with amplitude  $\alpha \times \gamma$  ( $\beta = 0.1$ ) are added.

## 5 Future Work

We have two future works. First future work is to develop new application area of generalized SSA. For that purpose, since there are large freedoms in the choice of the linear operator proposed in this paper, we need create new linear operator by considering the physical phenomenon of the target models. Second future work is to complete the theoretical framework of the generalized SSA.

## Acknowledgement

This work is partly supported by the 21st Century COE Program on Information Science and Technology Strategic Core, and the Grant-in-Aid for Scientific Research (S) of the Japanese Ministry of Education, Culture, Sports, Science and Technology.

## References

- [1] O. Benedens. Geometry-based watermarking of 3D models. In *IEEE CG*, pp. 46–55, 1999.
- [2] S. Broomhead and P. King. Extracting qualitative dynamics from experimental data. In *Physica D*, Vol. 20, pp. 217–2362, 1986.
- [3] B. Elsner, J. and A. Tsonis, A. *Singular Spectrum Analysis - A New Tool in Time Series Analysis*. Plenum Press, 1996.
- [4] S. Kanai, H. Date, and T. Kishinami. Digital watermarking for 3D polygons using multiresolution wavelet decomposition. In *Proceedings of the Sixth IFIP WG 5.2 International Workshop on Geometric Modeling: Fundamentals and Applications (GEO-6)*, pp. 296–307, 1998.
- [5] K. Matsui. *Basic of watermarks (in Japanese)*. Morikita Shuppan Publishers, 1998.
- [6] K. Murotani and K. Sugihara. Watermarking 3D polygonal meshes using the singular spectrum analysis. In *Proceedings of the 10th IMA International Conference on The Mathematics of Surfaces*, pp. 85–98, 2003.
- [7] K. Murotani and K. Sugihara. Watermarking 3d polygonal meshes using the singular spectrum analysis. In *ISM Symposium Statistics, Combinatorics and Geometry*, pp. 20–22, 2003.
- [8] K. Murotani and K. Sugihara. Generalized SSA and its applications to watermarking 3d polygonal meshes. In *METR METR 2004-17*, pp. 1–23, 2004.
- [9] K. Murotani and K. Sugihara. Watermarking 3d polygonal meshes using generalized singular spectrum analysis. In *NICOGRAPH International Conference 2004 in Taiwan*, pp. 121–126, 2004.
- [10] K. Murotani. Spectral decomposition method for three-dimensional shape models and its applications. In *Doctoral thesis (Information Science and Technology, University of Tokyo)*, 2004.
- [11] R. Ohbuchi, H. Masuda, and M. Aono. Watermarking three-dimensional polygonal models. In *Proceedings of the ACM International Conference on Multimedia '97*, pp. 261–272, 1997.
- [12] R. Ohbuchi, H. Masuda, and M. Aono. Targeting geometrical and non-geometrical components for data embedding in three-dimensional polygonal models. In *Computer Communications*, Vol. 21, pp. 1344–1354, 1998.
- [13] R. Ohbuchi, H. Masuda, and M. Aono. Watermarking three-dimensional polygonal models through geometric and topological modifications. In *IEEE Journal on Selected Areas in Communication*, Vol. 16, No. 4, pp. 551–560, 1998.
- [14] R. Ohbuchi, H. Masuda, and M. Aono. A shape-preserving data embedding algorithm for NURBS curves and surfaces. In *Proceedings of the Computer Graphics International'99*, pp. 7–11, 1999.
- [15] R. Ohbuchi, S. Takahashi, T. Miyazawa, and A. Mukaiyama. Watermarking 3D polygonal meshes in the mesh spectral domain. In *Proceedings of the Graphics Interface 2001*, pp. 9–17, 2001.
- [16] E. Praun, H. Hoppe, and A. Finkelstein. Robust mesh watermarking. In *ACM SIGGRAPH 1999*, pp. 69–76, 1999.
- [17] R. Vautard, P. Yiou, and M. Ghil. Singular spectrum analysis: A toolkit for short noisy chaotic signals. In *Physica D*, Vol. 58, pp. 95–126, 1992.
- [18] G. Wagner, M. Robust watermarking of polygonal meshes. In *Proceedings of Geometric Modeling & Processing 2000*, pp. 201–208, 2000.
- [19] B-L. Yeo and M. Yeung, M. Watermarking 3D objects for verification. In *IEEE CG&A*, pp. 36–45, 1999.
- [20] K. Yin, Z. Pan, J. Shi, and D. Zhang. Robust mesh watermarking based on multiresolution processing. In *Computers & Graphics*, Vol. 25, pp. 409–420, 2001.

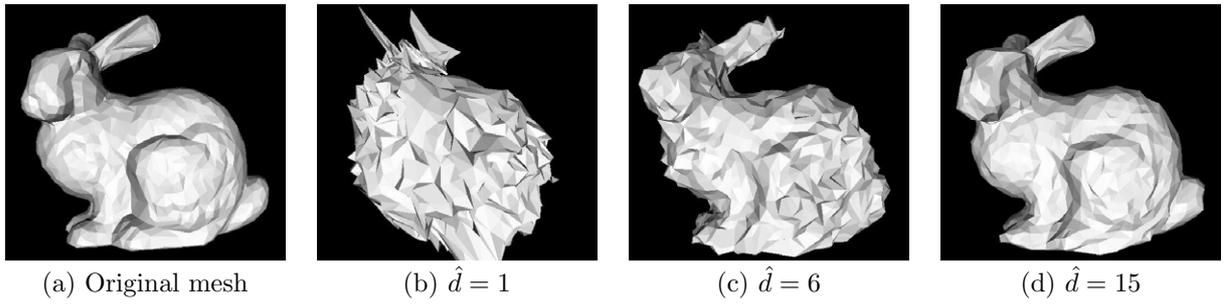


Figure 4. The decomposed bunny model of  $L = 21$ . The sum of  $\hat{d}$  lower frequency components.

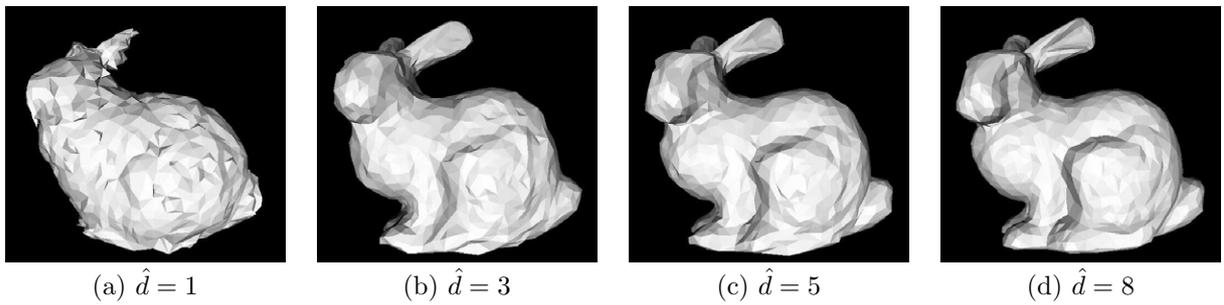


Figure 5. The decomposed bunny model of  $L = 10$ . The sum of  $\hat{d}$  lower frequency components.

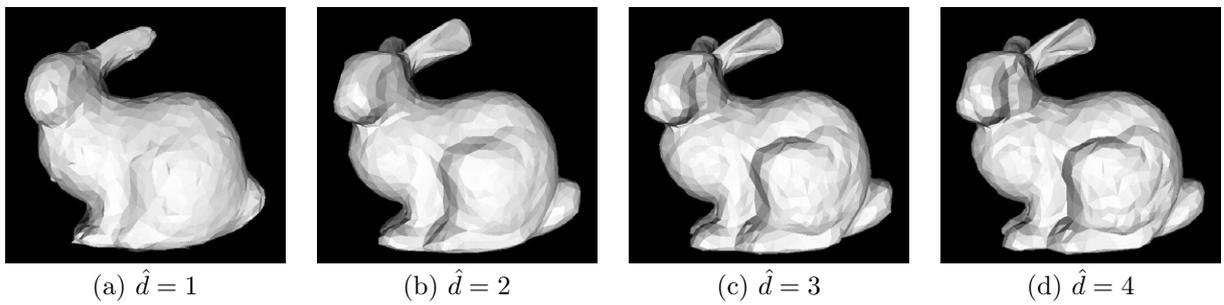


Figure 6. The decomposed bunny model of  $L = 5$ . The sum of  $\hat{d}$  lower frequency components.

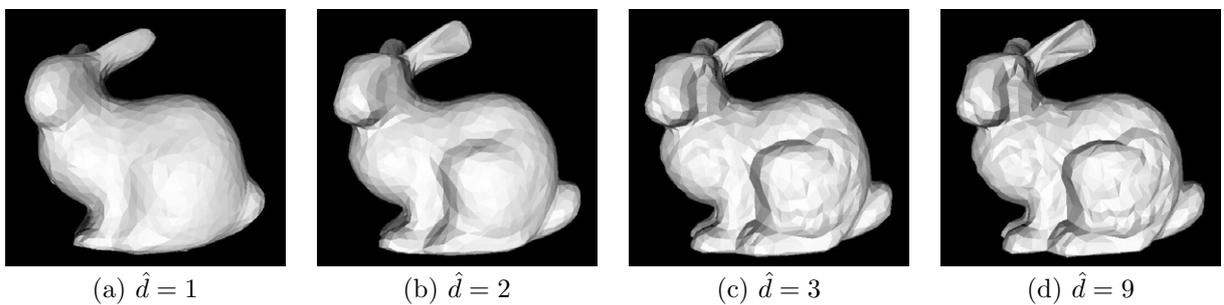


Figure 7. The decomposed bunny model of  $L = 5$  using the mesh whose vertices are transposed to laplacian. The sum of  $\hat{d}$  lower frequency components.

# A Subdivision Scheme to Model Surfaces with Spherelike Features

|                                    |                                    |                                    |
|------------------------------------|------------------------------------|------------------------------------|
| Koen Beets                         | Johan Claes                        | Frank Van Reeth                    |
| Limburgs Universitair Centrum      | Limburgs Universitair Centrum      | Limburgs Universitair Centrum      |
| Expertise Centre for Digital Media | Expertise Centre for Digital Media | Expertise Centre for Digital Media |
| Universitaire Campus               | Universitaire Campus               | Universitaire Campus               |
| B-3590, Diepenbeek, Belgium        | B-3590, Diepenbeek, Belgium        | B-3590, Diepenbeek, Belgium        |
| koen.beets@luc.ac.be               | johan.claes@luc.ac.be              | frank.vanreeth@luc.ac.be           |

## ABSTRACT

In this paper, we introduce a novel subdivision method able to generate smooth surfaces which locally tend to minimize variations in curvature. The method is based on a tensor product of a subdivision scheme for circle splines, which is then generalized to arbitrary quadrilateral meshes.

Although they involve a geometric construction, our rules are applied in a uniform way, without the need for applying different rules for different vertices or for different stages in the subdivision process. This results in a more general and natural way to obtain circular curvatures, unlike other approaches involving subdivision curves able to generate circles. Surfaces of revolution are just a basic example, as circular features can be distributed freely over the surfaces generated via our methods.

## Keywords

Curve and surface modeling, interpolatory subdivision, curvature minimization, circle splines

## 1. INTRODUCTION

Subdivision surfaces are widely used in the graphics community. A major advantage is their ability to generate surfaces with arbitrary topology in a uniform representation based on a freely editable coarse polygonal mesh. Due to their close relationship with multiresolution and wavelet analysis, their practical applications further benefit from a vast amount of theoretical knowledge. We refer the interested reader to the Siggraph 2000 course by Zorin et al. [Zor00] and to the book by Warren and Weimar [War02] for excellent introductions to subdivision techniques, with many pointers to the continuously developing literature.

In this paper, our attention goes to surfaces which locally resemble sphere regions. Conventional subdivision methods seem to be inadequate as

although the control points in a certain region are all located on the same sphere, the resulting surface usually exhibits a highly varying curvature. This variation results especially problematic for the schemes which directly interpolate the control points provided by the user instead of only approximating them.

Our study of the related literature started with curve representations based on circle blending. Compared to global optimization techniques, such as the Minimized Variation Curve [Seq92], local blending is much cheaper to compute. Furthermore, such global techniques have the additional disadvantage that a local change in the input may have a global impact on the resulting curve, something highly undesirable during interactive modeling.

Circular spline schemes usually employ 4 consecutive vertices  $P_0, P_1, P_2, P_3$  to generate a curve segment with minimal curvature variation between  $P_0$  and  $P_1$ . An interpolation scheme combines the segments resulting from the circle  $C_1$  through  $P_0, P_1, P_2$  and the circle  $C_2$  through  $P_1, P_2, P_3$  (see Figure 1). Various interpolation schemes have been proposed. Wenz blends the two circles using simple linear interpolation of 2 point positions on the base arcs [Wen96]. To improve the tangent continuity at the joints of segments, Szilvasi-Nagi and Vendel propose trigonometrically weighted interpolation, which guarantees  $G^2$  continuity [Szi00]. To further

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 2005 conference proceedings, ISBN 80-903100-7-9  
WSCG'2005, January 31-February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency – Science Press

improve the curve quality, Séquin et al. constructed a  $C^2$  circular curve blending scheme [Seq05]. Instead of interpolating point positions, they suggest a trigonometrical interpolation between tangent angles. As Séquin et al. argue, this approach remedies the cusps and sharp loops which can appear in the previous schemes, in particular when the control polygon features sharp corners.

In a theoretical work, Chris Doran shows how Clifford algebras help to define a circle blending with an arbitrary level of G continuity [Dor03]. His employment of conformal transformations leads to curves which are – in the  $G^2$  case – identical to the curves presented by Séquin et al. [Seq05]. For us this forms an extra argument to also adopt angle based interpolation. Doran furthermore extended his approach to sphere blending, but unfortunately this did not yet lead to practical methods for surface construction.

Circle blending techniques are interesting to generate curves, but for our subdivision surfaces, we also need a subdivision approach for the underlying curves. In 1987, Dyn et al. introduced the first interpolating subdivision scheme for curves, known in the literature as the four-point scheme or also as the DLG scheme [Dyn87]. Starting from a coarse control polygon, new points are recursively introduced between each pair of old points. The new point positions are defined as the central point of a spline which interpolates the four immediate neighbors.

As the four-point scheme generates curves with highly varying curvature, Séquin and Yen constructed a circular subdivision scheme. Their new point positions are now calculated to lie at the centre of an angle-based interpolated arc [Seq01].

Sabin and Dodgson provided yet another solution to create subdivision curves with more continuous curvature [Sab04]. With a particular definition of curvature as a kind of normalized cross product, they ensure that the new point's curvature averages the curvature of its immediate neighbors. Additionally, to obtain a more even spacing of vertices after subdivision, they position new vertices closer to the shortest edge adjacent to the current edge. The resulting scheme is  $C^2$  in practical situations, just as the four-point scheme.

In the literature, also some subdivision surface schemes incorporating circular arcs are described. Nasri et al. started from an interpolating subdivision algorithm for piecewise  $C^1$  circular spline curves, based on biarcs [Nas01]. This is used to create a modified version of the Doo-Sabin scheme for surfaces, where the standard rules are combined with the circular rules on user-defined edges. This way

surfaces with piecewise circular boundaries can be created as well as seamless connections between different surfaces along a common circular boundary. A disadvantage of their technique is that the curvature changes abrupt at the control vertices. Also, to create circular or spherical regions, extra vertices have to be added to the control polygon, while it is not always clear how to add these.

In a similar approach, Morin et al. describe a non-stationary subdivision scheme for surfaces of revolution [Mor01]. With their technique, the user has to mark the desired sections of the curve as circular. Between circular arcs, standard Catmull-Clark subdivision rules are applied. Both Morin et al.'s and Nasri et al.'s schemes are approximating, while we want to create a scheme that interpolates all of its control vertices. Also, we intend to have a more continuously varying curvature everywhere, more than only at certain indicated circular regions.

Our subdivision surface scheme is inspired by Kobbelt's interpolating scheme for quadrilateral meshes [Kob96]. Kobbelt first created a tensor product of the four-point scheme for curves and then extended this to meshes with arbitrary topology.

The rest of this paper is organized as follows: In section 2 we describe a subdivision scheme for curves, which minimizes local variations in curvature. Afterwards we employ the same idea to construct a subdivision scheme for surfaces, and explain how it is constructed. Next, we propose some applications for which the spherelike scheme is very well suited. Finally we present some results which we compare to results of well-known subdivision schemes, and we formulate our conclusion.

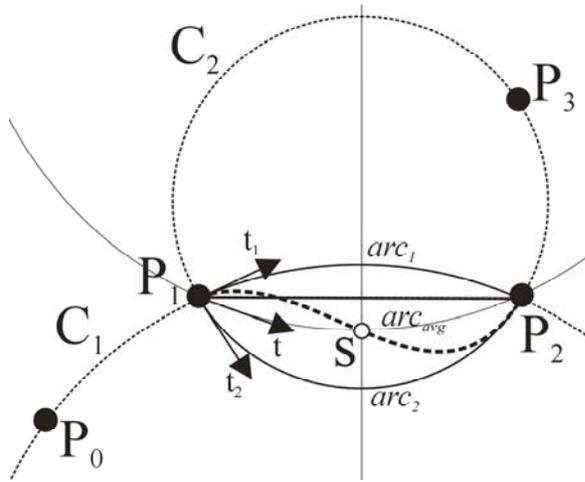
## 2. THE CURVE SCHEME

The subdivision scheme for curves works as follows: For every couple of adjacent vertices  $P_1$  and  $P_2$ , we consider the 4 consecutive vertices  $(P_0, P_1, P_2, P_3)$ . Since every circle can be determined by three vertices, one can fit exactly one circle  $C_1$  going through  $(P_0, P_1, P_2)$ , and also one circle  $C_2$  through  $(P_1, P_2, P_3)$ . This situation is illustrated in Figure 1. Between  $P_1$  and  $P_2$ , both circles have an arc  $arc1$  and  $arc2$ , which is parameterized to have parameter  $u=0$  at  $P_1$ , and  $u=1$  at  $P_2$ .

The scheme blends both arcs between  $P_1$  and  $P_2$ , creating a curve segment minimizing curvature changes. First we calculate the tangent vectors  $t_1$  and  $t_2$  in  $P_1$ , and their average  $t$ . The arc  $arc_{avg}$  which has a tangent vector equal to  $t$  in  $P_1$  is created. On this arc we take the central vertex to be  $S$ , and insert it into the new curve.

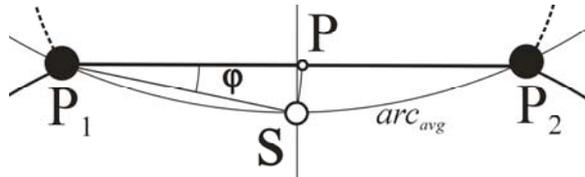
After several iterations of the recursive subdivision scheme, we obtain a smooth segment, which is

shown as a fat dashed line between  $P_1$  and  $P_2$  in Figure 1.



**Figure 1: Blending two circle segments between  $P_1$  and  $P_2$ .** Every iteration, the tangents  $t_1$  and  $t_2$  to the circles  $C_1$  and  $C_2$  are calculated in  $P_1$ . Then the arc passing through  $P_1$  and  $P_2$ , and having the average tangent  $t$  is calculated. The vertex  $S$ , lying in the middle of this  $arc_{avg}$  is added to the curve.

The algorithm we use is based on Séquin's circular subdivision scheme for curves [Seq01]. Séquin rotates a vertex  $P$  with distance  $f(u) = b * \sin(u * t(u)) / \sin(t(u))$  from  $P_1$  lying on  $P_1P_2$  an angle  $\phi(u) = (1-u)t(u)$  around the axis  $rot\_axis = P_0P_1 \times P_1P_2$ , where  $b = |P_1P_2|$ . The resulting vertex  $S$  lies on the average arc  $arc_{avg}$ . For  $u=0.5$ , this is shown in Figure 2.



**Figure 2: Obtaining  $S$  using matrix rotation**

We instead suggest not rotating this point  $P$  using matrix rotation, but instead we propose to use a geometric construction to obtain  $S$ , using vector calculation. Let  $P_{12} = (P_2 - P_1) / |P_2 - P_1|$ . We calculate the cross product  $N = rot\_axis \times P_{12}$ , which is a unity vector, because both  $rot\_axis$  and  $P_{12}$  are unity vectors and are perpendicular to each other.

Then we obtain:

$$S = P_1 + \cos(\phi) * f * P_{12} + \sin(\phi) * f * N$$

Advantages for using this method instead of matrix rotations are faster calculations of the new points, and higher accuracy.

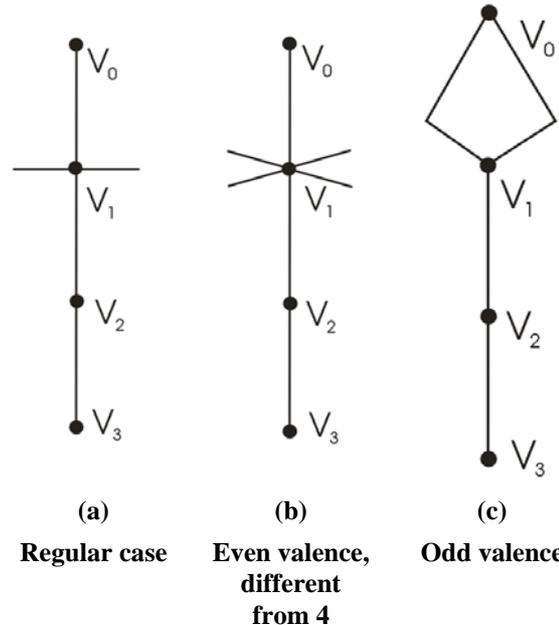
### 3. THE SURFACE SCHEME

In this paragraph we describe how we extend the techniques presented in the previous section to surfaces, generating smooth and interpolating surfaces of arbitrary topology. Starting from a coarse control mesh, the algorithm recursively refines the mesh. At each iteration, the number of faces is multiplied by four. The algorithm works as follows: First, all edges are split into two, while all original vertices are retained. Then, new face vertices are placed inside every face. Finally, the new mesh is reconnected, replacing every old  $n$ -sided face with  $n$  new quadrangles. In the next paragraphs we describe the algorithm in more detail.

First, all edges are split into two, using the rule for curves. For every edge which is not part of a boundary, we take the two end vertices  $V_1$  and  $V_2$ , and locate  $V_0$  and  $V_3$  (see Figure 3).

Figure 3a illustrates the situation in the regular case. We apply the curve algorithm to these four vertices, and split the original edge in two by inserting a vertex  $E$ . If, however, the valence of a vertex belonging to the curve is different from 4, we use other rules.

Suppose we have a vertex  $v$  with valence different from four. There are two different cases: If  $v$  has an even valence, we take the most central edge to obtain the other vertices used for calculating the edge split. This is illustrated in Figure 3b. If  $v$  has an odd valence, we choose  $V_0$  or  $V_3$  as the vertex which is the furthest away from  $V_1$ , and belonging to the central face. This is illustrated in Figure 3c.



**Figure 3: Different situations around an edge with vertices  $V_1$  and  $V_2$ .** In subfigure a, the situation with a regular vertex  $V_1$  is shown. Subfigure b

shows a vertex  $V_1$  with an even valence different from 4, while subfigure c displays a vertex  $V_1$  with an odd valence. In each situation, the vertices  $V_0$  and  $V_3$  are located.

When all vertices  $V_1, V_2, V_3$  and  $V_4$  are found, a new vertex  $E$  is inserted between  $V_1$  and  $V_2$ . This is illustrated in Figure 4.

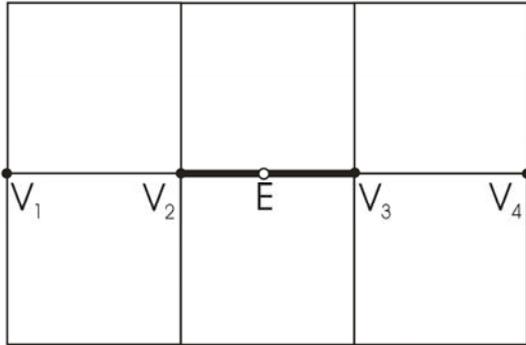


Figure 4: The mask for splitting edges using spherelike interpolating subdivision for surfaces. Vertex  $E$  is inserted in every edge, using the algorithm for curves. This algorithm is applied to the vertices  $(V_1, V_2, V_3, V_4)$ .

Secondly, a face vertex  $F$  is created inside every face. The creation of a face point in the regular case is illustrated in Figure 5. We apply the scheme for curves to the vertices  $(V_1, V_2, V_3, V_4)$  and to the vertices  $(V_5, V_6, V_7, V_8)$ . Note that both will not give the same result. Thus we add a new face vertex with the average coordinates. Since the scheme is interpolating, existing vertices are left unchanged. Finally, the old faces are discarded, and new faces are created by connecting every old vertex with its two adjacent edge splits, and with a face vertex of an adjacent old face.

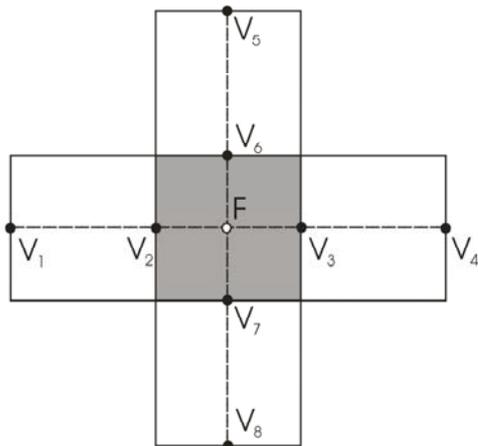


Figure 5: Creation of a new face vertex  $F$  inside the gray quadrangle, in the regular case. The points  $V_i$  are used for calculating  $F$ .  $F$  becomes the average of the curve subdivision applied to  $(V_1, V_2, V_3, V_4)$  and  $(V_5, V_6, V_7, V_8)$  respectively.

In the extraordinary case, we employ a different rule for generating new face vertices. In this case, we can not simply select 2 paths of edges passing through the centre of the face, since the number of edge splits in the face will be different from 4. So there is a problem picking the second edge vertex. We discern two different cases here: If the number of vertices  $n$  is even, we calculate the  $n/2$  curve subdivisions using the new edge vertices. Then we take the average of these results. This is shown in Figure 6 (left). If the face has an odd number of vertices, say  $n$ , we take the new face vertex to be the average of  $n$  calculations of the curve scheme, using the vertices of the face. This is illustrated in Figure 6 (right).

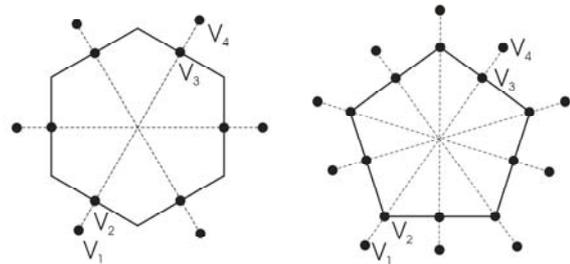


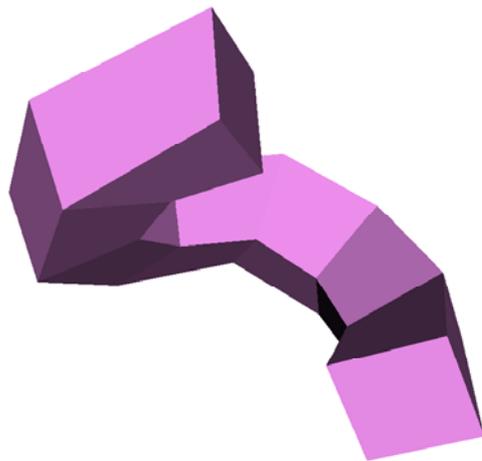
Figure 6: Calculating face points in the extraordinary case: a face with an even number of vertices (left), and a face with an odd number of vertices (right).

#### 4. MODELING APPLICATIONS

There are many applications which may benefit from using the interpolating spherelike subdivision scheme. We enumerate some examples.

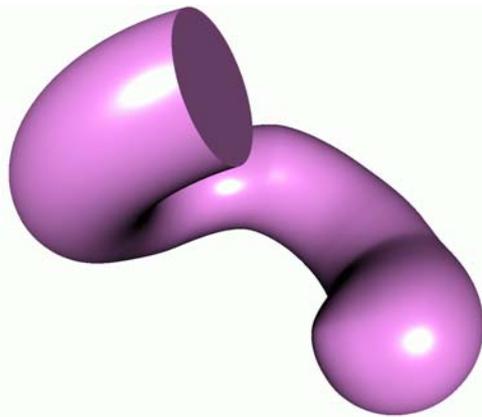
A first application would be the smoothing of polygonal objects, since the scheme does not shrink the object. Selective smoothing of edges is also possible.

Another application is the efficient generation of cylinder-like objects and tubes. Starting from a random curve in 3D, one can sweep a circle over this curve, with the curve going through the centre of the circle. Throughout this path, the diameter of the circle may change, or the circle may change shape. At the end points, one may use a sphere to produce a round end point, or a flat plane. An example object generated using this technique is shown in Figure 7.



(a)

Control mesh



(b)

Subdivided surface

**Figure 7: An example tubular object generated using the interpolating spherelike scheme.**

Also, objects like surfaces of revolution can be represented easily using the spherelike scheme. An example chess pawn is shown in Figure 8b, along with the control mesh in Figure 8a.

Finally the scheme can be used to support boolean operations. An example would be to create a smooth spherelike blending between two cylinders, where one cylinder cuts another cylinder under an angle.

## 5. RESULTS

Figure 8 shows a visual comparison of a chess pawn, subdivided using different subdivision schemes. Our spherelike scheme combines the advantages of both interpolating and approximating subdivision: it generates the smooth surface of approximating schemes like Catmull-Clark, while still interpolating the control points. This interpolation is an important

feature for application of subdivision surfaces in engineering applications. The interpolating spherelike scheme generates a round pawn while preserving the features well. The surface generated by Kobbelt's scheme is not round enough, while the Catmull-Clark surface lacks the necessary features. Clearly these schemes need a different – and more complex – control mesh to generate a realistic pawn.

## 6. CONCLUSION

We presented a new subdivision scheme for surfaces, which is interpolating, and which minimizes local variations in curvature. It is well suited to efficiently produce surfaces with spherelike regions.

For general use, subdivision surfaces are not suitable yet. Gonsor and Neamtu present a list of problems with subdivision in engineering applications [Gon01]. Several problematic properties of subdivision surfaces are mentioned which may be alleviated by our scheme: The scheme is interpolating, while still generating good quality surfaces, in contrast to existing interpolating schemes. Secondly, our scheme creates surfaces with good curvature. Finally, our scheme is locally refinable, while still maintaining exactly the same shape, unlike most other schemes.

Future work includes a thorough analysis of the behavior of the scheme, and adding support for adaptive subdivision.

## 7. ACKNOWLEDGMENTS

The authors are pleased to acknowledge that this work has been partially funded by the European Fund for Regional Development and the Flemish Government.

## 8. REFERENCES

- [Dor04] Doran, C. Circle and Sphere Blending with Conformal Geometric Algebra. Submitted to Computer Aided Geometric Design, 2004
- [Dyn87] Dyn, N., Gegory, J., and Levin, D. A 4-point Interpolatory Subdivision Scheme for Curve Design. Computer Aided Design, Volume 4, pp. 257-268, 1987
- [Gon01] Gonsor, D., and Neamtu, M. Can subdivision be useful for geometric modeling applications? Boeing Technical Report #01-011, <http://www.math.vanderbilt.edu/~neamtu/papers/report.pdf.gz>, 40pp, 2001
- [Kob96] Kobbelt, L. Interpolatory Subdivision on Open Quadrilateral Nets with Arbitrary Topology. Computer Graphics Forum, Volume 15, Issue 3, pp. 409-420, 1996
- [Mor01] Morin, G., Warren, J., and Weimer, H. A. Subdivision Scheme for Surfaces of Revolution. Computer Aided Geometric Design, Volume 18, Issue 5, pp. 483-502, 2001
- [Nas01] Nasri, A., van Overveld, C., and Wyvill, B. A Recursive Subdivision Algorithm for Piecewise

Circular Spline. *Computer Graphics Forum*, Volume 20, Number 1, pp. 33-45, 2001

[Sab04] Sabin, M., and Dodgson, N. A Circle-Preserving Variant of the Four-Point Subdivision Scheme, *Proceedings of the 6<sup>th</sup> International Conference on Mathematical Methods for Curves and Surfaces*, Accepted, 2004

[Seq92] Séquin, C., and Moreton, H. Functional Optimization for Fair Surface Design. *SIGGRAPH 1992 Proceedings*, pp. 167-176, 1992

[Seq01] Séquin, C., and Yen, J. Fair and Robust Curve Interpolations on the Sphere. *SIGGRAPH 2001 Sketch*, p. 182, 2001

[Seq05] Séquin, C., Lee, K., and Yen, J. Fair,  $G^2$ - and  $C^2$ -Continuous Circle Splines for the Interpolation of

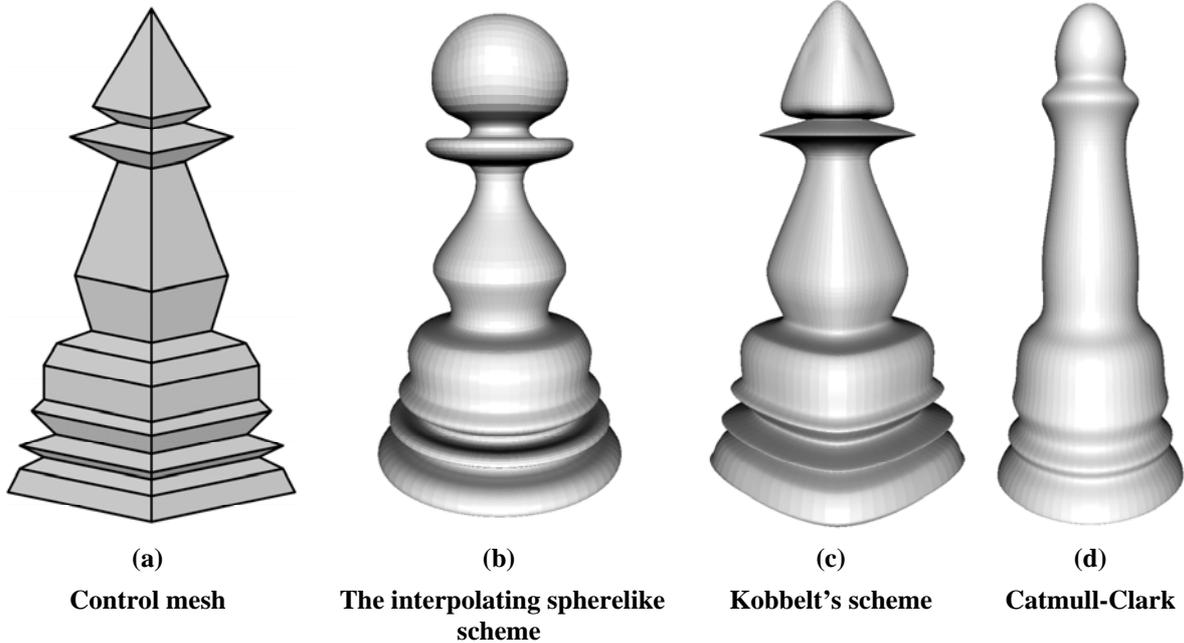
Sparse Data Points. *Journal of Computer-Aided Design*, Volume 37, Issue 2, pp 201-211, 2005

[Szi00] Szilvási-Nagy, M., and Vendel, T.P., Generating Curves and Swept Surfaces by Blended Circles, *Journal of Computer Aided Geometric Design*, Volume 17, Issue 2, pp 197-206, 2000

[War02] Warren, J., and Weimer, H. *Subdivision Methods For Geometric Design: A Constructive Approach*. published by Morgan Kaufmann. ISBN: 1558604464, 2002

[Wen96] Wenz, H.-J. Interpolation of Curve Data by Blended Generalized Circles. *Computer Aided Geometric Design*, Volume 13, Issue 8, pp. 673-680, 1996

[Zor00] Zorin D., Schröder P. *Subdivision for Modeling and animation*, SIGGRAPH 2000 course notes, 2000



**Figure 8: A visual comparison of an object of revolution, subdivided with various subdivision schemes.**

# Anisotropic Sampling for Differential Point Rendering of Implicit Surfaces

Florian Levet<sup>1</sup>   Julien Hadim<sup>1</sup>   Patrick Reuter<sup>1,2</sup>   Christophe Schlick<sup>1</sup>

<sup>1</sup> LaBRI - CNRS - INRIA - University of Bordeaux   <sup>2</sup> LIPSI - ESTIA  
{levet,hadim,preuter,schlick}@labri.fr

## ABSTRACT

In this paper, we propose a solution to adapt the *differential point rendering* technique developed by Kalaiah and Varshney to implicit surfaces. Differential point rendering was initially designed for parametric surfaces as a two-stage sampling process that strongly relies on an adjacency relationship for the samples, which does not naturally exist for implicit surfaces. This fact made it particularly challenging to adapt the technique to implicit surfaces. To overcome this difficulty, we extended the *particle sampling* technique developed by Witkin and Heckbert in order to locally account for the principal directions of curvatures of the implicit surface. The final result of our process is a *curvature driven anisotropic sampling* where each sample "rules" a rectangular or elliptical surrounding domain and is oriented according to the directions of maximal and minimal curvatures. As in the differential point rendering technique, these samples can then be efficiently rendered using a specific shader on a programmable GPU.

**Keywords:** *Geometric Modeling, Implicit Surfaces, Differential Geometry, Point Rendering*

## 1 Introduction

Implicit surfaces are an elegant surface representation to model 3D surfaces without explicitly having to account for topology issues. Moreover, it is possible to develop a complete modeling-animating-rendering pipeline with almost no topological constraints by using ray-tracing to render the corresponding surfaces. Unfortunately, to be able to provide a decent rendering of implicit surfaces at interactive framerates, there is usually no other choice than to convert them into polygonal meshes, which inherently reintroduces heavy topological constraints.

In 2001, Kalaiah and Varshney [17, 16] proposed an innovative technique to render parametric surfaces in higher quality. The basic idea of their *differential point rendering* technique is to generate a discrete sampling of the parametric surface, where each sample locally defines the differential geometry (i.e. the position, the

tangent plane, as well as the minimal and maximal direction of curvature). All the samples are individually rendered without any connectivity information by point rendering using a specific rectangular "splatter" accounting for the local differential geometry. This splatter can then be efficiently rendered by specific shaders on a programmable GPU (see also recent work of Botsch et al. [5]).

The differential point rendering technique is particularly well adapted to parametric surfaces since the samples can be generated explicitly using the local differential geometry, and the technique can be simply extended to triangular meshes by estimating the differential geometry at the mesh vertices.

Since the differential point rendering technique offers high quality rendering of surfaces using less surface samples, it is quite appealing to extend it for implicit surfaces as well. This was the initial motivation of the work we present here. Unfortunately, the extension is not as straightforward as it may appear at first glance. The main concern is that the sampling technique proposed by Kalaiah and Varshney is basically a two-stage process. In the first stage, a relatively dense point sampling of the surface is computed, either by direct sampling of the parameter space (in the case of a parametric surface), or by using the existing vertices (in the case of a triangular mesh). Then, during the second stage, many of the samples are removed using a simplification process that accounts for the local differential geometry, i.e. keeping more samples in 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.

Conference proceedings ISBN 80-903100-7-9  
WSCG'2005, January 31-February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency-Science Press

directions of high curvature and less samples in the directions of low curvature. The final result is a *curvature driven anisotropic sampling* where each sample "rules" a rectangular or elliptical surrounding domain, locally oriented according to the directions of maximal and minimal curvatures. The problem is, that this second stage is heavily based on the adjacency relationship between the samples. This relationship naturally exists in a triangular mesh or in a regular sampling of a parametric surface, but it has no natural counterpart for an implicit surface. As a consequence, even if it can theoretically be implemented, the two-stage process proposed by Kalaiah and Varshney is not well adapted to implicit surfaces.

The goal of this paper is to present an alternative solution for implicit surfaces to obtain a curvature driven anisotropic sampling that offers similar properties as the one generated by Kalaiah and Varshney. Our solution is based on a particle system, in the spirit of the one initially proposed by Witkin and Heckbert [31], but we also account for anisotropic sampling using local differential geometry.

The remainder of the paper is organized as follows: Section 2 presents some related previous work mainly dealing with sampling techniques for implicit surfaces. Section 3 details our new anisotropic sampling technique for implicit surfaces. Section 4 presents several experimental results that focus on the visual quality and the convergence speed. Section 5 concludes and presents some directions we are currently studying. Finally, the mathematical background to compute the principal curvature directions of an implicit surface as well as the corresponding curvature amounts is given in the Appendix.

## 2 Previous work

Since the groundbreaking work done in 1985 by Levoy and Whitted [20] which was revisited by Grossman and Dally in 1998 [13], point rendering has become a popular research domain in recent years. But as our goal is to adapt the differential point rendering technique to implicit surfaces, it is out of the scope of this paper to recall all existing point rendering techniques. We refer the interested reader to a recent overview [18] and tutorial [1].

We will thus focus more precisely on sampling techniques for implicit surfaces. Basically, existing work can be divided into two main families: tessellating techniques and particle system techniques.

### 2.1 Tessellating implicit surfaces

The tessellating techniques of implicit surfaces can themselves be divided into three categories. First, *spatial sampling techniques* subdivide the 3D space into

cells, commonly either cubes or tetrahedra, and search for the cells that intersect the implicit surface. One of the most commonly known spatial sampling techniques is the marching cubes algorithm [32, 21], that divides the 3D space into cubic cells, and triangles are generated according to the sign at the corners of the cells. Unfortunately, there are ambiguous configurations that have to be resolved [12]. Marching tetrahedra algorithms, e.g. by Shirley and Tuchman [23] or Hall and Warren [14], further divide the cubic cells into tetrahedra, and for each tetrahedra there are no ambiguous configurations. Nevertheless, marching tetrahedra algorithms create numerous, often over distorted triangles. In both the marching cubes and the marching tetrahedra algorithms, the cells are of constant size, so all these techniques may miss small features, do not adapt to the local geometry of the implicit surface, and require knowledge about the topology of the implicit surface. When the cell size is too small, an excessive number of polygons may be produced, and when it is too large, details may be obscured. To overcome this drawback, adaptive subdivision techniques that converge to the surface recursively have been developed [3], but with such techniques cracks may occur between triangles of adjacent cells of different size.

The second category are *surface fitting techniques* that create a seed mesh that roughly approximates the implicit surface and progressively adapt and deform it to better fit the implicit surface. For example, Velho [29, 30] starts with a coarse polygonal approximation of the surface and subdivides each polygon recursively according to the local curvature. But still there must be some a priori knowledge about the topology of the surface since the coarse polygonal approximation has to capture the correct topology of the implicit surface. Other surface fitting techniques either assume special classes of implicit surfaces created from skeletal elements [10, 7], or rely on a search for critical points [28, 6] suffering from inefficiency for complex implicit surfaces.

The third category are *surface tracking techniques* (or *continuation techniques*) that start from a seed element on the surface and iteratively grow a polygonal mesh that approximates the implicit surface. Cellular surface tracking techniques [2, 32, 4] start from a cell that intersects the implicit surface and iteratively find all intersecting cells among its neighbors. Since the cells are of constant size, cellular surface tracking techniques suffer from the same drawbacks as non-adaptive spatial sampling techniques, and furthermore, in the general case, it can be difficult to determine a seed cell.

## 2.2 Particle systems

The second way to sample implicit surfaces is to use so-called *particle systems* that evenly distribute samples over the implicit surface. The first time that particles were used to sample and control a surface was in a complete modeling tool using oriented particle systems by Szeliski and Tonnesen [24], but in their work there was no underlying implicit surface. Turk [26] used a similar process in order to generate textures using a reaction-diffusion method. Like the particle system, he used repulsion radii to simulate this reaction-diffusion, but the distribution of the particles is uniform and does not adapt to the local curvatures. Figueiredo et al. introduced a system to sample implicit surfaces using particles [9]. Even if the force applied to the particles is not the same as the one used by Turk, the authors used the relaxation process of Turk in order to achieve a uniform distribution of the points. Then these points are used to compute a polygonal approximation of the implicit surface. Turk's reaction-diffusion method can also be used to re-tile polygonal surfaces [27] according to the curvature with more points in regions of high curvature. But the method only takes into account isotropic curvature information and does not consider the principal directions of curvature of the surface.

Witkin and Heckbert [31] developed a powerful modeling application by putting together some of these existing ideas. Indeed, they demonstrated that particle systems are useful in order to both display and control implicit surfaces. They used two different types of particles: *floaters* that lie on the surface and that are used for rendering, and *control points* that are used to deform the surface. These two types of particles must resolve a set of constraints so that the floaters follow the implicit surface and that the surface follows the control points. Moreover, the authors defined an adaptive repulsion and a split/death condition so that particles could either split or disappear from the surface. Hart et al. [15] improved upon this particle system for automatic and numerical differentiation of the implicit surfaces. Indeed, in the work of Witkin and Heckbert, the derivatives for complex models can become computation-demanding and error-prone. Moreover, *shape adapters* were introduced that simplify surface deformations. One main limitation of both works is that no information about the curvature of the implicit surface is taken into account, thus only uniform distribution can be generated with spherical particles that all have the same repulsion radius.

Crossno and Angel [8] derived another extension based on the work of Witkin and Heckbert that can be used to sample an isosurface extracted from a 3D density image. A trilinear interpolation between the eight vertices of the voxel surrounding the particle location is

used to approximate the implicit function. They use the same repulsion forces and movement calculation as in [31], but they estimate the repulsion radius of each particle depending on the curvature at the sample. Consequently, particles in regions of higher curvature have a smaller repulsion radius. Nevertheless, Crossno and Angel only account for isotropic curvature information and thus do not consider principal directions of curvature.

Finally, Pauly et al. [22] used a particle system in order to simplify point-sampled surfaces. The same linear force as in [27] is used and the distribution of points depends on the curvature of a moving least squares (MLS) surface that approximates the points. Using a death condition combined with the repulsion forces enables them to simplify the number of points of the surface. Again, they only account for isotropic curvature information and do not consider principal directions and curvatures.

Some particle systems have also been used to polygonize implicit surfaces [9] via a Delaunay triangulation. Again, care must be taken that the particles are dense enough to create a topologically correct polygonal mesh.

## 3 Anisotropic sampling

Recall that our goal is to generate an anisotropic sampling technique for implicit surfaces that offers similar properties as the one generated by Kalaiah and Varshney for parametric surfaces. The density of the sampling should be related to the local curvatures as well as to account for the directions of maximal and minimal curvatures. In other words, each sample should be the center of an elliptical domain oriented along these directions and sized according to the maximal and minimal radius of curvature. To reach this goal, we propose to adapt the particle sampling technique developed by Witkin and Heckbert [31].

The basic idea of the sampling algorithm is outlined in Algorithm 1. In this section, we detail the choice we made for every step involved in this algorithm.

---

**Algorithm 1** The basic idea of our algorithm.

---

**Require:** An implicit surface

```
Create a set of particles lying on the surface
while Convergence is not reached do
  Compute the repulsion radii of the set of particles
  Compute the repulsion forces of the set of particles
  Update the position of the particles
  Split particles when necessary
end while
```

---

### 3.1 Initial set of particles

Actually, thanks to the particle splitting step, the algorithm converges even when starting with one sin-

gle initial particle, but it is more efficient to have an initial set of particles covering the surface. The easiest way to reach this is to use a scheme similar to the *shrink-wrap* technique [28]: first regularly sample either the bounding sphere or the bounding box of the implicit surface and then migrate the resulting particles by following the gradient of the implicit function until the surface is reached. Note that the diameter of this bounding volume is used as a normalization scaling factor during the entire process, so that every distance (radius, curvature, migration) can be computed in a scale-independent manner.

### 3.2 Repulsion radii

At each step of the particle migration loop, the repulsion radius for each particle has to be calculated. As stated above, we want an anisotropic repulsion process where particle are repelling more in directions of low curvature and less in directions of high curvature. So we actually compute two repulsion radii per particle (for the directions of maximal and minimal curvatures, respectively) by adapting the computation given in [17] to implicit surfaces. The mathematical background to compute the principal curvature directions of an implicit surface as well as the corresponding curvature amounts is given in the Appendix. Note that this calculation allows us to determine the variation of the implicit field of the implicit surface. The values that are found are not some distance measurements of the curvature that could be used directly in the application. Indeed, one has to multiply these values by a coefficient in order to scale them and to be able to use them as distance measurements that will define a repulsion domain around each particle. More specifically, *minCurv* and *maxCurv* will yield the curvature amounts in the two principal directions *minCurvDir* and *maxCurvDir*.

### 3.3 Repulsion forces

This step of the algorithm differs significantly from the rest of the literature. In [31, 8, 15], the authors compute the repulsion forces between the particles by using an energy measure. Even if it works well for isotropic repulsion, this process cannot be generalized for anisotropic repulsion. Another way to compute repulsion forces between particles has been proposed in [27, 22]. Again, the computation was proposed for isotropic repulsion between circular particles, but in contrary to the previous one, it is possible to extend this scheme for anisotropic repulsion between elliptical particles.

More precisely, an elliptical particle can be defined by combining the two repulsion radii *minRadius* and *maxRadius* and the two orthogonal directions of cur-

vature. By adding a radius in the third direction, we actually define ellipsoidal particles instead of elliptical ones. We have done this modification because in 3D space, it is much easier to compute repulsion forces between ellipsoids than between 2D ellipses defined on two different planar domains. Note that the radius given for this third direction (let us call it the *height* of the particle) does not really matter: we have tested either by using a small constant value for each particle, to get almost flat ellipsoids, or by using *minRadius* again, to get particles with a circular section, but the final sampling obtained after the particle migration process is very similar. The main reason is that the centroids of neighboring ellipsoids lie almost on the same plane during the last iteration steps, therefore the repulsion forces are more or less orthogonal to the height direction canceling the influence of the particle's height.

Once the ellipsoidal shapes of the particles have been set up, the repulsion forces can be computed according to the algorithm given below. It is important to note that we do not have to compute the repulsion force between any pair of particles. We rather use a space partitioning scheme that avoids computing the force between two particles that belong to distant areas. We use the same scheme as in [31], but any other hierarchical partitioning should work fine. Space partitioning reduces the overall complexity from  $O(n^2)$  to  $O(n \ln n)$  and thus significantly speeds up the computation involved in each step of the migration process.

#### 3.3.1 Computing repulsion forces by using spherical coordinates

We use Algorithm 2 in order to compute the repulsion force.

---

**Algorithm 2** Calculating the repulsion force between two particles.

---

**Require:** Two particles with their respective curvature information  
 Compute vector  $\mathbf{r}_{ij} = \mathbf{p}_j - \mathbf{p}_i$  between the centers of the particles  $i$  and  $j$   
 Compute the intersection  $\mathbf{m}_i$  of  $\mathbf{r}_{ij}$  and the ellipsoid of  $i$   
 Compute the intersection  $\mathbf{m}_j$  of  $\mathbf{r}_{ij}$  and the ellipsoid of  $j$   
 Determine whether the two ellipsoids intersect themselves  
 Compute their repulsion force.

---

This algorithm is really efficient as it only requires computing two line/ellipsoid intersections.

The intersection of the two ellipsoids can then be calculated. Starting from the two intersection points  $\mathbf{m}_i$  and  $\mathbf{m}_j$ :

$$res = \|\mathbf{m}_i\| + \|\mathbf{m}_j\| - \|\mathbf{r}_{ij}\| \quad (1)$$

A negative *res* means that the two particles  $i$  and  $j$  do not intersect and thus do not apply forces to each other.

Otherwise, the repulsion force of the particle  $j$  applied on the particle  $i$  is defined by:

$$F_{ij}(i) = res * (\mathbf{p}_i - \mathbf{p}_j) \quad (2)$$

We use the same linear repulsion force as in [27, 22] because of its compact radius of support. The total force exerted on  $i$  is then given by

$$F(i) = \sum_{j \in N_p} F_{ij}(i), \quad (3)$$

where  $N_p$  is the neighborhood of  $i$  that can be retrieved by the spatial partitioning.

### 3.4 Migration of the particles

After we have computed the repulsion forces for all the particles, the next step of the algorithm is to move the particle according to its repulsion force. Since the repulsion force  $F(i)$  of a particle is a vector, we just have to add this vector to the position of the particle in order to find its new position:  $\mathbf{p}_i += kF(i)$ . The constant  $k$  defines the rigidity of the particle's reaction with respect to the forces that are applied on it. In our implementation we use a constant value, but ideally  $k$  should be proportional to the average distance from a particle to its neighbors. Note that after this movement the particle does not exactly lie on the surface anymore, and we have to apply a step of the Newton-Raphson method to glue the particle on the surface:  $\mathbf{p}_i -= f(\mathbf{p}_i)\mathbf{n}_i$ , where  $f(\mathbf{p}_i)$  is the value of the implicit function  $f$  at the particle  $i$  and  $\mathbf{n}_i$  is the normal at the new position of particle  $i$ .

Once we have the final position of the particle, we have to compute its normal one last time for rendering.

### 3.5 Determining the fate of the particles

The final step of the algorithm is to determine whether the particle has to be split. We have seen that a good condition is to subdivide a particle when the sum of the norms of the forces that are applied on it becomes lower than a predefined threshold. Indeed, as the forces of repulsion are applied on a finite radius around the particles, the fact of having a particle with few forces applied means that it is in an under-sampled region. It is then natural to divide it in order to increase the local sampling density.

### 3.6 Rendering

After that the characteristics of the ellipsoids for all the particles  $i$  have been created, we use them in order to create the differential points that are rendered

as fragment-shaded rectangles. Since current graphics hardware does not support curved primitives as differential points, we make use of the possibility of directly programming new primitives in the GPU of these graphics cards.

More precisely, we use a rectangular primitive to represent a particle according to the local differential geometry. Similar to [17], the rectangle is defined in the tangent plane of the particle where the normal and the two perpendicular curvature directions define a local coordinate system.

The rectangle's extent is computed according to the maximum and minimum curvature amounts. Consequently, the higher the surface curvature, the smaller is the rectangle. In order to render the rectangle as a piece-wise smooth surface, an adequate normal distribution of the rectangle is required. In contrast to Kalahiah and Varshney, who select the best fitting normal out of 256 precomputed normals according to the principal curvatures, we interpolate the normals at the corners of the differential point's rectangle using *vertex shaders* and *fragment shaders*. Since we know the underlying implicit surface, we assign the normal to each of the four vertices of the rectangle. In other words, we define a normal field over the rectangle that locally approximates the appearance of the smooth implicit surface. The fragment shader interpolates the normals for each fragment and normalizes them by using a *cube map texture*.

The rectangles are fragment-shaded using the programmable graphics pipeline with vertex and fragment programs. In the vertex program, we do not compute the shading since we want per-pixel lighting. We write the normal, light and half vector in texture registers in order to interpolate them and define a normal distribution in screen-space. Then, in the fragment program, we normalize the interpolated normal, light and half vector and shade the fragment with both diffuse and specular components according to the Phong shading model. Finally, the set of these fragment-shaded rectangles gives a visual impression of a smooth surface.

## 4 Experimental results

All the images of this section were produced on a Pentium IV at 3.0 GHz with 1 GB of main memory and an NVidia GeForce Quadro FX. No code optimization effort has been done, the only acceleration technique is the spatial subdivision in order to determine the particles that are in the compact support radius of another particle. We implemented the differential point rendering using Cg's vertex and fragment shaders on a NV30 chipset graphics board from NVidia.

The only parameter that a user can modify is the scaling factor of the curvature amounts. Note that this factor must be chosen carefully. Indeed, the differential

point rendering allows having a smooth rendering of an implicit surface with a reduced number of points. Nevertheless, when the repulsion radii are too large, each differential point covers a too large amount of the surface resulting in a lower rendering quality. Figure 1 underlines this problem. Indeed, Figure 1(a) shows an ellipsoid rendered with a large repulsion radii. Artefacts appear at the silhouette of the surface as well as in the shaded regions compared to Figure 1(b), where the ellipsoids are rendered with a smaller repulsion radii and thus reducing the artefacts.

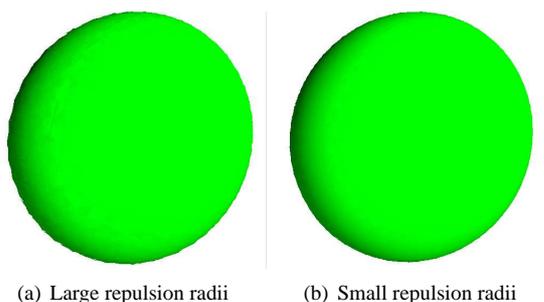


Figure 1: Differential point rendering

As explained in Section 3.6, differential point rendering renders the surface as a collection of overlapping fragment-shaded rectangles. In Figure 2, a random color has been used for each rectangle in order to outline the underlying structure. Figure 3(a) presents the rabbit rendered with a constant diffuse material but it should be noted that once the size and orientation of the rectangles have been defined, any fragment-shader can be used. As an example, Figure 3(b) shows a non-photorealistic rendering that is obtained by simply changing the shaders.

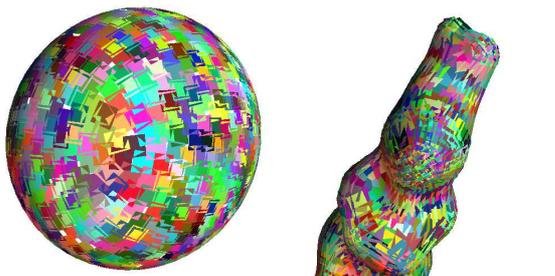


Figure 2: The differential points are rendered with random colors to outline the overlapping.

The principal problem of particles systems concerns the convergence detection: even if the surface seems to be well sampled, particles can still be created due to the splitting criterion of the particle system. Indeed, a slightly moving particle can lead to a change in the splitting criterion and thus to the generation of a new particle. This does not have a high impact on the rendering of the surface, but it shows that the convergence

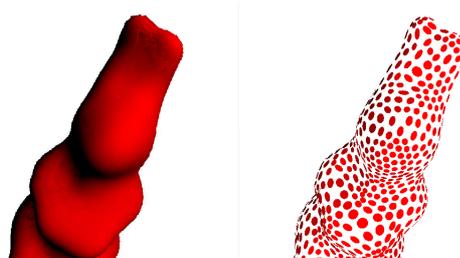


Figure 3: Different renderings of the rabbit

Figure 3: Different renderings of the rabbit

of particle systems is critical. A simple example of this problem is given in Figure 4.

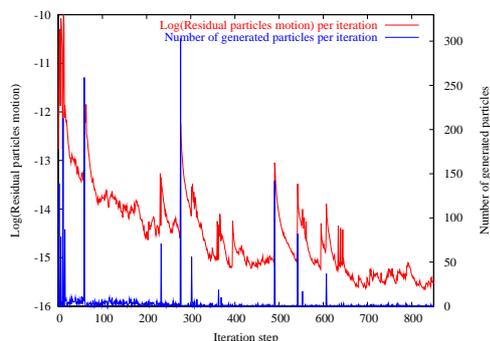


Figure 4: Residual particles motion and number of particles created per iteration.

Figure 4 underlines the fact that some particles are created even when the surface is well sampled. For example, Figure 4 shows that between iteration number 600 and iteration number 800, only 5 to 10 particles are created. So a quasi-equilibrium state of the system has been reached as soon as iteration number 200, but, because of the particle system behaviour, a small number of particles continues to split. Another evidence that lead us to believe that the equilibrium state has been reached is that, even if some particles have been created, the residual motion the particles decreases after a higher number of iterations. This means that the surface is well sampled and that the new particles only have a small influence on the motion of other particles. In order to resolve this problem, we are currently working on a more robust convergence criterion based on the residual motion of the particles instead of thresholding the forces.

## 5 Conclusion

In this paper, we presented an adaptation of the differential point rendering to implicit surfaces by anisotropically sampling the implicit surfaces using a particle system. Linking differential point rendering with particle systems provides an elegant way to sample and

render implicit surfaces. By pushing the information of the local differential geometry into each sample, we can describe the surface using fewer particles. This is particularly beneficial for remote rendering applications with limited bandwidth.

Another contribution of our work is the mathematical background that we provide to compute the principal curvature directions of an implicit surface as well as the corresponding curvature amounts (cf. Appendix). One drawback of the current implementation concerns the definition of a robust convergence criterion: in the case of spherical particle systems, an energy criterion can be used to identify the convergence. Unfortunately, energy criteria work well for isotropic particle systems (using spheres), but cannot be easily adapted for anisotropic systems (using ellipsoids).

We believe that the convergence time and the number of iterations of the relaxation process can be significantly reduced by first doing a global approximation step of the sampling and then running the particle system in a second step. More precisely, the global sampling step should determine the number of particles to sample the surface, and the second step determines the position of the particles on the surface.

Finally, we believe that particles systems are perfectly suited for interactive implicit surface modeling. As a consequence, the next step of our application will be to allow the user to deform the surface while the particles are following the surface deformation.

## References

- [1] M. Alexa, M. Gross, M. Pauly, H. Pfister, M. Zwicker, and M. Stamminger. Point based computer graphics. In *SIGGRAPH Course Notes*, 2004.
- [2] E. Allgower and S. Gnatzmann. Simplicial pivoting for mesh generation of implicitly defined surfaces. *CAGD*, 8(4):305–325, 1991.
- [3] J. Bloomenthal. Polygonization of implicit surfaces. *CAGD*, 5(4):341–355, 1988.
- [4] J. Bloomenthal. An implicit surface polygonizer. *Graphics Gems IV*, pages 324–349, 1994.
- [5] M. Botsch, M. Spornat, and L. Kobbelt. Phong splatting. In *Symposium on Point-based Graphics 2004*, pages 25–32, 2004.
- [6] A. Bottino, W. Nuij, and K. van Overveld. How to shrinkwrap through a critical point. In *Proc. of Implicit Surfaces '96*, pages 53–73, 1996.
- [7] B. Crespín, P. Guitton, and C. Schlick. Efficient and accurate tessellation of implicit sweep objects. In *Proc. of Constructive Solid Geometry '98*, 1998.
- [8] P. Crossno and E. Angel. Isosurface extraction using particle systems. In *IEEE Visualization '97*, pages 495–498, 1997.
- [9] L. de Figueiredo, J. Gomes, D. Terzopoulos, and L. Velho. Physically-based methods for polygonization of implicit surfaces. In *Proc. of Graphics Interface '92*, pages 250–257, 1992.
- [10] M. Desbrun, N. Tsingos, and M.P. Cani. Adaptive sampling of implicit surfaces for interactive modeling and animation. *Computer Graphics Forum*, 15(5), 1996.
- [11] M. DoCarmo. *Differential Geometry of curves and surfaces*. Prentice-Hall, 1976.
- [12] A. Van Gelder and J. Wilhelms. Topological considerations in isosurface generation. *ACM Transactions on Graphics*, 13(4):337–375, 1994.
- [13] J. Grossman and W. Dally. Point sample rendering. *Eurographics Rendering Workshop 1998*, pages 181–192, 1998.
- [14] M. Hall and J. Warren. Adaptive polygonalization of implicitly defined surfaces. *IEEE Computer Graphics & Applications*, 10(6):33–42, 1990.
- [15] J. Hart, E. Bachtá, W. Jarosz, and T. Fleury. Using particles to sample and control more complex implicit surfaces. In *Proc. of Shape Modeling International 2002*, pages 129–136, 2002.
- [16] A. Kalaiah and A. Varshney. Modeling and rendering of points with local geometry. *Trans. on Visualization and Computer Graphics*, 9(1):30–42, 2003.
- [17] Aravind Kalaiah and Amitabh Varshney. Differential point rendering. In *Proc. of Eurographics Workshop on Rendering 2001*, pages 139–150, 2001.
- [18] L. Kobbelt and M. Botsch. A survey of point-based techniques in computer graphics. *Computer & Graphics*, 2004.
- [19] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proc. of ACM SIGGRAPH 96*, pages 313–324, 1996.
- [20] M. Levoy and T. Whitted. The use of points as display primitive. Technical Report TR 85–022, University of North Carolina at Chapel Hill, 1985.
- [21] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics (ACM SIGGRAPH 87 Proc.)*, 21(4):163–169, 1987.
- [22] M. Pauly, M. Gross, and L. Kobbelt. Efficient simplification of point-sampled surfaces. In *IEEE Visualization 2002*, pages 163–170, 2002.
- [23] P. Shirley and A. Tuchman. A polygonal approximation to direct scalar volume rendering. *Computer Graphics*, 24(5):63–70, 1990.
- [24] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. *Computer Graphics (Proc. of ACM SIGGRAPH 92)*, 26(2):185–194, 1992.
- [25] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proc. of ICCV'95*, pages 902–907, 1995.
- [26] G. Turk. Generating textures for arbitrary surfaces using reaction-diffusion. *Computer Graphics (Proc. of ACM SIGGRAPH 91)*, 25(4):289–298, 1991.

- [27] Greg Turk. Re-tiling polygonal surfaces. *Computer Graphics*, 26(2):55–64, 1992.
- [28] K. van Overveld and B. Wyvill. Shrinkwrap: an adaptive algorithm for polygonizing an implicit surface. Technical Report 93/514/19, University of Calgary, 1993.
- [29] L. Velho. Adaptive polygonization made simple. In *Proc. of SIBGRAP '95*, pages 111–118, 1995.
- [30] L. Velho. Simple and efficient polygonization of implicit surfaces. *Journal of Graphics Tools*, 1(2):5–25, 1996.
- [31] Andrew P. Witkin and Paul S. Heckbert. Using particles to sample and control implicit surfaces. In *Proc. of ACM SIGGRAPH 94*, pages 269–278, 1994.
- [32] B. Wyvill, Craig McPheeters, and Geoff Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, 1986.

## Appendix: Principal curvature directions of an implicit surface

In the initial differential point rendering technique [17, 16], Kalaiah and Varshney proposed to extract the principal directions of curvature from parametric surfaces [11], triangular meshes [25], or NURBS surfaces that are fit to triangular meshes [19]. In this appendix, we show how to extract the principal directions of curvature for a point  $\mathbf{p} = [x, y, z]^T$  on the implicit surface  $\mathcal{S}$ , i.e.  $\mathbf{p} \in \mathcal{S}$ . To this end, consider the defining function  $f : \mathbb{R}^3 \rightarrow \mathbb{R}$  (that has second order partial derivatives) of the implicit surface  $\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 : f(\mathbf{x}) = 0\}$ . Recall, that the normal  $\mathbf{n}$  of a point  $\mathbf{p}$  is defined by the non-zero gradient of the defining function

$$\mathbf{n} = \nabla f(\mathbf{p}) = \left( \frac{\partial f}{\partial x}(\mathbf{p}), \frac{\partial f}{\partial y}(\mathbf{p}), \frac{\partial f}{\partial z}(\mathbf{p}) \right).$$

In order to derive second-order local geometry for the determination of the principal directions of curvature, we require the *Hessian matrix*  $\mathbf{H}$  of the second derivatives of the defining function  $f$ :

$$\mathbf{H} = \begin{pmatrix} \frac{\partial \mathbf{n}}{\partial x}(\mathbf{p}) \\ \frac{\partial \mathbf{n}}{\partial y}(\mathbf{p}) \\ \frac{\partial \mathbf{n}}{\partial z}(\mathbf{p}) \end{pmatrix} = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2}(\mathbf{p}) & \frac{\partial^2 f}{\partial x \partial y}(\mathbf{p}) & \frac{\partial^2 f}{\partial x \partial z}(\mathbf{p}) \\ \frac{\partial^2 f}{\partial x \partial y}(\mathbf{p}) & \frac{\partial^2 f}{\partial y^2}(\mathbf{p}) & \frac{\partial^2 f}{\partial y \partial z}(\mathbf{p}) \\ \frac{\partial^2 f}{\partial x \partial z}(\mathbf{p}) & \frac{\partial^2 f}{\partial y \partial z}(\mathbf{p}) & \frac{\partial^2 f}{\partial z^2}(\mathbf{p}) \end{pmatrix}$$

Note that  $\mathbf{H}$  is symmetric because of the equality of mixed partials. We use a local parameterization to extract the principal directions and curvatures on a point  $\mathbf{p} \in \mathcal{S}$  with an associated normal  $\mathbf{n}$  and a Hessian matrix  $\mathbf{H}$ . For the illustration of the following calculations, consider Figure 5.

Let us now approximate the defining function  $f$  of the implicit surface  $\mathcal{S}$  in a small vicinity of  $\mathbf{p}$  by using a small vector  $\mathbf{w}$  for a second degree Taylor expansion with an approximation error  $o(\|\mathbf{w}\|^2)$ :

$$f(\mathbf{p} + \mathbf{w}) = f(\mathbf{p}) + \mathbf{n}^T \bullet \mathbf{w} + \frac{1}{2} \mathbf{w} \bullet (\mathbf{H}\mathbf{w}) + o(\|\mathbf{w}\|^2)$$

Since  $\mathbf{p} \in \mathcal{S}$ , the defining function of the implicit surface  $\mathcal{S}$  in  $\mathbf{p}$  is  $f(\mathbf{p}) = 0$ , and we find

$$f(\mathbf{p} + \mathbf{w}) = \mathbf{n}^T \bullet \mathbf{w} + \frac{1}{2} \mathbf{w} \bullet (\mathbf{H}\mathbf{w}) + o(\|\mathbf{w}\|^2).$$

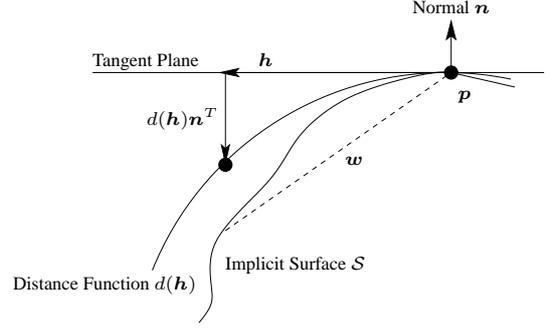


Figure 5: Curvature of an implicit surface  $\mathcal{S}$ .

Now, we split vector  $\mathbf{w}$  into a vector  $\mathbf{h}$  that is orthogonal to  $\mathbf{n}$ , i.e.  $\mathbf{n}^T \bullet \mathbf{h} = 0$ , and a distance function  $d$  to the tangent plane in  $\mathbf{p}$ :  $\mathbf{w} = \mathbf{h} + d(\mathbf{h})\mathbf{n}^T$ . We want to determine the distance function  $d$  so that  $f(\mathbf{p} + \mathbf{h} + d(\mathbf{h})\mathbf{n}^T) = 0$ . By combining the two previous equations and setting  $d(\mathbf{h}) = o(\|\mathbf{h}\|)$  since  $d'(\mathbf{0}) = 0$  by definition, we find

$$\mathbf{n}^T \bullet (\mathbf{h} + d(\mathbf{h})\mathbf{n}^T) + \frac{1}{2} (\mathbf{h} + d(\mathbf{h})\mathbf{n}^T) \bullet \mathbf{H}(\mathbf{h} + d(\mathbf{h})\mathbf{n}^T) = o(\|\mathbf{h}\|^2),$$

that we develop to

$$\mathbf{n}^T \bullet (\mathbf{h} + d(\mathbf{h})\mathbf{n}^T) + \frac{1}{2} \mathbf{h} \bullet (\mathbf{H}\mathbf{h}) + \frac{1}{2} \mathbf{h} \bullet \mathbf{H}d(\mathbf{h})\mathbf{n}^T + \frac{1}{2} d(\mathbf{h})\mathbf{n}^T \bullet (\mathbf{H}\mathbf{h}) + \frac{1}{2} d(\mathbf{h})\mathbf{n}^T \bullet (\mathbf{H}d(\mathbf{h})\mathbf{n}^T) = o(\|\mathbf{h}\|^2).$$

Since  $\mathbf{n}^T \bullet \mathbf{h} = 0$ , and since we can neglect the constant term with respect to  $\mathbf{h}$ , we find

$$d(\mathbf{h})\|\mathbf{n}\|^2 + \frac{1}{2} \mathbf{h} \bullet (\mathbf{H}\mathbf{h}) + d(\mathbf{h}) \mathbf{h} \bullet (\mathbf{H}\mathbf{n}^T) = o(\|\mathbf{h}\|^2),$$

and the second order approximation of  $d$  is

$$d(\mathbf{h}) = -\frac{\mathbf{h} \bullet (\mathbf{H}\mathbf{h})}{2\|\mathbf{n}\|^2}.$$

Now, we want to find the maximum and minimum of  $\mathbf{h} \bullet (\mathbf{H}\mathbf{h})$  for  $\mathbf{h}$  orthogonal to  $\mathbf{n}$ . This implies that the derivative of  $\mathbf{h} \bullet (\mathbf{H}\mathbf{h})$  has a component orthogonal to  $\mathbf{n}$  with the value 0, and hence  $\mathbf{H}\mathbf{h} = \mu\mathbf{h}^T + \lambda\mathbf{n}^T$ . To determine the principal directions and curvatures, we have to find the eigenvectors with associated non-zero eigenvalues of

$$\mathbf{H}\mathbf{h} - \frac{(\mathbf{H}\mathbf{h}) \bullet \mathbf{n}^T}{\|\mathbf{n}\|^2} \mathbf{n}^T = \mathbf{I} - \mathbf{n}^T \bullet \mathbf{n} \mathbf{H} \mathbf{I} - \frac{\mathbf{n}^T \bullet \mathbf{n}}{\|\mathbf{n}\|^2}.$$

Summing up, the principal curvature amounts  $u_{\mathbf{p}}$  and  $v_{\mathbf{p}}$  are the non-zero eigenvalues of this latter matrix, and the principal directions  $\mathbf{u}_{\mathbf{p}}$  and  $\mathbf{v}_{\mathbf{p}}$  are given by the corresponding eigenvectors.

# Simulation-Based Cartoon Hair Animation

|  |   |  |  |
|--|---|--|--|
| Eiji Sugisaki                                      | Yizhou Yu                                 | Ken Anjyo                                      | Shigeo Morishima                                   |
| Waseda University                                  | University of Illinois                    | OLM Digital Inc.                               | Waseda University                                  |
| 3-4-1 Okubo, Shinjuku-ku,<br>Tokyo Japan, 169-8555 | 201 N. Goodwin,<br>Urbana, IL U.S.A 61801 | 1-8-8 Wakabayashi,<br>Setagaya-ku, Tokyo Japan | 3-4-1 Okubo, Shinjuku-ku,<br>Tokyo Japan, 169-8555 |
| eijil1@toki.waseda.jp                              | yyz@cs.uiuc.edu                           | anjyo@olm.co.jp                                | shigeo@waseda.jp                                   |

## ABSTRACT

This paper describes a new hybrid technique for cartoon hair animation, one that allows the animators to create attractive and controllable hair animations without having to draw everything by hand except a sparse set of key frames. We demonstrate how to give a cel animation character accentuated hair motion. The novelty of this approach is that we neither simply interpolate the key frames nor generate the movement of the hair only using physical simulations. From a small number of rough sketches we prepare key frames that are used as indicators of hair motion. The hair movements are created based on a hair motion database built from physical simulations custom-designed by the animator. Hair animations with constraints from the key frames can be generated in two stages: a matching process to search for the desired motion sequences from the database and then smoothly connect them; the discrepancies between the database sequences and the key frames are interpolated throughout the animation using transition function.

## Keyword :

Cel Animation, Cartoon Animation, Hair Dynamics, 3D Animation

## 1 INTRODUCTION

Hair movement in cel character animation is sometimes inconsistent. For example, there may be inconsistencies in the number of strands or locks of hair, as can easily be seen in the images in Fig. 1. The representation of hair is thus a specialized work in cel animation. In fact, Cartoon hair representation is very difficult to achieve in computer graphics. This is because all of the hair attributes may not be consistent between camera positions. Although physics equations can be used to obtain physically correct movement, it is not always the movement for which the animator is looking. Since the movement of hair in cartoon animation sometimes carries meaning, the animator may be looking for something that exists only in his or her imagination. This means that the characteristics of the hair (modeled shape, number, etc.) between key frames may actually have not to agree. Even though the frames are physically inconsistent, the results can be quite convincing. This is the most difficult part of creating cartoons using computer graphics and is the reason

why cartoon hair animation has been done by hand. This requires a lot of human labor. In the full-cel animated movie "Princess Mononoke", [Dvd 01a] for example, it takes a month to complete five minutes animation in windy scenes.

The representation of hair motion is very important for computer graphics regardless of whether the graphic is simulation or cartoon animation. Moreover, while cartoon-like hair description plays a crucial role in making cartoon character animation impressive, hair simulation for cartoon animation is a challenging task because the human eyes discern the subtleties of hair motion and readily notices anything unnatural.

Although attractive cartoon hair animation is often seen on TV and in movies, there are few animators who can achieve impressive hair motion in cel animation. The hair motion as the camera angle changes is particularly hard to draw by hand. It requires the instincts of an expert animator and is very time consuming. While the work of a very skilled animator is very demanding, there has been little research directed at solving such time-consuming problems as the cel animation of hair motion. [Pno04a]



Figure 1. Example of cel images

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

*Conference proceedings ISBN 80-903100-7-9  
WSCG'2005, January 31-February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency - Science Press*

## 1.1 Overview

We develop a method for creating cartoon hair animation in computer graphics that easily retains the "anime-like" aspect of animation. Our goal is to produce an interesting cartoon hair animation that matches the hair designs shown in hand-drawn rough sketches. Our approach is a hybrid one taking advantages from both key-frame interpolation and physical simulation. The input is a sparse set of hand-drawn hair sketches for a cartoon character in the key frames. These sketches illustrate the target cartoon features.

A crucial step in our approach is building a motion database in advance. Building a database with an exhaustive list of motions is infeasible given the sheer number of hair strands and the number of vertices on each strand, so we build an "animator-directed motion database" with a chosen set of force impulses that can be used to generate sequences that potentially match the hair sketches. That is, we build a custom-designed database for each hair animation. It takes about 30 minutes to build such a database.

One way to reduce an animator's workload is to consider hair geometry in three dimensions. Although cel animation uses a two-dimensional structure, we use a three-dimensional structure so that hair motion can be created more easily. It is difficult, however, to use three-dimensional structures to fully express in two dimensions the inconsistencies that are peculiar to animation. We therefore propose a method for hair animation that matches data between the three-dimensional hair models and the key frames of rough sketches made by animators or directors that indicate hair motion. More specifically, we use the three-dimensional hair models and the impulse force to generate interactive and attractive motions, which we use to construct a hair motion sequence database. We then try to match the rough sketches to the three-dimensional hair motion sequence database data by projecting the sequence data onto the rough sketches. Once we find a match, we create a cartoon hair model to interpolate between the rough sketches.

## 2 Related Work

This overview of related work is limited to previous work on hair dynamics, focusing on explicit hair models. These models consider the shape and dynamics of each strand. While they are especially suitable for the dynamics of long hair, they do not consider cartoon simulations. Anjyo et al. [Ken92a] used a simplified cantilever beam to model hair and used one-dimensional projective differential equations of angular momentum to animate strands. Rosenblum et al. [Rer91a] and Daldegan et al. used sparse characteristic hair to reduce computation time. Kim and Neumann [Tae02a] presented an artful method for creating hairstyles that uses Multi-resolution Hair Modeling (MHM) system, which is based on

the observed tendency of adjacent hair strands to form clusters at multiple scales. Yu et al. [Yyu01a] also presented a method for creating hairstyles. These advances greatly improved hair expression in computer graphics.

Several researchers have proposed novel approaches to hair-hair interaction. Hadap and Magnenat-Thalmann [Had00a] proposed modeling dense dynamic hair as a continuum by using a fluid model for lateral hair movement. Hair-hair collision is approximated by the pressure term in fluid mechanics while friction is approximated by viscosity. Hair-air interaction is approximated by integrating hairs with an additional fluid system for the air. Chang et al. [Jtc02a] modeled a single strand as a multibody open chain expressed in generalized coordinates. Dynamic hair-to-hair collision is solved with the help of auxiliary triangle strips among nearby strands. The input to their simulation algorithm is an initial sparse hair model with a few hundred strands generated from their previous hair modeling method. Plante et al. [Epl01a] proposed a "wisps model" for simulating interactions in long hair. Bando et al. [Yba03a] proposed a method in which they model unordered particles that have only loose connections to nearby control points. By freeing particles from some constraints, they are able to animate hair including hair-hair interactions at a reasonable computational cost. Considering hair-hair interaction is also making significant contribution to hair expression in computer graphics.

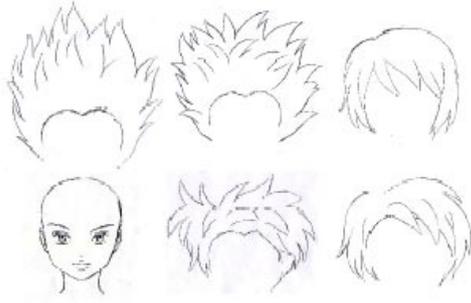
In terms of cartoon expression, Lasseter [Jon87a] is very likely the first paper to describe the basic principles of traditional two-dimensional hand-drawn animation and their application to three-dimensional computer animation. In this paper, he clearly describes cartoon animation and what it requires of an animator. Paul Noble and Wen Tang [Pno04a] achieved cartoon hair modeling and animation by using NURBS Surfaces to model the primary shape and motion of cartoon character hair.

Rademacher proposed the method that a three-dimensional structure is used in cel animation. [Pau99a] The reference hand-drawn image of the object or character often contains various view-dependent distortions that cannot be described with conventional 3D models. Therefore, to prepare view-dependent models, which consist of a base model, a set of key deformations created by the base model, a set of corresponding key viewpoints, and a given discretionary viewpoint, they interpolate the key deformations that are specific to the new viewpoint. They thus capture the view-dependent inconsistencies of the reference drawing.

## 3 Constructions of Hair Data

### from Original Input and 3D Geometry

Our method requires the preparation of various types of hair data before starting the simulation. We first need



**Figure 2** Rough sketches hand-drawn by an animator

to prepare a high-quality two-dimensional cel image of the animated character (like those shown in Fig. 1) and roughly sketched images of the character’s hairstyles (Fig. 2). These images are hand-drawn by a skilled animator and are used as original input. This is the only step in which a skilled animator draws images. We also need to prepare a three-dimensional model of the character’s head. This model is based on a wire frame model. We obtain the three-dimensional head model by texture mapping. This step is performed by users (animators).

### 3.1 Making Hair Strands

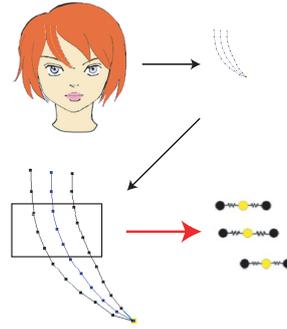
We also need to obtain position coordinates manually from high-quality two-dimensional cel images drawn by the animator. By obtaining the position coordinates and adjusting them to extract a hair model that is well adapted for the character animation, we create a three-dimensional sparse-hair model. This hair model has boundary lines to form the hair shapes and a centerline to control hair motion (see Fig. 3). These lines are expressed using Catmull-Rom splines. The centerline and boundary lines are connected by weak springs.

### 3.2 Constructing Initial 3D Hair Model and Converting The Rough sketch Images to data

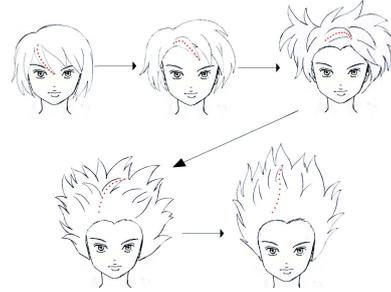
To construct an initial three-dimensional hair model that matches the character, we use a tool that allows the head model and hairs to be displayed simultaneously in three dimensions. A commercial tool can be used for this step. We also convert the rough-sketch hair images into quantitative data and match this data to hair data in the hair motion database (Fig. 4). Points on the sketched hair strands are plotted interactively. The plotted data is initially two-dimensional. Users perform all of these steps.

## 4 Creating Hair Motion

In this section, we describe how we create hair motion from a database and explain the construction of our “designed hair motion database”. We also explain the matching process, which uses the similarities between the angles derived from the rough sketches and the data obtained by projecting the three-dimensional hair data from the database sequences onto the image planes



**Figure 3** Example of making hair strand steps



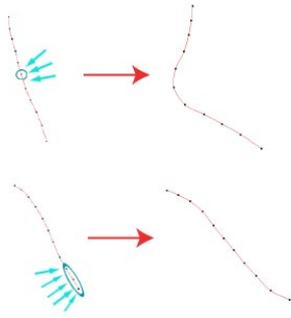
**Figure 4** Example of plotting the rough sketch images

defined for the rough sketches. We describe how we obtain the attractive hair shape by applying a deformation function to the differences between the angles. In addition, we describe how we interpolate between the database sequences to maintain smooth transitions between them.

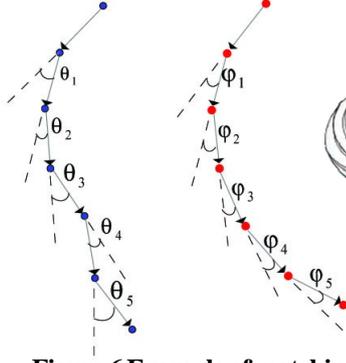
### 4.1 Designing Hair Motion Database

We apply forces such as those due to wind and head move to our hair model. After designing the forces, we simulate the hair dynamics using an implementation of Featherstone’s algorithm for *multi-body dynamic chains*. [Mul01a] [Pli94a] Only the centerline is controlled by this dynamics. Every sequence generates using this method specifies three-dimensional points with velocity vectors.

Compiling a database [Luc02a] [Jpl00a] [Dou03a] has become a common way of handling a corpus or scattered data. We also deal with the hair control point data as a database. A strong point of our method is that this database is custom-designed by an animator for a specific target animation. One unit of database sequence is about five frames long. All movements of hair strands in this database can be designed by an animator so as to reduce database size and enable target hair motions to be extracted. The animator can even define the specific forces needed to obtain a motion that does not conform to general physical laws. The size of the database depends on the trade-off between speed and quality. The database we use in our simulation is not so large, usually less than 5MB. (It depends on how many hairs a character has.) The force we apply is an impulse



**Figure 5 Example of designing hair motion**



**Figure 6 Example of matching aspect**



**Figure 7 Example of applying RBFs to matching aspect**

function. Since the animator needs to design hair that is close to the sketched hair in the key frames, we consider the rough sketches as the indicator of hair motion. For instance, to bend a hair strand dramatically, the animator should only apply a force to the middle part of the hair strand. Moreover, to make a motion in which only the hair tip moves, the animator should define forces applied only to the tip (see Fig. 5).

#### 4.2 Matching Process:

##### *Finding Similar Hair Strands in Database*

Our method requires animators to use their intuition to decide the camera positions for the rough hair sketches. Three-dimensional position points from the hair database sequence are then projected onto the image plane of the rough sketch. The data thereby become two-dimensional. Then we carry out matching in the 2D image plane with an x- and y-axes, comparing the angle between hair segments (Fig. 6). More specifically, we measure the difference in the angles between the data made from the rough hair sketches and the data of the hair sequences from the motion database projected onto the image plane. Then we compare this data using Eq. 1 from the hair root to the edge. We select the hair motion sequence that has the smallest error at the key frames.

$$d_{\min} = \frac{\sum_{i=1}^k \|\theta_i - \phi_i\|}{k} \quad \|\theta_i - \phi_i\| < th \quad (1)$$

where  $\theta$  is the angle made from the rough sketch data,  $\phi$  is the angle made from the data projected onto the rough sketch surface, and  $i$  is the angle number counting from the hair strand root. In this equation, the result of each subtraction must be lower than a certain threshold,  $th$ . If the subtraction result is higher than  $th$ , we do not choose the database sequence with the minimum error. We thereby obtain from the rough sketch the most similar hair form in the motion database.

#### 4.3 Deformation Function for Angles

Research on scattered interpolation has generally used radial basis functions [Jpl 00a](RBFs), which can be basically expressed as

$$\hat{d}(X) = \sum_{\psi}^N W_{\psi} \phi(\|X - X_{\psi}\|) \quad (2)$$

This interpolation is a linear combination of nonlinear functions of a distance from the data points. It uses the database sequence chosen in the matching process. To transform hair shapes to make them more similar to the hair sketches than to the database sequences obtained in the matching process, we use RBFs. Normally, impressive and exaggerated hairstyles cannot be obtained by using only physical equations. Initially, we used the RBFs to interpolate the discrepancies in the projected hair vertex positions on the image plane of the rough sketch. However, the hairs became awkwardly long (Fig. 7, center) and seemed unnatural. We therefore apply RBFs to the discrepancies in the angles in order to get the target shape. This is done using Eq. 3.

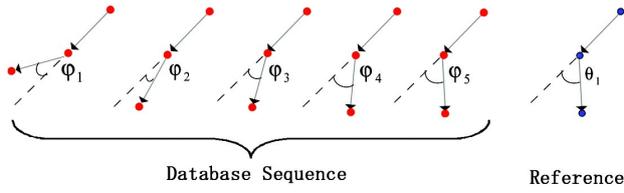
$$d(\phi) = \sum_{i=1}^N W_i \exp\left(-\frac{\|\phi_{ik} - \theta_i\|}{2r^2}\right) \quad (3)$$

where  $i$  is the number of links in a hair strand,  $N$  is the total number of links,  $k$  is the number of database sequences, and  $W_i$  is calculated by subtracting  $\theta$  from  $\phi$ . (The answer is an absolute value.) This calculation is carried out for the database sequence chosen in the previous step. We then repeat this step from the next hair root link to one point before the hair tip link, and adjust the database sequence. (Fig. 8)

To implement this interpolation, we can preserve the length of the hair segments. Once the angle differences at the first and last frames of a matching database sequence are computed, this interpolation is performed for all the frames within the sequence. It is repeated for all hair control points, thereby improving the strands, as shown in Fig. 7 (right). A second strong point of this process is that it preserves hair length and reduces the dimensionality of the calculations from two dimensions (for x and y) to one (the angle)

#### 4.4 Interpolation Between Hair Motions

We interpolate between two consecutive hair sequences in order to connect the hair motion smoothly. Since we use impulse forces to construct the hair motion



**Figure 8. An image of how to implement deformation to the database sequence**

database, the animation result may not be smooth without interpolation. We use Eq. (4) for the interpolation.

$$S = \frac{1}{2} \left( 1 + \frac{(1+c) \times (t \times 2 - 1)}{\|t \times 2 - 1\| + c} \right) \quad (4)$$

Where  $c$  is a variable parameter. If  $c$  is large, this interpolation becomes close to linear interpolation. If it is small, the interpolation becomes close to a step function.  $t$  represents the time interval; the smaller  $t$ , the smoother the interpolation. The advantage of this equation is that animators can control  $c$ . They can decide whether the interpolation should change dramatically or smoothly. Too much smoothness and meticulousness, however, is inconsistent with cartoon-like animation because it makes animation results more realistic. The animator thus must decide on the best parameters to use. Figure 9 shows the image of this interpolation.

## 5 Results

### Camera Position Interpolation

The camera position for each rough sketch is set to a key position, and we carry out linear interpolation between these positions. The number of segments depends on the number of frames obtained between the rough sketch images.

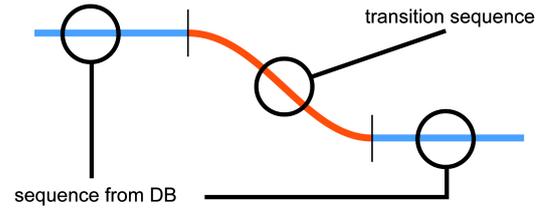
### Creating hair thickness

Our hair model does not consider thickness because we obtain this data from the two-dimensional images. To do the shading, we create the thickness of the hair. To create the thickness, we use the hair boundary lines. We obtain the center position from the boundary points and then calculate the vector from the center to right-boundary control point. To get the direction from the hair root to the end, we use an average vector made from both boundary points. We then calculate the outer product of these vectors. By controlling the calculated vector's magnitude, we can create and control hair thickness.

### Cartoon shading

We have implemented cartoon shading to obtain cartoon-like aspect character. [Adv02a]

Using our method, we have successfully animated our hair model in two animations using two kinds of inputs. For each animation, we have also used the 'on-tvos' animation method, which is commonly used for cartoons and is a means of getting more cartoon-like animation by



**Figure 9 An image of the transition between DB sequences**

using the same image for two frames. All animations were made at 24 frames per second, the standard rate for cartoon animation. Figure 10 shows the results using input that requires camera motion. It clearly shows the effectiveness of our method, since there are few animators who can achieve animation by hand. Figure 11 shows the results using input that does not require camera motion. This also shows that our method is clearly working for matching. The computations for these example results were executed on a computer with a 2.4 GHz Pentium 4 processor and were completed within five minutes for 120-frames animation.

## 6 Discussion

Using physical parameters obtained from a database of hair motions designed by a skilled animator, our method enables animators to use computer graphics to interactively design and generate cartoon hair animations with quality close to that obtained by hand. We easily produced target hair motions by using a simple hair model designed physically. Using our method we can achieve hair motions visualized from roughly sketched indicator images.

However, since the target cartoon expression is in the animator's mind, the animation results could be considered to be the right expression by one animator but not by another animator. An extreme way of saying this is that a cartoon character has a life in all scenes. What the character does must have the meaning the animator wants to express. One can argue that this is why various methods of cartoon expression have been developed.

Allowing this inconsistency is a huge advantage of cartoon expression and the most difficult thing to express in computer graphics. Even if the appearance of a cartoon characters' hair is physically impossible, the animation can still be fantastic and express the animator's intention. This is an advantage of drawing by hand. Since our model uses a three-dimensional hair structure to render hair, it cannot deal with such inconsistencies. The question of how to address these inconsistencies is future work. Automating the pre-computing steps, sections 3.1 and 3.2 is also left for future work. Another future project is consideration of how shadowing, rendering, etc. should be handled to enable discrepancy in cartoon expression.

## Acknowledgment

We would like to thank Yosuke Nakano, Kiyoshi Kojima, Shinji Sokawa and Akinobu Maejima for helping to make video. Additional thanks go to Jun Kurumisawa, Tatsuo Yotsukura, Mitsunori Takahasy and Shoichiro Iwasawa for their comments and suggestions. This research is supported by Japan Science and Technology Agency, CREST project.

## Reference

- [Ken92a] Ken. Anjyo, Y. Usami, and T.Kurihara. A simple method for extracting the natural beauty of hair. In *Proc. of SIGGRAPH 92*, pp. 111 - 120, 1992.
- [Dal93a] Daldegan, N.M.Thalmann, Kurihara, and D. Thalmann. An integrated system for modeling animation and hair renderin *Computer Graphics Forum (Eurographics 93)*, 12(3): pp. 211 - 221, 1993.
- [Had00a] Hadap and N. MagnenatThalmann. Interactive hair styler based on fluid flow. In *Computer Animation and Simulation 2000. Proceedings of the Eleventh Eurographics Workshop*, 2000.
- [Had 01a] Hadap and N. Magnenat-Thalmann. Modeling dynamic hair as continuum. In *Eurographics Proceedings. Computer Graphics Forum, Vol.20,No.3*, 2001.
- [Yyu01a] Y.Yu. Modeling realistic virtual hairstyles. In *Proceedings of Pacific Graphics*, pp. 295-304, 2001.
- [Joh02a] Johnny Chang, Jingyi Jin, and Yizhou Yu, A Practical Model for Hair Mutual Interactions, ACM SIGGRAPH Symposium on Computer Animation, San Antonio, July 2002, pp.73 - 80.
- [Rer91a] R.E. Rosenblum, W.E. Carlson, and E. Tripp. Simulating the structure and dynamics of human hair: Modeling, rendering and animation. *The Journal of Visualization and Computer Animation*, pp.141-148, 1991.
- [Rfe87a] R. Featherstone. *Robot Dynamics Algorithms*. Kluwer Academic Publishers, 1987.
- [Mul01a] Multibod Dynamics (Package software) <http://www.kuffner.org/james/software/index.html>
- [PLi94a] P. Litwinowicz, L. Williams. Animating images with drawings. SIGGRAPH 94, Orlando, FL, pp. 409-412,1994
- [Nbu76a] N. Burtnyk, M. Wein: Interactive Skeleton

Techniques for Enhancing Motion Dynamics in Key Frame Animation. SIGGRAPH 76, Orlando 564 - 569.

- [Pau99a] Paul Rademacher, "View-Dependent Geometry" In *Proceedings of SIGGRAPH 99*, L.A pp. 439-446
- [Luc02a] Lucas Kovar, Michael Gleicher, and Fred Pighin, Motion Graph. In *Proceedings of SIGGRAPH 2002 San Antonio*. pp 473-482.
- [Jpl00a] J. P. Lewis, Matt Cordner, Nickson Fong, "Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation" In *Proceedings of SIGGRAPH 2000 New Orleans*. pp 165 - 172.
- [Adv02a] Advanced RenderMan, A.A.Apodaca and L. Gritz. Morgan Kaufmann 2002
- [Dvd01a] DVD, Making process of "Princess of Mononoke" produced by Studio GHIBLI.
- [Pno04a] P. Noble and W. Tang: Modelling and Animating Cartoon Hair with NURBS Surfaces, *Proc. CG International 2004*, pp. 60 - 67.
- [Yba03a] Y. Bando, B.-Y. Chen and T. Nishita, Animating Hair with Loosely Connected Particles, *Computer Graphics Forum*, Vol. 22, No. 3, 2003.
- [Kwa03a] K. Ward and M. Lin, "Adaptive Grouping and Subdivision for Simulating Hair Dynamics", *Proceedings of Pacific Graphics*, 2003, pp.234-243.
- [Fbr03a] F. Bertails, T.-Y. Kim, M.-P. Cani, and U. Neumann, "Adaptive Wisp Tree - a multiresolution control structure for simulating dynamic clustering in hair motion", *ACM Symposium on Computer Animation*, 2003.
- [Tae02a] Tae-Yong Kim and Ulrich Neumann "Interactive Multiresolution Hair Modeling and Editing", In *Proceedings of SIGGRAPH 2002 San Antonio* pp 620 - 629.
- [Epl01a] E. Plante, M.-P. Cani, and P. Poulin. A layered wisps model for simulating interactions inside long hair. In *Proceedings of Eurographics Computer Animation and Simulation*, 2001.
- [Jtc02a] J. T. Chang, J. Jin, and Y. Yu."A practical model for hair mutual interactions" *Proceedings of ACM SIGGRAPH Symposium on Computer Animation 2002*, 73-80, 2002.
- [Jon87a] John Lasseter "Principle of Traditional Animation Applied to 3D Computer Animation" *Proc Sigggraph 1987*, pp 35-44.
- [Dou03a] Doug L. James and Kayvon Fatahalian, Precomputing Interactive Dynamic Deformable Scenes, In *Proceedings of ACM SIGGRAPH*, 2003.

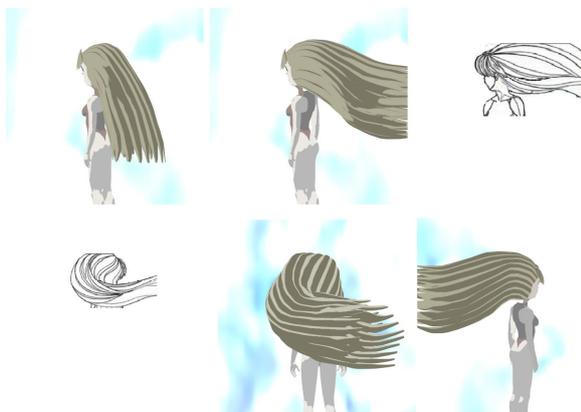


Figure 10 Animation sequence that requires camera motion

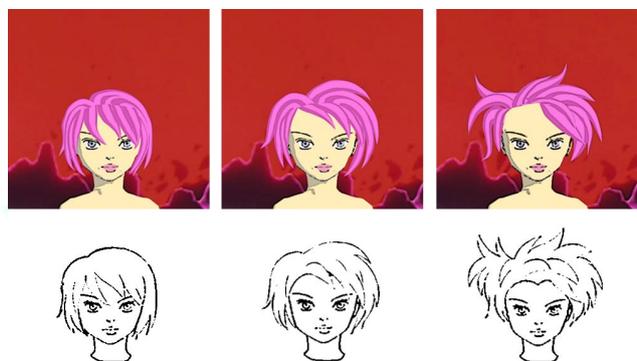


Figure 11 Matching sample of another character from animation sequence

# Motion Retargeting for the Hand Gesture

Chunbao Ge<sup>1</sup>  
Beijing 100020, China  
hagcb@163.com

Baocai Yin<sup>1</sup>  
Beijing 100020, China  
Ybc@bjut.edu.cn

Yiqiang Chen<sup>2</sup>  
Beijing 100080, China  
YiqiangChen@ict.ac.cn

Wen Gao<sup>2</sup>  
Beijing 100020, China  
WGao@ict.ac.cn

Changshui Yang<sup>2</sup>  
Beijing 100080, China  
Changshui@ict.ac.cn

## ABSTRACT

This paper presents a new technique for retargeting the sign language data captured from motion capture device to different characters with different sizes and proportions. Realistic and natural animations can be produced to express similar meanings to the original. The proposed method first defines many sensitive points on the human body and selects the key sensitive points through analyzing the importance of the sensitive points. Next a novel mapping method based on relative position is presented to adapt the original sensitive points to the target sensitive points. Finally we utilize an IK solver to realize the retargeting problem. Experimental results show that the proposed method dramatically improves the recognition rate about 30%.

## Keyword

Ehcharacter animation, motion retargeting, key sensitive points, IK, Chinese Sign Language

## 1. INTRODUCTION

Recently, motion capture has become one of the most promising technologies in character animation. Realistic motion data can be captured by recording the movement of a real actor with a motion capture system, and motion retargeting will adapt these motion data to new character. While the target character is different from the original one, the target character is likely to lose desire features of original motion. The problem can be solved through motion retargeting technology.

Many solutions to motion retargeting have been presented for different applications. Conventional retargeting techniques seldom consider the accurate meanings expressing in motion data. For example, in the sign language, very small changes in the hand postures will lead to wrong meanings. Our task is to retarget sign language motion data to new characters and guarantee the similarities of the meanings. The

most important characteristics for the sign language are the precise position relations between the hand and the other parts of the human body, so we must analyze and acquire these important characteristics.



Figure 1. Sign language of “human”, “eye”, “rectangle”

Figure 1 shows several key-frames of sign language (This model was downloaded from Miralab, and we only use it for demonstration as a standard virtual human model according to VRML). The left shows “human”. The middle shows “eye”. The right shows “rectangle”. For the left and the right the most important feature is the relative position between two hands. For the center the important feature is the relative position between the hand and the head, and the meaning will change if the forefinger points to other position.

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 2005 Conference proceedings ISBN 80-903100-7-9  
WSCG 2005, January 31-February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency – Science Press

1. Multimedia and Intelligent Software Technology Beijing Municipal Key Laboratory, Beijing University of Technology.
2. Institute of Computing Technology, CAS.

This paper presents a new technique for retargeting sign language data captured from motion capture device to different characters with different sizes and proportions. We can produce realistic and natural animations that express similar meanings to the original. We begin our discussion by providing a related work of our method in next section. Next we introduce the model of the human upper limb and reprocessing of the captured data. In section 3 we introduce feature analysis for the hand gesture. In section 4 we present a novel mapping method based on the relative position of the sensitive points. In section 5 we introduce our IK solver. Finally, we show several experimental results and conclude the paper.

## 2. RELATED WORKS

Several techniques have been proposed for reusing or altering existing motions. [Witkin95a] motion warping and [Bruderlin95a] motion displacement mapping discuss motion editing technique based on direct manipulation of data curves. [Bruderlin95a] and [Unuma95a] utilized signal-processing techniques for motion editing. [Wiley97a] proposed the interpolation synthesis algorithm that chooses and combines most relevant motions from the database to produce animation with a specific positional goal. Though some of the techniques above can be used for motion retargeting problem with user's extra efforts, they don't specifically address the motion-retargeting problem. [Boulic92a] presented the combined direct and inverse kinematics control technique for motion editing. The concept called coach-trainee metaphor is very similar to the motion retargeting problem formulation. A method, which is devoted to the motion-retargeting problem, was proposed by [Gleicher97a]. He used the space-time constraint method that minimizes an objective function  $g(x)$  subject to the constraint of the form  $f(x)=c$ . Since the whole interval has to be integrated to find the optimal solution, the method is intrinsically an off-line process. [Choi00a] adopted the idea of inverse rate control to compute the changes in joint angles corresponding to those in end-effectors positions while imitating the captured joint angles by exploiting the kinematics redundancy.

When the virtual character and performer have different sizes and proportions, not all aspects of the motions can be preserved during mapping. At the lowest level, it is simply not possible to mimic both the locations of the end-effectors and the joint angles. A system must make choices to which aspects of the motion should be preserved and which should be allowed to change. Gleicher's space-time motion editing [Gleicher98a] and retargeting system

[Gleicher97b] proposed the notion of preserving the important qualities of the motion by changing unimportant ones, where the important qualities were defined by constraints. Lee and Shin's hierarchical motion editing [Lee99a] provided similar results using a different underlying implementation. Popovic and Witkin demonstrated results that made the kinetic aspects of the original motion important to preserve [Popovic99a].

These methods mentioned above are all offline in that they examine the entire motion simultaneously in processing. Shin, etc [Shin01a] proposed an importance-based approach that retarget a performer to an animated character in real-time. They mapped as many of the important aspects of the motion to the target character as possible through importance analysis, while meeting the online, real-time demands. However, their approach addressed only the interaction between the end-effectors of a character and objects in the environment. In fact, there may also be interaction among the segments of a character, for example, sign language. In sign language we convey our meanings through the hand gesture, and small difference in the hand gesture may lead to wrong expressions. Due to the geometric difference between the character and the performer, the hand gesture of animation characters may deliver misleading meaning and even unrealistic motion through directly mapping. Existing algorithms don't deal with how to express precise meanings of the hand gesture. This paper presents a new retargeting method that can preserve accurate meanings of original motion for the upper limbs of the human body.

## 3. MODEL OF HUMAN UPPER LIMB AND REPROCESSING OF DATA

Our model has 18 joints and 32 degrees of freedom in each upper limb. The shoulder joint has three DOFs. The elbow and the wrist have two DOFs respectively. In order to perform complicated hand gestures there are three joints in each fingers and sum up to 25 DOFs in a hand (As shown in Figure 2).

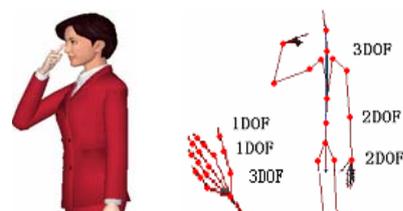


Figure 2. The controlled degrees of freedom for the dynamic model of the virtual human

We build our sign language motion library through data glove and location tracker device. Data glove is a kind of device for capturing the hand motion, and location tracker device can record the position of the shoulder, the elbow and the wrist. The captured data usually have many noises and cannot reflect the realistic motion in detail, so we must preprocess these captured data. A tool is developed to edit these original data through manual adapting, and finally we achieve a suit of standard sign language motion library for a fixed model proportional to the performer.

#### 4. ANALYZE FEATURES OF HAND GESTURE

The most important features for the sign language are the precise position relations between the hand and other parts of the human body. The small change occurring in the end-effectors may lead to mistake in the meaning. This section discusses how to get the most important features from the hand gesture. In figure 1, the middle shows “eye” when the forefinger points to the eye. And if the forefinger points to the nose, the gesture means “nose”. In this example the end of the forefinger is very close to the eye, so it is important to express the meaning of the gesture. The relative positions between the end of forefinger and the eye may be selected as the most important information. Similarly, we may analyze the important information involving the relation between two hands (see the left and the right pictures in figure 1).

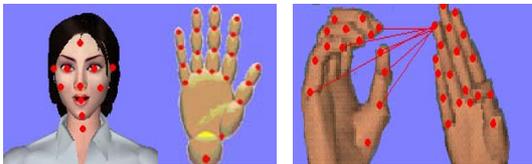


Figure 3. Feature points defined in the head and the hand

In order to find the feature information, we define three sets of special points called sensitive points for two hands and one head (see figure 3). Several notable points (for example eye, ear, nose, mouth, etc) in the head and two hands are selected as the characteristic points.

Usually, the importance is higher when the distance between two points is closer, and this feature must be preserved after retargeting. We call the first and the second closest points as key sensitive points and secondary key sensitive points defined as follows. Experiment shows that the main feature information can be got through selecting the key sensitive points and the secondary key sensitive points.

Here we only consider three parts including one head and two hands. We define three sets:  $H$  denotes sensitive points in the head;  $L$  and  $R$  denote the sensitive points in the left hand and the right hand respectively.

Let  $L = \{l_i, 0 \leq i \leq n\}$ ,  $R = \{r_i, 0 \leq i \leq n\}$ ,  $H = \{h_i, 0 \leq i \leq m\}$ . Here we define 22 points in each hand and 11 points in the head.

First we analyze the motion between two hands, and the motion between the head and the hand is similar. Given points  $r_i$  in  $R$  and  $l_j$  in  $L$ , let  $d_{ij}(t)$  be the Euclidean distance between them at  $t$  frame (We deal with each frame independently). Let  $D = \{d_{ij}(t), 0 \leq i, j \leq n\}$ . We sort the  $D$  and get the two minimal values  $d_1$  and  $d_2$  ( $d_1 \leq d_2$ ). Suitable  $r_i$  and  $l_j$  corresponding to the  $d_1$  and the  $d_2$  are selected as key sensitive points and secondary sensitive points. For example, the left gesture in figure 4, we select  $k_1$  and  $k_2$  as the key sensitive points, and select  $s_1$  and  $s_2$  as the secondary sensitive points. For the right gesture, there are only key sensitive points  $k_1$  and  $k_2$  (We consider the secondary sensitive points only if  $d_2 \leq 3$ ).

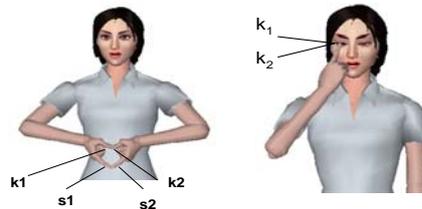


Figure 4. Analyze the key sensitive points and the secondary sensitive points: the left shows “heart” and the right means “eye”

#### 5. MAP SENSITIVE POINTS

From above analysis, we know that the most important thing is the relative position between the hand and other parts of the body for understanding the sign language. If we map the relative position to target object when retargeting, these important aspects for understanding the meaning will be preserved. We have got several couples of important sensitive points through above analysis. Next, we will discuss how to map the relative position for each couple sensitive points.

Usually the relative position can be denoted with a vector linking to two sensitive points, and we must retain identical vector corresponding to the key sensitive points after retargeting.

Two kinds of situations need to be considered when we compute the relative positions of two sensitive points. The first condition is that one point is fixed and the other point moves to the target position, for example, motion occurring in the hand and the head. The other condition is that two points both move to the target positions, for example, motion occurring in two hands.

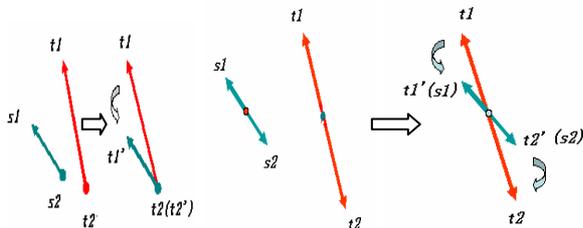
As shown in figure 5, here  $s_1$  and  $s_2$  are the sensitive points of the source object;  $t_1$  and  $t_2$  are the relative sensitive points of the target object without adapting. For first condition, we let  $s_2$  fixed points, and then we transfer  $s_1s_2$  to  $t_1t_2$  until  $s_2$  and  $t_2$  are coincident. For the second condition, we transfer  $s_1s_2$  to  $t_1t_2$  until their centers are coincident. New  $t_1'$  and  $t_2'$  are the target position.

We denote it with two equations as followings.

$$t_1' = (s_1 - s_2) + t_2, \quad t_2' = t_2$$

Or

$$t_1' = \frac{t_1 + t_2}{2} + \frac{s_1 - s_2}{2}, \quad t_2' = \frac{t_1 + t_2}{2} + \frac{s_2 - s_1}{2}$$



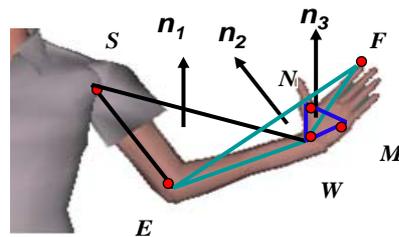
**Figure 5. The left shows that one is fixed and the other move to the new position. The right shows that two points both move to the new positions**

## 6. INVERSE KINEMATICS SOLVERS

Traditionally, inverse kinematics solvers can be divided into two categories: analytic and numerical solvers. Most industrial manipulators are designed to have analytic solutions for efficient and robust control. [Kahan83a] and [Paden86a] independently discussed methods to solve an inverse kinematics problem by reducing it into a series of simpler subproblems whose closed-form solutions are known. Korein and Badler [Korein82a] showed that

the inverse kinematics problem of a human arm and leg allows an analytic solution. Actual solutions are derived by Tolani and Badler [Tolani96a]. A numerical method relies on an iterative process to obtain a solution. Girard and Maciejewski [Girard85a] addressed the locomotion of a legged figure using Jacobian matrix and its pseudo inverse. Koga [Koga94a] made use of results from neurophysiology to achieve an “experimentally” good initial guess and then employed a numerical procedure for fine-tuning. Zhao and Badler [zhao94a] formulated the inverse kinematics problem of a human figure as a constrained non-linear optimization problem. Rose et al. [Rose96a] extended this formulation to handle variation constraints that hold over an interval of motion frames.

Here we adopt the analytical method based on the geometrical constraints. According to the Stokoe’s definition [Stokoe60a], each sign language can be broken into four parameters: hand shape, orientation, position and motion. These parameters as four important features play an important role in the sign language recognition. We build an objective function to satisfy these constraints (e.g. shape, orientation and position) for our special applications. Our method can find a suitable solution that maximizes the value of the objective function. Considering the IK chain displayed in the Figure 6 represents the human upper limb. This chain has three joints: the shoulder  $S$ , the elbow  $E$ , the wrist  $W$  and the end-effectors  $F$ . To guarantee orientation of the hand we select two points  $N$  and  $M$  in the hand together with the wrist joint  $W$  to define a plane in the hand, as shown in the Figure 6.



**Figure 6. Joints chain of the upper limb and normal vectors  $n_1, n_2, n_3$  defined by MNW, MEF, SEW**

We give several constraints for the IK chain: the position of the key sensitive points (hard constraint), the normal vector  $n_1$  of the plane  $MNW$  determining the hand orientation, the normal vector  $n_2$  of the

plane  $EFW$  and the normal vector  $n_3$  of the plane  $SEW$  determining the shape of the upper limb. In addition, each joint must meet the physiological constraints of the human body, so their activities should be restricted in a limited range. We give different weights for three orientation constraints according to their importance. The objective function is defined as in:

$$G = \alpha(n_1 \cdot n'_1) + \beta(n_2 \cdot n'_2) + \gamma(n_3 \cdot n'_3)$$

Here  $n_i$  and  $n'_i$  represent the original normal vectors and the target normal vectors respectively, and we specify  $\alpha=0.8$ ,  $\beta=0.15$ ,  $\gamma=0.05$ .

There are two elbow circles  $o_1$  and  $o_2$  defined in [Tolani00a]. One is made up of the elbow, the shoulder and the end-effector, and the other is made up of the elbow, the wrist and the end-effector, as shown in the figure 7. Let their swivel angles are  $\phi$  and  $\psi$ . When placing the end-effectors (F) at a desired point in space, there are an infinite number of solutions. When the swivel angles  $\psi$  of the elbow circles  $o_1$  is confirmed we can decide the position E, and hence we get another elbow circles  $o_2$ . Let its swivel angles is  $\phi$ , so we can get the position of W. Then we can get the length of EF and the DOF's values of the shoulder, the elbow and the wrist parameterized by  $\phi$ ,  $\psi$ . Furthermore we can get the reasonable range of  $\phi$ ,  $\psi$  and the length of EF according to the physiological constraints and the geometrical knowledge, and we can enhance the efficiency of our IK algorithm to a great extent. For example, let  $EF=d_2$ ,  $SF=d_1$ ,  $SE=l_1$ ,  $EW=l_2$ ,  $WF=h$ , we can deduce:  $d_1+l_1 \geq d_2 \geq d_1-l_1$ ,  $l_2+h \geq d_2 \geq h-l_2$ , and get the range of  $d_2$ :  $m \geq d_2 \geq n$ .  $m=\min\{d_1+l_1, l_2+h\}$ ,  $n=\max\{d_1-l_1, h-l_2\}$ .

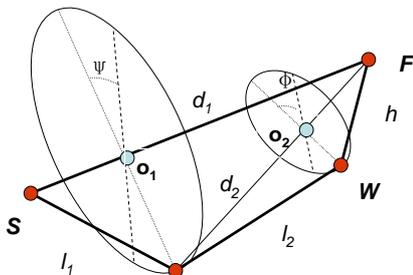


Figure 7. Two circles formed by the elbow and the wrist

Algorithm consists of two steps.

First we solve the DOF's values of the shoulder, the elbow, and the wrist according to the position of the key sensitive points. We preserve other DOF's values of joints in the hand if there is a lack of secondary sensitive points; otherwise we solve it again in a local joint chain in the hand formed by secondary sensitive points. It is described in the following steps.

- (1) Get the valid ranges of  $\phi$ ,  $\psi$ ,  $d_2$ .
- (2) For each  $d_2$  between  $m$  and  $n$ , we ascertain the elbow circle  $o_1$ .
- (3) For each  $\psi$  we compute the position of E and get the elbow circle  $o_2$ .
- (4) For each  $\phi$  we compute the position of F and get the value of  $G$ .
- (5) Select suitable E and F values corresponding to the maximal  $G$  and gets the values of DOFs.

As for secondary sensitive points, because we had ascertained the DOF's values of the shoulder, the elbow and the wrist, we can deal with it only in a finger chain.

## 7. EXPERIMENTAL RESULTS

Our method had been implemented on Chinese sign language synthesis system. It is a system using computer technology that translates text into animation of the virtual human in order to help hearing impaired people study sign language and communicate with the outsiders conveniently. For pursuing more harmonious interaction between human and the machine and applying it for more fields (for example, TV news broadcasting, internet, communication and film) so as to improve their life's quality we build many virtual human models for the users' choice. Our task is to produce animations for different virtual human models that express the same meanings and can be readily understood by members of the deaf population.



Figure 8. Virtual human models: Joe, Yuxin, Jali, Lisa, Susan, Lili

Our systems have six virtual human models that one is the standard model and others are different from

the standard in sizes and proportions (see figure 8). To evaluate the effectiveness of our work, two methods are adopted. First we conducted tests among deaf people for our retargeting results. In this experiment we selected 160 deaf people from four deaf schools and 100 typical examples of sign language including various kinds of contacting, crossing to test. Usually these sign languages will express wrong meanings if we don't retarget it. Test result is shown in the table 1.

| model | Lisa   | Jali   | YuXin  | Susan  | Lili   |
|-------|--------|--------|--------|--------|--------|
| R.R   | 97.54% | 98.33% | 98.43% | 97.35% | 96.26% |

**Table 1. R.R means recognition rate**

In addition, we invited some experts in sign language to examine all words for the five models. There are 3162 basic words in Chinese sign language. Test result is shown in the table 2.

| model | Lisa   | Jali   | Yu Xin | Susan  | Lili   |
|-------|--------|--------|--------|--------|--------|
| B.R.R | 47.28% | 65.03% | 61.52% | 52.43% | 63.25% |
| A.R.R | 96.54% | 94.33% | 96.43% | 95.35% | 95.26% |

**Table 2. B.R.R means recognition rate before retargeting, A.R.R means recognition rate after retargeting.**

It is very effective to preserve original meanings and can be readily understood by deaf people after retargeting from the test result. Experimental results show that the proposed method dramatically improves the recognition rate about 30%. Our methods can produce animation for the Sign language in real-time. We show some results in figure 9. These snapshots show some key-frames for several typical words in sign language, and animation for retargeting results can be got from our application for Chinese Sign Language Synthesis System.

## 8. CONCLUSIONS AND FUTURE WORK

We have presented a new approach for motion retargeting that transforms the upper limbs motions of a performer to the virtual characters with different sizes and proportions. First we define many sensitive points on a human body and select key the sensitive points and the secondary sensitive points through analyzing the importance of the sensitive points. Then we propose a novel mapping method based on relative position that adapts the original sensitive points to the target sensitive points. Finally we utilize an IK solver to realize the retargeting problem. Our methods had been

implemented on Chinese Sign Language Synthesis System.

Sign language, as a kind of most structured body language, is regarded as an indispensable means of everyday communication for deaf people. Research on sign language recognition will make for the communications between deaf people and common people. Conventional sign language recognition seldom utilizes the synthesis information of sign language. We can produce many suits of synthesis data for different models through our retargeting technique, and our future work is to implement the sign language recognition system based on synthesis information.

## 9. ACKNOWLEDGMENTS

This work has been supported by National Science Foundation of China (contract number 60303018), National Hi-Tech Development Program of China (contract number 2003AA114030), the Natural Science Foundation of Beijing of China (No.4011001), the Educational Committee of Beijing of China (No.01KJ-017, No. 2002KJ001)

## 10. REFERENCES

- [Boulic92a] R. Boulic and D. Thalmann. Combined direct and inverse kinematic control for articulated figure motion editing. *Computer Graphics Forum*, 11(4): 189–202, 1992.
- [Bruderlin95a] A. Bruderlin and L. Williams. Motion signal processing. In R. Cook, editor, *Computer Graphics (SIGGRAPH-APH '95 Proceedings)*, 97-104, August 1995. ACM-0-89791-701-4.
- [Choi00a] Kwang-Jin Choi and Hyeong-Seok Ko. On-line motion retargeting. *Journal of Visualization and Computer Animation*, 11:223-243, 2000.
- [Gleicher98a] M. Gleicher. Retargeting motion to new characters. In *SIGGRAPH 98 Conference Proceedings, Annual Conference Series*, pages 33–42. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.
- [Gleicher97a] Michael Gleicher. Motion editing with spacetime constraints. In *Proceedings of 1997 Symposium on Interactive 3D Graphics*, 139-148, 1997.
- [Girard85a] M. Girard and AA Maciejewski, "Computational modeling for the computer animation of legged figures," *Computer Graphics*, Vol. 19, No. 3, pp. 263-270, July 1985.
- [Kahan83a] W. Kahan. Lectures on computational aspects of geometry. Unpublished manuscripts, 1983.

[Koga94a] Y. Koga, K. Kondo, J. Kuffer, and J. Latombe. Planning motions with intentions. *Computer Graphics (Proceedings of SIGGRAPH 94)*, 28:395–408, July 1994.

[Korein82a] J. U. Korein and N. I. Badler. Techniques for generating the goal-directed motion of articulated structures. *IEEE CG&A*, pages 71–81, Nov. 1982.

[Lee99a] Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human likefigures. In *Proceedings of SIGGRAPH 99*, 39–48, 1999.

[Paden86a] B. Paden. *Kinematics and Control Robot Manipulators*. PhD thesis, University of California, Berkeley, 1986.

[Popovic99a] Zoran Popovic and Andrew Witkin. Physically based motion transformation. In *Proceedings of SIGGRAPH 99*, 11–20, 1999.

[Rose96a] C. Rose, B. Guenter, B. Bodenheimer, and M. F. Cohen. Efficient generation of motion transitions using spacetime constraints. *Computer Graphics (Proceedings of SIGGRAPH 96)*, 30:147–154, August 1996.

[Shin01a] Shin H. J., Lee, J., Gleicher, M., and Shin, S. Y. *Computer Puppetry: An Importance-Based Approach*. *ACM Transactions on Graphics*, Vol. 20, No. 2, April 2001, Pages 67-94.

[Stokoe60a] W. C. Stokoe, *Sign Language Structure: An Outline of the Visual Communication System of the American Deaf*. *Studies in Linguistics: Occasional Papers 8 (Revised 1978)*. Buffalo, NY: Linstok, 1960.

[Tolani96a] D. Tolani and N. I. Badler. Real-time inverse kinematics of the human arm. *Presence*, 5(4): 393–401, 1996.

[Tolani00a] D. Tolani, A. Goswami, and N. Balder. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical Models* 62(5), Sept. 2000, 335-388

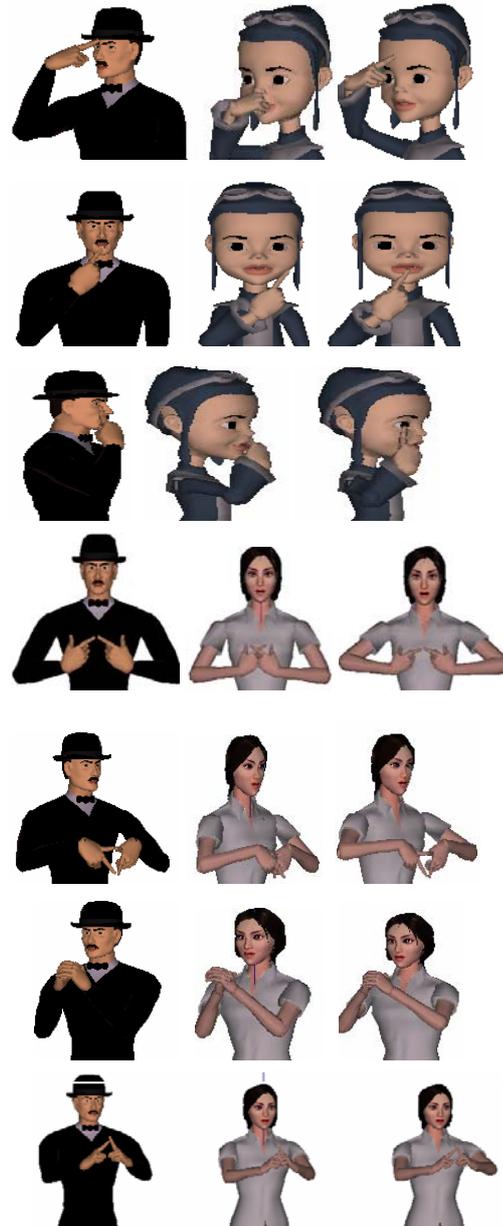
[Unuma95a] K. A. Munetoshi Unuma and R. Takeuchi. Fourier principles for emotion-based human figure animation. In R. Cook, editor, *Computer Graphics (SIGGRAPH '95 Proceedings)*, 91–96, August 1995. ACM-0-89791-701-4.

[Witkin95a] A. Witkin and Z. Popovic. Motion warping. In R. Cook, editor, *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 105–108, August 1995. ACM-089791-701-4.

[Wiley97a] D. J. Wiley and J. K. Hahn. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Applications*, 39–45, November/December 1997.

[Zhao94a] J. Zhao and N. I. Badler. Inverse kinematics positioning using nonlinear

programming for highly articulated figures. *ACM Transactions on Graphics*, 13(4): 3–13–336, 1994.



**Figure 9.** Some snapshots of typical sign language: The left shows standard sign language. The middle shows results without retargeting and the right shows results after retargeting. (They represent the meanings of “head”, “tongue”, “eye”, “fight”, “hesitate”, “bless” and “human” respectively.)



# The SenStylus: A Novel Rumble-Feedback Pen Device for CAD Application in Virtual Reality

Michele Fiorentino  
DIMeG

Politecnico di Bari  
Viale Japigia 182  
70100, Bari, Italy

m.fiorentino@poliba.it

Antonio E. Uva  
DIMeG

Politecnico di Bari  
Viale Japigia 182  
70100, Bari, Italy

a.uva@poliba.it

Giuseppe Monno  
DIMeG

Politecnico di Bari  
Viale Japigia 182  
70100, Bari, Italy

gmonno@poliba.it

## ABSTRACT

We have developed a pen device for CAD applications in virtual reality which provides novel features compared to existing systems. The SenStylus consists of a wireless pen designed to be ergonomically handled by the user for spatial interaction using a six degree of freedom optical tracking. In addition to the classic digital button(s) input, it provides analog multi-axial control, and a dual-rumble feedback output. We have integrated the device into an existing virtual reality CAD environment and extended the application functionalities with new device-specific features. The SenStylus vibration feedback improves perception in the virtual world by controlling frequency, amplitude, and duration of the feedback, simulating a variety of responses during collisions and selection tasks. This capability enforces the visual depth sensitivity, which is critical when working with complex CAD models. The multi-axial analog input provides a natural interaction paradigm to the user, thus simulating pen pressure and angle as in real world sketching and in real clay modeling. Dynamic tool-tip dimensioning and shaping are implemented as extra features. We present some applications to prove the added value of the SenStylus. The evaluation of the device received positive feedback by designers and engineers alike. The new features offered by this device can easily be extended to other VR applications using the API provided.

## Keywords

Virtual reality, user interface hardware, CAD, 3D interaction

## 1. INTRODUCTION

As established by many studies, one of the most limiting factors in desktop CAD is the use of two degrees of freedom devices for creating 3D forms. However, the use of fully 3D environments is limited by the barely explored virtual reality (VR) interface.

For an effective use of CAD in VR, several VR interaction techniques have been explored. In particular, among the many implementations, the so-called *pen&tablet* metaphor has proved to be effective in many applications [Zol97a].

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 80-903100-7-9  
WSCG'2005, January 31-February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency – Science Press

In the *pen&tablet* interface, the user holds in his/her non-dominant hand a transparent palette, on which menus and buttons are displayed; the other hand controls a stylus for application-related precision tasks.



Figure 1. The SenStylus.

In spite of the widespread use of the *pen&tablet* interface both in academia and in the industrial world, none of the stylus hardware implementations

fully fulfills our needs for our present and on-development applications. In addition to the classic stylus, providing six degrees of freedom (6DOF) and a few state buttons, we needed a multi-axial analog control for our VR CAD system, and a dual channel haptic feedback output.

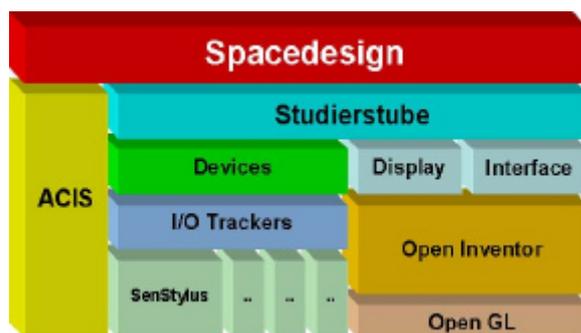
Therefore, we decided to design and prototype a novel stylus device called *SenStylus* (Figure 1).

Our *SenStylus* consists of a wireless pen which is designed to be ergonomically handled by the user for spatial interaction using a 6DOF tracking.

The first goal was to offer different controls in a stylus-shaped design, including buttons, analog joysticks and sliders. The second goal was to provide haptic feedback to add further insight to virtual environments (VE).

## Related Works

Spacedesign is an innovative test bed application developed by the authors to address CAD issues using virtual and augmented reality interface. Based on the Studierstube library [Sch96a] and the ACIS [Spa] modeling kernel, Spacedesign uses the *pen&tablet* metaphor (Figure 2).



**Figure 2. The Spacedesign VR CAD application architecture.**

The stylus is the main input device for interaction: it is used for 3D sketching, surfacing, navigation, selection, and manipulation. The user is also provided with a virtual palette (a tracked Plexiglas sheet device) that is used to display information as well as virtual menus and buttons.

In order to improve the CAD functions using the stylus, we started a preliminary research on new available devices suitable for our stylus needs.

Some non-standard interaction devices are commercially available for entertainment and research purposes. Besides the very well known Fastrack Stylus, Fakespace [Fak] produces the NeoWand™, with eleven buttons and ergonomic design, and the Cubic Mouse™ [Fro00a]. The Cubic Mouse - a tracked cube with state buttons and rods

protruding from the sides - is designed for 3D volume visualization and clipping planes control in space.

*Wanda* by Ascension [Asc] is a palm-sized navigation and interaction 6DOF-tracked tool, with 1 joystick and 3 buttons.

However, the 3D interaction efficiency in VE depends not only on the type of device, but also on external factors such as the type of feedback (visual, audio, or tactile), the number of degrees of freedom, and ergonomic and subjective aspects. In [Kon95a], [Lin99a], and [Lin02a], for example, it is demonstrated that the user's performance in VE can be improved with multi-channel feedback (e.g.: tactile and visual). Force-reflecting devices using exoskeletons or pantographs, such as PHANToM [Sen], provide very effective feedback, but their use is limited by their cost and cumber.

As alternative to this solution, the use of vibrating motors, like the ones in game pads or cell phones, is gaining popularity to provide inexpensive *vibrotactile* feedback [Che96a], [Oka98a], and [Cam99a]. Hughes and Forrest [Hug96a] coupled a standard desktop mouse with vibration elements and tested its application.

Logitech's [Log] proprietary *iFeel* technology, implemented in the *iFeel* mice, uses Immersion's Inertial Harmonic Drive engine and TouchSense API to produce vibrations in one axis. The API controls the vibration wave function through frequency, amplitude, and duration, while the hardware takes care of the rest, combining multiple effects to simulate a variety of responses, including quick pops and different textures.

In this preliminary research on devices suitable for our stylus needs, we found many products offering partial solutions, but none of them completely fulfilled our interface needs.

In the following sections, we describe the *SenStylus* design and some of its applications.

## 2. THE SENSTYLUS

### Interface Design and Requirements

To exploit the VR CAD potential, users must have the possibility to improve their communication to/from the virtual environment. Our underlying idea was to maximize the number and quality of the input/output channels.

We decided that the *SenStylus* design had to fulfill the following requirements:

- ergonomic and lightweight (it should be held in one hand and for continuative use);

- wireless;
- vibrating feedback;
- availability of buttons, sliders, and joysticks for digital and analog input.

### The SenStylus Prototype

We performed a market research on already existing products, in order to find some wireless, analog input and rumble-feedback pen-shaped device. Several products with characteristics similar to those required, especially oriented to the game industry, are available (i.e. gamepads and joysticks). Unfortunately, their shape is not optimized for VR applications, in which both hands are used independently. As far as the authors know, on the market there is no device meeting all the requirements mentioned above.

The commercial product which comes closest to our needs is the Logitech® *Wingman Cordless Rumblepad*™.

This device supplies 2 independent vibration channels, 11 push-buttons, 5 analog controls, and 1 wireless communication on a 2GHz bus with a range of 6 meters. We decided to modify this device to obtain a first prototype of the SenStylus. In the following sections, we describe the aforesaid hardware transformation.

### Ergonomic Issues

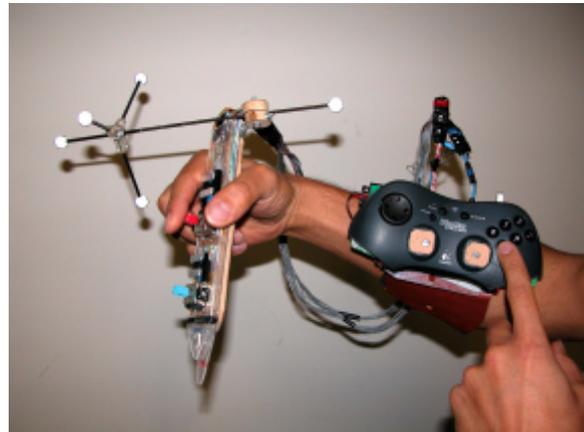
The SenStylus is designed for extensive use. Bad design and uncomfortable grasp can reduce the precision of the interaction, thus causing frustration in the user. The lack of limb support (as provided by the table during drawing) makes the 3D spatial interaction a big issue. In fact, the VR stylus - differently from other devices like the desktop mouse - requires both power and precision grip. For this reason, weight is critical. Researches on surgical instruments show that the tool weight is a trade-off between filtering hand vibrations (heavier tools) and reducing fatigue (lighter tools). In this first prototype, we decided to split the device in two parts: the first is the stylus itself and the second is the console attached to the user's forearm.

The pen is connected to the console by a soft wire plait.

This solution removes the heavier components (wireless transmitter, battery pack, etc.) from the hand-held device. Moreover, the front buttons on the console are of easy access for the other hand.

In the SenStylus design the thumb and middle finger provide the power grip, while the index reaches the controls located on the top (Figure 3). The stylus

cross-section is semi-elliptical for a comfortable grip and to permit adjustability to hand size. Both left- and right-handed people can use the device proficiently. For additional comfort, controls are rounded and covered with padded foam.



**Figure 3. The SenStylus ergonomics.**

In order to satisfy the ergonomic requirements mentioned above, some modifications have been made on the device based on the Wingman (Figure 4).

- The shape of the circuit board has been slightly modified for compactness reasons. The wireless transmission module has been removed and reconnected in a different location.
- Four buttons and the two joysticks have been moved from the console to the stylus in ergonomic position.
- The power supply has been modified. The original battery pack (four AA batteries) has been replaced with a single rechargeable 9V battery with tension regulator.
- The two vibrating motors have been moved from the console corners to the pen extremities.

### Wireless Connection

VR devices (stylus, tablet, glasses, etc.) must be tracked in order to provide the 6DOF position and orientation necessary for the 3D input. Most of the previous generation tracking systems, such as the magnetic- and acoustic-based ones, needed a wire connection between the tracked device and the central unit. However, a completely wireless device can disclose unknown freedom to the user, who can be free to move in the VR environment without wire jams. The latest optical tracking systems provide a much higher precision in conjunction with no physical connection. Regarding input device controls, such as buttons, joysticks, etc., a wireless communication is needed not to nullify the advantage of an untethered tracking system.

## Analog Multi-Dimensional Input

Apart from the 6DOF input, VR stylus devices are usually provided with simple controls: buttons, sliders, and joysticks. One single state button (on/off) is able to command simple operations (i.e. selection and navigation), but in order to be effective, more advanced CAD functions - such as shape modeling and editing - need other and more complex controls in the input interface.

Most of the devices available on the market are provided with discrete state buttons. Many applications though - such as those aimed at expressing ideas in conceptual design - go further than the on/off logic. A clear example of this is free sketching on paper, where the pressure and the inclination of the pen on the paper are used to give expression and “style” to otherwise simple and dull lines. Commonly available desktop interfaces (mouse and keyboard) do not usually gratify the designer’s artistic freedom. One of the main goals we want to achieve with the development of the SenStylus is to introduce analog (thus emotional) input into a VR-based conceptual design application.

We adapted two modified 2-axis joysticks of the Logitech Wingman to provide a continuous bi-dimensional analog input. The two joysticks, that we called *indexstick* because controlled by the index finger, are located on the top side of the stylus (Figure 4), just under the user’s natural index position. Some applications, especially developed to make use of the *indexstick*, are described in the application section.

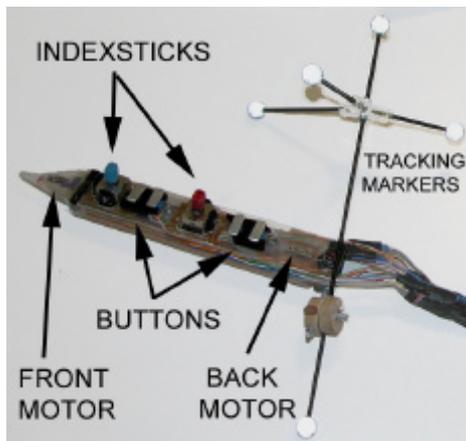


Figure 4. The SenStylus prototype.

## Rumble Feedback

Virtual reality technology, by means of stereo vision and tracked point of view, provides an enhanced visualization and an improved understanding of the digital model, especially for complex models.

Although visual stimuli play the major contribution to the human perception model, experiments have shown that users are not capable of judging the added depth dimension as naturally as they do the other two. Novice and experienced VR users alike find difficulties in localizing 3D targets along the depth direction. This can be explained by the following points:

- **occlusion issues:** hand and pen can cover the image on the screen;
- **2D interface influence:** the user thinks and acts on 2D as in desktop interface;
- **attention allocation:** the user concentrates his/her attention just on the plane of the screen.

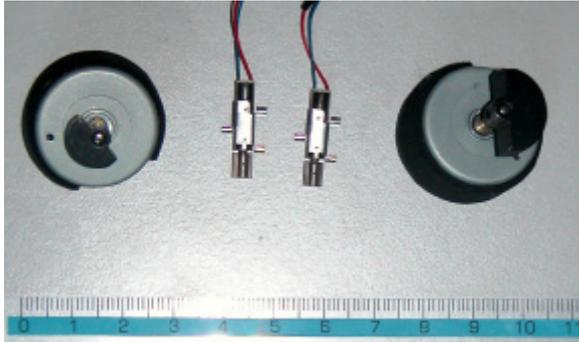
Force/Rumble feedback can be coupled with the visual stimuli to provide a better perception of the world. Many on-going studies are developing technologies for conveying force feedback in a VR environment. Unfortunately no one provided a definitive solution to issues such as complex set-ups, costs, and low user’s acceptance. The vibrating feedback technology instead is nowadays rather widespread. Rumble feedback is common in game controllers, but in some applications, such as the vibrating desktop mice, it is often reported to be uncomfortable and useless.

In VR, a controlled vibration can provide rendering of different material textures or different effects (i.e. collision, snapping, etc.).

The main idea is to use two vibration sources controlled by the application and located at the extremities of the pen (Figure 4). The final aim is to test the feasibility of rumble feedback for VR CAD applications.

Our early experiments with a first SenStylus prototype, using the rumble vibrator motors extracted from the Logitech Wingman, demonstrated that the vibrations commonly used in game controllers are excessive for immersive VR use, and not well accepted during modeling, because annoying and tiring. These considerations made us modify the Wingman motors, using smaller ones commercially available in the mobile phone market.

The effect, even if lighter than the previous one, is nonetheless experienced by the user. Moreover the motor substitution (Figure 5) reduced the encumbrance, the weight, and also the power absorption, allowing us to simplify the SenStylus design.

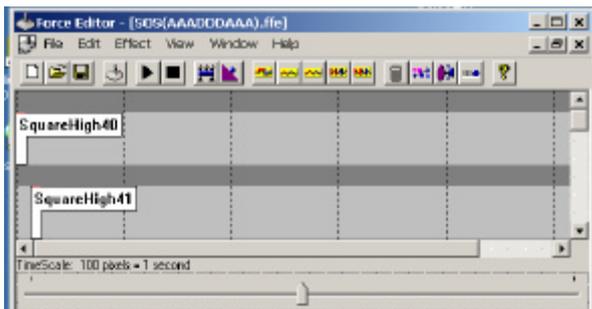


**Figure 5. The Wingman vs SenStylus rumble motors.**

### 2.1.1 Rumble effects

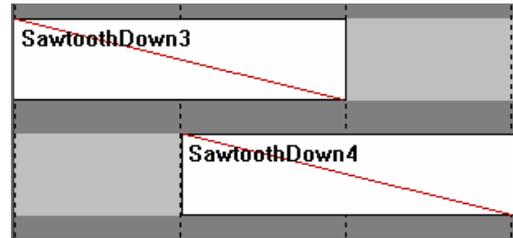
Since in our implementation we used two separate rumble vibrators, we can command two independent channels of feedback output. This solution allows to provide a wide spectrum of feedback effects without the need for more expensive haptic displays.

To determine the most effective waveform combination for the use in virtual environments, we tested several waveform effects, using a dedicated editor (Figure 6).



**Figure 6. Effects editor.**

Using the combination of simple waveforms, it was possible to create a wide list of different effects, clearly discerned by the users (Figure 7). In this way test subjects could accurately detect CAD events like collisions and snapping. Besides, by using the double channel it was possible to convey to the user direction information, such as that indicating collisions in and out of the virtual objects. Subjects could also easily associate a continuous change in the perceived vibration amplitude to a scalar value (i.e. temperature, pressure, or velocity) in a volumetric field data set.



**Figure 7. Example of a rumble effects wave shape in the two channels.**

Some examples described in the application section demonstrate the effectiveness of vibrotactile stimulation in conveying a wide range of information in a VR CAD system. We are continuously working to develop and test several dual channel effects for a more useful feedback.

### Software

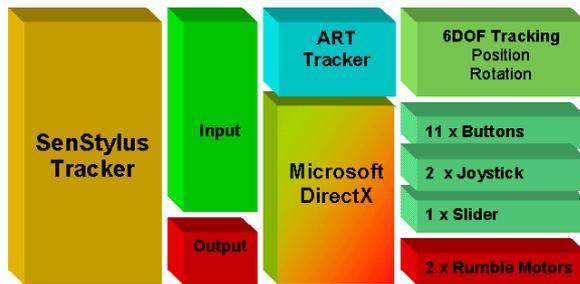
Integrating the SenStylus device into the Studierstube framework, which originally has no support for analog controls and force/rumble feedback output, required some effort to preserve the existing architecture and to maintain the compatibility with the previous applications.

In order to implement these new features, the Studierstube API Tracking class was extended with analog input and rumble feedback virtual methods. Then, a new tracker class called *SenStylusTracker* was implemented.

The SenStylusTracker architecture is displayed in Figure 8: the 6DOF input (translation and rotation) of the real stylus device is acquired from the 3D input tracker (i.e. ART Dtrack), and merged with the button and control states coming from the SenStylus driver, using a Studierstube *Buttonfilter*. The SenStylus I/O resources are accessed via a DirectX-based driver.

In the input device market, mostly pushed by the gaming industry, Microsoft *DirectX* has become a standard, thus making the development of new input devices very easy. *DirectX* enables the application to retrieve from each device the features provided and to control them accordingly.

Through action mapping, the applications can retrieve input data without the need to know what device is generating it.



**Figure 8. SenStylus Tracker.**

The SenStylus tracker maps the button input into an extended Studierstube tracker architecture.

SenStylus is provided with:

- **11 x state buttons,**
- **2 x analog controls,**
- **1 x slider.**

Rumble feedback features are activated according to different modalities. These modalities can be activated by selecting one of the following functions:

- **start constant vibration,**
- **load vibration effects from file (.ffe),**
- **start custom effect,**
- **stop effect.**

The rumble control is achieved in asynchronous mode, thus in a way transparent to the application.

For each custom effect, the user can instantly control the tension applied to the motor in order to vary:

- **phase,**
- **wavelength,**
- **effect shape and envelope,**
- **offset,**
- **max amplitude,**
- **time delay\duration,**
- **axes (front or back motor).**

Effects can be designed and tested using Microsoft Force Editor (provided with DirectX SDK) saving the files as “.ffe”, and then retrieving them from the application.

In the following section we present some applications where the added value of the SenStylus is evident.

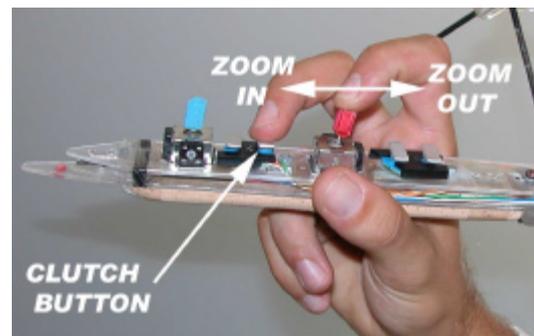
### 3. APPLICATIONS

The SenStylus has been integrated in Spacedesign via new software modules. These modules, as described in the next section, have been specifically designed

to exploit and test the innovative features provided by the new device.

### Enhanced Scene Navigation

In this module we have modified the VE navigation metaphor we had been using for years in SpaceDesign. The previous system was “clutch”-based, which means that the user, by pressing a status button on the stylus, attaches the virtual scene to the tracked device until he\she releases the button. The new navigation idea was borrowed from the AutoCAD “wheeled zoom”, which is very effective in 2D modeling. We added a fly-through function to the “clutch” navigation. In the fly-through function the speed is controlled by one of the indexsticks, and the direction is controlled by the orientation of the SenStylus. The swapping between the “clutch” and the fly-through modes is very fast (Figure 9). Our preliminary tests proved this solution to be very effective and more efficient for all users. We are currently working on extensive test cases to give a quantitative measure of the increased performances.



**Figure 9. Enhanced Scene Navigation using the SenStylus.**

### Object Snap

The *3D Object Snap* is the natural extension to the 3D input of the Object Snap tools already available on most Desktop CAD systems. The object snapping can be easily extended to a 3D input in a virtual environment, where they are very useful because of the tracking error, the user’s fatigue, the hand vibration, and the lack of limb support. Compared to 2D, the 3D object snapping uses a sensible volume instead of a flat region and the marker is displayed as a “wire framed” 3D geometry depending on the snapped topology (Endpoint, Midpoint, Perpendicular, Centre, etc.). With SenStylus we have added haptic effects while snapping was activated (Figure 10). The waveform of the associated effects varies according to the snapped topology. The use of a two-channel feedback can provide information about the pen movement direction towards the snap point. Another well known issue is the snap volume

dimensioning: a small volume increases precision but requires more interaction time for selection, especially in complex scenes. We have used one of the indexsticks to dynamically change the sensible volume dimension interactively, according to the complexity of the model.

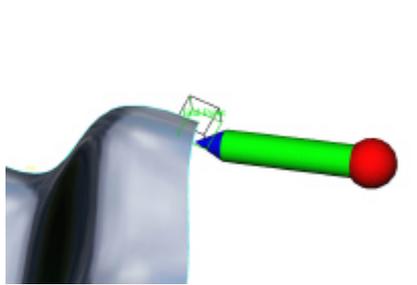


Figure 10. Endpoint snapping example.

### Object Collision Feedback

We have used SenStylus dual-channel rumble capabilities to improve the user's perception of the VR model. In our tests, object selection in a VR CAD has proved to be a critical task for complex models, especially in the depth direction. Therefore, we implemented a haptic proximity sensor with two different effects (in/out). A calibrated phase shift between the two channels was used to convey the information about the approach and withdrawal direction (Figure 11).

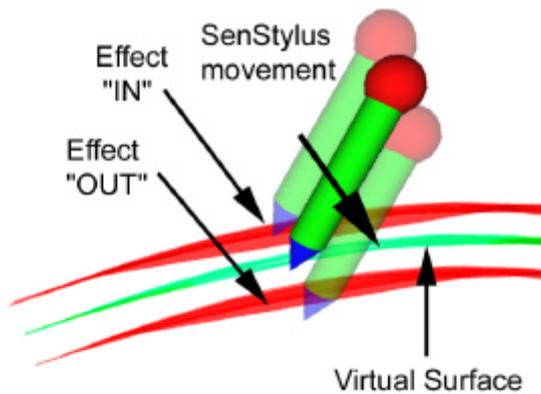


Figure 11. Dual effect collision.

### Multiple DOF “Solid Line” Sketching

In this module we tested the SenStylus analog control capabilities for 3D free sketching. With this Spacedesign function the user can draw tubular-shaped lines with multiple degrees of freedom (8DOF). Basically, the user sweeps a profile along the rail drawn in the space by the stylus movement (3DOF). The cross-section is an ellipsoid whose dimensions are constantly controlled by the user with one analog indexstick. During the sweeping, the

SenStylus orientation rotates the cross-section plane, in a way similar to the real-world marker pen. Tests showed positive results especially for 3D writing and logo sketching (Figure 12).



Figure 12. “Solid Line” sketching.

### Haptic Probing in Volumetric Fields

Virtual reality, through effective visualization and exploration, makes it possible to gain a quick and intuitive understanding of very complex datasets. To extend the data perception in VE we augmented the graphical clues with the two haptic channels available. Moving the SenStylus probe inside the volume dataset, the intensity of the vibration is associated to a selected variable value. Using two channels we convey the instantaneous variable value to the two extremities of the stylus where the vibrators are located. In this way the SenStylus is able to simulate directionality in the feedback without using more expensive force-reflecting devices (Figure 13).

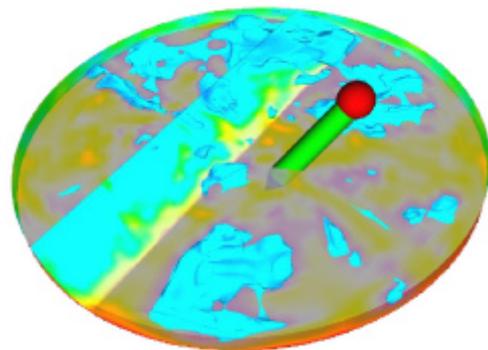


Figure 13. Dual-channel rumble rendering of a CFD dataset.

## 4. CONCLUSIONS AND FUTURE WORK

This work presents a novel VR device, the “SenStylus”, which is expressly designed for CAD in VR. At the moment no commercially available pen-

like interface satisfies in one product all the requirements for CAD interaction in virtual environment: ergonomically shaped, wireless, light weight, rumble feedback, analog input. The SenStylus prototype presented here is built using computer shop hardware. Two rumble feedback sources located at the pen's extremities can be activated separately in order to simulate a variety of responses. Analog input is provided by two dual axis joysticks, with which the user can control multiple degrees of freedom operations.

We have developed and implemented some applications especially conceived for testing the SenStylus potentials.

We plan to further test these potentials within a whole VR CAD design session in order to evaluate its global performances. Our intention is to make the SenStylus available to other VR research centers so as to have it tested in different frameworks. Beside the presented prototype, we are currently working on the redesign of the SenStylus in order to integrate all the components in one ergonomic device.

## 5. ACKNOWLEDGMENTS

We would like to thank Prof. Francesco Corsi, Dr. Angelo Dragone, Massimiliano Dellisanti, and Marco Landriscina for the essential help in building up the prototypes. The authors wish to thank Dr. Oliver Kreylos for the tube drawing implementation and Prof. Dieter Schmalstieg for the Studierstube library.

## 6. REFERENCES

- [Art] ART, "Advanced Realtime Tracking GmbH, ARTTrack1 & DTrack IR Optical Tracking System", [www.ar-tracking.de](http://www.ar-tracking.de).
- [Asc] [www.ascention.com](http://www.ascention.com).
- [Cam99a] Campbell C, Zhai S, May K, Maglio P., "What You Feel Must Be What You See: Adding Tactile Feedback to the Trackpoint", in: *Proc. of INTERACT'99: 7th IFIP Conference on Human Computer Interaction*, 1999; 383-390.
- [Che96a] Cheng L-T, Kazman R, Robinson J., "Vibrotactile Feedback in Delicate Virtual Reality Operations", in: *Proc. of the Fourth ACM Int'l. Conf. on Multimedia*, 1996; 243-251.
- [Fak] [www.fakespace.com](http://www.fakespace.com).
- [Fio02a] Fiorentino M., De Amicis R., Stork A., Monno G. "Spacedesign: Conceptual Styling and Design Review in Augmented Reality", in *Proc. of ISMAR 2002 IEEE and ACM International Symposium on Mixed and Augmented Reality*, Darmstadt, Germany, 2002, pp. 86-94.
- [Fro00a] B. Fröhlich and J. Plate, "The Cubic Mouse: A New Device for 3D Input," *Proc. ACM CHI 2000*, ACM Press, New York, Apr. 2000, pp. 526-531.
- [Hug96a] Hughes R, Forrest A., "Perceptualisation Using a Tactile Mouse." In: *Proc. Visualization '96* 1996; 181-186.
- [Kaw95a] Kawai, S. et al. "Effects of Varied Surface Conditions on Regulation of Grip Force During Holding Tasks Using a Precision Grip", *Japanese Journal of Physical Fitness and Sports Medicine* 44(5). 519-538. 1995.
- [Kon95a] Kontarinis D, Howe R., "Tactile Display of Vibratory Information in Teleoperation and Virtual Environments". *Presence: Teleoperators and Virtual Environments* 1995; 4(4); 387-402.
- [Lin99a] Lindeman R, Sibert J, Hahn J., "Towards Usable VR: An Empirical Study of User Interfaces for Immersive Virtual Environments". In: *Proc. of ACM CHI '99* 1999; 64-71.
- [Lin02a] Lindeman, R.W., Templeman, J.N., Sibert, J.L., Cutler, J.R., "Handling of Virtual Contact in Immersive Virtual Environments: Beyond Visuals", *Virtual Reality*, 6(3), 2002, pp. 130-139.
- [Log] [www.logitech.com](http://www.logitech.com).
- [Oka98a] Okamura A, Dennerlein J, Howe R., "Vibration Feedback Models for Virtual Environments". In: *Proc. of the IEEE Int'l. Conf. on Robotics and Autom.*, 1998; 674-679.
- [Ryu91a] Ryu, J. et al. "Wrist Joint Motion", in *Biomechanics of the Wrist Joint*, 27-60. 1991.
- [Sen] [www.sensable.com](http://www.sensable.com).
- [Spa] [www.spatial.com](http://www.spatial.com).
- [Sch96a] Schmalstieg D., Fuhrmann A, Szalavari Z., Gervautz M., "Studierstube - An Environment for Collaboration in Augmented Reality", in *Proc. of CVE 96 Workshop*, Nottingham, GB, 1996, pp. 19-20.
- [Zol97a] Zolt., Gervautz M. "The Personal Interaction Panel - A two Handed Interface for Augmented Reality", *Computer Graphics Forum* 16(3) C335-C346, 1997.

# Implementing Multi-Viewer Stereo Displays

Bernd Fröhlich,  
Jan Hochstrate,  
Jörg Hoffmann,  
Karsten Klüger

Bauhaus University Weimar

Bauhausstraße 11  
99423 Weimar  
Germany

Bernd.Fröhlich@medien.uni-weimar.de

Roland Blach  
Matthias Bues

CC Virtual Environments

Fraunhofer IAO

Nobelstr 12  
70569 Stuttgart

Germany

Roland.Blach@iao.fhg.de

Oliver Stefani

Center of Applied Technologies in  
Neuroscience

Wilhelm Klein-Strasse 27

4025 Basel,

Switzerland

ols@coat-basel.com

## ABSTRACT

In this paper we describe our implementations of multi-user stereo systems based on shuttered LCD-projectors and polarization. The combination of these separation techniques allows the presentation of more than one stereoscopic view on a single projection screen. We built two shutter configurations and designed a combined LC-shutter/polarization setup. Our first test setup was a combination of mechanical shutters for the projectors with liquid crystal (LC) shutters for the users' eyes. The second configuration used LC-shutters only. Based on these configurations we have successfully implemented shuttering of four projectors to support two users with individual perspective correct stereoscopic views. To improve brightness conditions and to increase the number of simultaneous users, we have designed a combined LC-shutter/polarization filter based projection system, which shows the most promising properties for real world applications.

## Keywords

Virtual Reality, Immersive Projection Systems, Stereo Displays, Multi User Systems, Multi Viewer Systems

## 1. INTRODUCTION

Perspective projection in combination with head tracking is widely used in immersive virtual environments to support users with correct spatial perception of the virtual world. However, most projection based stereoscopic systems show a correct perspective view for a single tracked viewer only. Other users share the same view, but from different positions, which results in an incorrect perception of the displayed objects. This limits the suitability of

projection-based stereoscopic systems for multi-viewer scenarios, particularly in cases where concurrent 3D-interaction of all users is desired.

Our intent is the development of a multi viewer projection system for local collaboration in immersive environments. We focus on projection based systems where all users operate in the same interaction space. A realistic application scenario for a team of collaborators in front of a single projection screen would incorporate not more than ten users due to space limitations in front of the screen. In most cases we expect only two to six users being involved in such scenarios.

In this paper we describe our implementations of multi-user stereo systems based on shuttered LCD-projectors and polarization. We discuss the results of our work and give a comparison of the three configurations. Additionally, our ideas for further improvement will be presented.

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 2005 conference proceedings ISBN 80-903100-7-9*

*WSCG'2005, January 31-February 4, 2005*

*Plzen, Czech Republic.*

Copyright UNION Agency – Science Press

## View Separation Techniques

The separation of different views is mainly used to separate the left eye view from the right eye view in stereo projection systems. There have been also examples for the separations of different user perspectives [Agr97, Blom02].

Following the classification of Paastor [Paa97] for separation techniques we will describe the common approaches in the field of immersive projection environments.

### 1.1.1 Time-Sequential or Shutter Techniques

There are two main approaches for shuttering projectors: mechanical shutters and liquid crystal (LC) shutters. Mechanical shutters are in the simplest form based on a spinning disc, which is half transparent and half opaque. [Fak04, Ham24, Lip01, Pal01] suggest this approach, which also seems to be used in a commercial product [Fak04]. Liquid crystal shutters are widely used for shutter glasses and they were also used for shuttering projectors [Kun01, Kun02]. They can be opened and closed electronically.

### 1.1.2 Color-Multiplexing

Anaglyphs, a common technique for stereo viewing, use different colors to provide different views. The perceived image appears monochrome. From the ergonomical point of view anaglyphs are more tiring and they are more straining for the eyes than other techniques. A new approach has been developed which is based on wavelength multiplexing which is a kind of multi channel color multiplexing for red, green and blue, the so called Infitec system [Jor04]. With this approach there are up to now some inherent problems with color matching delivered in the different views.

Color multiplexing uses appropriate filters in front of the projector and the eyes for the separation.

### 1.1.3 Polarization-Multiplexing

Polarized light has a defined oriented field vector in the plane perpendicular to the direction. Linearly polarized light has fixed direction. Circularly polarized light has a fixed rotation direction of the field vector. With appropriate filters, polarized light can be generated from unpolarized or undirected light. With polarization it is only possible to separate two views due to the nature of polarization where the filtering is based on the splitting of the light waves into two orthogonal parts. A linear polarization filter which is orthogonal to the light polarization direction theoretically blocks the light completely. Polarization filters are used in front of the projectors and the eyes to apply the separation. This is the standard technique for stereo projection in non interactive mass presentations.

### 1.1.4 Performance parameters

To evaluate the quality of a multi view projection system, three main parameters can be considered:

- Brightness per view
- Crosstalk; static and dynamic
- Perceived flicker, which depends on the shutter frequency, the video rate of the projector and brightness.

One of the main challenges is the delivery of sufficient light to the eye. The light which is emitted by the projection system is distributed over the amount of views and is therefore dependent on the overall view switching frequency, the initial brightness and the attenuation of the optical filter.

Another issue is crosstalk between different views which is generally disturbing and also strains the eyes. Crosstalk occurs when image parts belonging to other views are perceived, which should be ideally completely blocked. Crosstalk can be subdivided into static and dynamic crosstalk. Static crosstalk is based on the imperfection of the used materials. In the case of shutters the contrast ratio, that is the ratio between transmission in the open state to transmission in the closed state, is also an indicator for expected static crosstalk. Dynamic crosstalk is due to the timing behavior of the opening and closing of the shutter elements and only arises in the transition phases. Dynamic crosstalk can be reduced to nearly not existent with an adequate control system. In a system with low switching frequency, there will be always a trade off between dynamic crosstalk and brightness.

Our approach for the configuration of a scalable multi view system focuses on a hybrid configuration which combines shutter and polarization filter techniques.

## 2. RELATED WORK

Shuttering devices for time-sequential stereoscopic displays have a long history. Lipton provides an overview in [Lip91]. Interesting in this context is Lipton's reference to Hammond's work on the Teleview system from 1924 and 1928 [Ham24, Ham28]. Hammond used a spinning disc and two projectors to generate a field-sequential active stereo image. He also used a synchronized spinning disc in front of the user's eyes to provide each eye with the corresponding image. Palovuori's patent application from 2001 [Pal01a] presents basically the same approach based on the spinning disc and shows nearly identical images. In addition, Palovuori suggests the use of LC shutters in front of the users' eyes and/or in front of the projectors. Palovuori's patents also mention the extension of the shuttering approach to more than two projectors, which he calls multichannel images. In [Pal01b] Palovuori suggests

the development of pulsed projectors, which emit bright images only during their active cycle. They are dimmed down or turned off during the rest of the time.

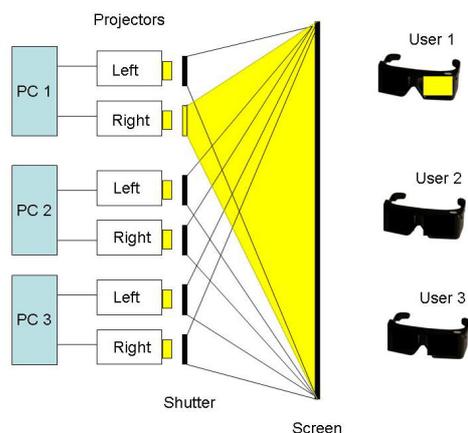
The application of polarization filters for stereo viewing systems was used since 1936, when three approaches were discovered for the economical and industrial production of polarization filters (Bernauer, Kaesemann, Land and Mahler). Thus, picture separation became possible even in color pictures [Waa85]. The technology has not changed much since. The main issues were the loss of light by the filter and crosstalk. Recently, new approaches for better exploitation of light for LCD-projectors was presented [Elk02, Ste05]. Kunz et al. [Kun01, Kun02] employed LC shuttered LCD-projectors to generate an active stereo display for their blue-c system. There have been a small number of other approaches to provide multiple users with individual stereoscopic images. The two-user Responsive Workbench [Agr97] displays four different images in sequence on a CRT-projector at 144Hz, which results in 36Hz per eye per user. They also developed custom shutter glasses for cycling between four eyes. Blom et al. [Blo02] extended this approach to support multi-screen environments such as the CAVE [Cru93]. Barco [Bar04] developed the “Virtual Surgery Table”, which provides two users with individual stereoscopic images by combining shuttered and polarized stereo into one system.

### 3. PROJECTION SETUPS

We describe three configurations which combine the polarization and shutter separation techniques.

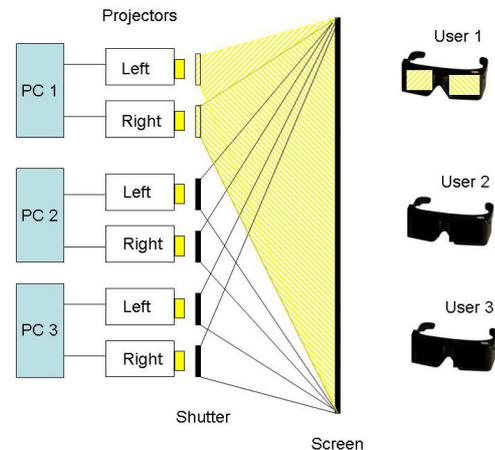
#### General Setup Considerations

A multi view setup which operates only with shutters can be described schematically as follows:



**Figure 1. General multi viewer setup based on shutters. The right eye of user 1 is active.**

A multi view setup which operates with shutters and polarization filters can be described schematically as follows:

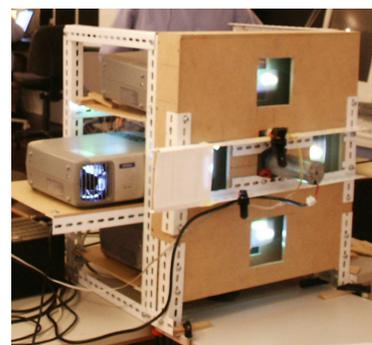


**Figure 2. General multi view setup with shutter and polarization filter; left and right view of user 1 are active.**

As described in [Agr97, Blo02] for pure shutter systems, there are basically two different open/close sequences; user interleaved or eye interleaved that is AL, AR, BL, BR, CL, CR, ... or AL, BL, CL, AR, BR, CR, ... (A, B, C are user indices, L and R are left and right eye index).

#### Combined Mechanical and LC-Shutter

For the mechanical shutter approach we used a spinning plexiglass disc in front of the projectors. For safety reasons the spinning disc is encased in a wooden cage (Figure 3).

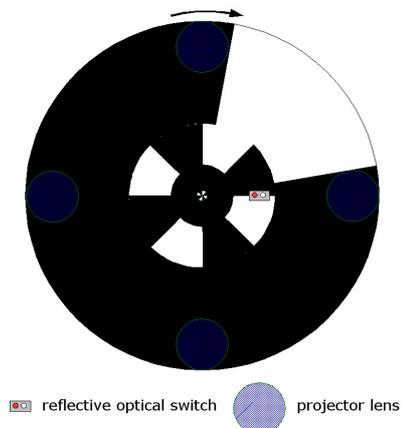


**Figure 3: The spinning disc is contained in a wooden cage separated from the projector rack to avoid vibrations of the projectors. The small motor in the middle spins the disc.**

The straight forward layout for the spinning disc would use three opaque quarters and one transparent quarter. If we open the shutters immediately once the transparent quarter reaches a lens, we introduce crosstalk since one of the other lenses is still open. If we reduce the transparent quarter such that it fits right in between the projector lenses such that only

one lens is open at a time, we reduce the crosstalk significantly (Figure 4). The overall brightness is appropriately reduced. Alternatively we could stick with the  $\frac{3}{4} / \frac{1}{4}$  and open shutters only during times when only one projector lens is open. This introduces phases during which all shutter glasses are closed.

Other layouts are possible, which divide the disc for example into eight zones. Two zones would be transparent, the others opaque. Such a setup would divide the required rotation speed in half, but decreases the actual light output if the disc size is not enlarged. The diameter of the exit pupil of the projectors in relation to the circumference of the disc should be small, since the actual shutter timing depends directly on it.



**Figure 4. Shutter disc for the two-user setup. Four projectors are located around the axis of the disc. One quarter of the disc is open, three quarters are closed. A reflective optical switch is used for generating the synchronization signals for the shutter glasses.**

We currently use a single reflective optical switch to generate the control signals for the shutters. The inner ring of the disc is separated into four black and four white zones, which generate the clock for the shutter glasses. Our current implementation requires that the spinning disc starts always in a defined orientation to switch the LC shutters in the correct order. We could also install an additional optical switch which detects the opening of the first video projector and provides an initialization for the clock signals of the inner ring.

We use the ATMEL ATmega32 [Atm04] micro controller to drive the shutter electronics for the projectors and glasses. The amplified digital outputs of the micro controller are used to drive the shutter glasses. The digital inputs read the signals from the reflective optical switches.

### LC-Shutter

We used standard tethered gaming shutter glasses (Elsa Revelator) for shuttering the users' eyes. For shuttering the projectors we took the same gaming

shutter glasses apart and mounted the shutters directly in front of the projectors. The shutters are a little too small to cover the whole image, but for a test setup they were quite sufficient. The original electronics of the shutter glasses were removed and we used also the ATMEL ATmega32 micro controller to generate the required signals.

LC-shutters are closed if a positive or negative voltage is applied. Otherwise they are open. For fast and continuous on/off switching of the shutters it is necessary to drive them with alternating polarity to avoid memory effects. Our experiments showed that our particular shutters provide the best results if  $\pm 15$  Volts are applied. We were able to run the shutters at up to 300Hz with only little cross talk. Currently we feed exactly the same signal to the shutters in front of the projectors and to the shutter glasses. As a consequence, the closing signal for a projector and the corresponding eye shutter arrive exactly at the same time as the opening signal for another projector and eye. This approach might contribute slightly to the cross talk, but we have not yet experimented with slight delays nor do we know the exact open and close timing behavior of the shutters.

For the final tests we used projectors with 1700 Lumens, which resulted in significant heat development in the shutters. We had to install a fan to cool the shutters down. Larger shutters would allow us to move away from the LCD-projectors, which would distribute the heat across the larger shutter surface. Smaller fans could be mounted near each shutter to avoid heat problems..

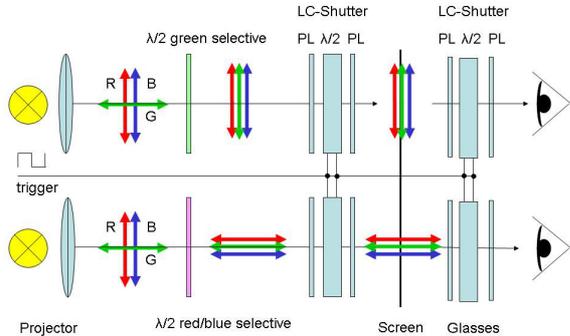
### Combined LC-Shutter and Polarization

For the combination of polarization and the LC-Shutter approach, two solutions are possible:

- Eye separation with shutters and user separation with polarization
- Eye separation with polarization and user separation with shutters

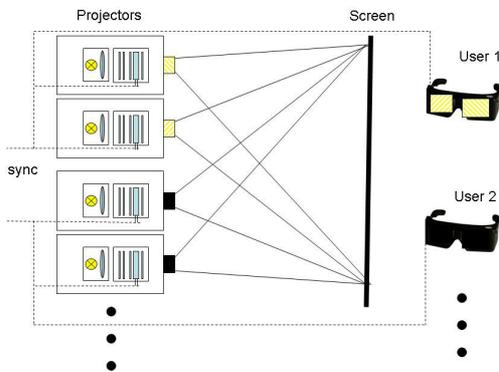
The second approach scales well, since users can be added one by one. For the maximum exploitation of light we used LCD-projectors with an extension of the filter optimization proposed by Stefani [Ste05]. Due to their internal structure most LCD projectors emit already linearly polarized light. Unfortunately, the polarization of the green beam is orthogonal to the polarization of red and blue beam. This problem can be solved by wavelength dependent  $\lambda/2$  retarders for the green channel and a red/blue combination, which rotates only the appropriate color channels by 90 degrees. These selective retarders can be obtained from projector filter manufacturers, e.g. ColorLink [Col04].

LC-Shutter elements use also polarization filter as an integral part of their function. A LC shutter is a combination of two linear polarizers and a voltage controlled retarder. For a single user the light path is shown in Figure 5.



**Figure 5.** The polarized light is emitted from the projector. A selective green  $\lambda/2$  retarder rotates the green channel on the upper projector. A selective red/blue  $\lambda/2$  retarder rotates the red and blue channel on the lower projector. The LC-shutters are then applied to open and close the views. All shutters for one user are opened and closed at the same time.

We plan to use ferroelectric liquid crystal (FLC) shutters for the user separation, which are significantly faster than standard LC shutters. FLC-shutters have to be driven differently than the above mentioned Elsa Revelator Shutters, which we also drive by an ATMEL microcontroller.



**Figure 6.** Combined LC-Shutter and Polarization Setup. View of user 1 is shown all others are closed.

In this configuration, four LC-Shutters are used for one user. All four shutters open and close at the same time. We have to trigger the next user after the shutters of the previous user are definitely closed to avoid cross talk. As projector we will use the Panasonic LB 10-NTE with 2000 ANSI lumens. For this setup a fan was also necessary to cool the optical elements.

## 4. DISCUSSION

We have evaluated the different setups regarding brightness, crosstalk and subjective perception of flicker. We are aware of the difficulty of comparing these setups formally. Nevertheless, they show the principles with their advantages and disadvantages very clearly.

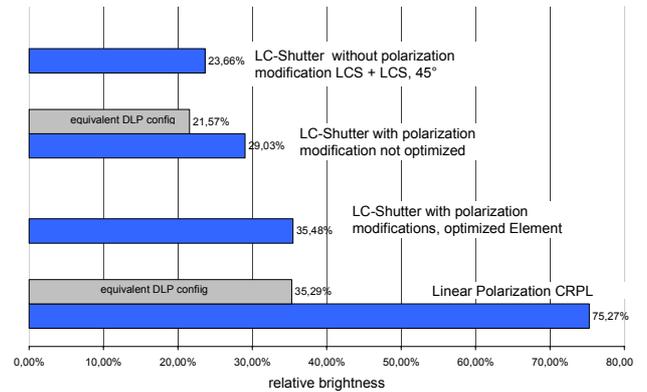
### Brightness considerations

We have measured the relative brightness of filter combinations which can be applied to the various configurations. For the measurement we used the Panasonic LB 10-NTE LCD-projector with 2000 ANSI lumens. As measurement device we used the *Universal Photometer* from Hagen. The following optical elements were used:

| Element | Description  |
|---------|--|
| LCS     | LC-Shutter element Stereographics Crystal Eyes 1   |
| PL      | Linear polarization filter heliopan ES 77  |
| RL2     | Retarder $\lambda/2$   |
| RL2g    | Selective Retarder for green $\lambda/2$   |
| CRPL    | Combined high quality element consisting of a selective $\lambda/2$ retarder, a $\lambda/2$ retarder, and a linear polarization filter |

**Table 1:** Used Elements in measurement.

We present here the results which are the building blocks for the three presented hardware combinations.



**Figure 7.** Relative brightness with a white test pattern. The DLP is shown only for comparison as a representative for non polarized light sources. The last row shows a optimized standard polarization.

The pure LC-shutter setup and the combined LC-shutter/polarization setup follow the same light path. Consequently the pure LC-shutter configuration can also benefit from the modification and optimization with the CRPL Element as a prefilter in front of the projector shutter. Nevertheless the combination of

shuttering and polarization provides twice the brightness as the shutter only approach, but it requires a polarization preserving projection screen.

### Combined Mechanical and LC-Shutter

The spinning disc approach leads to really fast rotations if it is used in its simplest form. For example if we want to achieve 200Hz, 50Hz per eye per user, we need to spin the disc at 50 Hz or 3000 rpm. We tried this approach and we were able to spin the disc at up to nearly 3000 rpm with a small DC motor. At the maximum rotation rate and 49.5 Hz per eye per user the image was basically flicker free. At 45Hz we saw minimal flicker. At lower refresh rates the flicker increased and below 40Hz the flicker was very noticeable.

Our current disc has a diameter of around 40 Centimeters and is made of 3mm plexiglass. The minimal size of the disc is mainly determined by the size of the projectors and the distances of the projector lenses, but the disc size affects directly the time it takes to close and open a projector lens. Larger discs reduce this time significantly, but it is difficult to fully avoid vibrations and noise of large spinning discs. We did not measure the noise of our system, but it was significant and annoying after a while. The cage around the spinning disc could be used to dampen the noise.

One of the main advantages of the mechanical shuttering approach is the possibility to completely avoid static crosstalk between projectors. There is always some cross talk due to the shutter glasses unless they would be replaced by mechanical shutters as well. Thus it is very worthwhile to look at mechanical shuttering systems, which provide 100% transmittance during their open period even though the LC shuttering approach is much easier to implement.

### LC-Shutter

Our first tests investigated the cross talk at different frequencies and supply voltages for the LC-shutters. The least cross talk was found at about 15 Volts. Over 15 Volts the shutters started to show some speckles, which indicates their voltage limitations. Nevertheless we were running the shutters at 15 Volts for many hours without any degradation in image quality, but it is possible that this is above the specs. There was slight crosstalk, which was barely perceivable while viewing stereoscopic images.

Our shutters are quite small and they barely cover the exit pupil of the projector lens. If we use the full resolution of the projectors, we are seeing refraction artifacts from the boundaries of the shutters, which results in some rainbow effects across the images. If we limit ourselves to about 80 percent of the shutter surface, these artifacts are no longer visible.

We experimented with different shutter switching frequencies in the range of 140Hz to 400Hz. We implemented the timing control for two, three, and four users. For two users, each shutter (eye) was open for one fourth of the time, for three users for one sixth, and for four users for one eighth. The tests were performed with two pairs of glasses and four projectors, but the timing was already correct for two, three and four users. The results of these tests:

|               | Two User                    | Three User                 | Four User           |
|---------------|-----------------------------|----------------------------|---------------------|
| <b>140 Hz</b> | flickering                  |                            |                     |
| <b>160 Hz</b> | Very little flicker         |                            |                     |
| <b>200 Hz</b> | no flicker                  |                            |                     |
| <b>240 Hz</b> | No flicker, good image      | slight flicker             | flickering          |
| <b>280 Hz</b> | No flicker, Very good image | barely flickering (270 Hz) | Slight flicker      |
| <b>300 Hz</b> |                             | No flicker                 |                     |
| <b>320 Hz</b> |                             |                            | very slight flicker |
| <b>360 Hz</b> |                             | dark image, pumping        | Dark image          |

**Table 2: Flicker impressions**

Above 320Hz our shutters did not open fully anymore and the images got quite dark. At around 400 Hz the shutters started to show some stripe patterns and did not work properly anymore.

It was amazing to see that these cheap LC-shutters worked quite well even at such high frequencies. For the two user scenario, our favorite frequency was 280Hz, which resulted in a stable and completely flickerless image. But even at 160 Hz the flicker was not really very disturbing, but we did not use the system for long working periods. We did not perceive any difference in brightness between 160Hz and 280Hz for two users, even though the state transition time of the shutter glasses should start to play a role. In particular the transition from the closed to open state is longer than the inverse transition. For three users, the image was slightly darker than for two users, since each eye was exposed to an image for only one sixth of the time. There was little flicker above 270Hz. For four users, the image was clearly darker and there was still slight flicker at 320Hz. At higher shutter frequencies the image got much darker, and it was hard to judge the image quality.

We have also investigated two different sequences of presenting the images to the left and right eye of each

user – similar to the approach in [Agr97]. The viewer interleaved sequences display the left eye images of all users in sequence and then the right eye images. The viewer sequential method displays the left and right eye images for each user directly in sequence. Surprisingly, we did not notice any perceivable differences, even when switching directly back and forth.

### Combined LC-Shutter and Polarization

It is obvious that proper orientation of shutters in front of the projector and in the glasses immediately leads to the desired polarization. The only difference to a purely shutter based approach is a different controller scheme for the shutters. The benefit is we need only half the shutter frequency and obtain double brightness. The FLC-shutters have much faster switching times than the Elsa revelator glasses, but they are also much more expensive. For real world configuration it will depend on the number of users. Based on the measurements a four user setup might already be possible with the Elsa-Shutter.

As of now, we only have used this configuration for pure proof of concept and have not built an entire working environment. So far our experiments look promising and they are confirmed by our measurements. Nevertheless, formal results can only be obtained with a working setup with more than two users.

### General Remarks on shutter techniques

When using LC-shutters in front of a LCD-projector one can benefit from the optimized optical elements described above. One advantage of a pure shutter configuration is that in principle it is not necessary to use a polarization preserving projection surface. The consequence is also that when depolarized on the projection surface, the system has no rotation restrictions anymore. The trade off is very low brightness. An equivalent for the polarization approach is the introduction of retarder ( $\lambda/4$ ) in the open light path to obtain circular polarization, which is also rotation invariant

It is important to notice the relation between the shutter element and the actual image formation inside the projection. As long as the shutters are not synchronized with the video signal, artifacts as image tearing or irregular flicker can occur. Also the usage of color wheels will introduce color artifacts. Off the shelf LCD-projectors seem to be very appropriate because they follow a three LC-chip approach and they are also slow enough to preserve the color information in the LC-cell until the next image will be generated.

Presenting the views with independent projectors has the nice property that the synchronization of the

shutter system can be independent from the computer graphics hardware, because no tight coupling of frame buffer swaps and shutter activity is necessary.

Previous approaches have mainly used quadbuffer stereo and active stereo components [Agr97][Blo02] where such a synchronization is necessary.

Shutter techniques as described here can also be easily combined with color multiplexing for left and right view separation. If the Infitec separation has overcome its color reproduction problems, it might be a powerful alternative to polarization techniques.

### Driving Software

The projection systems were driven by two different software systems Avango [Tra99] and Lightning [Bla98]. Both application frameworks are capable to support multiple views on multi pipe machines or on clusters in a very generic way. We implemented some basic test scenarios on both frameworks, which were basic 3D object viewers.



Figure 8. An image taken directly from the projection screen. It shows four images overlaid on top of each other. Two images are displayed for the left user's eyes and the other two images for the right user's eyes.

## 5. CONCLUSIONS AND FUTURE WORK

We have shown that multi view environments with more than two users are feasible and can be realized with a reasonable amount of hardware. Three different setups have been presented and discussed. The combination of LC-shutter and polarization has a shown to be the most promising approach considering scalability and brightness.

An interesting approach to enhance the projector shuttering is the usage of a pulsed light source which is synchronized with the users glasses which has been mentioned already in [Pal01b]. Major advantages are better exploitation of light, static crosstalk on projector side can be minimized to not existent because of best contrast ratio and dynamic crosstalk is not depending on mechanical properties of the shutter. It is a combination of the contrast properties of mechanical shuttering with the control properties of fast LC-shutters. Stroboscopic light

bulbs or LED-Technology might be an interesting path to follow.

We have experimented with a high luminous LED array. Some issues are already obvious: heat, beam guidance and the bundling of the light.

Besides further technical optimizations, we want to integrate known collaborative 3D-interaction tools and develop adapted tools for the new situation of local 3D-collaboration in the same interaction space.

## 6. ACKNOWLEDGMENTS

The micro controller board was developed within the VRIB project, which was funded by the German government. We thank David Paneque and Alexander Kulik for partly building the various shuttering setups.

## 7. REFERENCES

- [Agr97] Agrawala M., Beers A., Fröhlich B., Hanrahan P., McDowall I., Bolas M.: The Two-User Responsive Workbench: Support for Collaboration Through Individual Views of a Shared Space, *Computer Graphics (SIGGRAPH '97 Proceedings)*, volume 31, pp. 327–332, 1997.
- [Atm04] Atmel AVR microcontroller  
<http://www.atmel.com/products/AVR/>
- [Bar04] Barco: Virtual Surgery Table.  
<http://www.barco.com/VirtualReality/en/products/product.asp?element=523>
- [Bla02] Blach R., Landauer J., Rösch A., Simon A., A Flexible Prototyping Tool for 3D Real-Time User-Interaction. *Proceedings of the Eurographics Workshop on Virtual Environments 98*, 1998 pp. 195-203
- [Blo02] Blom K., Lindahl G., Cruz-Neira C.: Multiple Active Viewers in Projection-Based Immersive Environments. *Immersive Projection Technology Workshop*, March 2002
- [Cru93] Cruz-Neira, C., Sandin, D.J., and DeFanti, T.A. Surround-screen Projection-based Virtual Reality: The Design and Implementation of the CAVE. *Proceedings of SIGGRAPH '93*, 135-142, 1993.
- [Col04] Colorlink Filter  
<http://www.colorlink.com/products/pdfs/select.pdf>
- [Elk02] . Elkhov V.A, Ovechkis J.: Light loss reduction of LCD polarized stereoscopic projection, *Stereoscopic Display and Virtual Reality Systems X*, Proc. of SPIE Vol. 5006, pp. 45-48, 2003
- [Fak04] Fakespace Systems: Active Stereo Digital Projection Technology  
<http://www.fakespace.com/05162003.htm>
- [Ham24] Hammond L.: Stereoscopic Motion Picture Device. U.S. Patent No. 1,506,524, Aug. 26, 1924.
- [Ham28] Hammond L.: Stereoscopic Picture Viewing Apparatus. U.S. Patent No. 1,658,439, Feb. 7, 1928.
- [Jor04]. Jorke H., Fritz M.: INFITEC - A new Stereoscopic Visualisation Tool by Wavelength Multiplex Imaging, *Electronic Displays*
- [Krü94] Krüger, W., and Fröhlich B. The Responsive Workbench. *IEEE Computer Graphics and Applications*, 12-15, May 1994
- [Kun01] Kunz A., Spagno C.: Novel Shutter Glass Control for Simultaneous Projection and Picture Acquisition. *Immersive Projection Technology and Virtual Environments 2001*, Stuttgart, Germany, May 2001, pp. 257-266.
- [Kun02] Kunz A., Spagno C.: Technical System for Collaborative Work. *EGVE 2002*, pp. 73-80; May, 30-31 2002
- [Lip91] Lipton L.: Selection devices for field-sequential stereoscopic displays: a brief history. *Proc. SPIE Vol. 1457*, p. 274-282, *Stereoscopic Displays and Applications II*, John O. Merritt; Scott S. Fisher; Eds. Aug. 1991.
- [Lip01] Lipton L.: The Stereoscopic Cinema: From Film to Digital Projection, *SMPTE Journal*, pp. 586-593, Sept 2001
- [Pal01a] Palovuori, Karri: Apparatus based on shutter function for projection of a stereo or multichannel image. Patent number: WO03003750,  
<http://v3.espacenet.com/textdoc?DB=EPODOC&IDX=WO03003750>
- [Pal01b] Palovuori, Karri: Apparatus based pulsing for projection of a stereo or multichannel image. Patent number: WO03003751,  
<http://v3.espacenet.com/textdoc?DB=EPODOC&IDX=WO03003751>
- [Paa97] Paastor S., Wöpking M.:3 -D Displays: A review of current technologies, *DISPLAYS*, 17, pp. 100-110, 1997
- [Ste05] Stefani O., Bues M., Blach R: Low-loss filter for stereoscopic projection with LCD-projectors, to appear in *Proc. of SPIE Vol. 5008*, 2005
- [Tam99] Tramberend, H. Avocado: A Distributed Virtual Reality Framework. *Proceedings of VR'99 Conference*, Houston, Texas, 14-21, March 1999.
- [Waa85] Waack, F.G., *Stereo Photography*, London pp.72

# Virtual Destruction of a 3D Object with a Stick

Tohru Miyazaki, Toyohisa Kaneko, and Shigeru Kuriyama

Dept. of Information and Computer Sciences  
Toyohashi University of Technology  
{miyazaki,kaneko,kuriyama}@vcl.ics.tut.ac.jp

## Abstract

This paper is concerned with a real-time method for realizing virtual destruction of a 3D object with external force. A target object is a soft *tofu* (bean curd) cube which is easily destructable or breakable with a stick. A spring-mass network model is used to represent such an object. Destruction is realized by cutting a spring when its length exceeds its maximum length due to excess stretch force. A system was implemented on a PC with 2 CPU's and a PHANToM, a force feedback device. A set of optimal parameters are experimentally identified. It is concluded that real-time destruction realized with the presented method can provide destruction close to real one.

## Keywords

Virtual Destruction, Spring Network, Real-time Destruction, Bubble Mesh

## 1 Introduction

Virtual reality (VR) technology has found its important applications in such areas as medicine (e.g. virtual surgery), amusement (e.g. virtual driving), and manufacturing (e.g. virtual factory). It requires typically real-time interaction devices: high speed CPU, visualization devices (e.g. projector, display), and force feedback devices (e.g. PHANToM).

This paper addresses the problem of realizing

virtual real-time destruction of an object with external force. There have been a number of papers concerned with deformation [JP99, DDCB01, SP86, KH96] and cutting [BM02, ML03, MK00, THK98, WO04] in VR, but little works on destruction. For a cutting operation, the affected part on an object is localized only on its touching part of a cutting device (e.g. a knife)[THK98]. However, for destruction, the affected part could be on other parts.

In the area of Computer Graphics where real-time operation is not required, there have been works on deformable objects [TF88a] and fracture [TF88b] using a spring network model. Based upon a stress-strain FEM model, fracture of ductile objects has been treated by O'Brien et al. [OBH02]. A similar work is found on fracture on brittle objects [OH99]. Cracks on drying objects such as clay has been addressed by Hirota et. al. [HTK98].

This paper deals with real-time virtual destruction of soft, brittle objects such as tofu and soft cream cheese with a stick. We employ a PHANToM as a manipulator which enables force feedback with the acquisition capability of 3D position.

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 80-903100-7-9  
WSCG'2005, January 31-February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency. Science Press*

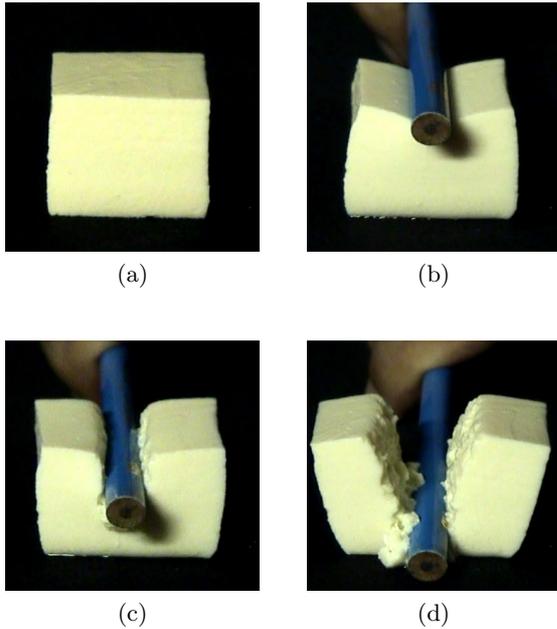


Figure 1: Real Destruction of a tofu lump.

## 2 Method

### 2.1 Approach

As an initial step, we observed the destruction process of a real tofu cube of 3cm edge length with a stick of 8mm diameter. Figure 1 shows real destruction scenes.

Destruction of a tofu cube is shown in Figure 1. Figure 1(a) shows the original shape. Figure 1(b) shows the initial phase where external force through a stick is placed on the top part of tofu to start sinking down slightly. Figure 1(c) and (d) show subsequent destruction by a half way and completely to the bottom, respectively. The phenomena will be physically simulated with a spring network model as follows.

### 2.2 Geometric Model and Network

To simulate destruction, we adopt an approach to represent an object with a network of springs. To construct a spring network, an object of interest is decomposed into a set of tetrahedrons with similar sizes. Take a cube of 3cmx3cmx3cm as an example. As shown in Figure 2(a), it is fitted with a close-packed structure where the mutual distance is 1.5mm. (The close-packed structure can pack the largest number of balls or atoms in a fixed volume.) It is analogous to carving a cube out of a closed packed crystal solid. Figure 2(a) shows

balls of 2.2mm radius and Figure 2(b) shows the resulting network. The number of balls is 306 on the surface and 294 in the internal domain. The resulting network contains 6994 edges or springs.

The second process is to move all the balls situated within a radius of 1.7mm from the surface to the surface. Then an iterative algorithm called the bubble mesh algorithm[SG95, MIKK04] is executed to align all the balls on the surface so that their mutual distances are as equal as possible (to 3.4mm). Then the same iteration aligns all the internal balls which are located three-dimensionally. The resulting ball alignment and network are shown in Figure 2 (c) and (d), respectively. See reference [SG95, MIKK04] for details of this iterative algorithm.

Since the mutual relationship between the balls (or bubbles) in the close-packed structure is known, the 3D structure resulting from the bubble mesh algorithm can be broken in a set of tetrahedrons, which are the basic building block of a 3D object. Then each tetrahedron is represented with a network of four edges as shown in Figure 2(e). A spring and a damper are allocated to each edge of a tetrahedron as shown in Figure 2(f). Note here only a set of a spring and a damper is assigned to an edge, although an edge is shared by a multiple of tetrahedron edges.

### 2.3 Kinematics

Once an object of interest is represented with a spring network, then the next to be investigated is its kinematics behavior. Consider node  $i$ . Its force  $\mathbf{f}_i$  is given as:

$$\mathbf{f}_i = m_i \frac{d\mathbf{v}_i}{d\tau} + c_i \mathbf{v}_i \quad (1)$$

$$c_i = 2\sqrt{k_{max} m_i \alpha^2} \quad (2)$$

where  $m_i$  is the mass assigned to node  $i$ ,  $\mathbf{v}_i$  is the velocity vector of node  $i$ ,  $\tau$  is a small incremental time slice,  $c_i$  is the viscosity of node  $i$ , and  $k_{max}$  is the maximum spring constant among the springs connected to node  $i$ . The damping coefficient  $\alpha$  will be treated as a variable whose optimal value is set experimentally (as will be described in Section 3). Equation (1) can be rewritten as:

$$\mathbf{v}_i = \mathbf{v}_i^p + \left( \mathbf{v}_i^p - \frac{\mathbf{f}_i}{c_i} \right) \left\{ \exp \left( -\frac{c_i}{m_i} \tau \right) - 1 \right\} \quad (3)$$

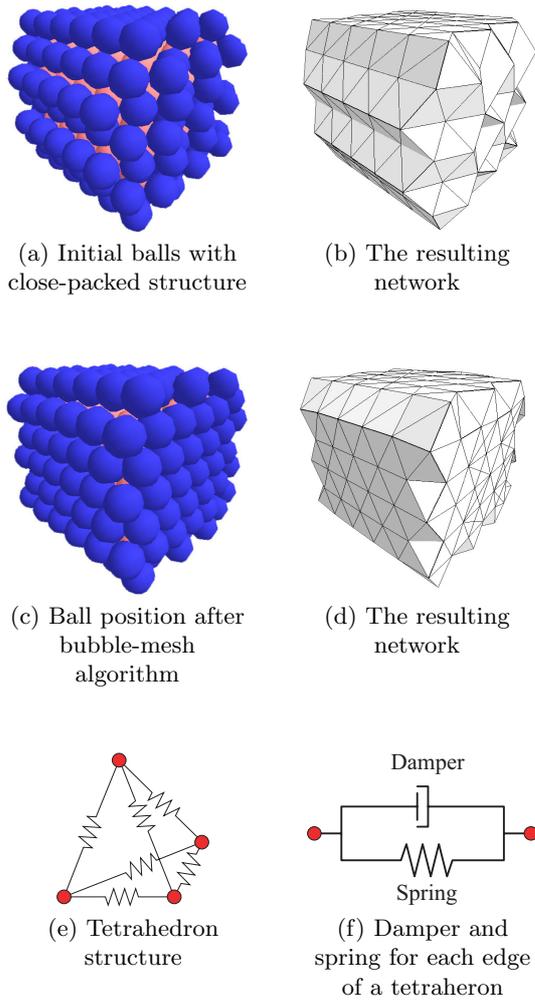


Figure 2: Modeling of a destructive object.

$$\mathbf{x}_i = \mathbf{x}_i^p - \frac{m_i}{c_i} \left( \mathbf{v}_i^p - \frac{\mathbf{f}_i}{c_i} \right) \left\{ \exp \left( -\frac{c_i}{m_i} \tau \right) - 1 \right\} + \frac{\mathbf{f}_i}{c_i} \tau \quad (4)$$

where  $\mathbf{v}_i^p$  and  $\mathbf{x}_i^p$  are the velocity vector and the position vector, respectively, at time  $(t - \tau)$ . Then  $\mathbf{f}_i$ , which is the sum of forces at node  $i$  from the connected springs from node  $j$ , is given as.

$$\mathbf{f}_i = - \sum_j k_{ij} \frac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|} (|\mathbf{x}_i - \mathbf{x}_j| - l_{0ij}) \quad (5)$$

where  $k_{ij}$  is the spring constant of node  $i$  to node  $j$ , and  $l_{0ij}$  is its natural length.

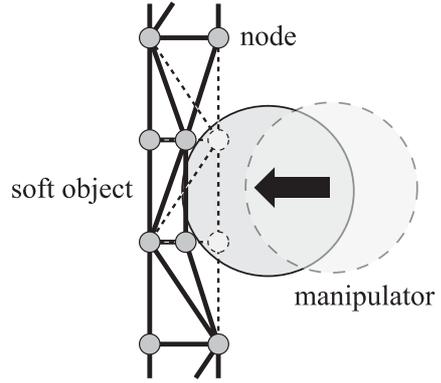


Figure 3: Manipulator position: some object nodes are inside the manipulator.

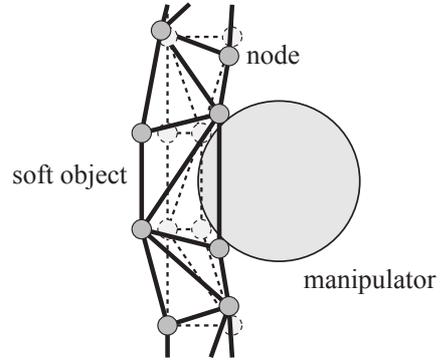


Figure 4: Manipulator position: Nodes are moved to the surface.

## 2.4 External Force

For the case of placing external force on an object of interest with a manipulator, the first step is to detect the collision between the object and the manipulator. In order to reduce computation time, the collision between nodes and surface polygons is detected rather than that between two sets of surface polygons, which is computationally very time-consuming. While all the surface polygons of the manipulator are considered, only those nodes on the object that collided are considered by identifying them in the following manner. At time  $T$ , some nodes shown with dotted circles in Figure 3 which are inside the manipulator can be detected based upon the decision on which sides each node is with respect to each polygon of the manipulator. This decision needs to be carried out by the

number of polygons of the manipulator. Then the nodes located inside are moved along the direction of the manipulator motion and placed on its surface as illustrated with real lines in Figure 3.

The locational displacement of these nodes changes the kinematics balance of the spring network. Equations (3) and (4) are executed iteratively to find the new equilibrium as shown in Figure 4. Here if some edges may be stretched beyond the maximum length allowed, they are cut in the middle as shown in Figure 5.

$$\mathbf{F} = \sum_{i=1}^n \mathbf{f}_i \quad (6)$$

The force given to the force feedback device is the average of the reaction forces between the current  $T$  and  $(T - 4\tau)$  in order to reduce possible noise. Namely  $\mathbf{F}_b$  is given as:

$$\mathbf{F}_b = \frac{\sum_{i=0}^4 \mathbf{F}_{T-i\tau}}{5} \quad (7)$$

where  $\mathbf{F}_{T-i\tau}$  is the reaction force at time  $(T - i\tau)$ .

## 2.5 Visualization

As was mentioned in section 2.1, the destruction of an object is represented by cutting a set of springs in the spring network model. A spring is cut when its length exceeds its maximum length  $l_{max}$ . Actually, it is measured based on the ratio as:

$$l_{max} = l \times l_p \quad (8)$$

As is shown in Figure 5(a), the reaction force is generated at either end of the terminals if  $l \leq l_{max}$  holds. For the case  $l > l_{max}$ , the spring is cut as shown in Figure 5(b). In this case, the reaction force due to the spring at either end of the terminal points becomes null.

When a spring is cut, a new end point is created at the point from each formerly connected end node point by a length of  $l_0/2$  as shown in Figure 2(b). Visualization is carried out for each tetrahedron as illustrated in Figure 6 where there are 10 different patterns shown, depending upon which springs are cut.

## 2.6 Process

The process for virtual destruction mentioned above is summarized as:

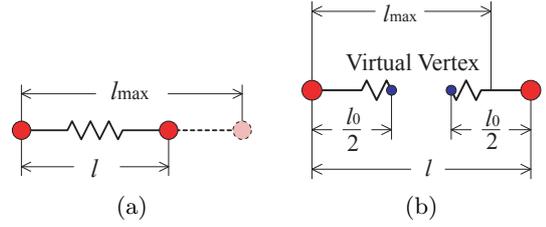


Figure 5: Cutting a spring to realize destruction.

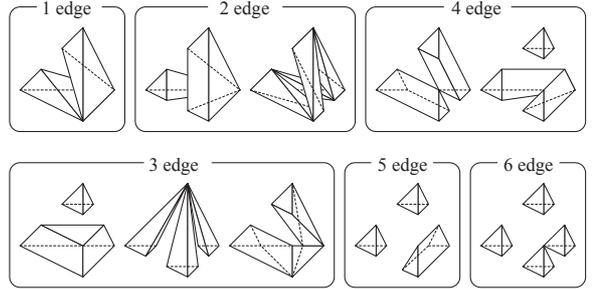


Figure 6: The drawing pattern of a tetrahedron.

- (1) The position and direction of the manipulator is updated based on the force feedback.
- (2) The collision detection is carried out. If some nodes are inside the manipulator, they are moved back to the manipulator surface.
- (3) The reaction force is computed based upon the above node motion. New node positions are iteratively computed using Equations (3) and (4).
- (4) If an edge exceeds its maximum length, it is cut in the middle.
- (5) The sum of reaction forces from all the nodes on the manipulator surface is fed back to the arm of the force feed device.

The above process must be carried out with a frequency exceeding 300Hz[CDA99]. On the other hand, the visualization speed needs to be in 60Hz.

## 3 Experiments

### 3.1 System

A system was implemented on a PC with 2 CPU's (Intel Xeon 2.8GHz) and Microsoft Windows XP. The software was thread-based in order to exploit full power of 2 CPU's. The force feedback device is a PHANTOM(SensAble Technologies Inc.). The total VR system is illustrated in Figure 7. The manipulator position can be controlled with the



Figure 7: The situation of execution.

PHANToM shown on the right and the visualization result is on the monitor on the left.

Three kinds of manipulator can be selected:(1) a round stick of 8mm diameter, a square stick of 6mmx6mm, and a diagonal stick of 3mmx10mm. The length of each stick is 5cm long.

The computation time per cycle (which is equal to  $\tau$  in Equations (3) and (4)) is proportional to the number of edges in the spring network. It was found that it is 2.9 msec. with 8000 edges, resulting to a frequency of 350Hz.

### 3.2 Optimal parameters

For destruction of a tofu cube, it is difficult to measure parameters such as spring constant  $k$ , viscosity  $\alpha$ , and maximum length ratio  $l_p$  without specially designed equipments. Therefore, we estimated these parameters experimentally by observing the way of destruction visually.

For conducting these experiments, we employed a cube of 3cmx3cmx3cm as an object and a stick of 8mm diameter as the manipulator. The same procedure described in section 2.2 was applied to get a spring network where the node lengths are reasonably similar. The weight of the object is 30 grams. The manipulator was moved downward with a speed of 1mm/sec. with a control from the application rather than manual operation in order to avoid unnecessary human errors. 1.5 and 3.0 are set as dynamic and static friction, respectively in order to account for the friction between tofu and the manipulator surface.

Experiments for 500 variations were carried out:(1) ten variations in spring constant  $k$  from 10 to 100 with an increment of 10, (2) ten variations

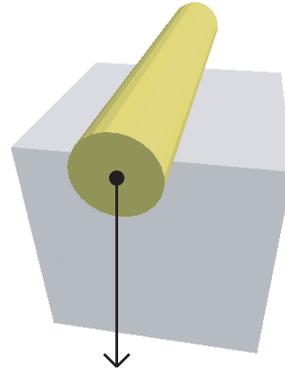


Figure 8: Simulation for deriving appropriate parameters.

Table 1: Parameter applied to a system.

| name                            | value |
|---------------------------------|-------|
| spring limit length ratio       | 1.3   |
| spring coefficient              | 30    |
| damper coefficient ( $\alpha$ ) | 30    |
| dynamic friction coefficient    | 1.5   |
| static friction coefficient     | 3.0   |

in viscosity  $\alpha$  from 5 to 50 with an increment of 5, and (3) five variations in maximum length ratio  $l_p$  from 1.1 to 1.5 with an increment of .1.

The 500 results were rated according to their visual similarity with the actual (see Section 2). Figure 9 shows the best, second best, and two failed examples and their parameters.

Figure 10 shows a temporal change of the total downward reaction force on the manipulator. The downward reaction force generally increases as the manipulator is pressed more downwardly. This trend of reaction force agrees with the actual case.

### 3.3 Experimental Results

Based upon the parameters selected in the above, experiments were conducted. In this experiment, a stick is operated by human. The object is a hexahedron with a dimension of 3cmx3cmx1cm, and a weight of 11grams. The bubble mesh algorithm was executed with the initial close-packed structure with a radius of 1.5mm. The total number of balls was 382 on the surface and 314 internally. And there are 7954 edges in the resulting spring network. A stick of 8mm diameter and 5cm length was used as the manipulator.

Figure 11 shows four phases of destruction sim-

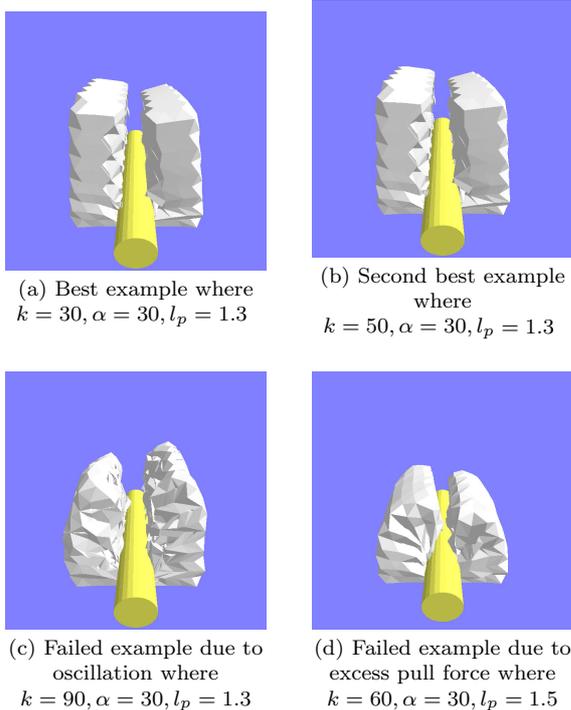


Figure 9: Simulation results

ilar to the real destruction shown in Figure 1. Figure 11(a) is the initial state of the cube, (b) shows the manipulator to start pressing the cube slightly, and (c) shows the phase that the manipulator moved downward about a half of the vertical length, and (d) shows the final phase that the manipulator was pressed to the bottom. The similarity between the four phase figures in Figure 1 and Figure 11 is reasonably good.

Figure 12 shows a temporal change of the total reaction force on the manipulator where the force along X, Y, and Z axis is indicated by thin dotted line, real line, and dense dotted line, respectively. The X, Y, and Z axis represent left-right, downward, and forward-back direction of the object, tofu, shown in Figure 11. The downward trend (Y-axis) agrees with that of Figure 10.

As an example of freeform objects, a tofu globe of 2cm radius was employed. Balls of 2.2mm radius are packed in the globe for the bubble mesh algorithm. The resulting globe contains 370 balls on the surface and 305 in the internal domain. The resulting network contains 8041 edges. The total weight is set to be 36grams. This globe was destroyed with a stick of 8mm radius with 10cm length. The resulting destruction is shown in Fig-

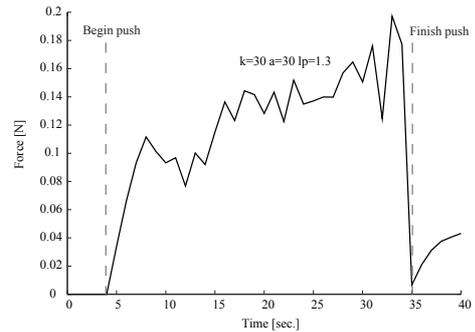


Figure 10: Reaction force along the downward direction.

ure 13. It was judged that this destruction is reasonably close to the real one, which is not shown here, though.

It is seen that real-time virtual destruction is possible using a force feedback device like PHANTOM.

## 4 Conclusion and Future Work

It has been shown that virtual real-time destruction of objects is possible by operating a manipulator. It utilizes a spring network model, where destruction is represented by cutting edges. Visualization is carried out on the basis of tetrahedrons, the basic unit of objects. Appropriate parameters such as spring constant and viscosity were selected experimentally by observing the manner of destruction with varying these parameters. A system with 2 CPU's and a force feedback device was implemented and its real-time operability was demonstrated.

Future works include treating more freeform objects of various sizes.

## References

- [BM02] C. D. Bruyns and K. Montgomery. Generalized interactions using virtual tools within the spring framework: Cutting. *Medicine Meets Virtual Reality*, January 2002.
- [CDA99] S. Cotin, H. Delingette, and N. Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions on Visualization*

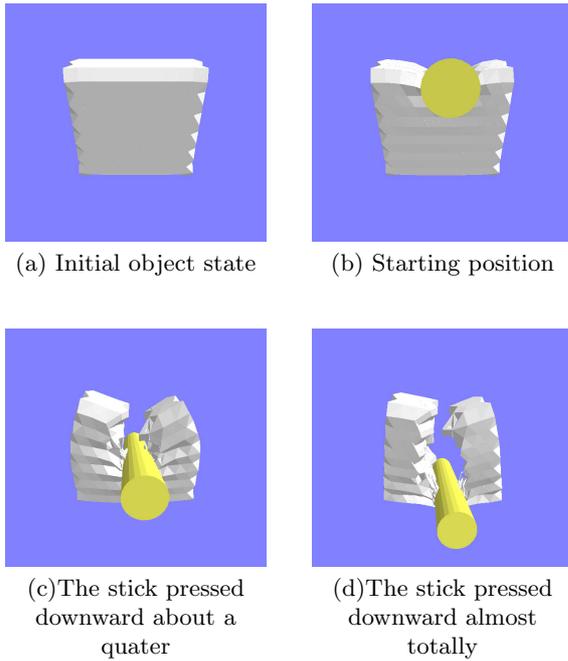


Figure 11: Simulation results.

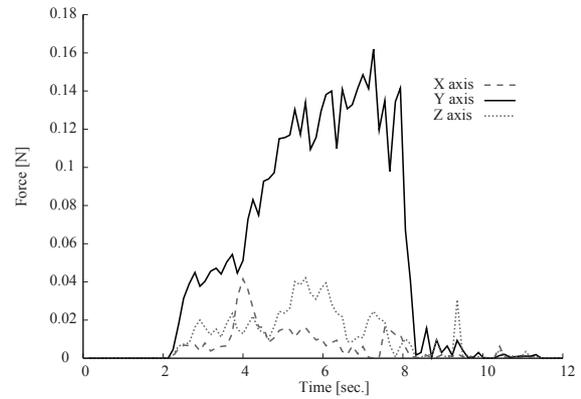


Figure 12: Temporal change of the reaction force along three directions

and *Computer Graphics*, 5(1):62–73, January–March 1999.

- [DDCB01] G. DeBunne, M. Desbrun, M. P. Cani, and A. H. Barr. Dynamic real-time deformations using space and time adaptive sampling. *ACM SIGGRAPH 2001*, pages 31–36, August 2001.
- [HTK98] K. Hirota, Y. Tanoue, and T. Kaneko. Generation of crack patterns with a physical model. *The Visual Computer*, 14:126–137, 1998.
- [JP99] D. L. James and D. K. Pai. Art-defo: Accurate real time deformable objects. *ACM SIGGRAPH'99*, pages 65–72, August 1999.
- [KH96] T. Kaneko and K. Hirota. Simulation of surgical operation onto soft and transforming tissues. *Proc. VSMM'96*, pages 283–287, 1996.
- [MIKK04] I. Mizuno, Y. Iwasaki, T. Kaneko, and S. Kuriyama. Volume preserving deformation of 3d elastic objects by external force. *IEICE Tr. o Information and Systems(Japanese Edition)*, J87-D-II(6):1319–1328, June 2004.

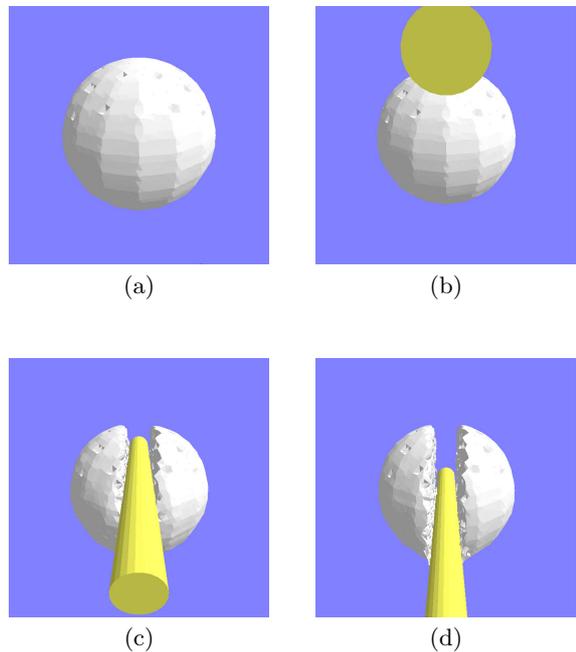


Figure 13: Four phases of a sphere destruction.

- [MK00] A. B. Mor and T. Kanade. Modifying soft tissue models: Progressive cutting with minimal new element creation. *Medical Image Computing and Computer-Assisted Intervention*, 1935:598–607, October 2000.
- [ML03] C. Mendoza and C. Laugier. Simulating cutting in surgery applications using haptics and finite element models. *IEEE Virtual Reality*, 2003.
- [OBH02] J.F. O’Brien, A.W. Bargteil, and J.K. Hodgins. Graphical modeling and animation of ductile fracture. *ACM SIGGRAPH 2002*, pages 291–294, July 2002.
- [OH99] J.F. O’Brien and J.K. Hodgins. Graphical modeling and animation of brittle fracture. *ACM SIGGRAPH 1999*, pages 137–146, August 1999.
- [SG95] K. Shimada and D. C. Gossard. Bubble mesh: automated triangular meshing of non-manifold geometry by sphere packing. *Proc. Solid Modeling and Applications*, pages 409–419, October 1995.
- [SP86] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. *ACM SIGGRAPH’86*, 20(4):151–160, August 1986.
- [TF88a] D. Terzopoulos and K. Fleischer. Deformable models. *The Visula Computer*, 1:306–331, 1988.
- [TF88b] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: viscoelasticity, plasticity, fracture. *ACM SIGGRAPH 1988*, pages 269–278, 1988.
- [THK98] A. Tanaka, K. Hirota, and T. Kaneko. Virtual environment for cutting operation with force feedback. *Proc. VSMM’98*, 1:164–169, 1998.
- [WO04] D. J. Weiss and A. M. Okamura. Haptic rendering of tissue cutting with scissors. *Medicine Meets Virtual Reality*, January 2004.

# Novel Path Search Algorithm for Image Stitching and Advanced Texture Tiling

Petr Somol  
Dept. of Pattern Recognition  
Inst. of Information Theory and Automation  
Pod vodárenskou věží 4  
182 08, Prague 8, Czech Republic  
somol@utia.cas.cz

Michal Haindl  
Dept. of Pattern Recognition  
Inst. of Information Theory and Automation  
Pod vodárenskou věží 4  
182 08, Prague 8, Czech Republic  
haindl@utia.cas.cz

## ABSTRACT

We propose a fast and adjustable sub-optimal path search algorithm for finding minimum error boundaries between overlapping images. The algorithm may serve as an efficient alternative to traditional slow path search algorithms like the dynamical programming. We use the algorithm in combination with novel adaptive blending to stitch image regions. The technique is then exploited in a framework for sampling-based texture synthesis where the learning phase is clearly separated and the synthesis phase is very simple and fast. The approach exploits the potential of tile-based texturing and produces good and realistic results for a wide range of textures.

## Keywords

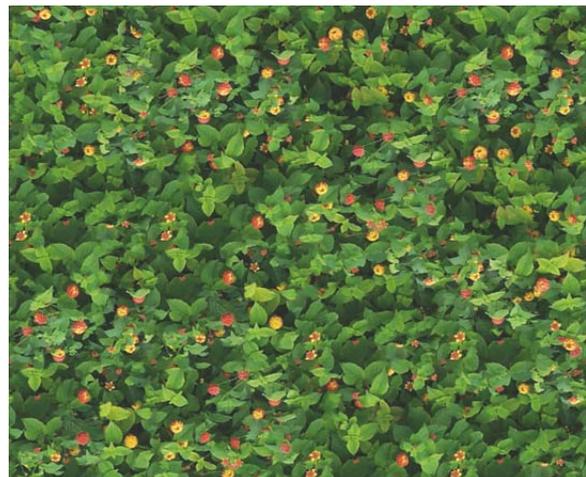
Path Search, Image Stitching, Image Transfer, Adaptive Blending, Texture Tiling, Texture Synthesis.

## 1. INTRODUCTION

Physically correct virtual model visualization can not be accomplished without naturally looking color textures covering virtual or augmented reality scene objects. These textures can be either smooth or rough (also referred to as BTF, see e.g. [MMu03]). The rough textures do not obey the Lambert law and their reflectance is illumination- and view-angle-dependent. Both types of textures occurring in virtual scene models can be rendered either through digitalization of natural samples or by synthesis from appropriate mathematical models. Exact sample digitalization may become prohibitive due to considerable memory requirements, particularly in case of BTFs where each texture is represented by a possibly high number of illumination and view-angle-dependent images. Therefore several texture synthesis methods have been defined to reduce the memory complexity. The related methods may be divided primarily to either intelligent sampling or model-based-

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 80-903100-7-9  
WSCG'2005, January 31-February 4, 2005  
Plzeň, Czech Republic.  
Copyright UNION Agency – Science Press*



**Figure 1. The picture is made of rectangular tiles. Can you guess, what is the tiling grid size and how many different tiles have been used ? (see Fig.10)**

analysis and synthesis. The model-based techniques (see, e.g., [Bes74], [Kas81], [BK98], [Hai91], [PJ00], [GH03], [HH00], [HH02]) describe texture data by means of multidimensional mathematical models and later use an extremely compact representation for seamless synthesis of arbitrarily sized texture images. Intelligent sampling approaches (see, e.g., [DB97], [EL99], [Efr01], [Hee95], [XGS00], [CS03], [KS03]) rely on sophisticated sampling from real texture measurements. Sampling based methods currently achieve better visual quality at a cost of less effective compression. Particularly the simpler intelligent-sampling methods have been receiving constant attention for their applicability in graphic hard-

ware. DeBonet’s method [DB97] constructs the texture in coarse-to-fine fashion, preserving conditional distribution of filter outputs over multiple scales, while another multi-scale method [Hee95] uses histograms of filter responses. The “image quilting” method [Efr01] by Efron et al. connects rectangular pieces of the texture sample together while minimizing the boundary cut error. Similarly the algorithm by Xu et al. [XGS00] uses regular tiling combined with a deterministic chaos transformation. Very good results can be achieved by employing Wang tiles [CS03] or the so-called graphcut textures [KS03]. All of these methods implement some sort of source texture sampling and the best of them often produce very realistic synthetic textures.

However, no texture synthesis method can be considered ideal for all potential applications. Either the performance, universality, visual quality of results or applicability in current hardware may become the prohibiting factor.

## Our Motivation

Many of the current sampling methods involve image operations that may result in visible seams, typically when combining incompatible pieces of texture. A good way to improve the visual quality in such cases is to find (possibly irregular) boundaries between the image pieces to minimize the visual error. In the following we propose a sub-optimal yet highly effective alternative to traditional slow path-search algorithms. Taking use of the algorithm we show a method of developing the texture by visually unrecognizable image transfers (to be referred to as patching). We also show how to utilize this technique in a simple way to obtain groups of mutually connectable tiles representing the given texture. However, the main part of the paper concentrates on the path search and seamless boundary creation problem as we believe the solution presented here is generally usable in many different contexts and applications.

The paper is structured as follows: Section 2 discusses in detail how a virtually invisible transition between two texture image regions can be created. Section 2.1 shows a novel sub-optimal algorithm for path search that can be used instead of slow exponential algorithms like the dynamical programming. Section 2.2 shows how to improve the visual transition quality in cases when minimum error path does not suffice to prevent discontinuities. Section 2.3 extends the stitching technique to enable seamless transfer of whole image regions (patching). In Section 3 we show a trivial yet well-performing way of seamless tile creation. Assuming one tile has been created, we show in Section 3.2 how new, visually different derivatives can be created based on it while all of the tiles remain mutually connectable. Such tile

sets can then be used to synthesize texture images of significantly higher quality than it is possible with simple tiling approaches, as shown in the Experiments Section 4. Section 5 summarizes the advantages and discusses the drawbacks and perspectives of the proposed methods.

## 2. IMAGE STITCHING

Consider *image stitching* a process of creating natural transitions between two image regions. This task is simpler for naturally self-similar (e.g. homogeneous) textured images. The transition is to be made as unnoticeable and indistinguishable from the neighboring image areas as possible. We define the technique based on the minimum error boundary cut idea, as used in the “image quilting” algorithm [Efr01]. Let us assume that each stitch between two equally sized overlapped image regions  $R_1$  and  $R_2$  is oriented. A right-oriented stitch image will consist mostly of pixels from  $R_1$  along its left side and mostly of pixels from  $R_2$  along its right side. Creating such stitch can be imagined as attaching a cropped part of  $R_1$  (source) to  $R_2$  (target) as demonstrated in Figure 2. The following sections show in detail how to crop and how to reduce unwanted visual errors for cases when cropping itself is not sufficient.



**Figure 2. Image stitching (right-oriented case).** The source image is cropped from the right along the minimum error path and placed over the target background image.

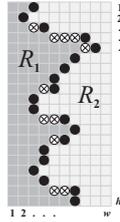
### Minimum Error Path Search

Let us consider a right-oriented stitch creation problem, as demonstrated in Figure 2. Suppose the source region  $R_1$  is to be placed over target region  $R_2$  where the overlap size is  $w \times h$  pixels. Width  $w$  is considered a user parameter that determines how relaxedly the transition between  $R_1$  and  $R_2$  should be constructed and thus trades the achievable visible quality for algorithm efficiency. To make the transition as invisible as possible,  $R_1$  is cropped from the right side along a minimum error path before attaching to  $R_2$ . The minimum error path is constructed to lead vertically from the top row to the bottom row of error map  $E$ , which represents the visual difference between  $R_1$  and  $R_2$  for each pixel of the overlap region:

$$E[i, j] = d(R_1[i, j], R_2[i, j]), \quad i = 1, \dots, w \quad j = 1, \dots, h$$

where  $d(., .)$  is, e.g., the Euclidean distance of two RGB pixel color values. Note: Error maps in Figures 2, 4, 5 and 6 depict higher error by darker grey lev-

els. We adopt a simplified path representation model, as shown in Figure 3. Only the pixels lying to the left of (and on) the path are to be copied from  $R_1$  to the underlying  $R_2$ .



**Figure 3. Simplified path representation model for the right-oriented stitch. Each row contains one control point (black dot). Complementary points (crossed dots) must be added to make the path continuous.**

Each path is represented unambiguously by a sequence of *control points*  $\mathbf{c}$ , one for each row:

$$\mathbf{c} = \langle c_1, c_2, \dots, c_h \rangle, \quad c_j \in \{1, \dots, w\}$$

However, the complete path as a vertically oriented, continuous sequence of pixels in  $E$  must include not only the control points, but also complementary points (marked by crossed dots in Figure 3). From several possible complete path definitions we have adopted the one that suits our oriented-stitch approach, i.e., where each control point becomes visibly the rightmost point in its row:

$$Path^c = \{(i, j) : j = 1, \dots, h; i = \mu_j, \dots, c_j\}$$

where

$$\mu_j = \min(c_{j-1} + 1, c_{j+1} + 1, c_j).$$

For each path a criterion can be evaluated to assess the expected visible transition inconsistency:

$$\varepsilon(Path^c) = \sum_{(i,j) \in Path^c} E[i, j]$$

Now we can define a sub-optimal minimum path search algorithm on error map  $E$  of  $w \times h$  pixels. The basic idea is to develop some initial  $Path^c$  in stepwise manner by conditional shifting of the control points. New control point position(s) get fixed only if  $\varepsilon(Path^c)$  would decrease. This ensures the algorithm to converge. The algorithm first evaluates single control point shifts in each step as long as the criterion value can be decreased. Next it attempts to bypass larger error areas by shifting small groups of consecutive control points forming a vertical line at once. Whenever such step improves the criterion value, fine tuning in form of single control point shifts follows. The only user parameter  $o_{\max}$  (where  $1 \leq o_{\max} \leq h$ ) depicts the maximum number of control points processed in one step. Higher  $o_{\max}$  values lead to better or equal solutions at a cost of longer computation.

*Oscillating search of minimum error path:*

Initialization: for  $j = 1, \dots, h$  let

$$c_j = \arg \min_{1 \leq i \leq w} E[i, j];$$

Let  $\bar{\varepsilon} = +\infty$ ; "initial error value boundary"

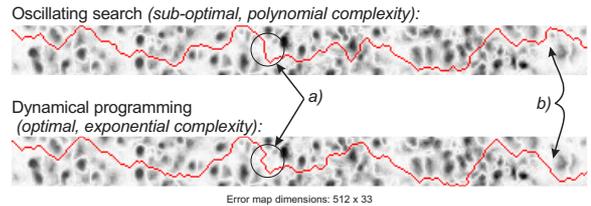
1: Let  $o = 1$ ; "initial step size"

2: For  $j = 1, \dots, h$

```
{
  Store  $c_j \dots c_{j+o-1}$  temporarily to  $c_j^{temp} \dots c_{j+o-1}^{temp}$ ;
  For  $i = 1, \dots, w$ 
  {
    For  $k = j, \dots, j + o - 1$  let  $c_k = i$ ; "vertical line"
    If  $\varepsilon(Path^c) < \bar{\varepsilon}$  update  $\bar{\varepsilon}$  and go to 1;
  }
  Restore  $c_j \dots c_{j+o-1}$  from  $c_j^{temp} \dots c_{j+o-1}^{temp}$ ;
}
```

Let  $o = o + 1$ ; If  $o \leq o_{\max}$  go to 2;

We use this algorithm as a fast alternative to slow optimal path search procedures like the dynamical programming. The main reason is computational speed. The *oscillating search* has polynomial complexity while optimal search is always exponential. The oscillating search is a step-wise procedure that sequentially improves some actual solution and thus can be stopped at any moment to yield a usable result. The visible differences between optimal and suboptimal search results can be considered marginal, as demonstrated in Figure 4.



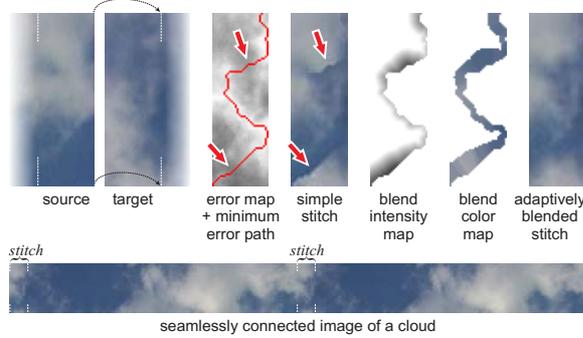
**Figure 4. Sub-optimality vs. optimality of path search.**

The suboptimal search as defined here prohibits returning path sections (Figure 4a). Differences, if any, occur in areas of evenly distributed error (Figure 4b) and thus remain visually unimportant. In the case depicted in Figure 4 the sub-optimal search was faster than dynamical programming by a factor of 5000 (depending on error map dimensions) while the numerical difference between the sub-optimal and optimal criterion values was about 10%. Moreover, experiments show that numerical optimality of found paths is not crucial for the visual appearance of transitions. It is more important to ensure that the overlap image region itself is positioned and sized not to rule out the existence of low error paths (for example of such a difficult error map see Figure 5).

Remark: In the context of off-line texture analysis to be discussed in the following (Section 4) the time factor is not crucial. However, it is of key importance in many other applications (see, e.g., [Efr01]).

## Adaptive Boundary Blending

The minimum path based stitching often produces good natural appearance of image transition areas. However, if no good path exists in the error map, visible artifacts can not be avoided (as demonstrated in Figure 5—simple stitch).



**Figure 5. Adaptive blending to improve visual consistency of stitched image areas.**

Therefore we have defined the *adaptive boundary blending* as an attempt to reduce the visibility of such unwanted and striking high-error artifacts, should they emerge during the stitching process. The idea is to interpolate between overlapped source region  $R_1$  and target  $R_2$  with a locally adjusted intensity while utilizing the *minimum error path* both as a boundary and as a coloring guideline. Our experiments show that to prevent unnaturally smoothed appearance it is better to keep the affected area minimal, just enough to mask the high error artifacts. In the following we consider a right-oriented stitch again. This is of importance now, because the blending process we adopt is targeted to one side (left in this case) of the path only what helps to better preserve the original image appearance.

Let us denote  $S$  the adaptively blended stitch region of  $w \times h$  pixels to be created from  $R_1$  overlapping  $R_2$ . We assume the minimum error path  $Path^c$  and error map  $E$  are known (see the previous Section). The *blending range* (maximum distance from the path where pixels get affected) is to be set as parameter  $\rho$ . The  $\rho$  value should be specified with respect to the properties of the processed source image. Higher image resolution should be reflected on higher  $\rho$ . However, with  $\rho$  being too high the blending effect can become visually too apparent. On the contrary, too small  $\rho$  may not be sufficient to suppress the worst visual stitching errors, should they appear. The stitch is created row-wise, i.e., for each  $j = 1, \dots, h$ :

$$S[i, j] = \begin{cases} R_2[i, j] & \text{for } c_j < i \leq w \\ R_1[i, j] * \alpha + R_2[m, n] * (1 - \alpha) & \text{for } 1 < i \leq c_j \end{cases}$$

where

$$(m, n) = \arg \min_{(a,b) \in Path^c} \sqrt{(a-i)^2 + (b-j)^2}$$

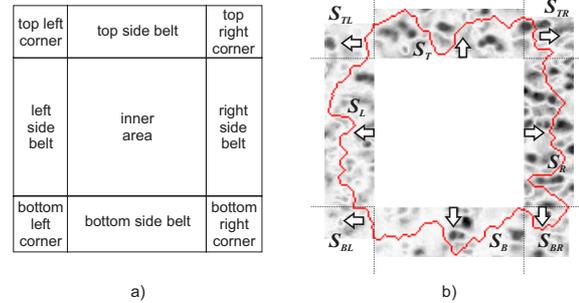
$$\alpha = \max\left(\frac{\sqrt{(m-i)^2 + (n-j)^2}}{\rho * E[m, n]}, 1\right)$$

The adaptive blending process can be visualized using the *blend intensity* and *blend color* maps (see Figure 5). The blend intensity map represents the weighing information on how  $R_1$  contributes to the blended result in the area left from the path. In Figure 5 the darker pixels depict lower contribution.  $R_2$  contributes to the same area indirectly using the blend color map that shows how color information is extracted from  $R_2$  path pixels. As seen on examples (Figures 5, 8 and 11), the boundary can be made almost unnoticeable in this way, except of cases when the transition is made between principally incompatible texture image areas.

Remark 1: We should note that simpler interpolation algorithms have been tested as well but led to worse results, usually emphasizing incompatibilities of  $R_1$  and  $R_2$  image contents alongside the minimum error path. Remark 2: A little better visual quality can be achieved if  $S[i, j]$  for  $1 < i \leq c_j$  would depend not only on the single closest on-path pixel  $(m, n)$  in  $Path^c$ , but on several close neighboring pixels in  $Path^c$ .

## Image Patching

The image stitching method described in the previous Sections can be extended to transfer general continuous image regions while keeping the transition between the old and new unnoticeable.



**Figure 6. Patch creation. The stitching technique is used to create sides and corners of the patch; the inner area is simply copied. White arrows show stitch orientations (requires optimization).**

For the sake of simplicity we define a patch as a part of image surrounded by a continuous minimum error path that does not extend out of a given surrounding rectangle. Our *image patching* algorithm is illustrated in Figure 6. The inner area is simply copied from source position to the target. Side stitches are then created inside the four side belts of a user-specified width with obvious orientation (see the white arrows in Figure 6). Finally the corner stitches are added,

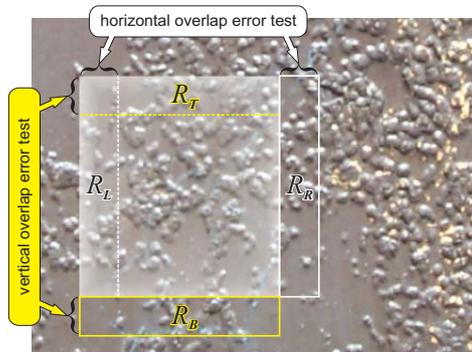
with one additional restriction: the path initial and final *control points* must be fixed to remain connected to those of the side stitches to ensure the patch is surrounded by one continuous path.

### 3. TEXTURE TILING

Texture tiling is extensively used for various purposes ranging from simple web page design to realistic 3D display of natural surfaces. Creating a single seamless tile out of some source image is thus a traditional problem for which numerous algorithms exist. One of the simplest is probably the “Photoshop clone tool” approach. The idea is to half-shift the image and then to use the manual clone tool to blot out the now apparent horizontal and vertical seams that have emerged inside the image. From the automated methods many take use of extensive blending in not very sophisticated manner, what often results in too striking visual change of the texture appearance along the tile borders. The image stitching technique described in previous sections is well suited for the purpose of seamless tile creation and can be expected to give considerably better results than simple blending methods.

#### Tile Overlap Optimization and Stitching

First, we search for such rectangular region in the source texture image, where the opposite border areas are the most visually consistent in both the horizontal and vertical direction.



**Figure 7. Tile template positioning on a source image.**

As depicted in Figure 7, the search procedure minimizes the visual difference among image regions  $R_L$  and  $R_R$ , and among  $R_T$  and  $R_B$ , respectively. The overlap width is to be decided by the user, the width and height of the candidate region is optimized by the procedure. The tile (brightened region) is then cut out and made seamless by overlapping and stitching image region  $R_R$  over  $R_L$ , and  $R_B$  over  $R_T$ , respectively. The tile sizing and positioning phase is trivial; its purpose is mainly to provide for better stitching results. We use the average RGB Euclidean distance as a criterion of visual consistency. Remark: From

performance reasons we split the tile positioning and sizing process to two sub-optimal independent phases, vertical and horizontal.

#### Deriving Mutually Connectable Tile Sets

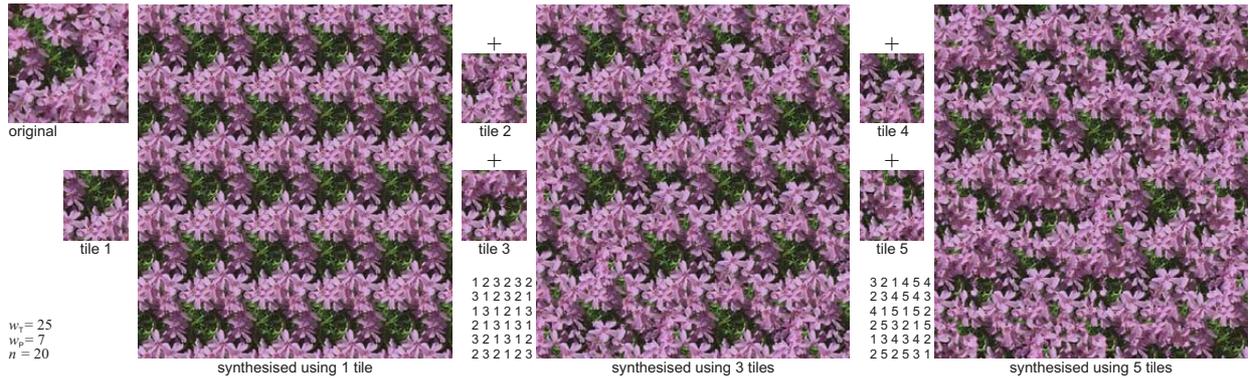
For many textures a single tile is not sufficient to synthesize naturally looking images. Simple tiling usually leads to unwanted emphasis of the rectangular grid, despite the seamlessness of tiles (see the left image in Figure 8). Moreover, the character of many textures makes it impossible to create a single tile that would sufficiently represent all the texture variability. Our idea is to make tiling more realistic by employing more than one tile per texture. The positive effect of increasing the number of tiles is demonstrated in Figure 8.

New tiles can be obtained using the *patching* technique described in Section 2.3. New tiles can be created by making a copy of the template tile and subsequently covering its inner area by patches taken from different positions in the source texture image. Different algorithms can be defined to accomplish this task, both deterministic and non-deterministic, with different properties. It is possible to define sophisticated algorithms aiming to build tile sets of specified properties, e.g. representing well the variability of original texture image contents. This algorithm definition problem can be considered outside the scope of this paper. Therefore, we suggest here only the simplest possibility. Each new tile is can be obtained by applying one patch only, sized little less or equal to the tile size and taken from a random source position. Even if the patch and tile sizes are equal, the original tile contents remain usually unaffected alongside its borders (and thus the tile remains connectable), because of the expected irregularity of minimum error paths. This effect can be seen in Figure 6b. Remark: The described tile set derivation procedure is obviously independent on the particular technique used to obtain the initial template tile.

### 4. EXPERIMENTS

We have tested the presented techniques on a set of textures, mostly from VisTex and UTIA databases. The benefit of using tile sets instead of single tiles is demonstrated in Figure 8. The picture of pink bloom over green grass can be considered a difficult texture. A single tile is clearly insufficient to obtain a naturally looking synthetic image. Adding two tiles improves the result. However, no less than 5 tiles seem sufficient to suppress the striking visibility of the regular tiling pattern.

The examples in Figures 1, 8 and 11 have been obtained as follows: the first tile for each texture has been created automatically (see Section 3) with the only restriction of some reasonable minimum and

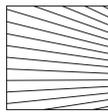


**Figure 8. Visual improvement of synthesis results by combining an increasing number of tiles.**

maximum tile size. The overlap width was set to ca. 1/10 of the expected tile size. The first tile was then used as a template for tile set generation (see Section 3.2). 30 patched tiles had been generated per texture from which the final tile subsets were selected manually. In cases where the results had been found unsatisfactory, the experiment was repeated with modified parameters. The stitching region width, both in the case of the first tile creation (see Section 3.1) and subsequent tile patching (see Sections 2.3 and 3.2) has proved to be important and is therefore marked in Figures 8 and 11 as  $w_T$  and  $w_P$ , respectively. Patch source positions were obtained randomly, but optimized subsequently on a small neighborhood of size  $n$  to avoid worst possible visual problems. For the adaptive blending the parameter  $\rho$  had been set to 3 in all cases, which has shown to be satisfactory in most cases. Note: The synthesized images in Figs. 8 and 11 are cropped to fit in the page.

Time complexity of all computations is low. Each tile can be generated typically in seconds on a 2GHz PC. Once a good set of tiles has been found, texture synthesis is reduced to assembling different tiles to the grid according to some index matrix. This can be done directly in the GPU at no additional time cost.

Expectably good results have been obtained with most of the regular textures (Fig. 11c). Very good results have been obtained even for some more difficult textures, where the irregular texture nature (chocolate, Fig. 11e) could have caused problems. Some of the most difficult textures displaying natural objects like tomatoes can be synthesized surprisingly well (Figs. 11b, 11d). However, with some textures it showed impossible to prevent unnatural artifacts (see in detail Fig. 11f, which looks fine at first sight). We have also experienced problems with textures where good overlapping regions could not be identified to create the initial tile, or with textures containing very distinct particles or structure that has a negative im-



**Figure 9. Example of a problem structure in source texture images.**

perfect on the tile grid visibility (Figure 9). Nevertheless, the technique proved to be well capable of synthesizing broad range of natural textures. Some of the possible visual problems follow from principal reasons and cannot be avoided; others can be removed or suppressed by repeating the experiments with modified parameters, in particular with different stitch widths and patching sources. The quality of output also depends on the size of the original texture sample. Too small a source image would result in too homogenous and regular results. Small source images usually imply the necessity to create more tiles to compensate the effect of denser and thus more apparent tile grid. To capture low frequency information, larger tiles are necessary. To prevent the visibility of the tile grid the tiles in a set must have sufficiently varied content, i.e., the patching process must affect most of the tile image surface. No tiling can overcome certain principal problems like, e.g., the problem of source texture images containing slightly rotated linear structures (Figure 9) from which no smoothly connectable tile can be derived. In general, if the texture exhibits some apparent linear structure then it should be either vertically or horizontally aligned. Skewed and rotated linear structures require sufficiently large samples or are not manageable at all.



**Figure 10. The front page image is composed of 8 different tiles in a 4x4 grid according to the index matrix: ((7,6,1,5), (3,7,5,8), (5,1,6,2), (4,3,2,3)).**

Finally, the answer to the front page Figure 1 question is in Figure 10.

## 5. CONCLUSION

We have presented a novel fast path search algorithm and adaptive blending technique that are suitable for seamless image transfer, in particular in the context of texture synthesis. Using these tools we have demonstrated a relatively simple technique that enables synthesis of naturally looking textures by means of advanced image tiling. We show how a set of mutually connectable yet differently looking rectangular tiles can be obtained for a broad range of source texture measurements. We show that even very irregular textures can be represented well using such tile sets. The main advantage of the presented technique is the clear separation of the off-line texture analysis, while the synthesis is reduced to trivial combination of pre-computed tiles. The visual quality of output is close or comparable to the best of current techniques as shown in Figures 8 and 11.

The tiling technique is scalable. The trade-off between the visual quality and computational complexity can be controlled by changing the number of tiles in the tile set. For each texture some minimum number of tiles is usually necessary to ensure sufficient quality of results. Regular (possibly rectangular) textures without much detail can be represented by fewer tiles than highly irregular stochastic textures. Most of the algorithms presented here are extendable or modifiable. We have found the technique to be extendable for BTF modeling (bidirectional texture fields, see, e.g. [MMu03], [MMe03]) to enable particularly accurate display of natural surfaces with respect to view- and illumination- angles.

## 6. ACKNOWLEDGMENTS

This work has been supported by the EC project IST-2001-34744 RealReflect, FP6-507752 MUSCLE, grant No. A2075302 and 1ET400750407 of the Grant Agency of the Czech Academy of Sciences. We thank Alexei Efros for permission to reproduce illustrations from [Efr01] (Figure 12).

## 7. REFERENCES

- [Bes74] Besag J.: Spatial interaction and the statistical analysis of lattice systems. *Journ. of the Royal Statistical Society*, B-36, 2 (1974), 192–236.
- [BK98] Bennett J., Khotanzad A.: Multispectral random field models for synthesis and analysis of color images. *IEEE Trans. on PAMI* 20, 3 (Mar. 1998), 327–332.
- [CS03] Cohen M.F., Shade J., Hiller S. and Deussen O.: Wang Tiles for image and texture generation. In *ACM Transactions on Graphics* 22, 3, SIGGRAPH 2003, 287–294.
- [DB97] De Bonet J.: Multiresolution sampling procedure for analysis and synthesis of textured images. In *Proc. SIGGRAPH 97* (1997), ACM Press, 361–368.
- [Efr01] Efros A.A., Freeman W.: Image quilting for texture synthesis and transfer. In *SIGGRAPH 01* (2001), Fiume E., (Ed.), ACM Press, 341–346.
- [EL99] Efros A. A., Leung T. K.: Texture synthesis by non-parametric sampling. In *Proc. Int. Conf. on Computer Vision* (2) (1999), 1033–1038.
- [GH03] Grim J., Haindl M.: Texture modelling by discrete distribution mixtures. *Computational Statistics & Data Analysis* 41, 3–4 (2003), 603–615.
- [Hai01] Haindl M.: Texture synthesis. *CWI Quarterly* 4, 4 (Dec. 1991), 305–331.
- [HH00] Haindl M., Havlíček V.: A multiresolution causal color texture model. In *Advances in Pattern Recognition*, LNCS 1876. Springer-Verlag, Berlin, (Aug. 2000), ch. 1, 114–122.
- [HH02] Haindl M., Havlíček V.: A multiscale color texture model. In *Proc. 16th Int. Conf. on Pattern Recognition* (2002), Kasturi R., Laurendeau D., (Eds.), IEEE Computer Society, 255–258.
- [Hee95] Heeger D.J. Bergen J.: Pyramid based texture analysis/synthesis. In *Proc. SIGGRAPH 95* (1995), ACM Press, 229–238.
- [Kas81] Kashyap R.: Analysis and synthesis of image patterns by spatial interaction models. In *Progress in Pattern Recognition 1* (North-Holland, 1981), Kanal L., A. Rosenfeld, (Eds.), Elsevier.
- [KS03] Kwatra V., Schödl A., Essa I., Turk G., Bobick A.: Graphcut Textures: Image and Video Synthesis Using Graph Cuts. In *ACM Transactions on Graphics* 22, 3, SIGGRAPH 2003, 277–286.
- [LLX\*01] Liang L., Liu C., Xu Y.-Q., Guo B., Shum H.-Y.: Real-time texture synthesis by patchbased sampling. *ACM Transactions on Graphics (TOG)* 20, 3 (2001), 127–150.
- [MMu03] Meseth J., Müller G., Sattler M., Klein R.: BTF Rendering for Virtual Environments. *Virtual Concepts* 2003, (2003), 356–363.
- [MMe03] Müller G., Meseth J., Klein R.: Compression and real-time Rendering of measured BTFs using local PCA. *Vision, Modeling, and Visualization*, (2003), 271–280.
- [PJ00] Portilla J. S. E.: A parametric texture model based on joint statistics of complex wavelet coefficients. *Int. Journal of Computer Vision* 40, 1 (2000), 49–71.
- [SP00] Somol P., Pudil P.: Oscillating search algorithms for feature selection. In *Proc. 15th Int. Conf. on Pattern Recognition* (2000), vol. 2, IEEE Computer Society, 406–409.
- [Wei01] Wei L. Levoy M.: Texture synthesis over arbitrary manifold surfaces. In *Proc. SIGGRAPH 01* (2001), ACM Press / Addison Wesley.
- [XGS00] Xu Y., Guo B., Shum H.: Chaos Mosaic: Fast and Memory Efficient Texture Synthesis. *Tech. Rep. MSR-TR-2000-32*, Redmont, 2000.

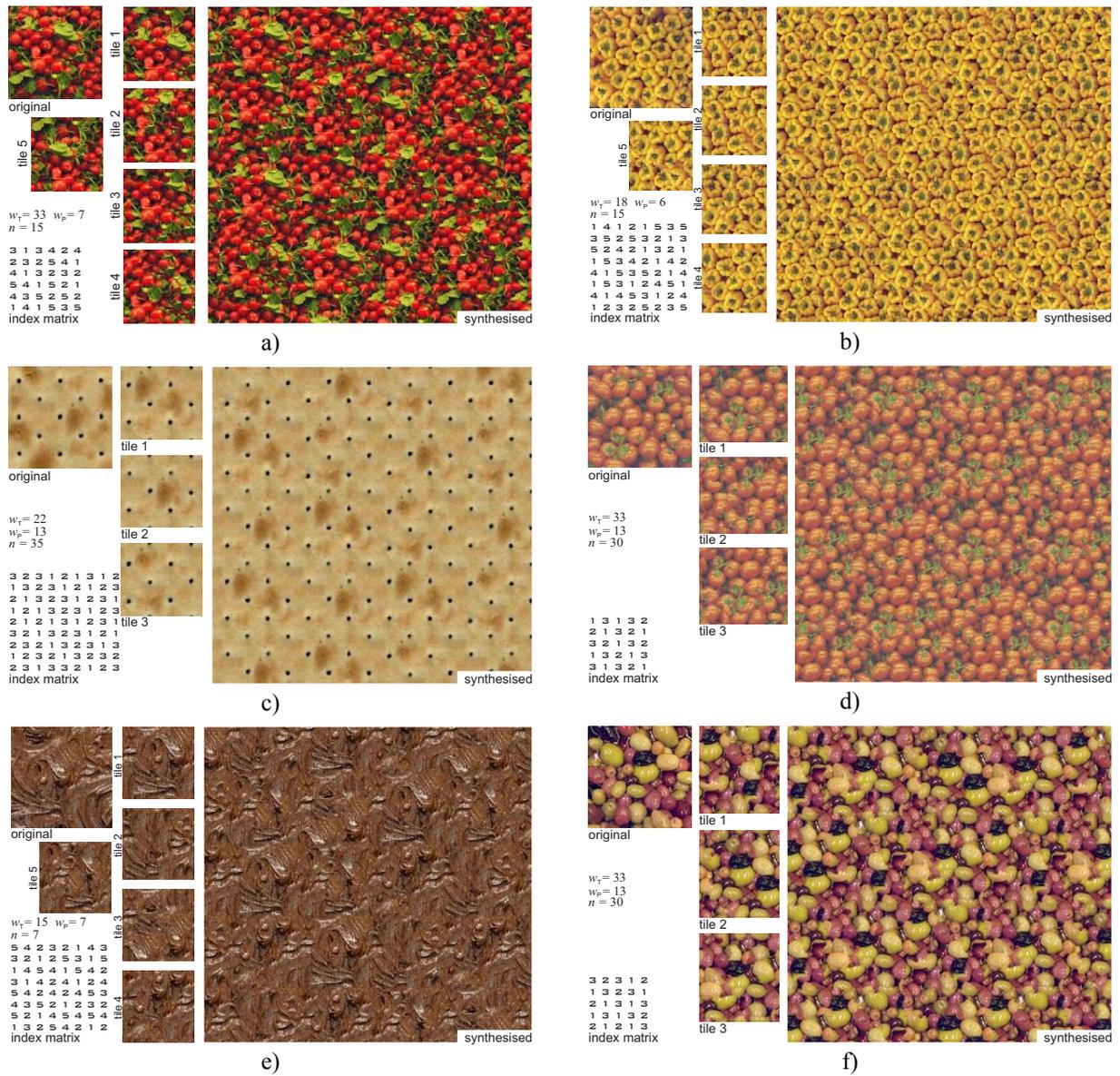


Figure 11. Examples of tilings obtained with the proposed method.

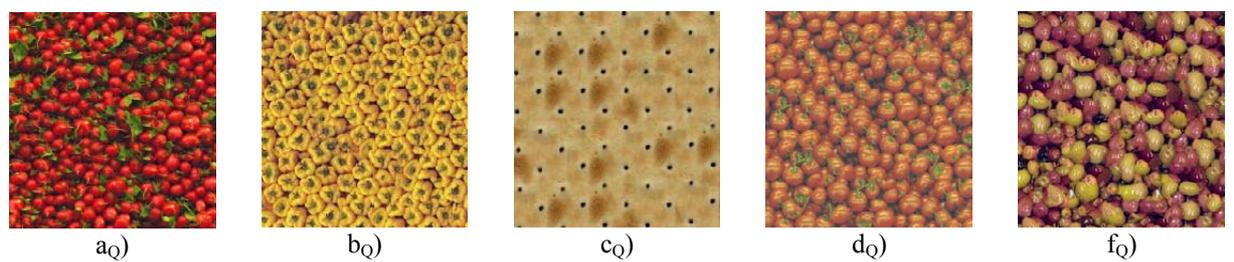


Figure 12. Image Quilting [Efr01] results for comparison.

# Interpolation Search for Point Cloud Intersection

Jan Klein

University of Paderborn, Germany  
janklein@uni-paderborn.de

Gabriel Zachmann

University of Bonn, Germany  
zach@cs.uni-bonn.de

## ABSTRACT

We present a novel algorithm to compute intersections of two point clouds. It can be used to detect collisions between implicit surfaces defined by two point sets, or to construct their intersection curves. Our approach utilizes a proximity graph that allows for quick interpolation search of a common zero of the two implicit functions.

First, pairs of points from one point set are constructed, bracketing the intersection with the other surface. Second, an interpolation search along shortest paths in the graph is performed. Third, the solutions are refined. For the first and third step, randomized sampling is utilized.

We show that the number of evaluations of the implicit function and the overall runtime is in  $O(\log \log N)$ , where  $N$  is the point cloud size. The storage is bounded by  $O(N)$ .

Our measurements show that we achieve a speedup by an order of magnitude compared to a recently proposed randomized sampling technique for point cloud collision detection.

**Keywords:** Collision detection, weighted least squares, proximity graphs, implicit surfaces.

## 1 INTRODUCTION

In the past few years, point clouds have had a renaissance caused by the wide-spread availability of 3D scanning technology. Interaction with objects thus represented often requires intersection tests between pairs of objects. Other applications, such as Boolean operations [1] or physically-based simulation [10], require fast construction of points on the intersection curves.

In order to do that, one must define an appropriate surface (even if it is not explicitly reconstructed). The simple weighted least-squares (WLS) definition of point cloud surfaces is quite attractive and can be evaluated very fast [3]. In order to overcome a problem caused by Euclidean distances in the weighting functions, [12] proposed a method that utilizes (conceptually) a Voronoi diagram and a geometric proximity graph to approximate geodesic distances between the query point and the cloud points.

In this paper, we present a method that can quickly find intersection points on objects represented by point clouds. It converges even if the sampling is sparse, compared to the surface areas, and even if the distance between the surfaces contains local minima.

The idea is to utilize a proximity graph over the point clouds and perform interpolation search along geodesic paths through these graphs. The search is initialized by randomized sampling that tries to find two points on

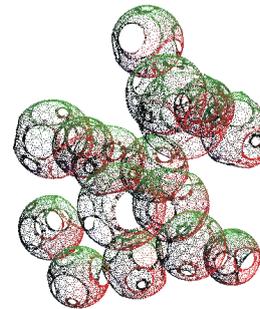


Figure 1: One of our point clouds for benchmarking our novel intersection method ( $> 137000$  points).

one object and on different sides of the other object. Then, our interpolation search converges quickly to an approximate intersection point. Finally, the space surrounding that is sampled to get very accurate (discrete) intersection points.

Our new algorithm can be combined very easily with any acceleration data structure for collision detection or intersection construction. For instance, with bounding volume hierarchies [11], the algorithm presented here would be invoked at the leaves.

In the following, we will first give a review of related work. Section 3 gives a quick recap of the WLS surface definition and the proximity graph we are using. Section 4 describes the details of our new algorithm while Section 5 shows its performance.

## 2 RELATED WORK

An attractive way of handling point clouds is to define the surface as the zero set of an *implicit function* that is constructed from the point cloud. Usually, this function is not given analytically but “algorithmically” [2, 3, 4].

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 2005 conference proceedings, ISBN 80-903100-7-9  
WSCG'2005, January 31–February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency – Science Press

This is a general method that can be used for reconstruction as well as ray-tracing or collision detection. Another very popular method is to define the surface as the set of fixed points of a projection operator based on local polynomial regression [5].

Geometric queries on point clouds have been studied extensively. An interesting result related to our problem can be found in [7, p. 908f]. They use a divide-and-conquer algorithm to find the closest pair of  $n$  points in time  $O(n \log n)$  which is, of course, not applicable to realtime collision detection.

However, there is very little literature on geometric queries on the implicit surfaces defined by such object representations. The work most related to ours is [21]. They sample an implicit function with a stochastic differential equation to detect intersections. Since it is a method for general implicit surfaces, they do not exploit the proximity graph available here. In addition, our new method is much simpler.

In [11] a bounding volume (BV) hierarchy for point cloud collision detection was proposed. The BV traversal first visits leaves where intersections are more likely. Then, a sampling technique similar to [21] determines the intersection points.

An algorithm to perform Boolean operations on solids was presented in [1]. However, their algorithm does not work for surfaces implicitly defined, and it requires closed surfaces.

As mentioned above, our method is based on proximity graphs, which have been studied extensively in the past decade. There is a broad spectrum of them, including the Delaunay graph, nearest-neighbor graph,  $\gamma$ -graph,  $\alpha$ -shape, and the spheres-of-influence graph, to name but a few; see [9] for a good survey.

### 3 IMPLICIT SURFACE MODEL

In this section, we give a quick recap of the weighted least-squares (WLS) method [2, 3], which was originally introduced by McLain [13] in the context of contouring, plus its geodesic extension based on proximity graphs [12].

#### 3.1 Weighted Least Squares

Let a point cloud  $\mathcal{P}$  with  $N$  points  $p_i \in \mathbb{R}^3$  be given. Then, an appealing definition of the surface from  $\mathcal{P}$  is the zero set  $S = \{x \mid f(x) = 0\}$  of an implicit function

$$f(x) = n(x) \cdot (a(x) - x) \quad (1)$$

where  $a(x)$  is the weighted average of all points  $\mathcal{P}$

$$a(x) = \frac{\sum_{i=1}^N \theta(x, p_i) p_i}{\sum_{i=1}^N \theta(x, p_i)}. \quad (2)$$

Usually, a Gaussian kernel (weight function)

$$\theta(x, p) = e^{-d(x,p)^2/h^2}, \quad d(x, p) = \|x - p\|, \quad (3)$$

is used, but other kernels work as well.

The bandwidth  $h$  of the kernel allows us to tune the decay of the influence of the points. It should be chosen such that no holes appear.

The normal  $n(x)$  is defined as the direction of smallest weighted covariance, which is the smallest eigenvector of the centered covariance matrix  $B(x) = \{b_{ij}(x)\}$  with

$$b_{ij}(x) = \sum_{k=1}^N \theta(x, p_k) (e_i(p_k - a(x))) (e_j(p_k - a(x))) \quad (4)$$

where  $e_i, i \in \{0, 1, 2\}$  is a basis of  $\mathbb{R}^3$ .

The above definition can produce artifacts in the surface  $S$ , which are mainly caused by the Euclidean distance function  $d(x, p)$  that does not take the topology of  $S$  into account. This problem can be solved by using a different distance function  $d_{\text{geo}}(x, p)$  in (3) that is based on geodesic distances on the surface  $S$ . Therefore, a geometric proximity graph can be utilized where the nodes are points  $\in \mathcal{P}$ . Then, geodesic distances between the points can be approximated by shortest paths on the edges of the graph.

We use the following *geodesic kernel*:

$$\theta(x, p) = e^{-d_{\text{geo}}(x,p)^2/h^2} \quad (5)$$

when computing  $f$  by (1)–(4).

#### 3.2 Geodesic Distance Approximation

There is a whole spectrum of different proximity graphs over a set  $\mathcal{P}$  of points. We decided to use the sphere-of-influence graph (SIG) as it has reduced artifacts in WLS point cloud surfaces dramatically [12]. In this section, we will give a short overview of this fairly little known proximity graph [6, 14]. Moreover, we will shortly summarize how to precompute and store the geodesic distances.

**The Sphere-of-Influence Graph (SIG).** The idea is to connect points if their “spheres of influence” intersect. More precisely, for each point  $p_i$  the distance  $d_i$  to its nearest neighbor (NN) is determined and two points  $p_i$  and  $p_j$  are connected by an edge if  $\|p_i - p_j\| \leq d_i + d_j$ .

As a consequence, the SIG tends to connect points that are “close” to each other relative to the local point density. In noisy or irregularly sampled point clouds, however, a lot of isolated “mini-clusters” can appear, even though there are no holes in the original surface. Because our root bracketing will utilize the graph, it would fail in such a situation.

Therefore, we use the  $r$ -SIG( $\mathcal{P}$ ): instead of computing the distance to the NN for each node, we compute the distance to the  $r$ -nearest neighbor and then proceed as in the case of  $r = 1$ . That means, the larger  $r$ , the more nodes are directly connected by an edge. In our experience, it seems best to choose  $r = 3$  or  $r = 4$ , and

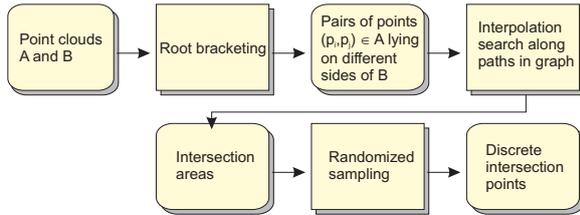


Figure 2: Outline of our point cloud collision detection.

then prune away all “long” edges by an outlier detection algorithm [22].

**Precomputing Geodesic Distances.** Computing shortest paths on-the-fly during the collision detection process would be, of course, prohibitively expensive, so we pre-compute and store them in a *close-pairs shortest-paths* (CPSP) map [12].

Since the Gaussian (3) decays fairly quickly, we need to store only paths up to some length for defining the surface. The contribution of nodes in Equations 2 and 4 that are farther away can be neglected. That means, for each point  $p_i$  we have to run a single-source-shortest path algorithm, but only for points whose influence in  $p_i$  is larger than some small threshold.

In [12] it is shown that all these geodesic distances for a whole point cloud of size  $N$  can be computed and stored in  $O(N)$  time and space.

## 4 CONSTRUCTING POINTS ON THE INTERSECTION

Given two point clouds  $A$  and  $B$ , the goal is to determine whether or not there is an intersection, i.e., a common root  $f_A(x) = f_B(x) = 0$ , and, possibly, to compute a sampling of the intersection curve(s), i.e., of the set  $\mathcal{X} = \{x \mid f_A(x) = f_B(x) = 0\}$ . Both can be achieved very quickly by exploiting the proximity graph.

First, our algorithm tries to bracket intersections by two points on one surface and on either side of the other surface (see Figure 2). Second, for each such bracket, it finds an approximate point in one of the point clouds that is close to the intersection (see Figure 3). Finally, this approximate intersection point is refined by subsequent randomized sampling. This last step is optional, depending on the accuracy needed by the application.

In the following, we describe each step in detail.

### 4.1 Root Bracketing

Finding common roots of two (or more) nonlinear functions is extremely difficult [17]. Even more so here, because the functions are not described analytically, but algorithmically.

As mentioned before, our algorithm starts by constructing random pairs of points on different sides of one of the surfaces. The two points should not be too far apart, and, in addition, the pairs should evenly sample the surface.

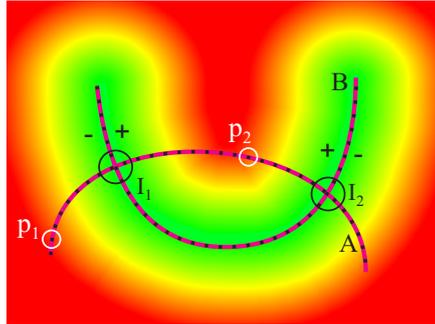


Figure 3: Two point clouds  $A$  and  $B$  and their intersection spheres  $I_1$  and  $I_2$ . Our root finding procedure, when initialized with  $p_1, p_2 \in A$ , will find an approximate intersection point inside the *intersection sphere*  $I_1$ .

An exhaustive enumeration of all pairs is, of course, prohibitively expensive. Therefore, we propose the following randomized (sub-)sampling procedure.

Assume that the implicit surface is conceptually(!) approximated by surfels (2D discs) of equal size [16, 19]. Let  $\text{Box}(A, B) = \text{Box}(A) \cap \text{Box}(B)$  and  $\bar{A} = A \cap \text{Box}(A, B)$ . Then, we want to randomly draw points  $p_i \in \bar{A}$  such that each surfel  $s_i$  gets occupied by at least one  $p_i$ ; here, “occupied by  $p_i$ ” means that the projection of  $a(p_i)$  along the normal  $n(p_i)$  onto the supporting plane of  $s_i$  lies within the surfel’s radius.

For each  $p_i$  we can easily determine another point  $p_j$  (if any) in the *neighborhood* of  $p_i$  so that  $p_i$  and  $p_j$  lie on different sides of  $f_B$ . We represent the neighborhood of a point  $p_i$  by a sphere  $c_i$  centered at  $p_i$ .

An advantage of this is that the application can specify the density of the intersection points that are to be returned by our algorithm. From these, it is fairly easy to construct a discretization of the complete intersection curves (for instance, by utilizing randomized sampling again).

Note that we never need to actually construct the surfels, or assign the points from  $A$  explicitly to the neighborhoods, which we describe in the following. Section 4.2 describes how to choose the radius of the spheres  $c_i$ .

In order to find a  $p_j \in A \cap c_i$  on the “other side” of  $f_B$ , we use  $f_B(p_i) \cdot f_B(p_j) \leq 0$  as an indicator. This, of course, is reliable only if the normals  $n(x)$  are consistent throughout space. If the surface is manifold and connected, this can be achieved by a method similar to [8].

Utilizing our proximity graph (which is a supergraph of the nearest-neighbor graph), we can propagate a normal to each point  $p_i \in A$ . Then, when defining  $f(x)$ , we choose the direction of  $n(x)$  according to the normal stored with the NN of  $x$  in  $A$ .<sup>1</sup>

<sup>1</sup> Surprisingly, the direction of  $n(x)$  is consistent over fairly large volumes without any preconditioning.

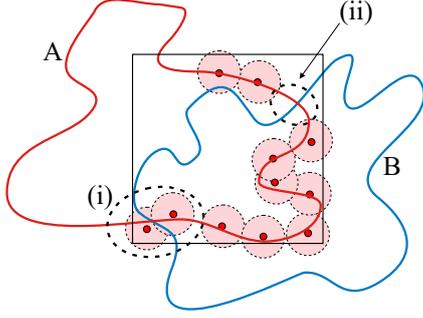


Figure 4: If the spherical neighborhoods  $c_i$  (red) are too small, not all collisions can be found. (i) adjoining neighborhoods do not overlap sufficiently, their intersection contains no cloud point. (ii) surface is not covered by neighborhoods  $c_i$ .

In order to sample  $A$  such that each (conceptual) surfel is represented by at least one point in the sample, we use the following

### Lemma 1

Let  $A$  be a uniformly sampled point cloud. Further, let  $S_A$  denote the set of conceptual surfels approximating the surface of  $A$  inside the intersection volume of  $A$  and  $B$ , and let  $a = |S_A|$ . Then, in order to occupy each surfel with at least one point with probability  $p = e^{-e^{-c}}$ , where  $c$  is an arbitrary constant, we have to draw  $n = O(a \cdot \ln a + c \cdot a)$  random and independent points from  $\bar{A}$ .

Proof: see Appendix A.

For instance, if we want  $p \geq 97\%$ , we have to choose  $c = 3.5$ , and if  $a = 30$ , then  $n \approx 200$  random points have to be generated.

Now, given a point  $p_i \in A$ , we have to determine another point  $p_j \in A \cap c_i$  on the other side of  $f_B$ .

This is done by testing  $f_B(p_i) \cdot f_B(p_j) \leq 0$  for all points  $p_j \in A \cap c_i$ . In the next section, we show that  $c_i \cap A$  is only a small, constant number of points. Therefore, a point  $p_j$  on the other side of  $f_B$  can be determined in time  $O(1)$  (if it exists). We utilize our proximity graph and a breadth-first search to access the points in the spherical neighborhood  $c_i$ .

## 4.2 Size of Neighborhoods

The radius of the spherical neighborhoods  $c_i$  has to be chosen so that, on the one hand, all  $c_i$  cover the whole surface defined by  $A$ . On the other hand, the intersection with each adjoining neighborhood of  $c_i$  has to contain at least one point in  $A$  to miss no collisions lying in the intersection of two neighborhoods. The situation is illustrated in Figure 4.

To determine the minimal radius of a spherical neighborhood  $c_i$ , we introduce the notion of *sampling radius*.

### Definition 1 (Sampling radius)

Let a point cloud  $A$  as well as a subset  $A' \subseteq A$  be given. Consider a set of spheres, centered at  $A'$ , that cover the surface defined by  $A$  (not  $A'$ ), where all spheres have equal radius. We define the sampling radius  $r(A')$  as the minimal radius of such a sphere covering.

Remember that we draw  $n = O(a \cdot \ln a + c \cdot a)$  random and independent points from  $\bar{A}$ . Let  $A'$  denote the point cloud consisting of these random points. Then, spheres with radius  $2 \cdot r(A')$  centered at points in  $A'$  contain always points of the neighboring spheres and, of course, cover the surface.

The sampling radius  $r(A')$  can obviously be estimated as the radius  $r$  of a surfel  $s_i \in S_A$ .

Let  $F_A$  denote the surface area of the implicit surface over  $\bar{A}$ . Then, the surfel radius  $r$  can be determined by

$$\frac{F_A}{a} = \pi \cdot r^2 \Rightarrow r = \sqrt{\frac{F_A}{a \cdot \pi}}.$$

Assume that the implicit surface over  $\bar{A}$  can also be approximated by surfels of size  $r(A)$ . Then,  $F_A$  can be estimated by

$$F_A = |\bar{A}| \cdot \pi r(A)^2.$$

Overall,  $r(A')$  can be estimated by

$$r(A') = r(A) \cdot \sqrt{\frac{|\bar{A}|}{a}} \approx r(A) \cdot \sqrt{\frac{\text{Vol}(A, B)}{\text{Vol}(A) \cdot a}} \cdot |A|.$$

The size of  $\bar{A}$  can easily be estimated depending on the ratio of  $\text{Vol}(A)$  and  $\text{Vol}(A, B)$ , the sampling radius  $r(A)$  can easily be determined in the preprocessing.

In [12] it has already been shown that for uniformly distributed points  $p_i \in \mathbb{R}^3$  and a sampling radius of  $r(A)$  only  $O(\lceil \sqrt{2} \cdot m \rceil^2)$  points  $\in A$  lie in a sphere with radius  $m \cdot r(A)$ . If we choose  $m = 2r(A')/r(A)$ , then at most  $O(\lceil \sqrt{2} \cdot 2r(A')/r(A) \rceil^2) = O(1)$  points  $\in A$  lie in a spherical neighborhood with radius  $2 \cdot r(A')$  because  $m = 2r(A')/r(A)$  is constant.

## 4.3 Interpolation Search

Having determined two points  $p_1, p_2 \in A$  on different sides of object  $B$ , the next goal is to find a point  $\hat{p} \in A$  “between”  $p_1$  and  $p_2$ , such that the approximate distance from  $B$  is small enough, i.e.,  $|f_B(\hat{p})| < \epsilon$ . In the following, we will call such a point *approximate intersection point* (AIP). The true intersection curve  $f_B(x) = f_A(x) = 0$  will pass close to  $\hat{p}$  (usually, it does not pass through any points of the point clouds).

Depending on the application,  $\hat{p}$  might already suffice. If the true intersection points are needed, then we refine the output of the interpolation search by the procedure described in Section 4.5.

If  $B$  does not have boundaries (e.g., holes) and  $A$  is sufficiently densely sampled, then there must be a point

---

```

 $l, r = 1, n$ 
 $d_{l,r} = f_B(P_l), f_B(P_r)$ 
while  $|d_l| > \epsilon$  and  $|d_r| > \epsilon$  and  $l < r$  do
     $x = l + \lceil \frac{-d_l}{d_r - d_l} (r - l) \rceil \{*\}$ 
     $d_x = f_B(P_x)$ 
    if  $d_x < 0$  then
         $l, r = x, r$ 
    else
         $l, r = l, x$ 

```

---

Algorithm 1: Pseudo-code of our root finding algorithm based on interpolation search.  $P$  is an array containing the points of the shortest path from  $p_1 = P_1$  to  $p_2 = P_n$ , which can be precomputed.  $d_i = f_B(P_i)$  approximates the distance of  $P_i$  to object  $B$ . (\*) Note that either  $d_l$  or  $d_r$  is negative.

$\hat{p} \in A$  lying on the shortest path between  $p_1$  and  $p_2$  for which  $|f_B(\hat{p})| < \epsilon$ . Let us assume that  $f_B$  is monotone along the path  $\overline{p_1 p_2}$  (this can always be ensured by making the surfels small enough). Then, instead of doing an exhaustive search along the path, we could utilize binary search to find  $\hat{p}$ . Better yet, we can utilize interpolation search, which makes sense here, because the “access” to the key of an element, i.e., an evaluation of  $f_B$ , is fairly expensive [20]. The runtime of interpolation search is in  $O(\log \log m)$ ,  $m =$  number of elements.

Algorithm 1 for our interpolation search assumes that the shortest paths are precomputed and stored in the CPSP map (Section 3.2). Analogously to [12], it is easy to see that the storage is still linear.

However, in practice, the memory consumption could be too large for huge point clouds. In that case, we can compute the path  $P$  on-the-fly at runtime by Algorithm 2. Theoretically speaking, the overall algorithm is now in linear time. However, in practice, it still behaves sublinear because the reconstruction of the path is negligible compared to evaluating  $f_B$  (see Section 5.3).

## 4.4 Models with Boundaries

If the models have boundaries and the sampling rate of our root bracketing algorithm is too low, not all intersections will be found (see Figure 5). In that case, some AIPs might not be reached, because they are not connected through the proximity graph.

Therefore, we propose to modify the  $r$ -SIG. After constructing the graph, we usually prune away all “long” edges by an outlier detection algorithm (see Section 3.2). Now, we only mark these edges as “virtual”. Thus, we can still use the  $r$ -SIG for defining the surface as before. For our interpolation search, however, we can also use the “virtual” edges so that small holes in the model are bridged.

---

```

 $q.insert(p_1); \text{ clear } P$ 
repeat
     $p = q.pop$ 
     $P.append(p)$ 
    for all  $p_i$  adjacent to  $p$  do
        if  $d_{geo}(p_i, p_2) < d_{geo}(p_1, p_2)$  then
            insert  $p_i$  into  $q$  with priority  $d_{geo}(p_i, p_2)$ 
until  $p = p_2$ 

```

---

Algorithm 2: This algorithm can be used to initialize  $P$  for Algorithm 1 if storing all shortest paths in the CPSP map is too expensive. ( $q$  is a priority queue.)

## 4.5 Precise Intersection Points

If two point clouds are intersecting, our interpolation search returns a set of AIPs. Around each of them, an *intersection sphere* of radius  $r = \|x - p_1\|$  where

$$x = \frac{1}{d_1 + d_2} (d_2 p_1 + d_1 p_2)$$

contains a true intersection point ( $p_1$  and  $p_2$  are the points  $\in A$  with smallest distance to  $B$  lying on different sides of  $B$ ,  $d_i = f_B(p_i)$ ). The idea is illustrated in Figure 6. If AIPs are not precise enough, then we can sample each such sphere to get more accurate (discrete) intersection points.

More precisely, if a precise collision point’s distance from the surfaces is to be smaller than  $\epsilon_2$ , we cover a given intersection sphere by  $s$  smaller spheres with diameter  $\epsilon_2$  and sample that volume by  $s \cdot \ln s + c \cdot s$  points so that each of the  $s$  spheres gets a point with high probability (see Appendix A). For each of these, we just determine the distance to both surfaces.

Rogers [18] showed that a sphere with radius  $a \cdot b$  can be covered by at most  $s = \lceil \sqrt[3]{a} \cdot a \rceil^3$  smaller spheres of radius  $b$ . Since we would like to cover the intersection sphere by spheres with radius  $b = \epsilon_2/2$ , we have to choose  $a = 2r/\epsilon_2$ , so that  $a \cdot b = r$ . As a consequence,

$$s = \lceil \sqrt[3]{2r} \cdot \frac{2r}{\epsilon_2} \rceil^3.$$

For example, if we would like to cover an intersection sphere with spheres of radius  $\epsilon$ , then  $\epsilon_2 = 2 \cdot \epsilon$  and  $s = \lceil \sqrt[3]{2r} \cdot r/\epsilon \rceil^3$ .

## 4.6 Complexity Considerations

In this section we analyze the runtime of our novel approach and the number of evaluations of the implicit function that are necessary to detect all intersections for a given sampling density described by the number  $a$  of surfels.

In general, evaluating  $f(x)$  takes  $O(\log N)$  time, even if the support of the kernel is bounded, because the NN of  $x$  has to be determined (using, for instance, a  $k$ D-tree). Here, fortunately, one evaluation can be done in

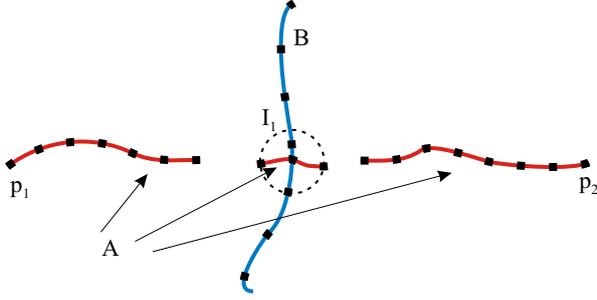


Figure 5: Models with boundaries can cause errors ( $I_1$  could remain undetected), which can be avoided by “virtual” edges in the proximity graph.

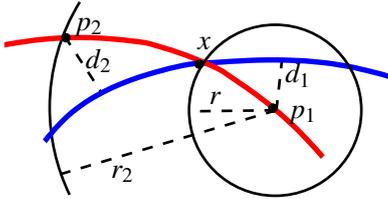


Figure 6: An intersection sphere centered at an AIP  $p_i$ . Its radius  $r$  can be determined approximately with the help of a second point on the other side of  $B$ .

only  $O(1)$  time: the root bracketing and interpolation search evaluate  $f(x)$  only at points  $x \in A \cup B$ , and computing the precise intersection points can use a brute force NN search in constant time, starting from the AIP.

Our root bracketing algorithm looks  $O(a \ln a)$  times for a pair  $(p_i, p_j)$  of points lying on different sides. Each time,  $f(x)$  has to be evaluated only  $O(1)$  times, as the spherical neighborhood around  $p_i$  contains only a constant number of points. As a consequence,  $O(a \ln a)$  evaluations of  $f_B$  have to be performed which is also the overall runtime of our root bracketing algorithm.

Then, for at most  $O(a \ln a)$  many pairs, our interpolation search has to be started. Each single interpolation search needs  $O(\log \log m)$  evaluations of  $f_B$  where  $m$  denotes the number of points along the shortest path between  $p_i$  and  $p_j$ .

Overall, the  $f_B$  has to be evaluated at most  $O(a \ln a \cdot \log \log m)$  times. As  $N \gg m$  and  $a$  is constant, this number can also be bounded by  $O(\log \log N)$ .

## 5 RESULTS

We implemented our new algorithm in C++. As of yet, the implementation is not fully optimized. All results were obtained on a 2.8 GHz Pentium-IV.

For timing the performance, we used a set of objects (see Fig. 11), most of them with several resolutions. Benchmarking was performed by the procedure proposed in [23], which computes average collision detection times for a range of distances between two identical objects, which are scaled uniformly so that they fit into a cube of size  $2^3$ .

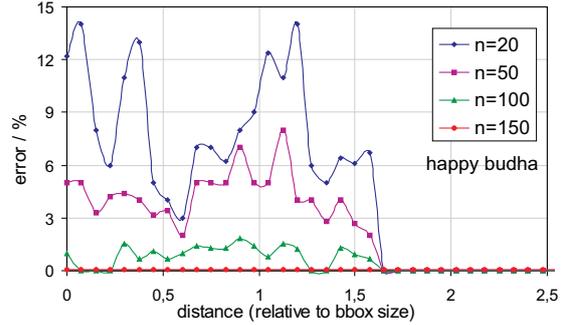


Figure 7: If the sampling density is too small, our approach can miss some intersections ( $n = O(a \ln a)$ , see Section 4.1).

### 5.1 Minimal Bracket Density

As mentioned before, if the number of (conceptual) surfels is too small, then the size of their neighborhoods can become too large, and, as a consequence, the likelihood can become too large that the normal  $n(x)$  flips its sign without  $x$  actually changing sides. In that case, our method could fail to find pairs of points on different sides of the surface.

Therefore, we propose to estimate the minimal number of surfels (which directly influences the radius of the spherical neighborhoods) by the following preprocessing procedure. For each distance, a large number of collisions tests is performed, each with a different constellation between the objects. A collisions test stops after the first intersection has been found. Each of these tests is performed with a different sampling density, expressed by the number  $n = O(a \ln a)$  (see Section 4.1). Then, we use the minimal sampling density for which all collisions have been found.

The results for one object can be found in Figure 7, which shows the error rate depending on different sampling densities. All our other models of our test suite show a similar behavior and it turned out that  $n_{\min} = 200$  is the minimal number, so that the error rate of all intersection tests for all our models is only 0.1%. This number was used for all further tests.

### 5.2 Interpolation Search vs Randomized Sampling

In order to evaluate the performance of our new algorithm, we compared it to the simpler randomized sampling technique (RST) proposed in [11]. No BV hierarchies were used.

The number of sample points  $n_s$  that have to be generated for the RST can be determined as proposed in Section 4.5, depending on the same  $\epsilon$  that is used for our new approach. As this number would always be large, we once again terminate both collision detection algorithms after the first intersection is found.

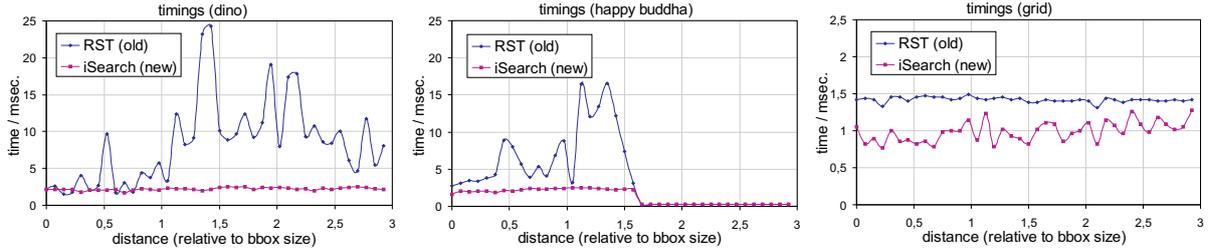


Figure 8: Timings for different models. Comparison of our novel technique and RST [11].

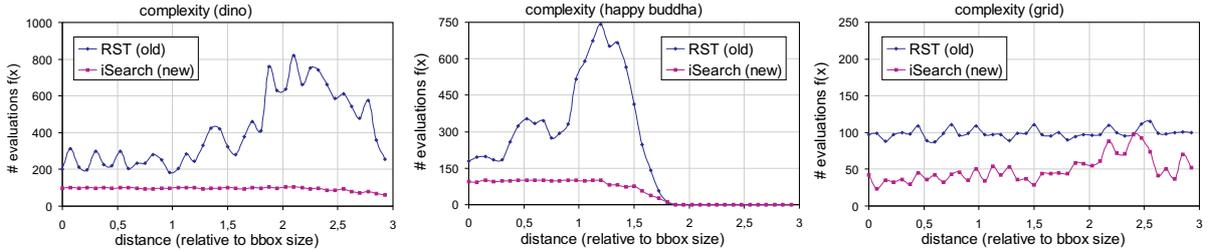


Figure 9: The number of evaluations of  $f(x)$  can be decreased by an order of magnitude by our new approach.

However, in the case of non-collision, in particular in the case of small distances between the objects, the runtime of the RST would be very long because of the large  $n_s$ , which is a big drawback of the old method. Therefore, if  $n_s$  is too large, we bound this number by 500. Note that in such cases the old method fails to report all intersection tests correctly, in contrast to our new method, which is another drawback of the old method.

Figure 8 shows that the collision queries can be answered much more quickly by our new approach.

The corresponding number of evaluations of the implicit function can be found in Figure 9. Note that the number of evaluations can exceed  $n_s$  in the case of the RST, since for each random point two evaluations are necessary.

### 5.3 Timings depending on Point Density

Figure 10 shows the runtime for detecting *all* intersections between two objects, depending on different densities of the point clouds. We define the density of an object  $A$  with  $N$  points as the ratio of  $N$  over the number of volume units of the AABB of  $A$  (which is at most 8 as each object is scaled uniformly so that it fits into a cube of size  $2^3$ ). This experiment supports our theoretical considerations of Section 4.6.

Note that the CPSP maps (see Section 3.2) were built so that the time for evaluating the implicit function remains constant.

We also measured the time that would be needed to compute all nodes on the shortest path between  $(p_i, p_j)$  used to initialize the interpolation search (see Algorithm 2). For all our models, this was at most 10% of the overall runtime. Therefore, one can save a significant amount of memory in the CPSP map by computing array  $P$  in Algorithm 1 during run-time.

## 6 CONCLUSION AND FUTURE WORK

We have presented a novel algorithm for sampling the intersection curves between surfaces defined implicitly by point clouds with the weighted least-squares method plus proximity graph. It can be used, for instance, to accelerate hierarchical collision detection or Boolean operations on this kind of object representation.

Our approach exploits the proximity graph by interpolation search along shortest paths in the graph. The technique of randomized sampling has proven to be efficient for initializing that search.

Our measurements show that the number of function evaluations is reduced by an order of magnitude and a speedup of factor 5–10 is achieved in many cases, compared to a previous randomized sampling technique.

Moreover, theoretical and experimental evidence is given that the runtime grows only as  $\log \log N$ , ( $N$  = the size of the point clouds).

We believe that this work opens up a number of further avenues for future work. Our new approach could be a way to handle *deformable* point clouds, since it does not utilize any spatial acceleration structure and the SIG can be updated in time  $O(\log^3 N)$ . From a theoretical point of view, a mathematically more rigorous estimation of the minimal sampling density would be appealing.

## ACKNOWLEDGEMENTS

This work was partially supported by DFG grant DA155/29-1 “Benutzerunterstützte Analyse von Materialflußsimulationen in virtuellen Umgebungen” (BAMSI), and the DFG program “Aktionsplan Informatik” by grant ZA292/1-1.

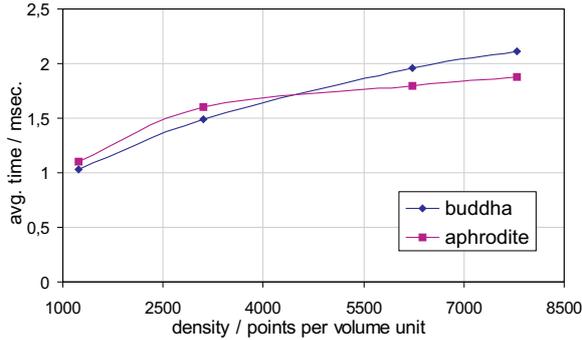


Figure 10: The plot shows the runtime depending on the size of the point clouds. The runtime is the average of all timings for distances between 0 and 1.5.

## A PROOF OF LEMMA 1

We can reduce the problem to a simple urn model. Given  $a$  bins (corresponding to the number of surfels), how many balls (corresponding to the number of points to be drawn) have to be thrown i.i.d. into the bins so that every bin gets at least one ball with high probability?

Let  $X$  denote the number of drawings required to put at least one ball into each bin. It is well known that the expectation value of  $X$  is  $a \cdot H_a$  where  $H_a$  is the  $a$ -th harmonic number [15, p. 57f].

Let  $c$  be an arbitrary constant. The  $a$ -th harmonic number is about  $\ln a \pm 1$  which is asymptotically sharp, and so  $c \cdot a$  additional balls are enough to fill each bin with probability  $p$  which depends on  $c$ . Therefore,  $n = a \cdot \ln a + c \cdot a$  points  $\in \text{Vol}(A \cap B)$  have to be generated.

To compute the dependence of  $p$  on  $c$ , we refer to the proof given by Motwani and Raghavan [15, p. 61ff]. They showed that the probability  $p = \Pr[X \leq n] = e^{-e^{-c}}$  for a sufficiently large number of bins.

## REFERENCES

- [1] Bart Adams and Philip Dutré. Interactive boolean operations on surfel-bounded solids. In *Proc. of SIGGRAPH*, volume 22, pages 651–656, July 2003.
- [2] Anders Adamson and Marc Alexa. Approximating and intersecting surfaces from points. In *Proc. Eurographics Symp. on Geometry Processing*, pages 230–239, Aachen, Germany, June 23–25 2003.
- [3] Anders Adamson and Marc Alexa. Approximating bounded, non-orientable surfaces from points. In *Shape Modeling International*, pages 243–252, 2004.
- [4] Anders Adamson and Marc Alexa. On normals and projection operators for surfaces defined by point sets. In *Eurographics Symp. on Point-Based Graphics*, pages 149–155, 2004.
- [5] Nina Amenta and Yong Kil. Defining point-set surfaces. In *Proc. of SIGGRAPH*, pages 264–270, 2004.
- [6] Elizabeth D. Boyer, L. Lister, and B. Shader. Sphere-of-influence graphs using the sup-norm. *Mathematical and Computer Modelling*, 32(10):1071–1082, 2000.
- [7] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

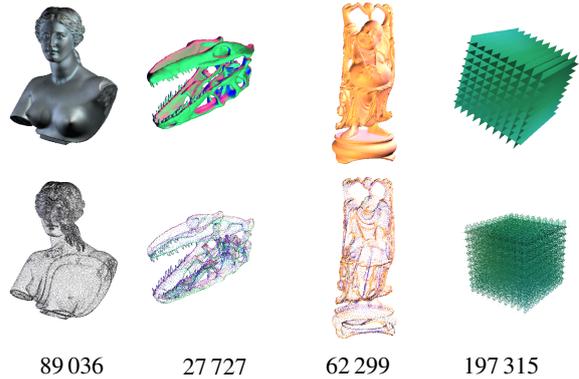


Figure 11: Some of the models of our test suite (courtesy of Polygon Tech. Ltd and Stanford). The numbers are the sizes of the respective point clouds.

- [8] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proc. of SIGGRAPH*, pages 71–78, 1992.
- [9] J. W. Jaromczyk and Godfried T. Toussaint. Relative neighborhood graphs and their relatives. In *Proc. of the IEEE*, volume 80, pages 1502–1571, 1992.
- [10] Richard Keiser, Matthias Mueller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. Contact handling for deformable point-based objects. In *Vision, Modeling, Visualization (VMV)*, pages 315–322, Stanford, USA, November 16–18 2004.
- [11] Jan Klein and Gabriel Zachmann. Point cloud collision detection. In *Computer Graphics Forum (Proc. of EUROGRAPHICS 2004)*, pages 567–576, 30 August - 3 September 2004.
- [12] Jan Klein and Gabriel Zachmann. Proximity graphs for defining surfaces over point clouds. In *Eurographics Symposium on Point-Based Graphics (SPBG'04)*, pages 131–138, June 2004.
- [13] D. H. McLain. Drawing contours from arbitrary data points. *Computer Journal*, 17(4):318–324, 1974.
- [14] T. S. Michael and Thomas Quint. Sphere of influence graphs and the  $l_\infty$ -metric. *Discrete Applied Mathematics*, 127(3):447–460, 2003.
- [15] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- [16] Hanspeter Pfister, Jeroen van Baar, Matthias Zwicker, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Proc. of SIGGRAPH*, pages 335–342, 2000.
- [17] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, England, 2nd edition, 1993.
- [18] C.A. Rogers. Covering a sphere with spheres. *Mathematika*, 10:157–164, 1963.
- [19] Szymon Rusinkiewicz and Marc Levoy. QSPat: A multiresolution point rendering system for large meshes. In *Proc. of SIGGRAPH*, pages 343–352, 2000.
- [20] Robert Sedgewick. *Algorithms*. Addison-Wesley, 1989.
- [21] Sotoshi Tanaka, Yasushi Fukuda, and Hiroaki Yamamoto. Stochastic algorithm for detecting intersection of implicit surfaces. *Computers and Graphics*, 24(4):523–528, 2000.
- [22] T. Lewis V. Barnett. *Outliers in Statistical Data*. John Wiley and Sons, New York, 1994.
- [23] Gabriel Zachmann. Minimal hierarchical collision detection. In *Proc. ACM Symp. on Virtual Reality Software and Technology (VRST)*, pages 121–128, Hong Kong, China, November 2002.

# Achieving Consistency in a Combined IK/FK Interface for a Seven Degree-of-Freedom Kinematic Chain

John McDonald, Rosalee Wolfe, Karen Alkoby, Jacek Brzezinski, Roymieco Carter,  
Mary Jo Davidson, Jacob Furst, Damien Hinkle, Bret Kroll, Glenn Lancaster,  
Lori Smallwood, Jorge Toro, Nedjla Ougouag, Jerry Schnepf

School of CTI, DePaul University  
{[jmcdonald@cs.depaul.edu](mailto:jmcdonald@cs.depaul.edu) | [asl@cs.depaul.edu](mailto:asl@cs.depaul.edu)}

## ABSTRACT

Many applications in computer animation portray the motion of a human arm and torso. Often such applications can benefit from a combination of Inverse (IK) and Forward (FK) Kinematics controls to manipulate the arms of the model. The human arm is a kinematic chain with seven degrees of freedom. The previous analytic solution to this kinematic chain gives highly detailed IK controls, but problems arise when integrating it with FK controls. These problems impede the artistic process when creating expressive animations.

This work improves on the previous analytic solution to create a hybrid FK/IK control interface for manipulating the chain, and enables the recalculation of all the parameters necessary for the IK solution. Thus IK and FK controls can interact seamlessly to manipulate the arm. The torso is modeled as a separate kinematic chain, and is integrated with the arm linkages. User tests demonstrate the effectiveness and efficiency of the combined FK/IK control interface.

## Keywords

Character Animation, Expressive Animation, Inverse Kinematics Forward Kinematics

## 1. Introduction

Many applications in computer animation portray the actions of a virtual human model. Such models are required to perform an incredible array of tasks, from manipulating objects in a virtual world to conveying emotional context and meaning. Much of the expressiveness of virtual actors is conveyed through the torso and arms. Therefore the control interface for the arms and torso is a key component of any system for animating a human model.

The flowing motions of the human arm and torso require a complex manipulation of a multitude of joints including the wrist, shoulder, elbow, collar, and spinal articulations [Van98]. To model these joints, computer graphics applications often use kinematic chains such as those found in robotic link-

ages, and rely on a variety of methods to control these joints to achieve the desired positions [Gir85] [Bad93] [Kog94] [Kon94] [Mur94].

Two general categories of controls exist for such kinematic chains, depending on the type of information they use as input:

1. *Forward Kinematics (FK) Controls.* These controls specify input data consisting of a collection of angles for the joints in the kinematic chain. The final orientation of any segment can be computed by multiplying the transformations in the joints.
2. *Inverse Kinematics (IK) Controls.* This class of control identifies an *end effector* in the chain, and a target at which the end effector should be positioned. Analytic or iterative IK solutions require the computation of the joint angles necessary to position the end effector at the target subject to various constraints, which restrict the space of possible solutions [Gir85].

The choice of control depends on the application. For example, in ergonomics simulations, the arms and torso need to be positioned so that the hands of the model can manipulate some object in space. These applications are most conveniently modeled using inverse kinematics [Bad93] [Tol00].

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 2005 conference proceedings, ISBN 80-903100-7-9  
WSCG'2005, January 31-February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency – Science Press

However, when positioning the body for expressive postures or narrative pantomime, the precise positioning of an end effector is not as important as the movements of the joints, which must look natural. In these kinds of applications, changes in the orientation of intermediate joints in the chain can dramatically alter the expression or emotion displayed by the figure. Animators must have complete control over all the degrees of freedom available in a model. Traditionally, for these types of applications forward kinematics has been the control of choice [Mae96] [Jon00].

## 2. Hybrid Interfaces

Many applications can benefit from a combination of these two types of controls. Character animation is a case in point. Consider a figure peering through a window with her fingertips touching the glass. If the character is curious or frightened, her posture will draw her elbows in towards her body, whereas if she is angry, her elbows will tend to flare out. Both postures place the character's fingertips on the glass in front of her. Therefore the animator would want IK control over the position of the fingertips, and direct control over the orientations of the intermediate joints.

To further illustrate the different kinds of controls required in expressive character animation, consider the following two examples:

1. Moving the arm so that a finger on one hand is placed in contact with the body, the face or the other hand, as in Figure 1A. An IK control easily achieves this contact.
2. Moving the hand away from the body in a natural arcing motion, as in Figure 1B. An FK control easily achieves this effect.

Forward kinematics has the advantage of being a straightforward process of building an object hierarchy. Rotating a single joint, as in Figure 1B, will move the joint and all of its children in a natural arcing motion. However, altering multiple joint angles causes the end effector to follow complicated arcs. While generally useful for positioning the body expressively, forward kinematics is cumbersome when used for placing an end effector at a specific location.

At the same time, these natural arcs may be precisely what the animator wants, but these arcs can be difficult to describe to an IK engine. If an IK rig animated the motion between the endpoints in Figure 1B, it would move the tip of the pinky along a straight line, and would introduce unwanted movement at the elbow. While one could describe this arc by a spline, the result is more cumbersome, and can involve more extraneous joint movement than simply letting FK have its way. On the other hand, when animating linear motions, as when the hands trace the

rectangular shape of an object, an IK solution is far more effective.

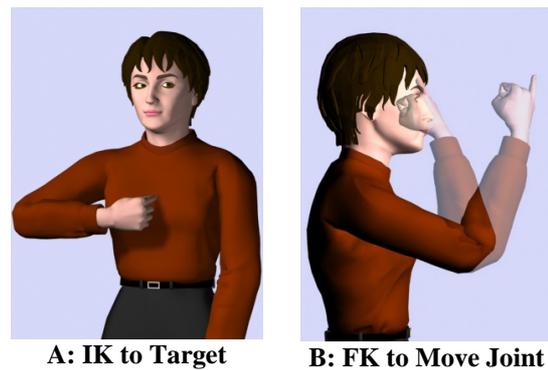


Figure 1: IK and FK Applications

For these reasons, a system for expressive animation must provide an integrated set of FK and IK controls to support both kinds of movements. Therefore, it must conform to the following criteria:

1. Provides IK controls for the arms, which allow the specification of a target for any point on the end effector.
2. Provides the animator direct control over all redundant degrees of freedom, while preserving end effector/target contact.
3. Provides FK controls for direct movement of each joint.
4. Provides the seamless integration of these FK and IK controls so that they may be used in any combination.

## 3. Available Techniques

Most comprehensive IK systems allow an animator to break out of IK control to manipulate the model with FK [Mae96]. However, the kind of control provided does not always conform to the criteria outlined in the last section.

Building a hybrid interface suitable for such applications requires the consideration of the available IK techniques in relationship to these four criteria. This includes investigating how well the IK solution provides for control over all the available redundant degrees of freedom and how well it integrates with FK controls.

### Inverse Kinematics

IK techniques fall into two categories: analytic and iterative solutions. A great deal of work has been done in developing both types of solutions for the kinematic chains in the arm and torso [Kon94] [McD00] [McD02] [Tol00].

Iterative IK solutions work from a given configuration of the model and incrementally move the end effector towards a given target. Such systems will generally try to calculate a set of “best” angles for the redundant degrees of freedom in the linkage. How-

ever, the criteria from Section 2 require that these angles be under direct control of the animator, allowing interactive exploration of the entire collection of postures that fix the end effector [McD02].

Another problem arises because certain targets on the body or in space will correspond to singular configurations for the iterative solution. When the joints approach such configurations the model will behave in a chaotic manner. Artists are quite familiar with the unfortunate choices that an iterative IK solution can make, as the final configuration of the model is not uniquely determined by the position of the end effector [Tol00]. While implementations reduce these effects using range limiting and optimization techniques [Gir85] [Mur94] [Zha94], analytic solutions, when available, are preferable [McD02] [Tol00].

Several analytic solutions have been investigated for different types of kinematic chains. The first analytic solutions were developed for robotic linkages [Mur94] and do not model the complete expressiveness of the human arm and torso.

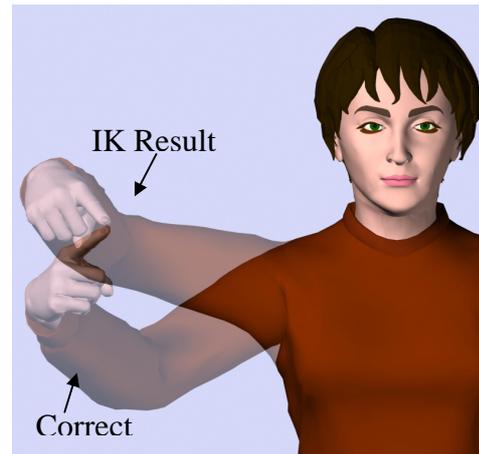
Most of the analytic solutions for the human arm in modern software work with linkages of just two bones, leaving the third segment, the hand, to be manipulated by FK or to be specified as a global orientation in space [Kon94] [Tol00]. This type of linkage has one redundant degree of freedom in the two joints of the linkage itself. This degree of freedom allows the user to raise and lower the model's elbow about the axis through the shoulder and wrist.

Such analytic solutions do not, however, take the full linkage of the human arm including the wrist into account in the solution itself. The full linkage has four redundant degrees of freedom. This case was considered in [McD02] where an analytic IK solution was presented for the entire seven degrees of freedom in the human arm.

[McD02] gives the animator direct control over the four redundant degrees of freedom in the linkage. The wrist orientation and the elbow elevation can both be altered while preserving end effector/target contact. Thus the solution satisfies three of the four criteria for an IK control outlined in Section 2.

Unfortunately problems arise when integrating this method with forward kinematic controls for the arm. Such integration was not considered in [McD02]. Forward kinematic manipulation of the shoulder, elbow, or wrist will cause the IK parameters to change. The system must recompute these parameters if the interface for manipulating the model is to work seamlessly. If it does not, and the user subsequently manipulates the IK controls, the model's arm will jump discontinuously as the recorded IK parameters reassert themselves.

For example, consider choosing the tip of the index finger with a bent wrist. Choose a target for the fingertip out in front of the body. Then straighten the elbow with an FK control. This moves the fingertip, and so the system records its new position. Then reapply the method from [McD02] using this new position as a target. Since the solution does not need to move the fingertip, it should not move the model at all. Instead, the position of the elbow jumps. Figure 2 shows the disparity between the resulting position and the correct one.



**Figure 2: Discontinuity Caused by [McD02]**

Such recalculations must always be considered when integrating an IK solution with FK. An advantage of iterative solutions is that their only input is the end effector's position, which can be easily recalculated from the model's joints. But, as described above, the unpredictable nature of iterative solutions makes them unsuitable for expressive animation.

The analytic solutions such as those outlined in [Tol00] and those found in many commercial packages must also recalculate all of the IK parameters if they are to be integrated with FK, and many do. However, they only work with the shorter two-segment IK chains, which do not extend to include the wrist's rotation in the IK solution itself.

#### 4. Extending the IK Solution

To see why problems arise in the current techniques, we must carefully investigate the method presented in [McD02].

##### The Previous Analytic IK Solution for the Arm

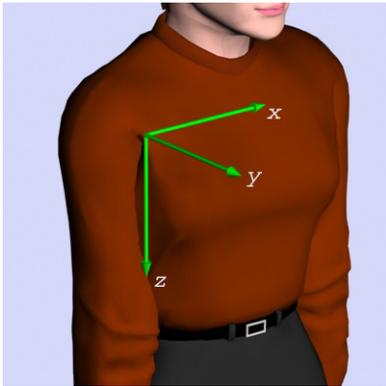
The human arm has three main joints: the wrist, elbow and shoulder. Together, these joints have a total of seven degrees of freedom. A discussion of these joints can be found in [McD02], where the following solution to the kinematic chain was developed. The algorithm requires the following input:

1. The position  $A$ , on the model's hand, called an *articulator*, to be used for targeting

2. The target point  $T$  in space or on the body
3. The local orientation of the model's wrist including the radial twist of the forearm
4. The elevation angle  $\delta$  for the elbow, which raises and lowers the elbow while keeping the chosen articulator fixed in space.

The solution then computes the remaining angles necessary to position the arm so as to place the articulator  $A$  at the chosen target  $T$ . Note that if the articulator is chosen at the wrist, then the system degenerates to the previous two-segment cases such as in [Tol00] and those implemented in several commercial packages.

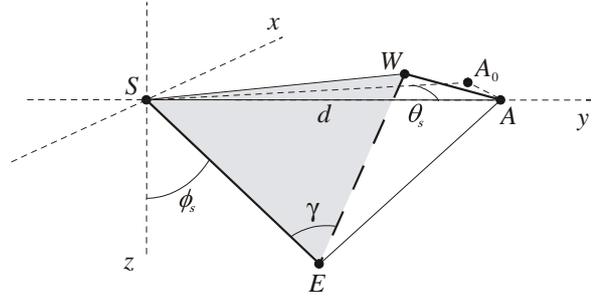
The method assumes that a coordinate system has been chosen with its origin at the model's shoulder, and with the  $z$ -coordinate axis corresponding to the primary vertical axis of the body but pointed downwards, i.e. running parallel to the line from the neck through the hips. The  $y$ -axis will protrude straight out of the body perpendicular to the plane formed by the two shoulders and the hips. See Figure 3.



**Figure 3: The Shoulder Coordinate System**

With this setup, the method calculates the transforms for both the elbow and the shoulder. Let  $S$  be the position of the shoulder in space and let  $d = \text{dist}(S, T)$  be the distance from the shoulder to the desired target. The solution is calculated in four steps:

1. Calculate the bend angle  $\gamma$  of the elbow, as shown in Figure 4. With this, the articulator will lie at a distance of  $d$  from the shoulder.
2. Position the articulator on the  $y$ -axis in front of the body by calculating two angles in a spherical coordinate system  $\phi_s$  and  $\theta_s$  for the shoulder, as shown in Figure 4.
3. Rotate the shoulder by the elevation angle  $\delta$  about the  $y$ -axis, which is currently the axis through the shoulder and the articulator.
4. Use the spherical coordinates of the articulator's target to calculate two more spherical angles  $\phi_a$  and  $\theta_a$  for the shoulder, which will rotate the articulator from the  $y$ -axis to the chosen target.



**Figure 4: The Spherical Angles of the Upper Arm in the Default Position**

Thus, the method decomposes the shoulder rotation into five angles,  $\phi_s$ ,  $\theta_s$ ,  $\delta$ ,  $\phi_a$ , and  $\theta_a$  about the axes  $x$ ,  $z$ ,  $y$ ,  $x$  and  $z$  respectively. The final transformation of the shoulder is computed as the product

$$M_s = R_D R_{\phi_s} R_{\theta_s} R_{\delta} R_{\phi_a} R_{\theta_a}$$

where  $R_D$  is the default rotation of the shoulder. The benefits of this parameterization were detailed in [McD02].

#### Effects of FK Controls

The method presented in [McD02] alone is sufficient if the only interface given to the user is:

1. The IK controls for the position of the articulator
2. The elbow elevation angle
3. The local orientation of the wrist.

It is important to note that wrist orientation, while under direct control, results in a call to the IK system so that the articulator position remains fixed as demonstrated in the left image in Figure 6 of Appendix A. This is distinctly different from the desired FK control for the wrist, which would move the articulator as in the left image of Figure 7.

In addition to these IK controls, we wish to allow the user to rotate the elbow, wrist and shoulder through FK. We will work with the underlying IK parameters as our basis since the IK solution will still be the primary means for controlling the model. Any forward kinematic moves made later will be recast in terms of the IK parameters so that if the user then moves back to the IK controls, the model will not jump as the IK system reasserts itself. This is the step that is missing from the previous method and which will satisfy the final criterion from section two.

Consider the effect of moving the model's arm with the above analytic IK method, followed by an adjustment of one of the joints with an FK control. *All* of the IK parameters will change:

1. The position of the articulator will change. This is not a problem since we can simply recompute

the articulator's position from the model and set the IK target to the new position.

2. The elbow bend angle and wrist orientation might change as the result of an FK move. Again, we can read their new values straight out of the FK data.
3. The elevation angle  $\delta$  will change. This can *not* be directly read from the FK data.

Certainly, the elevation angle will change if the shoulder is rotated, but even if the shoulder does not move, as when the wrist is flexed,  $\delta$  will change. Recall that this analytic solution decomposes the shoulder into five angles: two pairs of spherical angles and the elevation angle. There are many subtle interactions that can occur in the shoulder to change these angles, including  $\delta$ .

Suppose that the user has chosen an articulator on the tip of the index finger, and uses the IK solution to place the tip of the finger at a point out in space in front of the body. What happens if the user subsequently flexes the model's wrist via FK? The articulator target  $A$  will change position. Call the new target  $A'$ . This causes the spherical angles of the target,  $\phi_a$  and  $\theta_a$ , to change to match the new location.

Note that flexing the wrist does not actually change the orientation of the shoulder, and so the total shoulder transformation  $M_s$  remains constant. Since the default rotation of the shoulder has not changed, one or more of the other three angles  $\phi_s$ ,  $\theta_s$  and  $\delta$  have changed to compensate. In fact, most often the change in  $\phi_a$  and  $\theta_a$  will be compensated by a change in all three of the other angles.

In order to integrate FK and IK controls seamlessly for this method, we need to recompute all five of the angles in the shoulder transformation's decomposition,  $\phi_s$ ,  $\theta_s$ ,  $\delta$ ,  $\phi_a$ , and  $\theta_a$ . All five must be recalculated each time the user moves the model using the FK control. This will complete the recalculation of the IK parameters in this analytic method and will correct the jumping problem seen in Figure 2.

### Extending the Method: Recomputing the Angles

To recompute the five angles in the IK solution, we need to take a close look at the decomposition of the shoulder transformation. Suppose that the user has made a sequence of FK changes to the arm's joint rotations. These FK moves on the model result in a new local transformation  $M'$  at the shoulder.

We know from the development of this analytic IK solution that whatever this matrix is, it can be decomposed into a product of six rotations:

$$M' = R_D R_{\phi_s} R_{\theta_s} R_{\delta} R_{\phi_a} R_{\theta_a}$$

The first transformation in this product is the default rotation for the shoulder, which is a constant and is

therefore known. It remains to recompute the other five angles in the decomposition.

On the right side of the chain, we have the rotations for the two spherical angles corresponding to the new articulator position  $A' = (x', y', z')$ . Therefore, these angles may be calculated from the coordinates of  $A'$ .

$$\theta_a' = \arctan\left(\frac{-x'}{y'}\right)$$

$$\phi_a' = \frac{\pi}{2} - \arccos\left(\frac{z'}{\sqrt{(x')^2 + (y')^2 + (z')^2}}\right)$$

The differences in these formulas from their normal spherical coordinate presentation are due to the fact that the spherical coordinates of the articulator's target are measured from the  $y$ -axis.

Next, the first stage of the original analytic solution can be run to find the values of  $\phi_s'$  and  $\theta_s'$ . See Figure 4. Determining these angles is accomplished by first calculating the projection of  $A$  to the plane  $SEW$ . Call this projection  $A_0$ . Then  $\phi_s'$  and  $\theta_s'$  can be calculated as

$$\phi_s' = \pi/2 - \arccos\left(\frac{SE \cdot SA_0}{|SE||SA_0|}\right)$$

$$\theta_s' = \arccos\left(\frac{SA \cdot SA_0}{|SA||SA_0|}\right)$$

These calculations are most conveniently accomplished in the elbow's coordinate system, in which  $A'$  and  $W$  have a known representation in terms of the initial input data.

Finally, we come to the elevation angle  $\delta'$ . While we could try to use elementary trigonometry to calculate the new elevation angle, the decomposition of  $M'$  admits a more elegant solution with fewer special cases. Since each of the transformations in the decomposition is a rotation and is therefore invertible, the rotation matrix for the elevation angle can be calculated from the decomposition

$$M' = R_D R_{\phi_s} R_{\theta_s} R_{\delta} R_{\phi_a} R_{\theta_a}$$

by multiplying both sides of the equation by the inverses of the rotations for  $\phi_s$ ,  $\theta_s$ ,  $\phi_a$ ,  $\theta_a$  and the default rotation. Thus,

$$R_{\delta'} = (R_{\theta_a'})^{-1} (R_{\phi_a'})^{-1} (R_D)^{-1} M' (R_{\theta_s'})^{-1} (R_{\phi_s'})^{-1}$$

This gives us the transformation as a matrix. To calculate the elevation angle for the elbow, we use a surprising fact arising from the construction of the analytic solution.  $R_{\delta'}$  is a rotation about the  $y$ -axis, and will therefore be of the form

$$R_{\delta'} = \begin{pmatrix} \cos \delta' & 0 & \sin \delta' \\ 0 & 1 & 0 \\ -\sin \delta' & 0 & \cos \delta' \end{pmatrix}$$

Therefore, we can find  $\delta'$  by taking

$$\delta' = \text{sgn}\left((R_{\delta'})_{1,3}\right) \arccos\left((R_{\delta'})_{1,1}\right)$$

This completes the recalculation of the IK parameters after a forward kinematic move. Applying this recalculation finishes the computation of the five IK parameters.

## 5. The Integration with the Torso

The completion of the control interface requires the integration of the arm's IK chain with the linkages for the torso and back. The FK and IK controls should continue to work seamlessly as the animator manipulates the model's torso.

The interface uses a simplified model of the torso consisting of a sequence of three evenly spaced joints starting at the hips, and having three degrees of freedom, corresponding to flexion, abduction and radial twist about the local tangent to the spinal column.

The sterno-clavicular joint in the collar is modeled as a rotational joint at the top of the spine and articulating with the shoulder joint. It has three degrees of freedom, a rotation that moves the shoulder forwards and backwards, a rotation that moves the shoulder up and down, and a small amount of radial twist. Though simplified, this model is capable of representing many movements of the shoulder and torso.

This method is also compatible with more realistic, physically based models, as described in [Mau00] and [Sha03]. Other torso and neck models such as [Nef04], which use IK methods incorporating balance control into the torso, are also compatible with this method.

The first step of the new IK solution calculates the coordinates of the target relative to this hierarchy, i.e. relative to the arm's rest orientation, which has been rotated by the torso and collar joints. Figure 5 displays two examples of the added expressivity enabled by integrating the FK/IK system with the torso.

## 6. The Interface

We implemented the FK and IK controls enabled by these techniques in a custom software package allowing animators to control a human model interactively. The integrated FK/IK interface adds FK controls for the wrist, elbow, shoulder and torso to the IK controls. Figures 6 and 7 in Appendix A show the effects of the wrist and elbow controls in the combined FK/IK interface, emphasizing the differences in effect from the FK and IK controls for these joints.

The remaining shoulder and torso controls are slider controls accounting for all the degrees of freedom.



Figure 5: Torso Movement

## 7. Testing the Interface

A usability test evaluated the effectiveness of the integrated FK/IK interface for positioning a model's arms. The test compared the integrated interface with the controls found in the most recent version of a widely used, commercially-available animation package, which allowed the subjects to use either FK, a traditional IK solution, or a combination of both to move the model.

The test participants had previous experience with the commercial package ranging from six months to three years. While three of the participants also had substantial experience in using the integrated FK/IK interface, three of them had minimal prior exposure. See Table 1 in Appendix B.

The participants created two versions of three American Sign Language (ASL) signs: FOOD, IDEA and CLOTHES. Each sign contained successively more complicated motion. The first, FOOD, required a small arcing of the forearm and a localized oscillation of the wrist along a single axis. The second, IDEA, required a twisting and arcing motion of the forearm. The last sign, CLOTHES, required a circling motion of the wrists as well as a twisting and arcing motion of the forearms. For a background on ASL animation see [McD00] and [Wol99]. Each participant created one version of the sign using the commercial package, and another using the integrated FK/IK interface, yielding a 3x2 experimental design.

As a guide for creating the animations, participants received videotaped demonstrations of each sign recorded from side and front views. For each sign, participants received a model whose fingers were already in the correct position. They began the task of creating the animation from that point. Only the time required to position the arms was recorded.

Each participant completed the two versions of a sign, and then proceeded to complete two versions of the next sign. To control for transfer of learning, the

order in which they used the commercial software and the integrated FK/IK interface was randomized.

After the participants completed the animations, a team of two animators familiar with ASL critiqued the results jointly. This team had access to both the completed animations and the reference video footage. They examined the accuracy of the arm placement. The reviewers were unaware of which software was used to create each animation.

Table 2 and Table 3 of Appendix B display the results. For the simplest sign, FOOD, the average completion times were the same. However, for the more complex signs, participants required less time when using the integrated FK/IK interface. For the sign IDEA, participants using the integrated interface required only 57 percent of the time that they needed to complete the same sign using the commercial software. This percentage dropped even further with the sign CLOTHES. When using the integrated FK/IK interface, participants required less than 50 percent of the time they needed when using the commercial software.

While the completion times examined the efficiency of the new interface, the reviewer's critique evaluated its effectiveness. Table 3 in Appendix B shows the results. The preferred versions of each animation are listed for each test participant. When both versions were judged to be of equal quality, the word "tie" appears.

For all three signs, the new integrated interface produced better results, but it is particularly striking for the most complex sign, CLOTHES. For all six participants, the preferred animation was the one created with the new FK/IK interface. Since five of the six participants were more familiar with the commercial package, this clearly demonstrates the advantages of the new approach.

## 8. Conclusion and Next Steps

The techniques in this paper extend the results from [McD02] to build an integrated FK/IK control interface for a three segment kinematic chain with seven degrees of freedom, such as the human arm. These controls allow animators to create expressive motions in less time than with previous FK/IK techniques.

Next steps include the incorporation of a system of joint correlations similar to those described in [McD00] and [Sha03] to help animators coordinate the movements of various segments of the arms and torso. This would increase the efficiency of the interface. Also, while the controls for the arm are integrated into a simple model for the spine and shoulders, a more realistic model of the torso, collarbone and neck as in [Mau00] would further expand the expressiveness of the model.

We would also like to investigate the added expressiveness that could be gained from sophisticated balance-control methods such as the one in [Nef04].

## 9. Acknowledgements

The authors wish to express their thanks to Tahseen Basheeruddin, Ryan Burnett, Cynthia Dwyer, Bret Kroll, Jerry Schnepp and Prabhakar Srinivasan for their participation in the user test.

## 10. References

- [Bad93] Badler N, Phillips C, Webber B. *Simulating Humans: Computer Graphics Animation and Control*. Oxford University Press. New York, NY. 1993.
- [Gir85] Girard M, Maciejewski A. *Computational Modeling for the Computer Animation of Legged Figures*. *Computer Graphics*. 19(3), July 1985, Pages 253-270.
- [Jon00] Jones A, Bonney S. *3Dstudio Max 3 Professional Animation*. New Riders, Indianapolis, 2000.
- [Kog94] Koga Y, Kondo K, Kuffner J, and Latombe J. "Planning Motions with Intentions," *Proc., SIGGRAPH'94*, Orlando, FL, July 24-29, 1994. Pages 395-407.
- [Kon94] Kondo K. *Inverse Kinematics of a Human Arm*. Technical Report STAN-CSTR-94-1508, Dept. of Computer Science, Stanford University.
- [Mae96] Maestri G. *Digital Character Animation*. New Riders, Indianapolis, 1996.
- [Mau00] Maurel W, Thalmann D. *Human Shoulder Modeling Including Scapulo-Thoracic Constraint and Joint Sinus Cones*. *Computer & Graphics*, New York, Vol 24, No 2, p. 203-218, 2000.
- [McD00] McDonald J, Toro J, Alkoby K, Berthiaume A, Carter R, Chomwong P, Christopher J, Davidson MJ, Furst J, Konie B, Lancaster G, Roychoudhuri L, Sedgwick E, Tomuro N, Wolfe R. *An Improved Articulated Model of the Human Hand*. *Proceedings of the 8th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media*. 2000. Pages 306 – 313.
- [McD02] McDonald J, Alkoby K, Carter R, Christopher J, Davidson MJ, Ethridge D, Furst J, Hinkle D, Lancaster G, Smallwood L, Ougouag-Tiouririne N, Toro J, Xu S, Wolfe R. *A Direct Method for Positioning the Arms of a Human Model*. *Graphics Interface 2002*, Proceedings, 99-106, 2002.
- [Mur94] Murray R, Li Z, and Sastry S. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., 1994.
- [Nef04] Neff M, Fiume E. *Methods for Exploring Expressive Stance*. *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 2004, pp. 49-58.
- [Sha03] Shao, W, Ng-Thow-Hing, V. *A General Joint Component Framework for Realistic Articulation in Human Characters*. *Proceedings of the 2003 sym-*

posium on Interactive 3D graphics, 2003, pp 11-18.

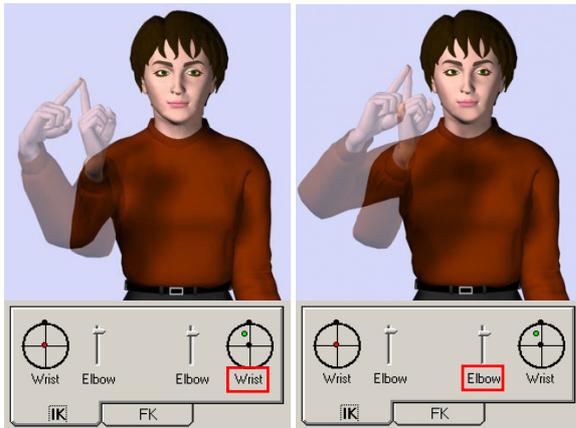
[Tol00] Tolani D, Goswami A, and Badler N. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical Models* 62 (5), September 2000, Pages 353-388.

[Van98] Van De Graaff, K, *Human Anatomy* 5th Edition. WEB McGraw-Hill. Boston, MA. 1998.

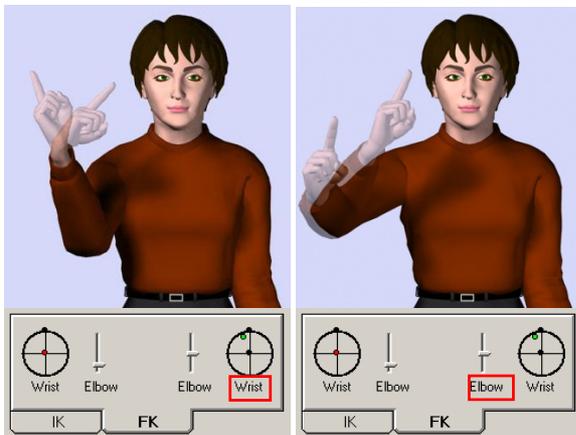
[Wol99] Wolfe R, Alkoby K, Berthiaume A, Chomwong P, Christopher J, Davidson MJ, J. Furst, Konie B, Lancaster G, Lytinen S, McDonald J, Roychoudhuri L, Sedgwick E, Tomuro N, Toro J. An Interface for Transcribing American Sign Language. SIGGRAPH 99 Sketches, August 11, 1999, Page 229.

[Zha94] Zhao J, Badler N, Inverse kinematics positioning using nonlinear programming for highly articulated figures, *ACM Transactions on Graphics*, Vol 13, No 4, October 1994, pp. 313-336.

**Appendix A: Interface Examples**



**Figure 6: The IK Interface**



**Figure 7: The FK Interface**

**Appendix B: Data from User Test**

| Participant | Commercial Package | Integrated FK/IK control |
|-------------|--------------------|--------------------------|
| 1           | 18 Months          | 1 Hour                   |
| 2           | 36 Months          | 18 Months                |
| 3           | 36 Months          | 1 Hour                   |
| 4           | 12 Months          | 12 Months                |
| 5           | 6 Months           | 2.5 Hours                |
| 6           | 36 Months          | 20 Hours                 |

**Table 1: Previous Experience of Test Participants**

| Sign    | Commercial package | Integrated FK/IK control |
|---------|--------------------|--------------------------|
| FOOD    | 20.2               | 20.7                     |
| IDEA    | 21.8               | 12.5                     |
| CLOTHES | 42.2               | 21.0                     |

**Table 2: Average Completion Times.**

| Participant | FOOD       | IDEA       | CLOTHES |
|-------------|------------|------------|---------|
| 1           | Commercial | Commercial | FK/IK   |
| 2           | FK/IK      | Commercial | FK/IK   |
| 3           | FK/IK      | Tie        | FK/IK   |
| 4           | FK/IK      | FK/IK      | FK/IK   |
| 5           | Tie        | FK/IK      | FK/IK   |
| 6           | FK/IK      | FK/IK      | FK/IK   |

**Table 3: Results of Animation Critique, Software Resulting in Preferred Animation**

# A Multi-Scale Singularity Bounding Volume Hierarchy

Kerawit Somchaipeng  
3D-Lab, School of Dentistry  
Dept. of Pediatric Dentistry  
University of Copenhagen  
Nørre Alle 20, DK-2200  
Copenhagen N, Denmark  
kerawit@lab3d.odont.ku.dk

Kenny Erleben  
Dept. of Computer Science  
University of Copenhagen  
Universitetsparken 1  
DK-2100, Copenhagen N  
Denmark  
kenny@diku.dk

Jon Sparring  
Dept. of Computer Science  
University of Copenhagen  
Universitetsparken 1  
DK-2100, Copenhagen N  
Denmark  
sparring@diku.dk

## ABSTRACT

A scale space approach is taken for building Bounding Volume Hierarchies (BVHs) for collision detection. A spherical bounding volume is generated at each node of the BVH using estimates of the mass distribution.

Traditional top-down methods approximate the surface of an object in a coarse to fine manner, by recursively increasing resolution by some factor, e.g. 2. The method presented in this article analyzes the mass distribution of a solid object using a well founded scale-space based on the Diffusion Equation: the Gaussian Scale-Space. In the Gaussian scale-space, the deep structure of extremal mass points is naturally binary, and the linking process is therefore simple.

The main contribution of this article is a novel approach for constructing BVHs using Multi-Scale Singularity Trees (MSSTs) for collision detection. The BVH-building algorithm extends the field with a new method based on volumetric shape rather than statistics of the surface geometry or geometrical constructs such as medial surfaces.

## Keywords

Collision Detection, Bounding Volume Hierarchies, Gaussian Scale Space

## 1 INTRODUCTION

In physics-based animation, collision detection often becomes the bottleneck, since a collision query needs to be performed in every simulation step in order to determine contacting and colliding objects. Animations can have many objects, all of which may have a complex geometry, such as polygonal soups of several thousands facets, and it is therefore a computationally heavy burden to perform collision detection especially for real-time interaction.

Bounding Volume Hierarchies (BVHs) are widely used in computer graphics, e.g. for ray tracing [GS87], and they are quite popular in animation (e.g. [BMF03] uses them for cloth animation), since they are applica-

ble of handling more general shapes than most feature-based and simplex-based algorithms, they tend to generate smaller hierarchies than spatial subdivision algorithms, and they offer a graceful degradation of objects, which is highly useful when accuracy is to be traded for performance. New performance improvements of BVHs is therefore of great practical and theoretical interest to the computer graphics and animation community.

The main contribution of this paper is a novel algorithm for bottom-up construction of spherical approximating BVHs. We prefer our hierarchies, firstly because they save memory, and therefore increases simulation performance, when compared to traditional BVH, and secondly because they are a direct implementation of the mass of objects rather than their boundary representation.

In this article we will restrain ourselves from the n-body problem and only consider narrow phase [Hub93] collision detection of solid non-deformable objects.

### 1.1 PREVIOUS WORK

There is a wealth of literature on collision detection, and many different approaches have been investigated. Spatial subdivision algorithms like Bi-

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 80-903100-7-9  
WSCG'2005, January 31-February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency-Science Press

nary Space-Partitioning (BSP) tree [Mel01], octree [TC96, GDO00, ES03], k-d trees and grids [GDO00, ES03], feature-based algorithms like polygonal intersection [MW88], Lin-Can [PML97], VClip [Mir98], SWIFT [EL01], recursive search methods [SL00], simplex-based such as GJK [GJK88, vdB01], generalized Voronoi diagrams [HKL<sup>+</sup>99], and signed distance maps [GBF03, BMF03, Hir02]. Finally there are algorithms based on BVHs such as ours.

BVHs have been around for a long time. Consequently there is a huge wealth of literature about BVHs. Most of the literature addresses homogeneous BVHs and top-down construction methods. A great variety of different types of bounding volumes have been reported: Spheres [Hub96, Pal95, DO00], axed aligned bounding boxes (AABBs) [Ber97, LAM01], oriented bounding boxes (OBBs) [GLM96, Got00], discrete orientation polytypes (k-DOPs) [KHM<sup>+</sup>98, Zac98], Quantized Orientation Slabs with Primary Orientations (QuOSPOs) [He99], Spherical shell [KPLM98], and swept sphere volumes (SSVs) [LGLM99]. In general, it has been discovered that there is a trade-off between the complexity of the geometry of a bounding volume and the speed of its overlap test and the number of overlap tests in a query.

In contrast to bounding volumes types, there has only been written little on approximating BVHs. To our knowledge [Hub93] pioneered the field, where octrees combined with simulated annealing were used to construct a sphere tree, followed by [PG95, Pal95], cumulating with a superior bottom-up construction method based on medial surface (M-reps) [Hub96]. More recently [OD99, DO00] used approximating sphere-trees built in a top down fashion based on an octree for time critical collision detection, and [BO04] used an adaptive m-rep approximation-based top-down construction algorithm.

There have been written even less about heterogeneous bounding volume hierarchies, although object hierarchies of different primitive volume types are a widely used concept in most of today's simulators [Ode, Vor, Kar]. The SSVs [LGLM99] are one of the most recent publications. The general belief is, however, that heterogeneous bounding volumes does not change the fundamental algorithms, but merely introduces a raft of other problems. It is also believed that heterogeneous bounding volumes could provide better and more tightly fitting bounding volumes resulting in higher convergence towards the true shape volume of the objects. This could mean an increase in the pruning capabilities and a corresponding increase in performance.

Most of the work with BVHs has addressed objects that are represented by polygonal models. Many experiments also indicate that OBBs (and other rectangular volumes) provide the best convergence for polygo-

nal models [GLM96, Got00, Zac98, LGLM99], while spherical volumes are believed to converge best towards the volume. The underlying query algorithms for penetration detection, separation distance and contact determination of BVHs have not changed much. In its basic form, these algorithms are nothing more than simple traversals.

To our knowledge, the trees based on the deep structure of Gaussian Scale-Space has not been used previously for generating BVHs in collision detection. An alternative to Gaussian scale-space is curvature scale-spaces, from which M-reps are derived. M-reps based methods are state of the art for bottom-up construction method [Hub96] and top-down construction [BO04]. For deformable objects such as cloth, bottom-up construction based on mesh topology [VM95, VMT00, BFA02] are the preferred choice. In [Ber97] a median based top-down method was proposed for building an AABB tree. [LAM01] suggested using a mesh connectivity-tree in a top-down construction method.

## 2 GAUSSIAN SCALE-SPACE

The  $N + 1$  dimensional Gaussian scale-space,  $L : \mathbb{R}^{N+1} \rightarrow \mathbb{R}$ , of an  $N$  dimensional image,  $I : \mathbb{R}^N \rightarrow \mathbb{R}$ , is an ordered stack of images, where each image is a blurred version of the former [Iij62, Wit83, Koe84]. The blurring is performed according to the diffusion equation,

$$\partial_t L = \nabla^2 L \quad , \quad (1)$$

where  $\partial_t L$  is the first partial-derivative of the image in the scale direction  $t$ , and  $\nabla^2$  is the Laplacian operator, which in 3 dimensions reads  $\partial_x^2 + \partial_y^2 + \partial_z^2$ .

An example of the scale-space of a three-dimensional solid cow is shown in Fig. 1. The continuous scale parameter enables smooth degradation of the object detail.

The Gaussian kernel is the Green's function of the heat diffusion equation, i.e.

$$L(\cdot; t) = I(\cdot) \otimes g(\cdot; t) \quad , \quad (2)$$

$$g(x; t) = \frac{1}{(4\pi t)^{N/2}} e^{-x^T x / (4t)} \quad , \quad (3)$$

where  $L(\cdot, t)$  is the image at scale  $t$ ,  $I(\cdot)$  is the original image,  $\otimes$  is the convolution operator,  $g(\cdot; t)$  is the Gaussian kernel at scale  $t$ ,  $N$  is the dimensionality of the problem, and  $t = \sigma^2/2$ , using  $\sigma$  as the standard deviation of the Gaussian kernel. The *Gaussian scale-space* is henceforth called the scale-space in this article. The information in scale-space is logarithmically degraded, the scale-parameter is therefore often sampled exponentially using  $\sigma = \sigma_0 e^T$ . Since differentiation commutes with convolution and the Gaussian

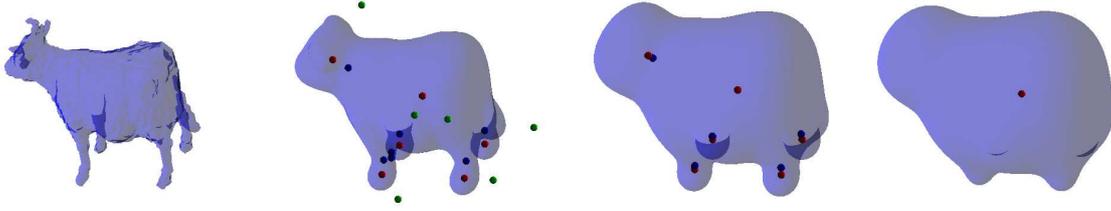


Figure 1: An example of the scale-space of a solid cow [Bra]. From left to right, the images show the zero iso-surfaces of the solid cow at scales  $\sigma = 0, 2, 3.75, 5$ . The small red, green, and blue spheres denote maxima, minima, and saddles, respectively

kernel is infinitely differentiable, differentiation of images in scale-spaces is conveniently computed,

$$\partial_{x^n} L(\cdot; t) = \partial_{x^n} (I(\cdot) \otimes g(\cdot; t)) = I(\cdot) \otimes \partial_{x^n} g(\cdot; t) . \quad (4)$$

Alternative implementations of the scale-space are multiplication in the Fourier Domain, finite differencing schemes for solving the heat diffusion equation, additive operator splitting, and recursive implementation [Der92]. We prefer the spatial convolution, since it is guaranteed not to introduce new extrema in homogeneous regions. Typical border conditions are Dirichlet, Cyclic repetition, and Neumann boundaries. We use Dirichlet boundaries, where the image is extended with zero values in all directions.

Although the dimensionality of the constructed scale-space is one higher than the dimensionality of the original image, *critical points*, in the image at each scale are always points. A critical point is e.g. an extremum,  $\partial_x L = \partial_y L = \partial_z L = 0$ . Critical points are classified by the eigenvalues of the Hessian matrix, the matrix of all second derivatives, computed at that point. As we increase the scale parameter, the critical points move smoothly forming *critical paths*. Along scale, critical points meet and annihilate or are created. Such events are called catastrophe events, and the points where they occur are called catastrophe points. The collection of events is called the *deep structure* of the image. The notion of genericity is used to disregard events that are not likely to occur for typical images, i.e. generic events are stable under slight perturbation of the image. There are only two types of generic catastrophe events in scale-space namely pairwise *creation events* and *annihilation events* [Dam97], and it has further been shown that generic catastrophe events only involves pairs of critical points where one and only one eigenvalue of the Hessian matrix changes its sign, e.g. the annihilation of a minimum (+, +, +) and a saddle (+, +, -). The implementation detail of the method for extracting critical paths and catastrophe points in 3+1D scale-space can be found in [SSKJ03].

### 3 MSSTs

Multi-Scale Singularity Trees (MSSTs) are scale-space based multi-scale image representation. They are constructed based on the nesting of image features in the scale-space to represent the deep structure of the original image. Two kinds of MSSTs are introduced in [SSKJ05]: Extrema-Based MSSTs and Saddle-Based MSSTs. Extrema-Based MSSTs will be discussed in this article. The method produces rooted ordered binary trees with catastrophe points as nodes. In 3+1D scale-space, catastrophes are also possibly caused by creations or annihilation of saddle points, e.g. between critical points with eigenvalues of the Hessian matrix (+, +, -) and (+, -, -). These saddle-saddle annihilation catastrophes together with all creation catastrophes are ignored.

Other scale-space based methods that produce tree structure but only for up to 2+1D scale-space can be found in [LP90, Kui02].

#### 3.1 Extrema Partitions

Given an image at any scale, we would like to partition the image at one scale into segments so that each segment contains only and exactly one extremum. Let  $\Omega \subset \mathbb{R}^N$  be a compact connected domain and define  $I : \Omega \rightarrow \mathbb{R}^+$  to be an image,  $\vec{e} \in \Omega$  as an extremum, and  $\vec{x} \in \Omega$  as an image point in the domain. Consider a set of continuous functions  $\gamma : [0, P] \rightarrow \Omega$  for which  $\gamma(0) = \vec{e}$  and  $\gamma(P) = \vec{x}$ ,  $\gamma \in \Gamma_{\vec{e}\vec{x}}$ , where  $\Gamma_{\vec{e}\vec{x}}$  is the set of all paths in the domain from the extremum  $\vec{e}$  to the point  $\vec{x}$ , and  $\gamma$  is parameterized using Euclidean arclength. We define the energy  $E_{\vec{e}}(\vec{x})$  with respect to an extremum  $\vec{e}$  evaluated at  $\vec{x}$  as,

$$E_{\vec{e}}(x) = \inf_{\gamma \in \Gamma_{\vec{e}\vec{x}}} \int_0^P \sqrt{(\alpha - 1) \left| \frac{\partial \gamma(p)}{\partial p} \right|^2 + \alpha \left| \frac{\partial I(\gamma(p))}{\partial p} \right|^2} dp . \quad (5)$$

Note that the energy functional is independent of parameterization. When  $\alpha = 1$ , the energy functional is

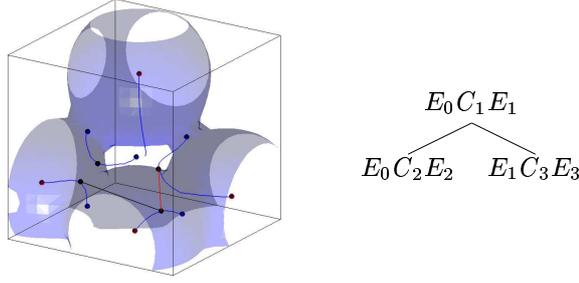


Figure 2: The Extrema-Based MSST of a three-dimensional image of four Gaussian blobs. The 2.0 iso-surfaces of the image at scale  $\sigma = 3$  is shown in blue, on the left panel. The small red and blue spheres are the maxima and saddles respectively. The blue lines are the critical paths (the scale axis is projected away) and the small black spheres are the catastrophe points. The black line and the red line denote the left-child linking and the right-child linking in the tree. A schematic drawing of the extracted MSST is shown on the right panel. Note that there is a saddle-saddle catastrophe which is ignored

also known as the path variation, a generalization of the total variation [AC03]. The path variation depends solely on the image intensity and is invariant to affine transformation of the underlying space. Moreover, it is co-variant with scaling of the image intensity. If  $\alpha \rightarrow 0$ , the energy functional will increasingly depend on the spatial distance, and therefore become increasingly localized in space.

Let  $\mathcal{E} \subset \Omega$  be the set of all extrema in the image. The *extrema partition* [AC03],  $Z_i$ , associated with an extrema  $\vec{e}_i \in \mathcal{E}$  is defined as the set of all points in the domain, where the energy  $E_{\vec{e}_i}(\vec{x})$  is minimal,

$$Z_i = \{ \vec{x} \in \Omega \mid E_{\vec{e}_i}(\vec{x}) < E_{\vec{e}_j}(\vec{x}), \forall \vec{e}_j \in \mathcal{E}, i \neq j \} . \quad (6)$$

An approximation of the energy map  $M_i : \Omega \rightarrow \mathbb{R}^+$ , which defines the energy at every point in the image associated with an extremum  $\vec{e}_i$ , can be efficiently calculated using the *Fast Marching Methods* [Set99].

### 3.2 Constructing MSSTs

MSSTs are defined by nodes and their relations. Each MSST node consists of three components: The image segment(i) that immediately covers the area of the image segment(ii) disappearing at the catastrophe(iii). For algorithmically convenience we denote the ‘surviving’ image segment the *leftport*, the catastrophe for the *body*, and the disappearing image segment for the *rightport*. Because there is exactly one image segment associated with an extremum and exactly one extremum disappears at an annihilation catastrophe, then exactly one image segment also disappears.

A node  $E_{left}C_{body}E_{right}$  is generated if an image segment of  $E_{right}$  disappears at the catastrophe  $C_{body}$  inside an image segment of  $E_{left}$ . The inclusion is easily determined by calculating the energy map with respect to the catastrophe  $C_{body}$ : the image segment of

$E_{right}$  is nested inside the image segment of  $E_{left}$  if the energy evaluated at  $E_{left}$  is minimal among all extrema existing at that scale.

Assuming that critical paths and catastrophe points in the scale-space are already and correctly detected, then the MSST building algorithm is as follows:

1. Set the root of the tree as  $E_{last}C_{top}E_{ann}$ , where  $E_{last}$  denotes the last extremum that remains in the scale-space,  $C_{top}$  denotes the highest catastrophe in scale, and  $E_{ann}$  denotes the extremum that annihilates at the catastrophe.
2. At the highest unprocessed catastrophe  $C_{next}$  in scale, calculate the energy map with respect to the catastrophe and create a node  $E_{cover}C_{next}E_{ann}$ , where  $E_{ann}$  is the extremum that disappears at  $C_{next}$ , and the energy evaluated at the extremum  $E_{cover}$  is minimal among all extrema existing at that scale.
3. Link the new created node as the leftchild of a node in the tree that does not have the leftchild and where  $E_{cover}$  equals its leftport, or as the rightchild of a node in the tree that does not have the rightchild and where  $E_{cover}$  equals its rightport.
4. Repeat 2., 3., and 4. until all catastrophe points are processed.

An example of the Extrema-Based MSST constructed from a simple three-dimensional image of four Gaussian blobs is shown in Fig. 2. The constructed MSST has three nodes corresponding to the three relevant catastrophes in the scale-space. An annihilation catastrophe of a pair of saddles is ignored.

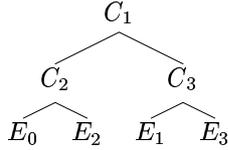


Figure 3: An extended Extrema-Based MSST example

## 4 BVHs from MSSTs

Given a MSST, we produce a BVH as follows. The MSST is extended with a set of leaves according to the leftport and the rightport representing each extremum in the original image. The newly added leaves represent the finest scale for the BVH. All free ports are extended with a leaf for the corresponding extremum, and then all ports are removed. The result is that the MSST is extended with one leaf for each extremum. All the extrema will appear in the extended MSST one and only one time. The extended MSST of the image in Fig. 2 is shown in Fig. 3.

We can denote the catastrophe scale as a size measure of the corresponding extremum. That is, at the catastrophe scale, the corresponding Gaussian will have a size that dominates the underlying image structures. We may also give a statistical interpretation using Tchebycheff’s inequality [BM93]. It states that for a random variable  $X$  with standard deviation  $\sigma$ , the probability of finding values outside a bound proportional to its standard deviation is inversely small:

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2} \quad (7)$$

We take this as a guide to set the size of the leaf bounding volumes, i.e. a leaf will be given a sphere, whose radius is proportional to the catastrophe scale. There will be one extremum, which does not disappear in a catastrophe, which is the last extremum in the scale-space. We set the bounding volume of the final extremum to be proportional to the distance to its only sibling in the MSST minus the already known sibling’s radius in the BVH.

Since the BVH is binary, we find bounding volume for the non-leaf nodes in the tree as the smallest sphere that encloses the two child spheres. Although tighter bounds may be found, this is left for further development.

## 5 RESULTS

Currently, our algorithm is capable of producing trees from objects that are sampled on a  $256^3$  grid, for a reasonable computation time, we only use  $64^3$  grids. We demonstrate our algorithm on the cow polygonal mesh [Bra]. Figure 4 shows a schematic drawing of the extracted BVH of a solid cow and Fig. 5 shows a

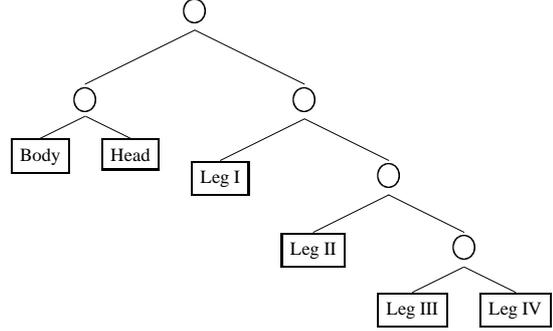


Figure 4: A schematic drawing of the extracted BVH of a solid cow

solid cow together with the spherical bounding volume at each level in the hierarchy.

In the scale-space of the cow, the legs of the cow appears in a sequential manner from coarse to fine. This makes the tree building process simple, however, in this particular example, it would possibly be more natural to let the leg-nodes appear at the same time in a 4-ary tree node. In our tree, such decisions can be enforced by post-processing, and a useful indication in this case would be that the catastrophes occur within a very narrow scale-band.

There are many properties which are interesting when evaluating the quality of a BVH. Unfortunately some of them are contradicting each other.

- Smallest possible bounding volumes
- Smallest possible overlap between volumes at the same depth in the hierarchy
- Small sized BVH, i.e. as few nodes as possible
- Complete coverage versus sampling based coverage
- “balanced” trees

The last property is one we challenge, although it has been proved that balanced trees provide best worst case queries, a balanced tree do not represent the scale of the object. Working with time critical or approximating queries this become an important property. We suggest that the tree should be balanced with respect to the density of the object.

## 6 DISCUSSION

Most recent work with BVHs has focused on: Trying out new kinds of bounding volumes, figuring out better methods for fitting a bounding volume to a subset of an object’s underlying geometry, finding faster and better overlap test methods, and comparing homogeneous BVHs of different bounding volume types.

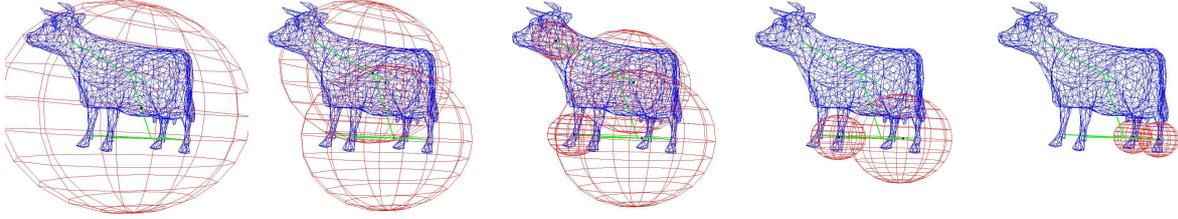


Figure 5: A solid cow and the hierarchical bounding volumes at each level of the BVH. The surface of the original cow, the links between catastrophes in the scale-space and the spherical bounding volumes are shown from left to right for the level one to five of the BVH

In order to improve the performance of traversal algorithms, depth control, layered bounding volumes, caching bounding volumes, and shared bounding volumes have been studied. We have chosen to classify our method as being a mixed bottom-up and top-down method, because the scale-space is built bottom-up, and the MSST are found in a top-down manner. The corresponding BVH is then built in a straightforward incremental way, by doing an order traversal of the MSST, and creating bounding volume nodes as catastrophes are encountered.

The computational complexity for our algorithm is currently high. Using  $N^3$  as the number of pixels in the image,  $S$  as the number of scales to be evaluated,  $\sigma$  as the largest scale,  $K$  as the number of critical line-pieces found, and  $E$  as the number of extrema at the lowest scale, the computational complexity for each part of our algorithm is as follows:

**Computation of the Scale-Space:**  $\mathcal{O}(S\sigma^3 N^3)$

It may be possible to improve the calculation time for the Gaussian scale-space, e.g. using sub-sampled image for approximating scaled image at high scales or using faster alternatives to spatial convolution. However, we have not yet found alternatives that does not introduces spurious extrema in homogeneous regions.

**Storage of the Scale-Space:**  $\mathcal{O}(2N^3)$

The most memory intensive part of our algorithm is the storage of the scale-space. We only require the storage of two neighboring scales in order to find the critical paths in our current implementation.

**Extracting Critical Paths:**  $\mathcal{O}(SN^3 + K^2)$

The critical paths can be extracted considerably faster by tracking each extremum from the finest scale, however this would require either to store the full Scale-Space or perform local calculations during the tracking process. Since this is by far not the slowest part of our algorithm, we have left this for further research.

**Finding a Euclidean Tree,  $\alpha = 0$  in (5):**  $\mathcal{O}(E^2)$

It is fastest to use the Euclidean metric in (5), for  $0 < \alpha \leq 1$  see below.

**Finding a General Tree:**  $\mathcal{O}(EN^3 \log N^3)$

This is the most computationally expensive part of our algorithm. However, we expect that the speed of the Fast Marching Method can be improved by a narrow band implementation.

Gaussian scale space provides us with a continuous degradation of an object, other algorithms fail completely on this point, they typical control their scale by saying that at the next level of the BVH should have 50% less number of volumes, or at the next level the volumes should fit 20% better. A direct study of scales seems to be a more proper representation.

Medial surface (M-reps) based methods for building BVHs have been the approach to use for bottom-up construction. Our method differs from M-reps significantly by being a density based method, whereas M-reps is more a surface-based method. Furthermore our method provides us with a natural scale that is easily used to determine both bounding volumes and the topology of the hierarchy. M-reps do not provide this scale information nor can they tell one about the density of an object.

The well-established foundation on scale-space theory provides us with a well-defined concept of scale, shape, and detail of an object. These concepts are valuable tools as our work hopefully demonstrates.

The main contribution of our work is a new method for building bounding volume hierarchy, however, there is still much need to be done. So far our work has been a feasibility study showing that the construction of BVHs from MSSTs actually can be done. We have not yet made any attempt towards comparing the quality of the multi-scale singularity BVH with other algorithms. Future research will be on the tightening of the bounding volumes utilizing information in scale-space.

## 7 ACKNOWLEDGMENTS

This work is part of the DSSCV project sponsored by the IST Programme of the European Union (IST-2001-35443).

## References

- [AC03] P. A. Arbelaez and L. D. Cohen. The Extrema Edges". In *Scale Space 2003, LNCS 2695*, pages 180–195, 2003.
- [Ber97] G. van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools*, 2(4):1–13, 1997.
- [BFA02] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. *Proceedings of ACM SIGGRAPH*, 21(3):594–603, 2002.
- [BM93] L. Brøndum and J. D. Monrad. *Statistik I - Sandsynlighedsregning og statistiske grundbegreber*. Den private ingeniørfond, 1993.
- [BMF03] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 28–36. Eurographics Association, 2003.
- [BO04] Gareth Bradshaw and Carol O’Sullivan. Adaptive medial-axis approximation for sphere-tree construction. *ACM Transactions on Graphics*, 23(1), January 2004.
- [Bra] Gareth Bradshaw. Polygonal mesh of a cow. .
- [Dam97] James Damon. Local Morse theory for Gaussian blurred functions. In Jon Sporrying, Mads Nielsen, Luc Florack, and Peter Johansen, editors, *Gaussian Scale-Space Theory*, chapter 11, pages 147–163. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.
- [Der92] R. Deriche. Recursively Implementing the Gaussian and its Derivatives. In V. Srinivasan, Ong Sim Heng, and Ang Yew Hock, editors, *Proceedings of the 2nd Singapore International Conference on Image Processing*, pages 263–267. World Scientific, Singapore, 1992.
- [DO00] John Dingliana and Carol O’Sullivan. Graceful degradation of collision handling in physically based animation. *Computer Graphics Forum*, 19(3), 2000.
- [EL01] Stephan A. Ehmman and Ming C. Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition. In A. Chalmers and T.-M. Rhyne, editors, *EG 2001 Proceedings*, volume 20(3), pages 500–510. Blackwell Publishing, 2001.
- [ES03] Kenny Erleben and Jon Sporrying. Collision detection of deformable volumetric meshes. In Jeff Lander, editor, *Graphics Programming Methods*. Charles River Media, 2003.
- [GBF03] E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. *ACM Transaction on Graphics, Proceedings of ACM SIGGRAPH*, 2003.
- [GDO00] Fabio Ganovelli, John Dingliana, and Carol O’Sullivan. Buckettree: Improving collision detection between deformable objects. In *Spring Conference in Computer Graphics (SCCG2000)*, pages pp. 156–163, Bratislava, April 2000.
- [GJK88] E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4:193–203, 1988.
- [GLM96] S. Gottschalk, M. C. Lin, and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. Technical Report TR96-013, Department of Computer Science, University of N. Carolina, Chapel Hill, 8, 1996.
- [Got00] Stefan Gottschalk. *Collision Queries using Oriented Bounding Boxes*. PhD thesis, Department of Computer Science, University of N. Carolina, Chapel Hill, 2000.
- [GS87] Jeffrey Goldsmith and John Salmon. Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics and Applications*, 7(5):14–20, May 1987. see Scherson & Caspary article for related work.
- [He99] Taosong He. Fast collision detection using quospo trees. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 55–62. ACM Press, 1999.
- [Hir02] Gentaro Hirota. *An Improved Finite Element Contact Model for Anatomical Simulations*. PhD thesis, University of N. Carolina, Chapel Hill, 2002.
- [HKL<sup>+</sup>99] Kenneth E. Hoff, III, John Keyser, Ming Lin, Dinesh Manocha, and Tim Culver. Fast computation of generalized voronoi diagrams using graphics hardware. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 277–286. ACM Press/Addison-Wesley Publishing Co., 1999.
- [Hub93] P. M. Hubbard. Interactive collision detection. In *Proceedings of the IEEE Symposium on Research Frontiers in Virtual Reality*, pages 24–32, 1993.
- [Hub96] Philip M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15(3):179–210, 1996.

- [Iij62] T. Iijima. Basic theory on normalization of a pattern (in case of typical one-dimensional pattern). *Bulletin of Electrotechnical Laboratory*, 26:368–388, 1962. (in Japanese).
- [Kar] Karma. MathEngine Karma, <http://www.mathengine.com/karma/>.
- [KHM<sup>+</sup>98] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of  $k$ -DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.
- [Koe84] J. J. Koenderink. The Structure of Images. *Biological Cybernetics*, 50:363–370, 1984.
- [KPLM98] S. Krishnan, A. Pattekar, M. Lin, and D. Manocha. Spherical shell: A higher order bounding volume for fast proximity queries. In *Proc. of Third International Workshop on Algorithmic Foundations of Robotics*, pages 122–136, 1998.
- [Kui02] Arjan Kuijper. *The deep structure of Gaussian scale space images*. PhD thesis, Image Sciences Institute, Institute of Information and Computing Sciences, Faculty of Mathematics and Computer Science, Utrecht University, 2002.
- [LAM01] Thomas Larsson and Tomas Akenine-Möller. Collision detection for continuously deforming bodies. In *Eurographics*, pages 325–333, 2001.
- [LGLM99] Eric Larsen, Stefan Gottschalk, Ming C. Lin, and Dinesh Manocha. Fast proximity queries with swept sphere volumes. Technical Report TR99-018, Department of Computer Science, University of N. Carolina, Chapel Hill, 1999.
- [LP90] L. M. Lifshitz and S. M. Pizer. A multiresolution hierarchical approach to image segmentation based on intensity extrema. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 12(6):529–541, 1990.
- [Mel01] Stan Melax. Bsp collision detection as used in mdk2 and neverwinter nights. *Gamasutra*, March 2001. Online article.
- [Mir98] Brian Mirtich. V-clip: Fast and robust polyhedral collision detection. *ACM Transactions on Graphics*, 17(3):177–208, July 1998.
- [MW88] M. Moore and J. Wilhelms. Collision detection and response for computer animation. In *Computer Graphics*, volume 22, pages 289–298, 1988.
- [OD99] C. O’Sullivan and J. Dingliana. Real-time collision detection and response using sphere-trees, 1999.
- [Ode] Ode. Open Dynamics Engine, <http://q12.org/ode/>.
- [Pal95] I.J. Palmer. Collision detection for animation: The use of the sphere-tree data structure. In *The Second Departmental Workshop on Computing Research*. University of Bradford, June 1995.
- [PG95] I.J. Palmer and R.L. Grimsdale. Collision detection for animation using sphere-trees. *Computer Graphics Forum*, 14(2):105–116, 1995.
- [PML97] M. K. Ponamgi, D. Manocha, and M. C. Lin. Incremental algorithms for collision detection between polygonal models. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):51–64, 1997.
- [Set99] J. A. Sethian. Fast Marching Methods. *SIAM Review*, 41(2):199–235, 1999.
- [SL00] K. Sundaraj and C. Laugier. Fast contact localisation of moving deformable polyhedras. In *IEEE Int. Conference on Control, Automation, Robotics and Vision*, Singapore (SG), December 2000.
- [SSKJ03] K. Somchaipeng, J. Sporning, S. Kreiborg, and P. Johansen. Software for Extracting Multi-Scale Singularity Trees. Technical report, Deliverable No.8, DSSCV, IST-2001-35443, 15. September 2003.
- [SSKJ05] K. Somchaipeng, J. Sporning, S. Kreiborg, and P. Johansen. Extrema-Based Multi-Scale Singularity Trees: Soft-linked Scale-Space Hierarchies. In *Submitted to Scale-Space 2005*, 2005.
- [TC96] C. Tzafestas and P. Coiffet. Real-time collision detection using spherical octrees : Vr application, 1996.
- [vdB01] G. van den Bergen. Proximity queries and penetration depth computation on 3d game objects. *Game Developers Conference*, 2001.
- [VM95] Pascal Volino and Nadia Magnenat Thalmann. Collision and self-collision detection: Efficient and robust solutions for highly deformable surfaces. In Dimitri Terzopoulos and Daniel Thalmann, editors, *Computer Animation and Simulation '95*, pages 55–65. Springer-Verlag, 1995.
- [VMT00] Pascal Volino and Nadia Magnenat-Thalmann. *Virtual Clothing, Theory and Practice*. Springer-Verlag Berlin Heidelberg, 2000.
- [Vor] CMLabs Vortex. <http://www.cm-labs.com/products/vortex/>.
- [Wit83] A. P. Witkin. Scale-space filtering. In *Proc. 8th Int. Joint Conf. on Artificial Intelligence (IJCAI '83)*, volume 2, pages 1019–1022, Karlsruhe, Germany, August 1983.
- [Zac98] G. Zachmann. Rapid collision detection by dynamically aligned dop-trees, 1998.

# Dynamic Coordinated Email Visualization

Simone Frau  
Computing Laboratory,  
University of Kent,  
Canterbury,  
Kent CT2 7NF, UK  
sf31@kent.ac.uk

Jonathan C. Roberts,  
Computing Laboratory,  
University of Kent,  
Canterbury,  
Kent CT2 7NF, UK  
j.c.roberts@kent.ac.uk

Nadia Boukhelifa  
Computing Laboratory,  
University of Kent,  
Canterbury,  
Kent CT2 7NF, UK  
n.boukhelifa@kent.ac.uk

## ABSTRACT

Many computer users receive hundreds (if not thousands) of emails per week; users often keep these emails and have many years of personal emails archived: users use their stored emails to manage appointments, to-do lists, and store useful information. In this paper we present an interactive email visualization tool (Mailview) that utilizes filter and coordination techniques to explore this archived data. The tool enables users to analyze and visualize hundreds of stored emails, it displays the emails on time-dependent plots enabling users to observe trends over time and perceive emails with similar features. Interaction is an important aspect of finding meaning within information, hence the tool utilizes focus+context views, dynamic filters, detail-on-demand techniques and coordinated views, finally, we discuss various methods that enable the system to be designed such that it can display hundreds of objects at interactive rates.

## Keywords

Email visualization, information visualization, coordinated views, exploration, email archive

## 1. INTRODUCTION

The use and growth of email is staggering, even with the divergence and growth of other communication technologies such as the mobile phone and short messaging, still the use of email is growing. According to the University of California Berkeley "How much information?" 2003 evaluation, about 31 Billion emails were sent per day in 2002 and a prediction of 60 billion will be sent per day in 2006 [HMI03]. Many users store past emails for reference: archiving them in folders for future observation, they may store hundreds and thousands of emails in these archives; indeed the authors themselves have a joint archive of approximately 100,000 emails. They are stored such that information can be referenced for future use.

This personal email archive provides a rich and diverse dataset, and it is both interesting and potentially useful to analyze and visualize this information. First, visualization can help with

information retrieval. There are known problems with email archiving [Whi96] that could be overcome by visualization. Such problems include, generating appropriate folder names, reconstructing these labels when they are needed to search for the required data, and being consistent in grouping similar material in the same folder, indeed, there is an ontology issue (as some mails could be legitimately stored under different categories). Second, visualization can be used to help the user analyze the information for trends or make observations. These observations could be utilized to control and influence work-patterns or to potentially benefit the effectiveness of spam-filters or automatic mail transfer (such as Exim) and archiving engines. For instance, if it was observed that the majority of spam messages arrived between a certain times of day then the spam filter could be dynamically adapted appropriately to catch more spam during that period.

Many email programs, like for example Outlook Express, Exmh, Mutt or Pine, besides showing the content of the emails display some features of the emails themselves; such as the receiving date and time, sender name, email length, and information about whether the email has attachments, etc. They are very good at displaying information about single emails, but do not traditionally depict aspects or trends of multiple emails. Furthermore, time is an important factor in emails: they are downloaded in chronological order, often stored in order, email

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 80-903100-7-9  
WSCG'2005, January 31-February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency – Science Press*

addresses may change when senders change employers, and minutes of meetings or mail-shots may be weekly occurrences. Thus, in contrast to current email visualization tools, the aim of our system is to visualize hundreds of emails displayed in a chronologically based visualization.

This paper describes the type of data that can be represented (section 2), related work (section 3), and our Mailview application (the remainder of the paper). Mailview provides a visualization of emails focused on showing the emails chronologically and it uses focus+context techniques and multiple views. It includes (a) an overview of the multiple emails represented by glyphs, (b) two zoomable and coordinated views depicting the arrival time, specifically the chronological order in which they arrived, and the relative size of the emails and (c) depicts specific details of selected single emails. Mailview is not meant to replace a normal email viewer, but to be used in partnership with their standard mail reader; Mailview may enhance the users understanding of their stored emails.

## 2. EMAIL DATA

Email data provides a rich and interesting source of information. Indeed, there are many attributes and statistics that can be calculated and visualized. We classify this information into four categories: meta-information, content, intra-email and inter-email.

First there is the straightforward information about the mail itself, which is merely stored in the email header. This meta-information includes the senders email address (their familiar name may be retrieved from an address book), recipient email address, delivery date/time received date/time, the path of the email through various servers, content mime types and whether it contains attachments, subject line and the content of the email body. Not only is it interesting to investigate trends over every email received, but also it would be useful to filter and depict emails from specific senders, or by subjects.

Second, it is useful to view the content of the emails. The content of the email may be stored in different types (text, html, pdf etc), and the challenge of any mail viewer is to seamlessly view each of these different types. Moreover, although the average size of an email may be 59KB [HMI03] some emails are much larger. This means that many emails do not fit into a single screen, the information needs to be scrolled. Hence, there are opportunities to use focus+context techniques to abstract and better represent this text information.

Third, statistical information can be generated from each email individually. We name this intra-email analysis (within an email). Intuitively, we can easily

calculate the length or size of the email, e.g. we can calculate the number of words or letters in the text, the number of pages, and the amount of memory it takes up. But, other statistical calculations can be made on a single email. For instance, word frequencies can be calculated to find common and recurrent words in the email, or emails can be classified based on an analysis of the content (such email classification is a necessary part of spam filtering).

Finally, observations and calculations can be made on a collection of emails. We classify this as being an inter-email analysis (between emails). Such analysis is often used to help navigate news articles. For example, some news reading programs display threaded information, depicting an initial post with all occurrences of replies to that post, more generally threaded-visualization depicts all objects that mention the same topic (thread) of information.

## 3. RELATED WORK

Electronic mail is a widely used communication medium. Its role, however, has expanded from a mere information carrier into a dynamic medium where users can have conversation threads, delegate tasks, plan meetings and exchange minutes and multimedia files. As a result of the growing usage of emails, users are overwhelmed by the increasing traffic on their mailbox and the varied nature of emails (from professional, personal and sales to spam and harmful emails). Again referring to the "How much information?" report [HMI03], by May 2003 almost 55% of emails sent were classified as spam. Consequently, researchers are continuously working on new interfaces to help users better manage their emails.

Much recent development, that the general public would be aware of, is the continued development of facilities that allow users to browse, manage and query emails more efficiently. Certainly, most modern mail viewers permit users to filter or highlight mail messages that conform to certain search criteria (such as displaying all the messages sent by a specific user, or on a particular subject). The tools allow the user to quickly and efficiently search through multiple folders and display exact and partial matches to the queries. Other tools or add-ons allow users to manage their mail, such as automatically archiving or deleting emails.

In the research domain, one important strand of email visualization research is the analysis and visualization of relationships between emails (detailed in section 2 as inter-email visualization and analysis). One important area of study focuses on clustering emails into meaningful groups. For example, Sudarsky and Hjelsvold [Sud01] make use

of the hierarchical nature of the domain names present in email addresses for the clustering criteria. While others have investigated modeling and characterizing email conversations (e.g. Venolia and Neustaedter [Ven03]).

Another area of research is visualizing these threads of conversation depicting, for instance, where users hold ongoing discussions by email. Similar techniques are applied to analyze and visualize social networks such as determined from Internet relay chat (IRC) and other forms of data. For example, Mutton [Mut04] infers associations between IRC participants based on parsing the conversation text. He utilizes graphs to visualize the inferred relationships, with the nodes being the participants and the edges the associations.

EmViz [Hec97] also uses a graph-based layout to visualize correspondence from email traffic within an organization. EmViz uses a cone-tree to depict the hierarchy of the organization. Additional information is annotated onto this reference structure, including, the quantity of emails sent/received by an individual is denoted by the size of the node, and the color of the edge depicts the frequency of peer-to-peer correspondence. Similarly, the aim of eArchivarius [Leu03] is to highlight existing communities of people. The eArchivarius tool visualizes and organizes collections of emails in various ways, one example demonstrates a cluster-based visualization, where each sphere glyph represents a person and the more emails two people exchange the closer the glyphs become. Colors can represent various attributes such as the topic (where the confidence of 'an email being correctly classified' is realized by the intensity of the color).

Thread Arcs [Ker03] is another graph-based tool that visualizes relations between emails. In this representation the threads are chains of emails where each one (except the root) is a reply to another belonging to the chain. Arcs link each 'child' (an email being the reply to another one) to its 'parent' (the email the child replies to) showing the connections among them and the progress of a conversation. The user can interact selecting any email in the thread.

The emphasis of the aforementioned related work focuses on intra-email visualization, however the focus of our work in this paper is on visualizing the personal email archive especially displaying the emails on time related plots. So far, there has not been much research investigating methods to visualize this archive. Jovicic [Jov00] does describe a system to visualize personal email data and indeed discusses that time is important. In fact, email data and metadata has a temporal nature, and thus visualizing the mailbox can benefit greatly from the

already existing visualization tools for temporal information.

We are in agreement with Jovicic [Jov00], who states that users tend to ignore time as a crucial factor in email communication. She added that most emails tend to have personal and informational qualities. When an email involves personal events, activity, people involved and place of the event are typically well remembered explicitly [Jov00]. This is often known as episodic memory, which details the human ability of remembering things that happened at a particular time and place. Jovicic, discussing work by Friedman [Fri93], mentions that the memory reconstruction relies on 'temporal cycles' which are used to estimate the time elapsed since the event and to provide a frame of reference within which an event can be placed. Jovicic plots the mails on periods of 'days' and 'weeks' particularly highlighting the weekends. In another study Begole et al [Beg02] monitors computer activity minute-by-minute in order to establish rhythms. Begole et al display computer activity timelines coupled with information about the location of the activity, online calendar appointments, and email activity. This linkage helps find patterns of individuals according to time of day, location and day of the week.

Temporal based charts have been used to visualize historic, and other chronological data in other datasets. Email data has much similarity with this information and thus we briefly mention other related temporal/historic visualization research. For instance, Weber et al [Web01] describe some of the popular visualization tools for time-series data; these include sequence charts, point charts, bar charts, line graphs and circle graphs. One of the most utilized techniques to visualize temporal data across various fields is timelines.

A timeline is a linear visual representation of time-varying events. According to Kumar et al [Kum98], the earliest use of timelines in the published literature can be traced to William Playfair. Timelines have been used to display historical information very efficiently. Extensions to this technique resulted in 3D dynamic timelines [Kul96], spiral timelines [Web01], RiverThemes [Hec97] and lifelines [Pla96]. Moreover, Kumar et al [Kum98] presented a framework and interface for representing temporal information. Finally, Karam [Kar94] suggested a model to automate and generalize timelines.

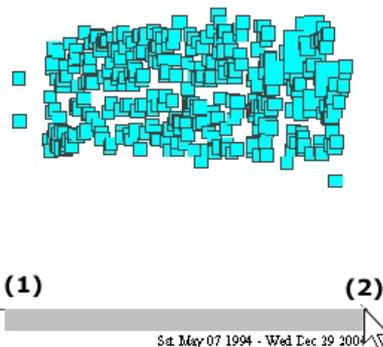
## 4. DESIGN & IMPLEMENTATION

### 4.1. Data Gathering and Preparation

Various email readers store the emails in different forms (from human readable files with one file per message, tagged file including multiple messages, or proprietary databases). In this paper we assume that



**Figure 1** A screenshot of Mailview, depicting emails from the spam directory from our personal email archive.



**Figure 2** The user can zoom into a particular area either by selecting a bounding region directly on the visualization, or by dragging the mouse along an axis line (position 1 to 2) as shown by this screenshot of Mailview.

the emails are archived in individual files, the files may be grouped together in folders, that the files are MIME encoded (the Multipurpose Internet Mail Extensions format allows enriched content such as images, audio, and attachments) but they are stored in essentially text documents.

Gathering and preparing data consists of first choosing the mails to be viewed, then scanning the mails to create a data-structure containing an abstract representation of the mail data required for visualization. The data-structure is hierarchically organized mirroring the folders hierarchy, and each node corresponds to an email and includes some of

its features, such as sender, receiver and so on. It also contains fields concerning the emails presentation (such as layout coordinates). Besides extracting some basic fields (date, sender, receiver and subject), we calculate the size of the email, and scan the whole body of the email to obtain a frequency analysis of the most common words and their percentage.

#### 4.2. Visualization

The overarching design was to depict the emails in plots that demonstrated temporal attributes. Hence we display the emails in various temporal based scatter plots that can be scaled and zoomed. Figure 1 shows Mailview.

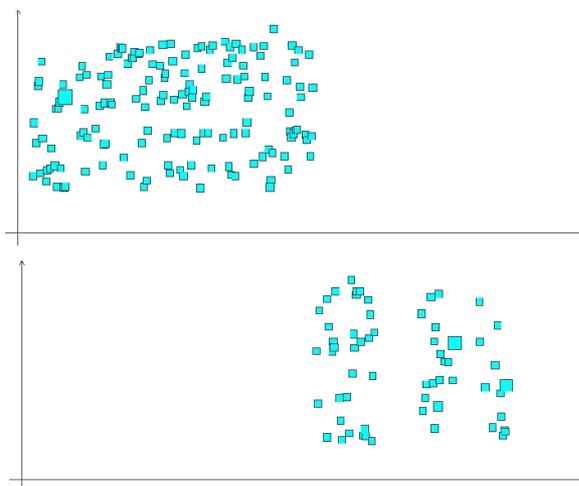
There are three main layers in the tool, the upper layer (a) depicts details about a specific email (sender, recipient, date, time, and the frequency analysis) the second (b) and third (c) layers depict two views: one context view that shows an overview of the whole display (right), and the other is a zoomed view (left). Each of the plots display dates (days, weeks and months) along the bottom (the x-axis). The plot in the middle layer (b) displays time along the y-axis (from 00:00 at the bottom to 24:00 midnight at the top) and the plot on the bottom layer (c) represents a stacked bar-chart representation (with the y-axis representing the quantity of emails that day).

Each email is represented by a glyph (the user can choose the glyphs) either vertical lines, circles or squares. The relative size of the email is realized by the size of the glyph (with larger emails being realized by larger glyphs), the emails are also colored, the color depends upon the folders in the archive and is automatically allocated, but the user can edit the allocated colors.

We took a design decision to not display too many labels, as there could be potentially hundreds of emails being display, there could be hundreds of labels each denoting an email: which would quickly become unreadable. However, labels are important and a balance must be met. Thus, text information about both the axis and particular details of the highlighted email get displayed as the user brushes over an email glyph. In fact, the use of glyphs enables a large amount of information to be displayed in a small space (which was one of the original goals). Obviously, it would be useful to link a regular email browser to this visualization (or to embed this into such a browser) and again coordinate everything together. However, we leave this



**Figure 3** This screenshot shows Mailview displaying emails of laboratory meetings. After a quick observation it is easy to see that most of the emails have arrived during traditional work hours (8.30am to 5.30pm).



**Figure 4.** This figure shows a subset of the data from Figure 3, (Upper) depicting the emails that have been sent by one member of staff, and (Lower) by a second member of staff. The visualization clearly depicts when the first member of staff left and the second one took over the role of sending out the regular meeting minutes.

integration of the traditional mail reader and the Mailview visualization for future work.

In fact, the tool was developed using Java, and although Mailview currently includes four plots, glyphs and mappings, each of the plots are inherited from the same abstract class, because they have similar properties and attributes, hence the system is easily extensible.

#### 4.3. Interaction, Coordination & Filtering

As the user brushes over the plots so the underlying glyph is highlighted, the glyph doubles in size, summary details of the appropriate email are updated in the top view (Figure 1a), and the same email (represented in the other window-view) is also simultaneously highlighted. In fact, the user can choose which views are coordinated together to allow the users to compare disparate parts of the plots. Figure 1 shows some emails, which are designated as spam from our email archive; the plot demonstrates that spam arrives throughout the day, and throughout the week.

Furthermore, users can zoom into any area. Zooming can be controlled either by dragging out a bounding box directly on the plot (allowing the user to change the date and time range together), or by dragging the mouse along an axis line (allowing the user to zoom into either a date range or a time range

independently) as shown in Figure 2. Again, these views can be coordinated together to allow the same focused area to be visualized in each view. The zoomed area is also depicted in the overview plots.

Every operation that has an effect to the display (such as zooming, changing what is coordinated, or changing the appearance of the glyphs) is stored in a history list. The user can undo any operation that they have made; this encourages the user to try out scenarios and makes comparison easier.

The system allows emails to be filtered and selected so trends about particular senders or subjects can be spotted. This is different to the above zooming and selection operations, which enable the user to see trends in time. For instance, the user may wish to observe repeating patterns such as to see the email of weekly meetings, or occurrences when employees have left a company. Filtering is achieved through selection. The email is selected when the user clicks on a glyph, this fixes the current detail information, thus when the user thereafter brushes over a detail field such as subject or sender, only those emails that have been sent by that sender are displayed (all the others are filtered out); the user can then select another field and the information is thereafter constrained by two fields.

We demonstrate this filtering mechanism through a simple example. Figure 3 shows some emails that relate to laboratory meetings, they are all stored in one folder in the archive. From an initial observation, it is easy to notice that the emails mostly arrive between 8.30am and 5.30pm. There are some outliers, including one sent at 7.15 am (under the displayed cursor in the screenshot of Figure 3). The user can now brush over the elements to discover different aspects of the emails. In browsing this dataset we discovered that most of the emails on the left-hand-side were sent by one member of staff, and those to the right by another. This discovery is shown in Figure 4 and in fact demonstrates the time when one secretary left and another one joined the laboratory. At any stage the user can undo the operation to explore another constraint or scenario.

#### **4.4. Performance challenges and solutions**

With hundreds of emails it takes time to refresh the display. The bottleneck seemed to be calculating positions at runtime, so in order to speed up the refresh rate we cache the positions in the hierarchical data-structure. This way the positions only need to be recalculated when a range change occurs. Further problems occurred with brushing at interactive speeds (especially finding the closest email to the mouse pointer). Creating a grid, containing a two-dimensional array of lists of emails, solved this. Since we set the cell size to the maximum radius of

the shapes, we knew for sure that if we are hovering with the mouse over a shape its center must be either in that cell or in one next to it.

If the feedback is coordinated among windows, then corresponding emails need to be highlighted in the other diagrams. Even though emails datastructure contains a unique identifier field that makes them unambiguously distinguishable, it still would take time to search for the corresponding element to highlight. The chosen solution was to develop a new class, similar to the coordination space of Boukhelifa et al [Bou03], to manage the coordination. Practically, this class stores a list of the email unique identifiers as an associated array, so if on a check the element contains -1, there is no similar email in that diagram, otherwise the number contained will represent the position the email has in the hierarchical datastructure.

## **5. SUMMARY & FUTURE WORK**

We have successfully developed a visualization tool that displays email archive data. The tool enables users to see trends and details of emails within time and date plots. Users can interact, zoom and filter the information in a coordinated exploratory environment. We believe chronological information is important within effective email data perception. Most of our program's functionalities have been developed to be as extendable as possible (both for visualization and coordination), such that the tool can be further developed in the future. Although we have tested the tool on various users, we have yet to accomplish a full user study. We plan to do this in the near future.

There is much functionality, other views and techniques that could be added to the system. In fact, we believe aggregation is an important extension that is missing from other tools and also missing from Mailview at present, such ideas are important and have been used by Larsen et al [Lar96] and Begole et al [Beg02]. Additionally, we know that users do not often remember exact dates of events (as discussed in the related work, section 3) rather they remember periods of time, and hence it would be useful to allow the user to explore the data through a rich set of aggregation commands.

## **REFERENCES**

- [Beg02] J.B. Begole, J.C. Tang, R.B. Smith, and N.Yankelovich, "Work rhythms: analyzing visualizations of awareness histories of distributed groups," Proc ACM CSCW'02, pp.334-343, 2002. New Orleans.
- [Bou03] N. Boukhelifa, J.C. Roberts, and P.Rodgers, "A Coordination Model for Exploratory Multi-View Visualization," Proc International Conference

- on Coordinated and Multiple Views in Exploratory Visualization (CMV 2003), pp.76-85. J.Roberts, ed., IEEE, July 2003.
- [Fri93] W.J. Friedman, "Memory of time for past events," *Psychological Bulletin* 113(1), pp.44-66, 1993.
- [Hec97] B. Heckel and B.Hamann, "Emviz - a visual e-mail analysis tool," *Proc New Paradigms in Information Visualization and Manipulation Workshop*, pp.36-38, 1997. Las Vegas, Nevada USA.
- [HMI03] "How much information? 2003," [www.sims.berkeley.edu/research/projects/how-much-info-2003/](http://www.sims.berkeley.edu/research/projects/how-much-info-2003/)
- [Jov00] S.Jovicic, "Role of memory in email management," *Proc ACM CHI 2000, Interactive posters*, pp.151-152, 2000. The Netherlands.
- [Kar94] G.M. Karam, "Visualization using timelines," *Proc International Symposium on Software Testing and Analysis ISSTA*, T.Ostrand, ed., pp.125-137, 1994.
- [Ker03] B.Kerr, "THREAD ARCS: An Email Thread Visualization," in *IEEE Symposium on Information Visualization*, pp.211-218, 2003. Seattle, Washington.
- [Kul96] R.L. Kullberg, "Dynamic timelines: Visualizing the history of photography," *Proc ACM CHI 96 Conference on Human Factors in Computing Systems, VIDEOS: Visualization*, pp.386-387, 1996.Vancouver, Canada.
- [Kum98] V. Kumar, R. Furuta, and R.B. Allen, "Metadata visualization for digital libraries: Interactive timeline editing and review," in *DL'98: Proc ACM International Conference on Digital Libraries*, pp.126-133, 1998. Pittsburgh, USA.
- [Lar96] S.F. Larsen, C.P. Thompson, and T.Hansen, "Remembering our past," in *Studies in autobiographical memory time in autobiographical memory*, D.C.Rubin, ed., Cambridge University Press, 1996.
- [Leu03] A.Leuski, D.W. Oard, and R.Bhagat, "eArchivarius: Accessing Collections of Electronic Mail," *Proc ACM SIGIR Conference on Research and Development in Information Retrieval, Demos*, p.468, 2003. Toronto, Canada.
- [Mut04] P.Mutton, "Inferring and visualizing social networks on internet relay chat," *Proc Information Visualization, IEEE Computer Society*, 2004. London, UK.
- [Pla96] C.Plaisant, B.Milash, A.Rose, S.Widoff, and B.Shneiderman, "Lifelines: Visualizing personal histories," *Proc ACM CHI 96 Conference on Human Factors in Computing Systems Papers: Interactive Information Retrieval*, pp.221-227, 1996.
- [Sud01] S.Sudarsky and R.Hjelsvold, "Visualizing electronic mail," *Proc Information Visualization IV'02*, pp.3-9, IEEE Computer Society, 2002. London, UK.
- [Ven03] G.D. Venolia and C.Neustaedter, "Understanding sequence and reply relationships within email conversations: a mixed-model visualization," *Proc ACM CHI*, pp.361-368, 2003.
- [Web01] M. Weber, M. Alexa and W. Müller "Visualizing timeseries on spirals," *Proc InfoVis'01*, pp.21-28, IEEE Computer Society, 2001.
- [Whi96] S.Whittaker and C.Sidner, "Email overload: Exploring personal information management of email," *Proc of ACM CHI 96*, pp.276-283, 1996.



# Cellular Automata for 3D Morphing of Volume Data

SK Semwal

Department of Computer Science  
University of Colorado

Colorado Springs, CO. 80933, USA

semwal@cs.uccs.edu

K Chandrashekhar

Department of Computer Science  
University of Colorado

Colorado Springs, CO. 80933, USA

kchandra@uccs.edu

## ABSTRACT

Morphing involves the smooth transformation of one model, called the source to another, called the target. Several methods have been employed in this field both for two and three dimensional morphing. This paper performs morphing through the usage of cellular automata. The goal was to develop morphing algorithms that would minimize the need for both the user input and correspondence specification between source and the target. Two algorithms, *the bacterial growth model* and the *core increment model* have been designed and implemented in C++. Both algorithms utilize simple automata rules and are stable over dissimilar data. Results are presented that display the efficiency of the approach.

## Keywords

Animating Volume Data, Local Interaction creating global phenomena.

## 1. INTRODUCTION

Morphing is a technique in graphics that results in the transformation of an object, called the source model, into another, called the target model, in a gradual and smooth fashion. Apart from many movies, morphing now finds usage in 3D games that are in the market such as Alter Echo [outrage2003] and Perimeter [K-DLab2004]. The concept of morphing extends to other applications as well. Some example applications are: Visualization during cranio-facial surgery [Fang1996]; evolution by morphing the skulls of primates and modern humans [Rodier1997]; environmental changes on sea levels and forest cover [geoplace2004]; continental drift [Bourke2001] or erosion; and understanding biological processes such as plant growth and fetal development [pbs2004].

Both 2D and 3D morphing methods have been developed. Several good papers can be found on 2D Morphing 1992 [Beier1992,Sederberg1992] and

recently in [Abraham2004]. The biggest benefit of 3D morphing over 2D is that it is independent of lighting and other environmental effects which are inherent in the images. In addition, the view of the camera can be changed in real-time in order to provide a much clearer understanding of the morphing process. We focus on the 3D variety.

Many 3D morphing algorithms require a correspondence that maps features of the source model to that of the target model. We wanted to investigate approaches that are free of this restriction. Most morphing algorithms also rely on user-defined control points that guide the way the source model morphs to the target model. While this is useful in guiding morphing in the manner that the user desires, we felt that there is room for exploring techniques using minimal user input because this correspondence process can be tedious and tiring for the user. While this perhaps takes away from the artistic impressions that users are allowed when the correspondence is defined manually, minimal manual specification has its own benefits. Our approach uses the concept of cellular automata in order to perform morphing. Cellular automata are dynamic systems where an N dimensional space is created with each cell containing a value which changes according to pre-determined rules depending on the neighborhood. From this simple local concept, complex global patterns and behavior emerge as the morphing animation considers the collective response of the cells within the lattice. We developed two new

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 80-903100-7-9

WSCG'2005, January 31-February 4, 2005

Plzen, Czech Republic.

Copyright UNION Agency – Science Press

algorithms morphing algorithms using the CA the *core-increment* and the *bacterial-growth* model. The rules that have to be written for each cell in the automata to perform this morphing are simple and hence easy to update or replace. There is no correspondence required between models except that identical 3 dimensional volume sizes are required of the source and destination volumes. No control points are needed to drive the morph. Our approaches work really well in situations where there is no pre-defined transformation path required between the source and target.

## 2. PAST RESEARCH

Cohen et al [Cohen-Or1998] explain the problem of morphing or metamorphosis as follows: ‘Given a source model  $S$  and a target model  $T$ , morphing constructs a series of transformations  $\{W_t \mid 1 \leq t \leq [0,1]\}$  such that  $W_0 = S$  and  $W_1 = T$ ’ [Lerios1995]. A sample morph from one of our algorithms is displayed below (Figure 1).



Figure 1: Gradual transition from source to target.

Mesh Morphing and Volume Data Morphing based upon the data which they use. Polygonal morphing is the process of metamorphosis where the source and target are polygonal or polyhedral meshes. A mesh can be defined as a linear surface that consists of a set of polygons that can be described either as a set of vertex/face/edge/graph structures or as a set of vertices [Kanai2000]. One of the important characteristics of a mesh model is that it describes the topology of the model. It is the basis of intense research with the majority of papers in the morphing field concerning this area. Several methods have already developed for entertainment industry. The input mesh-models are easy to create and many packages support model [Maya2004] creation in this format and light effects are computationally faster to process and render as they are well supported. However, complex topologies *are* difficult to morph, especially if they require user control. In addition, many applications such as medical and geological world produce large amount of volume data and may have to be converted into polygonal mesh. Both correspondence and interpolation problems are documented in Kent et al [Kent1992]. Kanai et al use the concept of merging of two meshes in a common

domain [Kanai2000] and use *harmonic mapping* method [Kanai2000]. Lee et al [Lee1999] describe a process of correspondence that uses both the source and target meshes at several resolutions and coarse *base domains* or simplified meshes.

## 3. VOLUMETRIC MORPHING

Volume morphing uses three dimensional volumes as input for the morphing process. Models are described in terms of voxels (short for volume pixels, the smallest box shaped unit of volume). Chen et al [Chen1995] define a volume as a collection of scattered voxels, with each voxel being associated with a set of values of size  $S$ , i.e, the volume  $V$  is given by:

$$V = \{(x_i, v_i) \mid x_i \in \mathbb{R}^3, v_i \in \mathbb{R}^S, i = 1 \dots n\}$$

The most popular format for representing volume data is in the form of a 3 dimensional grid. Each  $(i,j,k)$  position represents a voxel which has a value associated with it.

The volumetric approach is not as popular as polygonal formats in the entertainment industry. It is computationally intensive to process and render. However, volumetric approach is free of restrictions of topology and geometries. Volume morphing can easily be applied to meshes by converting them to volumetric data while the reverse can result in topologies which are difficult to morph. A large amount of data in the medical, geological and energy fields is generated in the volumetric format and needs to be morphed directly. Most volumetric algorithm do not need a bijective mapping between vertices of the source and target formats like mesh morphing techniques. The simple format allows implementation ease.

## 4. CELLULAR AUTOMATA

Cellular Automata (CA) were originally proposed as formal models of self-reproducing organisms [Sarkar2000]. CAs are dynamical systems that occupy a uniform, regular lattice, work in discrete steps of time and are characterized by local interactions [Wolfram1984]. They utilize a discrete lattice of sites Discrete time steps drive the simulation. Each site can only take a finite set of values. Each site evolves according to the same deterministic rules. The evolution of a site only depends on the neighborhood. The main advantage of CAs is that complex patterns and behaviors can be achieved using simple local interactions. CAs have been used in many applications [Sarkar2000, ] , Bezzi2000, Sloot2002, Hamagami2003, Forsyth2002, Droun2003]. Sosič and Johnson [Sosič1995] use the

concept of CA to describe a growing automaton. Sloot et al [Sloot2002] describe a non-uniform model used to simulate drug treatment for HIV infection. Bezzi [Bezzi2000] describes the simulation of several biological processes using CAs. Claudia et al [Claudia2001] discuss the use of the CAs for simulating the effects of a landslide. [Droun2003] uses a cellular automaton to deforming 3 dimensional objects, not 3D morphing.

Most 3D morphing techniques utilize the idea of correspondence, which is mapping where each point in the source model will end up in the target model [Kanai2000, Lee1999, Kent1992, Chen1995, Cohen-Or1998, Leros1995]. This becomes inconvenient if there are complex topologies that would require many control-points to describe the morphing accurately.

We looked at volumetric morphing as a collection of voxels comprising the source model trying to achieve similarity with the voxels in the target model. The cellular automata which was used in our design has the following characteristics: It is a 3 dimensional lattice. The dimension of the lattice is that of the volume. Each cell can either be empty or contain one value. All cells are equal, in the sense that a change of value within a position does not change the behavior of the entities occupying the automata. The cellular automaton is non-circular.

Using cellular automata as a base, we have developed 2 algorithms that perform morphing. In both cases, the volume is treated as a cellular automaton. Each non-empty voxel is treated as an independent agent.

## 5. CORE-INCREMENT ALGORITHM

The core-increment algorithm works using the intersecting parts of the morphs as a base. The intersection part of the source and the target is used to create a core. The core is then grown or incremented in a step-wise fashion so that it fills the space of both the source and target models. More specifically, for each voxel position present in the core, the source and target models are checked to see if any voxels within them surround the position. If so, the voxels are added to the core at the same position that they were found in the source or target. At each step, the voxel positions that are needed to occupy the space of the source and target are recorded in the delete-array and add-array respectively. The arrays contain the points added to them as separate sets during each iteration. The core-increment process completes when there are no more positions either in the source or target that the core can grow into.

Next, the source model is loaded into a new volume. The add and delete-arrays are scanned, one forward and the other in reverse. At each step, the voxel positions mentioned in the delete-array for that iteration is removed from the source model and the corresponding voxels are added from the add-array. In this way, gradually, voxels are removed from the source model where they do not intersect with the target model and voxels added where the target model is supposed to be. The forward and reverse iterations give the morphing a smooth, organic quality. The pseudo-code is as follows:

```

proc core-increment
    // Loading of volumes and tests
    Load source volume as srcVol;
    Load target volume as tgtVol;
    If ( dimension(srcVol) != dimension(tgtVol) )
        print error and exit endif
    Create core with dimensions of srcVol
    // Initialization of core
    for each voxel position (i,j,k)
        if ( both tgtVol and srcVol's has object present)
            add (i,j,k) to the core at (i,j,k)
        endii
    end for
    // Iteration to create add and delete arrays
    do
        for each non-empty voxel position (i,j,k) in core
            if voxel found surrounding (i,j,k) in srcVol
                add to core at (i,j,k)
                add (i,j,k) to del-array
            end if
            if voxel found surrounding (i,j,k) in dstVol
                add to core at (i,j,k)
                add (i,j,k) to add-array
            end if
        end for
    until core cannot increment further
    Load source volume as morphVol
    // Morph iterations.
    for i = 0 to sizeof(add-array)
        // iterating through the add-array

```

```

    get position at add-array[i] as (i,j,k)
    add voxel at (i,j,k) to morphVol
    // iterating through del-array in reverse
    get position at del-array[size(add-array)-i] as
        (i,j,k)
    add voxel at (i,j,k) to morphVol
    // Rendering the deformed volume
    Render morphVol
endfor
endProc

```

The rules that define the behavior of the cellular automata that makes up the core are simple, hence easy to upgrade or replace. The algorithm uses 3 dimensional arrays containing the position data from volumetric models. This means that most popular formats of representation of volumetric data can be used directly. No complex data-types or intensive pre-processing is required. There is no correspondence required between the source and the target models. In the above example, it is clear that there is no correspondence information present.

The morph can be controlled because of the add and delete arrays containing the information of each iteration as separate sets. In cases where many points are added in the add array as compared to the del array or vice versa, by controlling the sets released per iteration from the arrays, the morph can be made to be a gradual process. This is important in cases where the source model is very small in comparison to the target model or vice versa. In the normal case, if the source were small, the non-intersecting parts of the source would either disappear quickly while the target would grow slowly, or if the target were small, the target would grow to completion while the source was still disintegrating. By coordinating the release of points this problem can be avoided.

By using random probabilities in the points being selected for each iteration, the morphing gains an organic quality (Figures 4 and 5). The growth of the morph can be made to start with an uneven texture to the surface that clears up during the end of the morph to give the texture of the target model. The method requires the creation of four volumes, two for the source and target models to be loaded, one for the core and one for the source model during the morph iterations. Since the implementation of the method results in the first instances of the source and target models being destroyed, the source cannot be reused during the morph iteration. This makes the implementation memory-heavy if very large models are used. The algorithm requires the volumes

containing the source and target models to be of the same dimensions. However, there is no restriction on the size of the models themselves.

## 6. BACTERIAL GROWTH MODEL

Several papers during my research into cellular automata have mentioned its use for simulating the behavior of bacteria given certain environmental conditions. Each bacterium is modeled as an entity within a lattice and rules govern its reaction to the environment and other bacteria. Researchers have succeeded in simulating complex behavior for bacteria using the simple rules required for CAs. This gave rise to the idea of using the bacterial growth model as a method of morphing (Figures 1 and 2).

The following rules govern the behavior of bacteria:

(a) Bacteria are non-motile. (b) All bacteria are of the same type and governed by the same rules. (c) A bacterium has 2 needs, the need for food and the need to reproduce, the latter being dependent on the former. (d) A bacterium will reproduce if it finds food and has space to place its offspring by making a copy of itself. (e) A bacterium creates only one offspring per iteration. (f) A bacterium with food at its current location will live and reproduce infinitely given enough space. (g) A bacterium will die if food is not present in its current position. (h) Bacteria cooperate to keep each other alive. If a bacterium is completely surrounded by other bacteria, it does not die even if its current position contains no food. (i) Each non-empty voxel within the target volume is considered as a source of food. Each source contains an infinite supply of food.

Each 'bacterium' within the source volume checks to see if it has food in its current position. If not, and if it is not completely surrounded in 26 directions by other bacteria, it dies with a certain probability. If it finds food, it looks for a empty place in the neighborhood to reproduce and place its progeny, with a certain probability. In this way, bacteria in parts of the source volume that do not intersect with the target volume begin to die out, thus removing the feature. Bacteria that intersect the target volume begin to breed, placing their progeny in places where the target volume is supposed to be. This results in features of the target growing to form the final shape of the target volume. The pseudo-code is as follows:

```

Proc core-increment
    // Loading of volumes and tests
    Load source volume as srcVol;
    Load target volume as tgtVol;
    if ( dimension(srcVol) != dimension(tgtVol) )

```

```

    print error; exit;
endif
for each non-empty voxel position (i,j,k) in srcVol
do
    if (voxel at position (i,j,k) is non-empty )
        // food at current position
        if ( voxels surrounding (i,j,k) have a
            non-empty position (i1,j1,k1) )
            reproduce by placing copy of voxel at
                (i,j,k) in (i1,j1,k1) with probability p1.
            else if (not completely surrounded by
                voxels at position (i,j,k) )
                die by removing voxel at (i,j,k)
                    with probability p2
            endif
        endif
    endif
    render srcVol;
enddo
end core-increment

```

## 7. RESULTS AND ANALYSIS OF THE PROPOSED APPROACHES

The above two algorithms that were developed were implemented in C++ using the Visualization Tool Kit [vtk2004] library as the rendering engine. The Visualization Toolkit VTK [VTK2004] was free and provided open-source C++ library that supports several graphics related activities including image processing and 3D visualization. It has inherent support for volume data and runs on all popular machine-platforms. The tests of algorithms were done with the following datasets with certain characteristics on a PC with dual Pentium III processors running at 1 GHz with 1.5 Gb memory. In both Table 1 and 2, these cases are identified as (a)-(d) as follows: Case (a) is the morph sequence where source is *Input.bin* and target is *Fuel.bin*. Both these data sets are 64 x 64 x 64 datasets with about 17,000 non-empty voxels (Figure 1). The source model intersects to a large extent with the target. Case (b) is the morph between *Cube256x256x256.bin* to *aneurism.bin*. These are 256 x 256 x 256 sized datasets. There are about 1.1 million voxels in total. The target (*aneurism.bin*) model is dissipated throughout the volume, being branch-like. There is no central core volume as in other models, hence there is very small amount of

overlap between the source (cube) and target (aneurism) model. This leads to a large amount of voxel additions and deletions. Case (c) morphs *Cube256x256x256.bin* to *MRI-head.bin*. Once again the data sets are of size 256 x 256 x 256. They have around 7.1 million non-empty voxels between them. The source is a cube that is centered across the volume. The target volume is a MRI of a head that envelops the cube; hence most of the voxel manipulations are addition operations. Finally, Case (d) is the morph between *MRI-head.bin* to *bonsai.bin*. These are again 256 x 256 x 256 sized datasets. The total of the non-empty voxels of both source and target is 7.3 million. The source model overlaps the target to a large extent and hence, most of the operations in this morph involve the deletion of voxels. Figures 2-5 show the results of our two algorithms.

The best and worst case complexity of this algorithm is  $n^3$  where  $n$  is the size of one dimension of the source or target volume. As shown in Tables 1 and 2, normally we find that the amount of time taken for each iteration as well as the number of iterations depend on the size of the volume and the number of non-empty voxels within it..

In case of test Case (b), the small amount of overlap leads to a large amount of additions and deletions. In this case, the number of iterations became large for the bacterial growth algorithm, with the iterations during the end of the morph yielding very small numbers of voxels. These do not contribute significantly to the quality of the morph. The performance of this algorithm is better than the core increment method described earlier. This should not be assumed to be a reflection of the efficiency of the algorithm. The main reason for this is that the bacterial growth algorithm incorporates an iso-surfacing algorithm. This means that only the voxels on the surface of the intermediate volumes are processed. The core increment algorithm does not easily support such a scheme and hence the current implementation processes all the voxels present in its core. Bacteria growth algorithm seems to do well in cases where there is a large percentage of overlap between volumes. The quality of the morphs is in general worse than the core-increment algorithm and this can be assessed by looking at Figures 2-5.

Table 1 and 2 also show the time taken to create morphing sequences, and the average time taken to complete a sequence during morph.

Our implantation results indicated that the core increment algorithm is more stable and provide better visual results with a varied type of source and target models than the bacterial growth models. In comparison to other existing 3D algorithms, the

solution provides a morph which does not require any human-intervention, and the morphing sequences has better visual appearance as in both cases morphing sequences are expected to grow gradually in spatial domain, avoiding frequency interpolation based aliasing completely.

## 8. CONCLUSIONS AND FUTURE RESEARCH

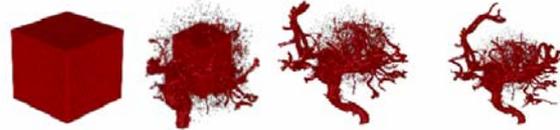
We have presented two algorithms which successfully demonstrate the 3D Morphing. The methods presented in this paper can handle branching structures and topological mismatches, which have been a problem for the past algorithms, without any human-intervention. Our current design requires that the volumes intersect. An important improvement would be to handle is that non-intersecting volumes. Our method can be extended by merging the current design with the distance field metamorphosis technique [Cohen-Or1998] when the volumes do not intersect. Parallelization has been performed on cellular automata based models before [Telford1999] and our method can benefit from that as well. We will like to also consider non-homogenized mixture of bone, tissue etc) in future as well. In addition, we also plan to develop methods to handle color (rgb) volume data sets.

## 9. REFERENCES

- [Abraham2004] Abraham AW. Image View-Shift: Three dimensional representation from a photo. MS Thesis, University of Colorado, Colorado Springs, pp. 1-274, 2004.
- [Beier1992] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis, International Conference on Computer Graphics and Interactive Techniques, pp. 35 – 42, 1992
- [Bezzi2000] Bezzi M. Modeling Evolution and Immune System by Cellular Automata. <http://citeseer.nj.nec.com/429312.html>
- [Breen2001] D. E. Breen and R. T. Whitaker. A level-set approach for the metamorphosis of solid models. IEEE Transactions on Visualization and Computer Graphics Volume 7, Number 2, pp. 173, 2001
- [Bourke2001] Paul Bourke. Terrain morphing. <http://astronomy.swin.edu.au/~pbourke/terrain/tmorp>
- [Calionna2001] Claudia R Calionna ,Claudia Di Napoli, Maurizio Giordano, Mario Mango Furnari and Salvatore Di Gregorio. A network of cellular automata for a landslide simulation. International Conference on Supercomputing Proceedings of the 15th international conference on Supercomputing, pp. 419 – 426, 2001
- [Chittarao2001] Chittaro L. Information Visualization and its Application to Medicine. Artificial Intelligence in Medicine, vol. 22, no. 2, pp. 81-88, 2001.
- [Chen1995] M. Chen and M. Jones and P. Townsend. Methods for Volume Metamorphosis. In Image Processing for Broadcast and Video Production, Y. Paker and S. Wilbur (Eds.), Springer-Verlag, London, 1995.
- [Cohen-Or1998] Daniel Cohen-Or, Amira Solomovic, David Levin. Three-Dimensional Distance Field Metamorphosis. ACM Transactions on Graphics (TOG), Volume 17, 2, pp. 116 – 141, 1998.
- [Droun2003] Druon S, Crosnier A, Brigandat L. Efficient Cellular Automata for 2D / 3D Free-Form Modeling. Journal of WSCG (Winter School of Computer Graphics), 11, 1, pp. 102-108 : 2003
- [Fang96] S. Fang and R. Raghavan and J. Richtsmeier. Volume Morphing Methods for Landmark Based 3D Image Deformation. SPIE International Symposium on Medical Imaging, 1996
- [Forsyth2002] Cellular Automata for Physical Modeling. Game Programming Gems 3, 2002.
- [Gagvani2001] Nikhil Gagvani and Deborah Silver. Animating volumetric models. Volume modeling Volume 63, Issue 6, November 2001, pp. 443–458, 2001.
- [Hamagami2003] Tomoki Hamagami and Hironori Hirata. Method of crowd simulation by using multiagent on cellular automata. IEEE/WIC International Conference on Intelligent Agent Technology, pp. 46 – 53, 2003
- [He1994] T. He and S. Wang and A. Kaufman “Wavelet-Based Volume Morphing” Proceedings of Visualization 94, Pages: 85-92, : 1994
- [Java3D2003] Java 3D API ® Sun Microsystems <http://java.sun.com/products/java-media/3D/>
- [Kanai2000] Takashi Kanai Hiromasa Suzuki Fumihiko Kimura. Metamorphosis of Arbitrary Triangular Meshes with User-Specified Correspondence. IEEE Computer Graphics & Applications, 20, 2, pp. 62-75, 2000.
- [Kent1992] James R Kent ,Wayne E Carlson and Richard E Parent. Shape transformation for polyhedral objects. International Conference on Computer Graphics and Interactive Techniques, pp. 47 – 54, 1992
- [Kazakov2003] Maxim Kazakov, Alexander Pasko and Valery Adzhiev. Interactive metamorphosis and carving in a multi-volume scene. Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia, pp. 103, 2003.

- [K-DLab2004] Perimeter Developer: KD-LAB / IC Company [www.codemasters.com/perimeter](http://www.codemasters.com/perimeter)
- [Lerios1995] Apostolos Lerios, Chase D. Garfinkle, Marc Levoy. Feature-Based Volume Metamorphosis. International Conference on Computer Graphics and Interactive Techniques Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, pp. 449 – 456, 1995
- [Lee1999] Aaron Lee and David Dobkin and Wim Sweldens and Peter Schroder. Multiresolution Mesh Morphing. Siggraph 1999, Computer Graphics, pp. 343-350, 1999.
- [Lee1988] F. Sweldns, W. Schorder, P. Cowsar, and L., Dobkin. Multiresolution Adaptive Parameterization of Surfaces. Computer Graphics, pp. 95–104, 1998.
- [Maya2004] Aliaswavefront Maya ® <http://www.alias.com/eng/products-services/maya/index.shtml>
- [Outrage2003] Alter Echo Developer: Outrage Games Publisher :THQ [www.outrage.com](http://www.outrage.com)
- [Sarkar2000] Palash Sarkar. A brief history of cellular automata. ACM Computing Surveys (CSUR) Volume 32 , 1, pp. 80 – 107, 2000.
- [Sederberg1992] Thomas W Sederberg and Eugene Greenwood. A physically based approach to 2–D shape blending. International Conference on Computer Graphics and Interactive Techniques pp. 25 – 34, 1992
- [Sloot2002] Peter Sloot, Fan Chen, and Charles Boucher. Cellular Automata Model of Drug Therapy for HIV Infection. Lecture Notes In Computer Science Proceedings of the 5th International Conference on Cellular Automata for Research and Industry, pp. 282 – 293, 2002.
- [Sosić1995] R. Sasic and Robert R. Johnson. Computational properties of self-reproducing growing automata. BioSystems, Volume: 36, pp. 7-17, 1995.
- [Sussman2004] Alan Sussman, Michael Beynon, Mary Wheeler, Steven Bryant, Malgorzata Peszynska, Ryan Martino, Joel Saltz, Umit Catalyurek, Tahsin Kurc, Don Stredney, Dennis Sessanna. Exploration and Visualization of Oil Reservoir Simulation Data, 2004.
- [Treece2001] Graham Treece and Richard Prager and Andrew Gev. Volume-based three-dimensional metamorphosis using sphere-guided region correspondence. The Visual Computer, volume 17, 7, pp. 397-414, 2001.
- [Ulgen1997] F Ulgen. A step towards universal facial animation via volume morphing. Robot and Human Communication, 1997. RO-MAN '97 6th IEEE International Workshop, pp. 358 – 363, 1997

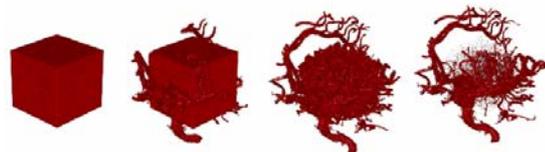
- [Wolfram201984] Wolfram ,Stephen. Universality and Complexity in Cellular Automata. Physica D 10, pp. 1-35, 1984.
- [vtk2004] The Visualization ToolKit (VTK) <http://www.vtk.org/>



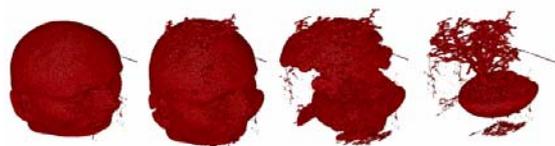
**Figure 2: using bacterial growth model (Cube to Aneurism)**



**Figure 3: using bacterial growth model (Head to Bonsai)**



**Figure 4: Using core increment model (Cube to Aneurism)**



**Figure 5: Using core increment model (Head to Bonsai)**

|   | <b>Volume size</b> | <b>Source non-empty Voxel Count</b> | <b>Target non-empty Voxel Count</b> | <b>No of iterations</b> | <b>Avg Morphing Time per iteration (secs)</b> | <b>Total Time taken for morphing (secs)</b> | <b>Avg Rendering Time (secs)</b> |
|---|--------------------|-------------------------------------|-------------------------------------|-------------------------|---|---|----------------------------------|
| a | 64 X 64 X 64       | 4096                                | 13731                               | 32                      | 0.02  | 0.64  | 0.76                             |
| b | 256X256X256        | 1000000                             | 168948                              | 387                     | 1.16  | 448.92                                      | 3.16                             |
| c | 256X256X256        | 1000000                             | 6198649                             | 132                     | 3.38  | 446.16                                      | 2.14                             |
| d | 256X256X256        | 6176412                             | 1298598                             | 195                     | 1.33  | 259.35                                      | 2.71                             |

**Table 1: Core element results**

|    | <b>Volume size</b> | <b>Source non-empty Voxel Count</b> | <b>Target non-empty Voxel Count</b> | <b>No of iterations</b> | <b>Avg Morphing Time per iteration (secs)</b> | <b>Total Time taken for morphing (secs)</b> | <b>Avg Rendering Time (secs)</b> |
|----|--------------------|-------------------------------------|-------------------------------------|-------------------------|---|---|----------------------------------|
| a. | 64X 64 X 64        | 4096                                | 13731                               | 32                      | .0334   | 1.07  | 1.27                             |
| b. | 256X256X256        | 1000000                             | 168948                              | 248                     | 2.5027  | 620.67                                      | 4.46                             |
| c. | 256X256X256        | 1000000                             | 6198649                             | 120                     | 8.7642  | 1051.71                                     | 3.32                             |
| d. | 256X256X256        | 6176412                             | 1298598                             | 175                     | 11.24   | 1966.24                                     | 3.33                             |

**Table 2: Core element results**

# Enhancing Visual Exploration by Appropriate Color Coding

Petra Schulze-Wollgast  
University of Rostock  
Institute for Computer Science  
A.-Einstein-Str. 21  
18059 Rostock, Germany  
psw@informatik.uni-  
rostock.de

Christian Tominski  
University of Rostock  
Institute for Computer Science  
A.-Einstein-Str. 21  
18059 Rostock, Germany  
ct@informatik.uni-rostock.de

Heidrun Schumann  
University of Rostock  
Institute for Computer Science  
A.-Einstein-Str. 21  
18059 Rostock, Germany  
schumann@informatik.uni-  
rostock.de

## ABSTRACT

Visualization is an effective means for exploring and analyzing complex data. Color coding is a fundamental technique for mapping data to visual representations. Although color coding is widely used in a large variety of visualizations, it is often provided in a limited way only or it is not used effectively. Therefore, we describe in this paper how appropriate (automatic) color coding can enhance the visual exploration of spatial-temporal data. We demonstrate our techniques with a system for visualizing human health data by means of choropleth maps. Furthermore, we focus on how to use color coding for facilitating comparison tasks in visualization.

## Keywords

Visualization, Color Coding, Perception-Based Color Scales, Comparison.

## 1. INTRODUCTION

Visualization is an effective means for exploring and analyzing complex data. Regarding this, color plays an important role. Color coding is a fundamental technique for mapping data to visual representations. Although, color coding is widely used in a large variety of visualizations, it is often provided in a limited way only or it is not used effectively. Furthermore, adapting color scales automatically by applying a simple minimum-maximum-scaling often results in visual representations of different views on the data which cannot be compared with each other.

Therefore, we describe in this paper how appropriate automatic color coding can enhance the visual exploration of spatial-temporal data. This is achieved by taking into account:

- Perception-based color schemes,

- User aims, and
- Characteristics of the data.

We use perception-based color schemes suggested in [Bre94] and [Ber95], which have proven to be effective. From a collection of such schemes we choose the most appropriate one with respect to the users' visualization goal. In this context, we focus on comparison; on the one hand comparison is a major visualization task for interactive data exploration, on the other, comparison is not supported sufficiently by most existing color-based visualization systems. By considering data characteristics we have the ability to combine our color scale legends with Box-Whisker plots. By doing so, users intuitively get more insight into the data. We demonstrate our techniques considering color coded maps, which represent human health data.

The paper is structured as follows. In Section 2 we give a general overview on color scales, describe problems regarding the use of color in visualization and give some guidelines on how color can be used efficiently. Known approaches addressing these issues are reviewed before presenting our approach in Section 3. In Section 4 we describe a system for visualizing spatial-temporal human health data on maps. It is shown how our approach can enhance visual exploration of such data. Section 5 concludes the paper and gives an outlook for future 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.

*WSCG 2005 conference proceedings, ISBN 80-903100-7-9*  
*WSCG'2005, January 31-February 4, 2005*  
*Plzen, Czech Republic.*  
Copyright UNION Agency – Science Press

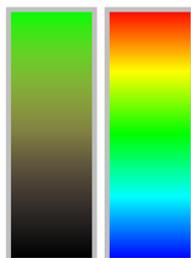
## 2. COLOR SCALES

Color is a retinal variable, which is very effective for mapping (abstract) data [Ber83]. This is due to the spontaneous perception of color by the human visual system. Since for now the interrelation between the physical phenomena color and its perception by the human visual system is not fully understood, color scales have to be chosen carefully to fully utilize effectiveness of color-based visual representations and furthermore, to avoid misinterpretations.

A color can be described as a point in a 3-dimensional color space. In technical applications the primary colors red, green and blue (RGB) span the dimensions of the color space. Hue, saturation and brightness (HSB) are used as dimensions in perception-based applications. A color scale provides a range of colors varying in hue, saturation, and/or brightness. Color scales are defined by:

- A set of control points and
- A mapping function describing the transition between colors.

Control points associate parameter values with colors. They are utilized to add colors to a color scale according to a parameter range  $t: 0.0 \leq t \leq 1.0$ . A color scale consists of at least two control points, one for  $t = 0.0$  and one for  $t = 1.0$ ; however, more control points could be used to create more sophisticated color scales. The mapping function describes how color is interpolated between two control points. A requirement for color coding is that the value range to be displayed must be scaled to  $[0.0; 1.0]$ , the range of parameter  $t$ .

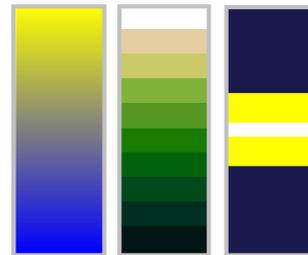


**Figure 1** A standard RGB-based color scale and a rainbow color scale.

Nowadays visualizations mainly use, on one hand, color scales that are created by linearly interpolating two colors from the RGB color space (e.g.  $t(0.0) = black$  and  $t(1.0) = green$ ; cp. Figure 1). On the other, a rainbow color scale is used. This scale contains all spectral colors from blue to red (i.e. the colors appearing in a rainbow). Though these scales are intended to linearly map colors to a scalar value range, the resulting color scales are not perceived as linear. Quite the contrary, users perceive differently

sized regions in the color scale, which show variance not only in hue, but in saturation and brightness as well (cp. Figure 1). Therefore, providing only standard color scales for visualization is not sufficient.

In literature several approaches are known addressing the creation of color scales or giving guidelines for the use of color. Brewer describes the use of color for mapping data on cartographic maps [Bre94, Bre99]. Regarding this, binary, qualitative, sequential and diverging color schemes are described. The schemes are based on human perception and are, therefore, a good basis for creating effective visualizations. Additionally, the suggested color schemes have been evaluated considering different output devices like CRT screens, TFT displays or LCD projectors. However, all these scales aim on categorized (i.e. segmented) maps; special scales for continuous quantitative data are not provided. Such scales can be found in PRAVDAColor [Ber95] developed by Bergman et al.. The authors describe a rule-based mechanism for supporting users in choosing appropriate color scales. In order to decide what scale fits best, data type (ratio or interval data), spatial frequency (low or high), and representation task (isomorphic representation, segmentation, highlighting) are taken into account. Figure 2 shows how color scales for different visualization tasks may look like.



**Figure 2** Color scales for isomorphic representation (left), segmentation (middle), and highlighting (right).

Though these works highly support developers in designing as well as users in choosing effective color scales for visualization, most of today's visualizations barely utilize them. Another aspect currently still underestimated is the dependency on properties of the data (e.g. the distribution of data values). Though a general solution is hard to find – if this is possible at all – an integration of known concepts complementing one another is a vital step to further facilitating the use of colors in data representation.

A modern visualization system, therefore, should provide all: potentially effective color scales, methods for choosing and adapting color scales according to data characteristics, and intuitive interactive tools to enable users to adapt color scales according to their needs.

### 3. ENHANCED COLOR CODING

In this section we will review factors influencing color scaling. Based on this, we introduce our approach for enhancing effectiveness of color-based visualizations.

#### 3.1. Factors influencing color scaling

Creating effective color scales for data visualization is not a trivial task. There exist no general guidelines or methods for choosing color scales automatically. This is due to the complexity of factors influencing the decision for a concrete color scale.

We identified 3 main categories of factors:

- Data properties,
- Visualization goal, and
- General context.

In the following, we will focus on these categories in more detail. Regarding data properties we subcategorize: scaling of the value range and statistical characteristics.

The *scaling of the value range* is considered for each variable within a data set. Data variables can be of nominal, ordinal, or quantitative scaling. While for nominal variables no ordering of data values is given, ordinal variables comprise an order of the data values. This has to be considered for the visualization. For achieving effectiveness, color scales for nominal variables must NOT and color scales for ordinal variables must imply an ordered perception of colors used. Furthermore, quantitative variables allow for distances between data values. Therefore, perceptual distances within a color scale must reflect distances in the data. Moreover, a color scale should reveal whether a data variable contains a special zero value sectioning the value range (i.e. ratio data).

By considering *statistical properties* of a variable (e.g. element count, minimum, maximum, average, mean, quartiles, etc.) the effectiveness of color scales can be improved. This can be realized mainly by adapting the control points or the mapping function rather than by choosing certain colors.

The goal a user wants to achieve when using visualization has much influence on the choice of color scale. Bergman et al. [Ber95] differentiate between isomorphic representations (user seeks an exact image of the data), representations of segmentation (user intends to detect segments within the data), and representations for highlighting (user is interested in particular values). Besides these goals a variety of further tasks are possible (e.g. comparison, detection of correlation or clusters, etc.). Our interest especially regards comparison tasks. Since comparison is one of the main tasks in data exploration, we do not

limit our considerations on intercomparison within a single visual representation but detail also on comparison of different representations of varying portions of the data (e.g. different time steps or different regions).

The third category of influencing factors is related to general context. Regarding this we identified the following aspects: perception of color, colorblindness, output device, and user preferences.

Regarding *color perception* the capabilities of the human eye can be considered. Though this issue is hard to grasp due to the complexity of the human visual system, some perception aspects can yet be considered for designing color scales. So it is possible to take visual resolution into account. Visual resolution regards to what degree small and differently colored spatial structures can be visually distinguished. Furthermore, paying attention to adaptation mechanisms of the eye (e.g. a negative afterimage of what has been previously seen shortly remains on the retina), to the relation of color and size (i.e. differently sized objects of same color are perceived as differently colored), as well as to time dependent color perception (i.e. perception changes over time and under different environmental lightings) could enhance color scales; though these aspects are hard to integrate into a real system. However, *colorblindness* and the addressed *output device* can be taken into account easily by providing parameters users can adjust. Even if users do not know whether they are colorblind or not, simple tests could reveal this (cp. [Mey88]).

| Mandatory factors           | Optional factors                |
|-----------------------------|---------------------------------|
| Scaling of the value range  | Colorblindness                  |
| Statistical characteristics | Output device                   |
| Visualization goal          | Cultural environment            |
| Visual resolution           | Adaptation mechanisms           |
|                             | Relation of color and size      |
|                             | Time dependent color perception |
|                             | Interaction of different colors |

**Table 1: Factors influencing color scaling.**

By considering *users preferences* (e.g. favorite color) visual analysis can be enhanced in general. When visualization is used to communicate facts found in the data among users of different cultural backgrounds, the relevance of user preferences is even increased. To be more concrete, in each culture different colors may be associated to different things.

The color red, for instance, is associated in Germany with danger, in Egypt with death, in India with life, and in China with happiness. This example underlines the difficulty of solving the general color coding problem.

In order to assess all the mentioned factors we distinguish mandatory and optional factors (cp. Table 1). This distinction is based on the relation between effort for integrating a factor into a real system and resulting benefits for the effectiveness of the color scales.

### 3.2. Automatic color coding

We have developed our approach (cp. [Rut03] for detailed description) based on previous work by Brewer [Bre94, Bre99] and work by Bergman, Rogowitz, and Treinish [Ber95, Rog96, Rog98]. Namely, we use a collection of color scales suggested in these publication. It is important to mention that all of these color scales are perception-based and are, therefore, potentially effective for visualization tasks. Furthermore, we follow [Ber95] in using their rule-based approach. Depending on scaling of the value range (nominal, ordinal, quantitative, or ratio data) and the user's visualization goal (isomorphic, segmentation, or highlighting) the most suitable color scale is chosen. We extend the approach of Bergman et al. [Ber95] by:

- Extracting statistical metadata from the data set,
- Adapting the chosen color scale according to the metadata, and
- Creating an expressive legend for the chosen color scale.

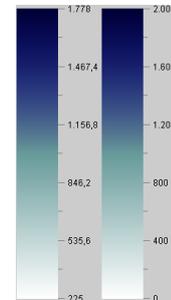
**Extracting metadata** Statistical characteristics can be easily extracted from a data set. We use average, median, mode, minimum, maximum, skewness, and quartiles as metadata for the automatic adaptation of the mapping function for a chosen color scale. While all of these statistics can be calculated for quantitative data, for ordinal data media, mode, and quartiles can be determined only. In case of nominal data mode is the only characteristic being considered.

**Adapting the color scale** Based on these metadata we allow for the following automatic adaptations of a chosen color scale:

- Expansion of the mapped value range,
- Adjustment of control points, and
- Alteration of the mapping functions.

*Value range expansion* is used to create an adequate value range for the color scale mapping. Additionally, the lower and upper bounds of the value range are intended to be intuitively comprehensible. When considering dynamic data sets lower and upper

bounds of the same variable might change. Especially in this context, value range expansion allows for coherent visualization. For realizing value range expansion the lower and upper bounds are calculated according to the minimum and maximum values of a variable. An example (cp. Figure 3) for this is the expansion of a variables' range from 225 to 1778 to a range of values reaching from 0 to 2000.

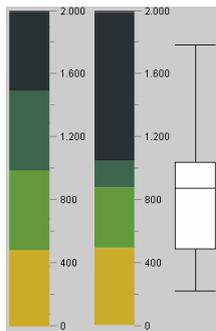


**Figure 3** Range expansion is used to increase the comprehensibility of color scales.

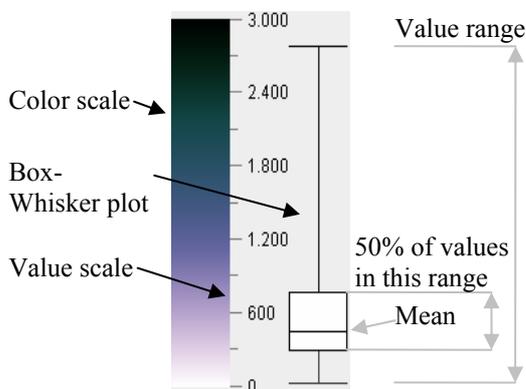
*The adjustment of control points* is mainly used for improving color scales for ratio data as well as for segmentation and highlighting scales. Color scales for highlighting tasks are adjusted by setting a special control point (i.e. a control point denoting the value to highlight) according to average, median, or mode of a variable. For ratio data it is also possible to consider a “real” zero for adjustment of control points. Color scales for segmentation could be adjusted based on quartiles. With respect to the number of segments to be differentiated, quartiles are calculated, containing the same number of values each. The control points of the color scale are then positioned according to these quartiles. This results in a color scale that supports the user in the detection of similar regions within the data. Figure 4 depicts how a standard segmentation color scale (for 4 segments) can be adapted according to quartiles.

*An alteration of the mapping function* might become necessary for data with certain value distributions. Usually, the mapping function performs a linear interpolation between the colors associated to the control points. Since these colors are perception-based, this is effective and the structure of the data is accurately represented. However, linear interpolation leads to problems if the values of a variable are not uniformly distributed (e.g. if outliers are present). In this case a wide range of the color scale represents a small number of values and the majority of values has to cope with only a narrow range on the color scale. Similar problems are known from the field of computer vision. There histogram equalization is used to handle unfavorable color distribution in images. We deal with this problem by means of nonlinear mapping functions. In order to decide how to adjust the mapping function, we take skewness of the

value distribution into account. Depending on whether a variable has positive or negative skewness, we apply an exponential or logarithmic mapping function. By doing so, we “stretch” the range of colors used for the majority of data values. The visualization is improved in a way that differences between values can be more easily detected for values from the range of high value density (cp. Section 3.3 for examples). It is very important that the color legend clearly reveals the color scale as exponential or logarithmic. Otherwise, misinterpretations are inevitable.



**Figure 4** Adaptation of a segmentation color scale according to quartiles. Note that on the adapted scale (right) the majority of values (50% of the value range) are exactly encoded by the two green segments.



**Figure 5** A color legend enhanced with a Box-Whisker plot.

**Creating a color legend** In order to achieve an easy comprehensibility of the data and the used color-coded visualization, a color legend has to be provided. The color legend should show all used colors and an additional scale, which allows an association of characteristic data values to a color. By using the method of value range expansion, we ensure that the scales of our color legends represent characteristic data values. Moreover, we provide a Box-Whisker plot attached to the color legend (inspired by [And01]). Such a color legend is presented in Figure 5. By doing so, users get better insight to the distribution of data values.

### 3.3. Considering comparison tasks

Comparison is an essential task when visualization is used for data exploration/analysis. In literature, only few publications explicitly focus on this essential task. Therefore, we will describe comparison in more detail.

Comparison is used to assess characteristics of the data. Before comparing an aspect of the data has to be chosen regarding which comparison is performed. A requirement for comparison of objects is the equality of the objects according to a common basis (e.g. objects from the same domain). If this requirement is satisfied objects are comparable. Then an object of reference has to be chosen. Other objects can then be compared with respect to the reference object. This means that the process of comparison consists of four steps:

1. Choose an aspect for comparison
2. Check for comparability
3. Choose a reference object
4. Check for equality or differences.

Comparison facilitates 3 basic tasks regarding the assessment of data characteristics. Depending on the scaling of the value range the following basic tasks can be performed. For nominal variables only equality or inequality can be checked ( $a = b$ ;  $a \neq b$ ), regarding ordinal variables the ordering of objects can be determined ( $a < b$ ;  $a > b$ ), and for quantitative data the detection of an amount of difference is possible ( $a = b + \Delta$ ).

The duration of comparability is another aspect that has to be taken into account. Regarding this aspect we distinguish:

- Intercomparison in one single visualization,
- Comparison in one single visualization session,
- Comparison among multiple visualizations sessions, and
- Long term comparisons.

The difficulty of supporting comparison tasks by appropriate color coding increases with the duration of comparability. If the data is represented in only one view a single effective color scale has to be created (cp. Section 3.2). To support comparison in one visualization session a single color scale should be used during the whole session. This ensures that visual representations created later in a session can be compared to views, which have already been analyzed. Human color memory is notoriously poor, which makes comparisons across different visual representations difficult. To alleviate the problem of

comparison among multiple visualization sessions and long term comparison, it is necessary to save color scales used in one session and import them into another session. Moreover, a special “color memory” could be used. This means that the visualization system remembers which color scales have been used to visualize which variables. This memory can then be used to ensure that the same variables are encoded using the same color scales (i.e. ensure visual continuity) and to avoid using the same color scale for different variables (i.e. avoid visual misinterpretations). By doing so, a binding between color scale and encoded variable is established in the users’ mind. Since this procedure is limited to a rather small number of variables (6-8), a “color memory” should be created for each data set. By using the concept of “color memory” we support comparison tasks among different visualization sessions and long term visualization tasks are facilitated.

For visualization of spatial-temporal data on maps the following specific comparison tasks can be refined:

1. Comparison of different regions of a map.
2. Comparison of different time steps of a data set.
3. Comparison of different variables of a data set.

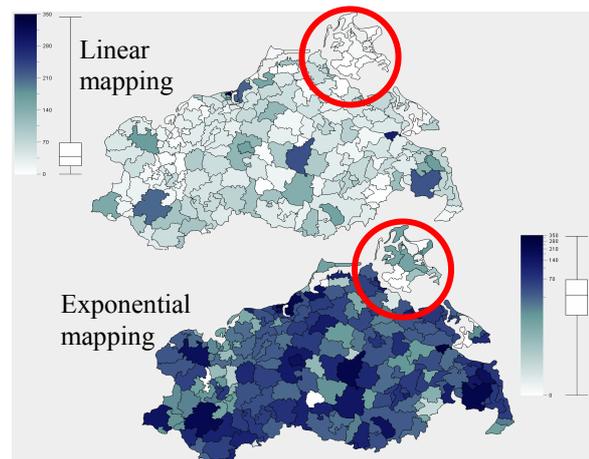
Intercomparison of regions of the map is supported by adjusting the mapping functions for variables with positive or negative skewness. This can be seen in Figure 6. When using a linear mapping, the regions of the island Rügen seem to have equal values. However, when using exponential mapping users can clearly detect that differences exist. Note that the attached Box-Whisker plots give an idea of the distribution of the data values.

For comparison of different variables or time steps multiple view techniques are suitable, where each of the views shows a map color coding one time step or one variable. However, if more than one view is used, the construction of a color scale is more difficult. This is due to the requirement for effectiveness of the color scale not only for a single but for all views.

When comparing different time steps two opposing goals exist. On the one hand, a global color scale can ensure comparability among the views, but regions within a single view might become hard to distinguish. On the other hand, local color scales can encode each view effectively, but comparison of time steps is hardly possible. In order to alleviate this problem, we collect metadata for the entirety of data represented in the views to be compared and select the most suitable color scale. Furthermore, value range expansion is applied to the data range common to all views. Based on this, we set a global color

scale for all views. Additionally, each view is equipped with its own local color legend including a Box-Whisker plot representing the statistical characteristics of the respective view.

In order to support comparison of different variables, the described “color memory” can be used. Moreover, if the variables to be compared have similar value ranges it is also possible to generate a global color scale and to provide local color legends (analogous to time step comparison).



**Figure 6 Using an exponential mapping function supports comparison of different regions.**

#### 4. COLOR CODING FOR VISUALIZING HUMAN HEALTH DATA ON MAPS

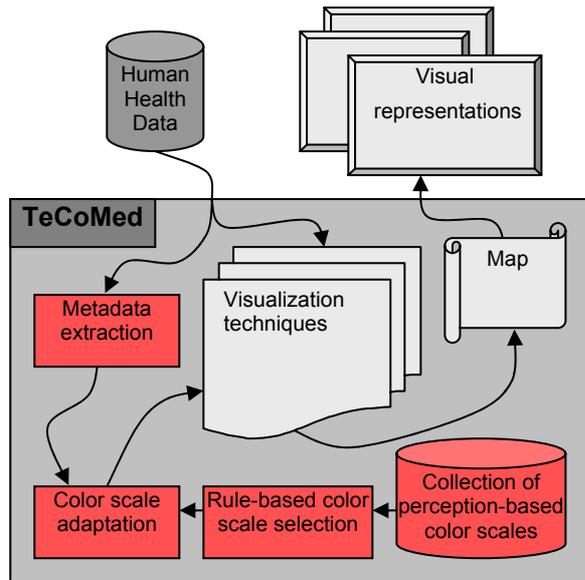
The human health data we consider consists of the number of cases for a variety of diseases. These data depend on time (i.e. number of cases per week) and space (i.e. number of cases in different regions). The system TeCoMed (*Tele Consultation for Medics*) [Sch03] has been developed for visualizing such data via the Internet. TeCoMed provides a rich functionality to select diagnoses, time steps (e.g. day, week, month, quarter and year) and geographical regions interactively.

A variety of concepts for visualizing human health data according to their spatial and temporal dependencies within different levels of granularity have been realized. Among these visualization facilities, color-coded maps (i.e. choropleth maps) are a key feature of the system. Our approach of choosing appropriate color scales automatically has been integrated to TeCoMed in order to facilitate the effectiveness of color coding. Figure 7 shows the integration of the approach into the architecture of the system TeCoMed.

The metadata extraction performs a statistical analysis of the data to be visualized. Since our human

health data have a quantitative scaling, we consider minimum, maximum, mean, and quartiles (for Box-Whisker plots).

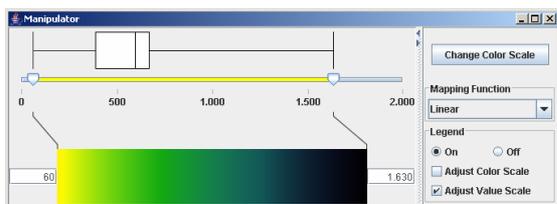
The collection of color scales has been built up by color scales provided by Brewer [Bre94] and Bergman [Ber95]. All color scales are evaluated regarding value range scaling, visualization task, colorblindness, and output device. The use of this collection is not limited to the visualization of human health data and could, therefore, be integrated into other systems as well.



**Figure 7** Integration of the approach into TeCoMed.

Since TeCoMed targets physicians and apothecaries, it was our aim not to confuse users with lots of parameters to set. Therefore, we created predefined parameterizations and associated them with verbal questions. During visualization we then provide a selection of questions users can choose from. Regarding to what question was chosen, the associated parameterization is used to select an appropriate color scale.

Here, the extracted metadata are used to further adjust the chosen color scale. However, not all issues could have been taken into account for automatic color scale adjustment. Therefore, color scales can be adjusted interactively as well. We provide a color manipulator for altering a variety of parameters (e.g. value range adaptation; cp. Figure 8).



**Figure 8** The color manipulator.

In the following example we demonstrate how appropriate color coding can enhance comparison tasks for the visual analysis of human health data in the system TeCoMed. In Figure 9 an isomorphic color scale for quantitative data was chosen to color code 3 time steps of the same disease. The upper 3 maps of the figure have been created without considering comparison. To a user it seems that all time steps contain similar data values. The 3 maps in the lower part of the figure have been created according to a user's decision to compare the 3 time steps. Value range expansion has been applied in order to be able to adjust the color scale to the data range common to all time steps. It can be seen that by doing so, regions of one map can be easily evaluated regarding corresponding regions on the other maps and differences between regions can be quite correctly determined. A general decrease of data values from time step 1 to time step 2 and an increase from time step 2 to time step 3 can be revealed as well. Furthermore, the enhanced color legend for each map further supports the understanding of the data. By utilizing statistical metadata, Box-Whisker plots can be provided for each time step. This helps users in comprehending the distribution of the data values of a certain time step with respect to the created common data range.

## 5. CONCLUSION AND FUTURE WORK

Color coding is an effective means for visual representation of (abstract) data. However, color scales have to be carefully chosen to facilitate this effectiveness. Though previous work addresses this problem, the proposed approaches are barely used for today's visualization techniques.

For our approach, we reviewed factors influencing the decision what color scale to use. Based on [Bre94] and [Ber95] we have developed an integrated rule-based approach for choosing appropriate color scales automatically. Our approach is enhanced by automatic adjustments of the chosen color scale. This is realized based on statistical metadata extracted from the data to be visualized.

One of the main tasks in visualization is comparison. Therefore, we aimed to support this particular task in our system. Several concepts have been introduced addressing this problem.

Our approach has been integrated into a system for visualizing human health data. Here, comparison tasks are highly relevant. By using appropriately chosen and adjusted color scales, we have been able to enhance the visual exploration of such data.

In the future we intend to review the rule-based process for color-scale selection. We try to formulate more questions users might have regarding the analy-

sis of health data (e.g. “Where are extreme values?”). Moreover, we are going to apply further automatic adjustments to a chosen color scale. Especially, the adjustment of highlighting scales may be improved.

Finally, it must be mentioned that user evaluations are still outstanding. This is a special focus for future research.

## ACKNOWLEDGMENTS

Our thanks go to Thomas Ruth for his valuable work for this paper; he did most of the implementation work. Further thanks go to René Rosenbaum and Georg Fuchs for proof-reading the paper.

## REFERENCES

[And01] Andrienko, G. and Andrienko, N.: Interactive Maps for Visual Data. International Journal of Geographical Information Science, Vol. 13, No. 4, pp. 355-374, 1999.

[Ber83] Bertin, J.: Semiology of Graphics. The University of Wisconsin Press, 1983.

[Ber95] Bergman, L., Rogowitz, B.E., and Treinish, L.A.: A Rule-based Tool for Assisting Colormap Selection. Proceedings of IEEE Visualization '95, pp. 118-125, 1995.

[Bre94] Brewer, C.A.: ColorUse Guidelines for Mapping and Visualization. In: Visualization in Modern Cartography. Elsevier Science, Tarrytown, NY, pp. 123-147, 1994.

[Bre99] Brewer, C.A.: Color Use Guidelines for Data Representation. Proceedings of the Section on Statistical Graphics, American Statistical Association, Alexandria VA, pp. 55-60, 1999.

[Har03] Harrower, M.A. and Brewer, C.A.: ColorBrewer.org: An Online Tool for Selecting Color Schemes for Maps. The Cartographic Journal, Vol. 40, No. 1, pp. 27-37, 2003.

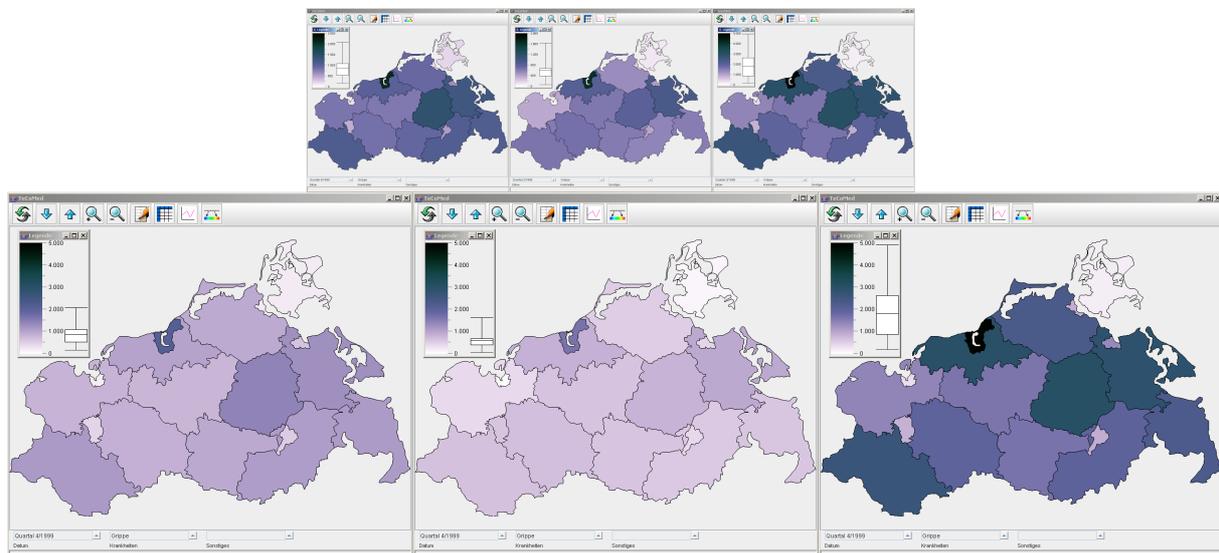
[Mey88] Meyer, G.W. and Greenberg, D.P.: Color-Defective Vision and Computer Graphics Displays. Computer Graphics and Applications, Vol. 8, No. 5, pp. 28-40, 1988.

[Rog96] Rogowitz, B.E. and Treinish, L.A.: How NOT to Lie with Visualization. Computers in Physics, Vol. 10, No. 3, pp.268-274, 1996.

[Rog98] Rogowitz, B.E. and Treinish, L.A.: Data Visualization: The End of the Rainbow. IEEE Spectrum, Vol. 35, No. 12, pp. 52-59, 1998.

[Rut03] Ruth, T.: Möglichkeiten und Grenzen der automatischen Farbskalierung unter dem Gesichtspunkt der Vergleichbarkeit von Visualisierungen. Diploma thesis, Institute for Computer Science, University of Rostock, 2003.

[Sch03] Schulze-Wollgast, P., Schumann, H. and Tominski, C.: Visual Analysis of Human Health Data. Proceedings of the International Resource Management Association 14<sup>th</sup> International Conference, Philadelphia, 2003.



**Figure 9** Comparison of time steps. The upper maps show 3 time steps each visualized without taking care of comparison aspects. The time steps seem to have similar underlying data. The lower maps have been created based on a user’s decision to compare the 3 time steps. The common color scale allows the comparison of the maps. Contrary to the upper maps it can be seen, how data changes over time. Moreover, it can be clearly stated that the data values decrease from the first time step (left) to the second (middle) and then increases from the second time step to the third (right). The Box-Whisker plots further ease the understanding of the data for each view.

# A Motion Constrained Dynamic Path Planning Algorithm for Multi-Agent Simulations

T. R. Wan

Department of EIMC,  
School of Informatics,  
University of Bradford,  
Bradford, West Yorkshire, UK,  
BD7 1DP  
T.Wan@Bradford.ac.uk

H. Chen

Department of EIMC,  
School of Informatics,  
University of Bradford,  
Bradford, West Yorkshire, UK,  
BD7 1DP  
H.Chen3@Bradford.ac.uk

R.A. Earnshaw

Department of EIMC,  
School of Informatics,  
University of Bradford,  
Bradford, West Yorkshire, UK,  
BD7 1DP  
R.A.Earnshaw@Bradford.ac.uk

## ABSTRACT

In this paper, we present a novel motion-orientated path planning algorithm for real-time navigation of mobile agents. The algorithm works well in dynamical and un-configured environments, and is able to produce a collision-free, time-optimal motion trajectory in order to find a navigation path. In addition to the motion constraint path planning, our approach can deal with the unknown obstacle-space terrains to moving agents. It therefore solves the drawbacks of traditional obstacle-space configuration methods. Multi-agent behaviour has been explored based on the algorithm. In the simulation a simple physically-based aircraft model has been developed, which is addressing the manoeuvring capabilities of the moving agents, while the moving agents' accelerations and velocities are always continuous and bounded. The generated motion path is constituted smoothly and has continuous curvature on the whole state space of the motion, thus satisfying the major requirement for the implementation of such strategies in real-time animation or in simulation applications in VR environments.

## Keywords

Motion modelling, Constraint motion, path planning, trajectory generation, multi-agents.

## 1 INTRODUCTION

Path planning with motion modelling is an important and challenging task that has many applications in the fields of AI, virtual reality, autonomous agent simulation, and robotics. The various approaches reported have different criteria to be met, which result in a number of algorithms, and provide solutions to specific application problems. The basic task for the motion constraint path planning is to perform navigations from one place to another by co-ordination of planning, sensing and controlling whilst maintaining a smooth motion trajectory. Navigation may be decomposed into three sub-tasks: mapping and modelling the environment; path planning and generation; and path following and collision avoidance. Path-finding is properly the most popular and frustrating game AI problem in computer game

industries [Ari01, Tat91]. Early work was concentrated on offline planners. These methods used the map of the environment to produce a configured path [Oom87, Lum90]. The common ground for this type of method is that the planner must have full information about the environment [Pad00, Lat91]. Most of the current successful approaches lead to some sort of graph search strategy [Jos97]. An approach called line intersection was proposed when the data consisted only of geometrical objects. The objects here were solid and all the space not occupied by an object was considered unobstructed. There was no variation in vehicle speed or other parameters. The idea was to construct the convex hull of all objects using vertices and connect all vertices with edges. These edges are then filtered to find the shortest valid path between source and destination along a series of edges using standard graph algorithms. These methods suffer from the problem of a rapid increase in computation time and memory for large and complex maps [Mon87, Hol92]. Another popular approach is called weighted graph [Woo97], which divides the search space into a number of discrete regions, called cells, and restricts movement from a particular space cell to its neighbour. Neighbouring cells are those that can be directly reached from a particular cell. A weight function is defined by the cost of the connection

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 80-903100-7-9  
WSCG'2005, January 31-February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency – Science Press*

between neighbour cells [Woo97]. A\* is the most popular algorithm of this kind, which uses the weighted graph idea. Recently an approach called path planning algorithm D\* [Ste95, Yah98] was reported, which resembled the A\* algorithm for applications in partially known environments, and only achieved limited success [Yah98].

Given that motion factors must be taken into consideration, the main problem is to generate a smooth trajectory curve by joining two distinct configurations in the space with constraints using interpolated piecewise polynomials. Curves have been an object of mathematical study and also a tool for solving technical problems and applications. The construction of smooth curves is the trajectory generation for moving agent path planning and motion control, for example, the simulation of an autonomous car or aircraft in a virtual environment, and the robot motion trajectory control in the real world. A  $C^2$  smooth curve is necessary for the agent or its control system to track either in a virtual environment or in the real world. Motion trajectory was discussed by L.E.Dubins [Dub57], who proposed the solution using straight segments connected with tangential circular arcs of minimum radius and proved that the shortest distance between two configurations was such a path. However it is important to note that the curvature along such a trajectory curve is discontinuous; the discontinuities occur at the transitions between segments and arcs. The non-continuous curvatures may result in difficulties in agent control. Continuous-Curvature Curve (3C) generation has become a key technique for on-going research in this area. A few types of splines have been proposed to solve this problem. Yamamoto et al [Yam99] studied the B-spline-based path planning for finding the time optimal trajectory; Tomas et al [Ber03] used the bezier curve in path planning, having considered minimizing the square of the arc-length derivative of curvature along the curve. Y. Kanayama and N.Miyake [Kan86] suggested that using clothoid curve would form a smooth path with continuous curvature, and the resulted curve curvature varies linearly along the path. Later, Y.Kanayama and Hartman, B. I [Kan89] proposed another solution using a cubic curve. A. Scheuer and Th. Fraichard [Sch96, Fra01] used clothoid curves in their vehicle control experiment. Bryan Nagy and Alonzo Kelly [Nag01] adopted cubic splines in their trajectory generation algorithm. However, previous work has mainly been focused on the static trajectory generation problem and on finding the solutions for 2D applications.

In this paper, we propose a new approach for motion modelling for autonomous moving agents in virtual environments. We improve the conventional A\*

method by developing a dynamical visible point detection system, which allows the system to update its optimised node system based on the viewed vision field, rather than the pre-configured environments, to find a smooth motion path in real-time. Our method is capable of dealing with dynamic unknown environments using motion constraints and is very efficient in terms of computational cost.

## 2 UNKNOWN ENVIRONMENTS

One of our objectives in the current work is to study and develop a path planning algorithm for autonomous agent navigation or exploration in unknown environments. The task can be divided into three parts, plan a main path according to the pre-information, keep tracking the difference between the map and the real environment, and then locally amend the pre-designed path. This strategy can efficiently use the available information and reduce the re-planning time. Navigation in an unknown environment is a more challenging topic; at the same time it is also the most promising technology that could be used for generic applications. For example, unmanned robots with navigation ability in unknown environments could achieve tasks in many dangerous places that humans would not wish to entry for safety or health reasons. Navigation is an important part of AI. Navigation in an unknown environment means no pre-information is available before the path-planning algorithm has been executed. The self-guided agent uses the sensor equipped to detect the surrounding environment and obtain the local information. It then uses the local information to generate the path to the destination. In our current work, a virtual environment has been built, which consists of a terrain with unknown configuration and obstacles. A hierarchical strategy has been developed in the current work for creating a navigation environment using raw terrain data.

## 3 MOTION DYNAMICS

Representing all the motion characteristics by analytical equations can be unpractical. A simplified motion model is considered in the current work. The moving agent model developed has six degrees of freedom and the dynamics of the model can be represented as a set of motion parameters in terms of mass, accelerations and steering angles as well as external force conditions, such as air resistance or ground frictions. The dynamics of a moving autonomous agent must follow the basic law of motion dynamics, which may be represented as a set of general ordinary differential equations in the form:

$$\frac{d^2 X}{dt^2} = f(X, \dot{X}, \mathbf{d}) \quad (1)$$

where,  $X$ ,  $\dot{X}$ ,  $\hat{\mathbf{I}}$ ,  $\hat{\mathbf{A}}$  which is the motion state of the agents and its first derivative, and  $\mathbf{d}$  is the motion

control input. We can recast the equation for our motion optimisation problem in the form:

$$\frac{d^2 X}{dt^2} = \hat{f}(X, \dot{X}, \mathbf{d}) + \Delta(\mathbf{d}) \quad (2)$$

$$\Delta(\mathbf{d}) = f(X, \dot{X}, \mathbf{d}) - \hat{f}(X, \dot{X}, \mathbf{d}(\hat{\mathbf{a}}, \hat{\mathbf{q}})) \quad (3)$$

where,  $\hat{X}$ ,  $\hat{\mathbf{a}}$  and  $\hat{\mathbf{q}}$  are approximate values of motion velocity, acceleration and direction of motion (i.e. a steering angle) respectively, and the motion control  $\mathbf{d}$  is a function of acceleration  $a$  and moving direction  $\mathbf{q}$ . A desired or predicted motion state of the moving object is pre-estimated by a set of approximate functions according to the state of moving object and the environment conditions related to the surrounding obstacle-space. The actual motion track is then computed. The difference between the predicted motion and the actual motion will be used for estimating the control input to the motion system above.

#### 4 PERCEPTION AND CONTROL ARCHITECTURE

In our system the "visual sensor" captures the information about the environment. It is a simple method to compute which parts of objects can be seen from the location of the agent. A virtual camera performs perspective projection, all points along a line pointing from the optical centre towards the location of the agent are projected to a single point. We assume all the obstacles are opaque. The mutual occlusion of objects and self-occlusion are analysed. The maximum detection distance and view angle are then calculated. All the obstacles out of the maximum detection distance or view angle are assumed to be invisible. If in one direction there is no obstacle within the maximum detection distance, we can use the point at the end of the detection distance as the flag in this direction.

The motion control for the agent moves through a field of obstacles to a goal, which includes finding and predicting an optimised path and controlling its motion parameters in order to follow this path. We

use the information from the virtual vision sensor to identify the key obstacle points and edges, then create and add the obstacle nodes and path nodes to the vision system. One of the advantages of our approach is that the generated desired or predicted path is dynamic but it is not necessarily the ones the mobile agent must pass exactly at a given time and the actual motion track is therefore smoother in terms of curvature. The position errors between exact desired path nodes and the actual motion track are then used to modify the motion parameters. Another advantage over other methods is that our approach is quite robust with respect to errors and external disturbances. If both errors and the disturbances are within certain bounds, the algorithm can still work effectively. The architecture of the system is shown in Figure 1.

#### 5 PATH-PLANNING ALGORITHM

The Path-Planning in our work can be stated as follows: given an arbitrary rigid polyhedral object, P, and polyhedral environment, find a continuous collision-free and smooth motion trajectory path taking P from some initial configuration to a desired goal configuration. For a path planning problem, the A\* algorithm appears to be an obvious option, and it is so far the most widely used searching algorithm associate with heuristic search. A\* explores paths from an initial state in a systematic manner whilst paying a little attention to the path finding cost. It is the optimal solution under certain conditions. The A\* path planning algorithm will execute actions (sequence of actions from a state to another in the state space) after the shortest path has been found. However, it is basically an offline search algorithm and can not be used for unknown or dynamic environments. Our method uses dynamically allocated points through on-line searching, sensing and reasoning in the environment. At any location of the environment, we could find the points of visibility that are concerned with the co-ordination of the agent. The information perceived is analysed and the resultant path points are recorded and used to

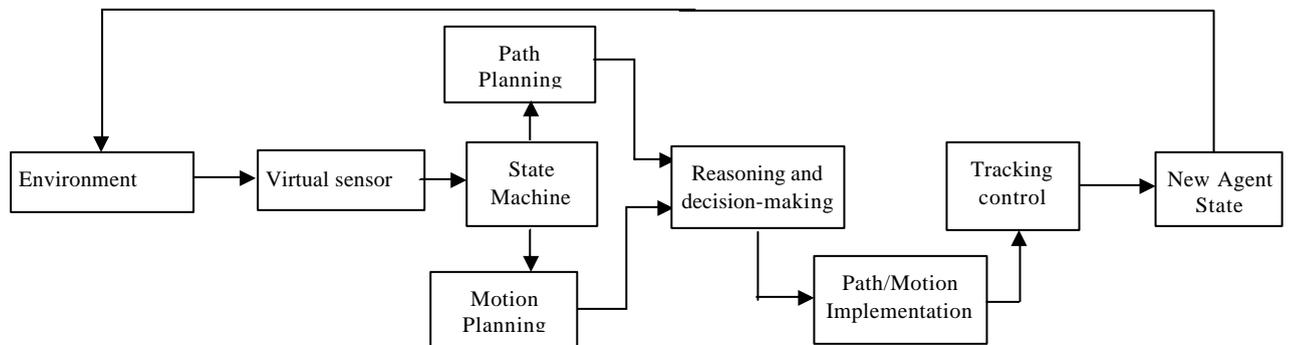


Figure 1: The architecture of the motion control system

construct memorised path nodes. The algorithm uses the angle to partition the obstacle-space, while keeping a safe margin, which allows the agent to plan its path and motion trajectory at any location in the environment. Penalty functions were introduced to make the state node with no obstacles a higher priority. It then chooses the lowest cost state as the local goal and keeps iterating until it reaches the destination goal.

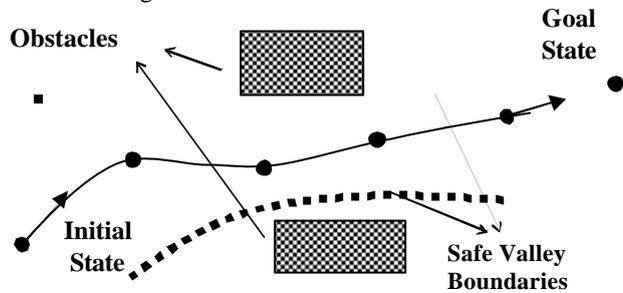


Figure 2: Creating a smooth motion path trajectory in a dynamically allocated safe-pass valley.

As noted earlier, in order to deal with unknown or dynamic environments, a suitable path planning algorithm must be a real-time one. The agent executes actions before finding the global solution. So it is an exploration in an unknown environment or a dynamic environment. A number of heuristic functions were defined, which could provide the least cost path estimated from the current state to a goal state and the actual cost estimated from the initial state to the current path state. The actions defined will return a list of possible solutions in the state space. Assuming that an action is deterministic, the agent might have access to an admissible heuristic function, which estimates the distance from the current to a goal state. The objective of this consideration is to reach the goal with shortest distance with a motion constrained minimum cost. For example, the agent is required to go through a complex obstacle block to reach a destination in a reasonable time whilst maintaining smooth motion characteristics. Figure 2 shows an example of generating a smooth path in the range of a safe valley identified by the agent's perception and reasoning.

## 6 BUILDING UP A DYNAMIC TREE FOR PATH FINDING

A dynamic path tree must be created for the searching process. The planning algorithm searches in a state space for the least expensive path from a start state to a goal state by examining the neighbouring or adjacent states of particular states (the states are formed according to the map representation). By repeatedly examining the promising unexplored location area, the algorithm will reach an end if a configuration is the final goal. Otherwise, it takes notes of all that location's neighbours for further exploration.

In order to avoid the agent being halted in a dead-end, we assume the path or actions executed are reversible. So if our real-time path planning reaches a dead-end state, where no goal state is reachable, the agent could seek to reverse its actions. It should be noted that no algorithm can avoid dead-ends in all state spaces. In cases where the agent reaches a dead-end, it must find a way back by its own reasoning according to the information available. According to the requirements of motion dynamics, it can not simply travel back to the state it previously visited. Instead a number of extra actions must be executed, and an extra set of states must be added, in order to return the motion back to the state. After that it should follow the path in a reverse direction until it finds a branch node, which was executed before. The algorithm does not make the agent follow the action path tree in a reverse order, because of the motion constraints. We must guarantee the motion is smooth or at least C1 continuity. Once the agent reverts to the nearest branch path node, the sequence of the path node state will be pruned from the path tree. The searching algorithm uses two data structures to record the path information, one is a list called Close to record the passed states, and the other is called a tree called PathTree to record the path map. At the start, Close is empty, and PathTree has only the starting state. In each iteration, the algorithm removes the most promising neighbour

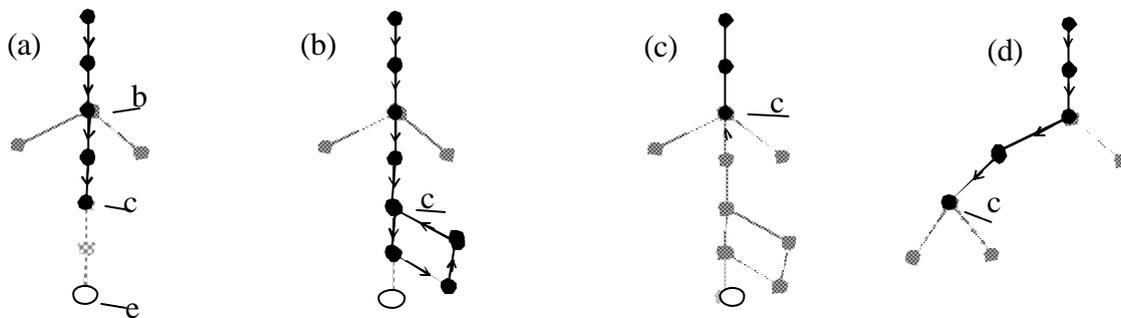


Figure 3: An illustrating example of building up a dynamical tree  
b: Branch node; c: Current node; O: Dead-end.

state for examination. If the state is not a goal, all the other neighbouring states are sorted. If they are already in Closed, they are ignored. Otherwise they are kept in temporary locations. The algorithm will then perform a procedure called Merging to reduce the useless neighbour states. If some states could survive, then the current state is recorded as a split node, and the neighbours are recorded as crosses in PathTree. If no neighbour state is available, the PathTree will trace back to the nearest split node, and switch to a new cross. If all the crosses in the PathTree have been tested before the goal is reached, it concludes that there is no path to the goal from the specified start configuration. Figure 3 shows an illustrating example, in which an agent performed an on-line path search: It reached a dead-end in (a), and found a way back, (b), reached a branch node (c), and pruned the path sequence unsuccessful and planed the path in another route (d).

## 7 MOTION TRAJECTORY CURVE

Trajectory generation is important for the motion constraint path planning, because the moving agent will have to adjust or control its motion state to follow the trajectory reasonably closely whilst maintaining good motion characteristics. The clothoid curve is very useful because its curvature varies linearly along the arc. Kanayama [Kan89] proposed to use this curve for motion trajectory design. It is now the most commonly used curve type for highways and railroad design [Esv01]. It is chosen for generating a smooth path since it satisfies all the requirements for agent motion control tasks and modelling. The Clothoid curve is an intrinsic spline, it can not be expressed in a close form, and this is the biggest disadvantage and results in calculation difficulty. However, its curvature varies linearly along the arc, and the curve can be constructed from its curvature. On the other hand, since the parameter  $t$  is proportion to the length of the arc, it can be used directly as a trajectory. The derivative of the curvature of clothoid curve is a constant which is identical with the Bang-bang control theory in aiming at giving solution to the optimal-control problem. The clothoid curve is defined as following form:

$$Cv(s) = k * s + Cv_0 \quad \dots (3)$$

where  $s$  is the arc length,  $Cv(s)$  is the curvature and  $k$  is a constant. The direction of the tangent vector is the integration of the curvature and expressed by

$$\mathbf{q}(s) = \int_0^s (k * s + Cv_0) ds = \frac{1}{2} * k * s^2 + Cv_0 * s + \mathbf{q}_0 \quad \dots(4)$$

In our work, three dynamically allocated points in a 3D space are identified at each state for each basic curve element configuration. Unlike a conventional

approach, we do not restrict the curve as a symmetric pair since it may encounter difficulties in a global configuration. The global trajectory is made up of a set of local curve pieces and we should not only guarantee the smooth agent movement along local curve, but also the smooth transition between these curve elements. Our approach is to use an un-symmetric clothoid curve element plus two extra dynamic control points  $S_2$  and  $S_4$  to offer the more flexible and powerful solution to the trajectory generation problem. In practice, a symmetric pattern cannot always be guaranteed to be the optimal trajectory to follow, and the un-symmetric pattern is the general situation and offers more flexibility for the control process. In order to achieve a successful smooth connection at the joints between curve elements, keeping in mind that each destination state is also the new initial state for the next move step, the direction of the motion trajectory at the current destination location should be the same as at the next new initial agent location. The curvatures at the both locations should also be the same for smooth curvature transition. To meet these requirements, two transition points are introduced which are determined by the motion kinematical states of the agent to produce a smooth transition between curve elements.

As shown in Figure 4, the agent starts at  $S_1$ .  $S_3$   $S_6$  are the points perceived, and  $S_2$ ,  $S_4$  are the two points added to control the agent movement to make a smooth transition between adjacent clothoid curve elements, which are derived from the agent motion requirements that satisfy the agent kinematical conditions and the equations (3) and (4). The first local curve element ends at the destination point  $S_4$ , which is also the new initial position of the next clothoid curve element. At  $S_4$ , the curvature is decreased to zero and the direction is from  $S_3$  to  $S_6$ . The trajectory generation will keep going as long as subsequent path points are supplied.

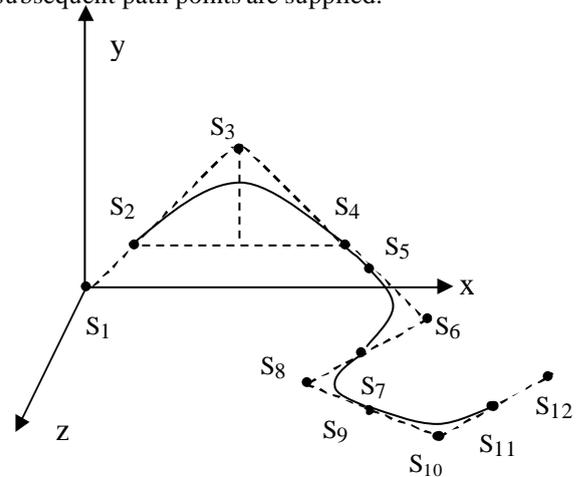


Figure4: An example of the global configuration

## 8 GROUP BEHAVIOUR

Our multi-agent path planning algorithm in unknown environments is developed based on the single agent path planning environments. In the multi-agent environments, introducing more than one agent simply will lead to poor performance if they are just simply added. Therefore the agent path planning algorithms must be changed to introduce elements for joint activities. The observation and prediction about other agents will be included in order to operate effectively and in a timely manner. Our method is to plan actions jointly via coordination and communication. In a simulation, the agent also acts as an obstacle or part of the environment, although they are not static most of the time. The path planning problem and strategy for joint activities is best constructed according to the goals specified for the location and motion states of each agent. The algorithm is summarised as follows:

**Agent A:** Initialise agents A, which including agents' state location, behaviours etc.

**Goals:** Assign specific goals to A, including a final goal and sub-goals

**Actions:** Move, how to move, perceiving environment, recognise other agents and the environment. Predict other's intention etc. Approaching other agents

**Agent B:**

...

**Plan and strategy:** Coordination, communication, competition and collaboration etc.

The algorithm has been implemented in our simulation and a basic exploration test was conducted at this stage.

## 9 SIMULATION RESULTS

A number of simulations have been conducted in our work for evaluating the motion trajectory generation algorithm. An auto-pilot-aircraft was created as an intelligent agent and it has the ability to perceive the visual information about the environment when it is in a navigation. Sensors have been created and attached to the aircraft.

Figure 5 show three instances of a path planning simulation. using the algorithm discussed above. Figure 5a shows that the moving agent moves to the entrance of an unknown alley, where there are three potential paths that have been created and a split node is created and added to the path tree. Figure 5b shows the agent moving back to the nearest split node. Figure 5c shows that the moving agent's successful navigation through the alley.

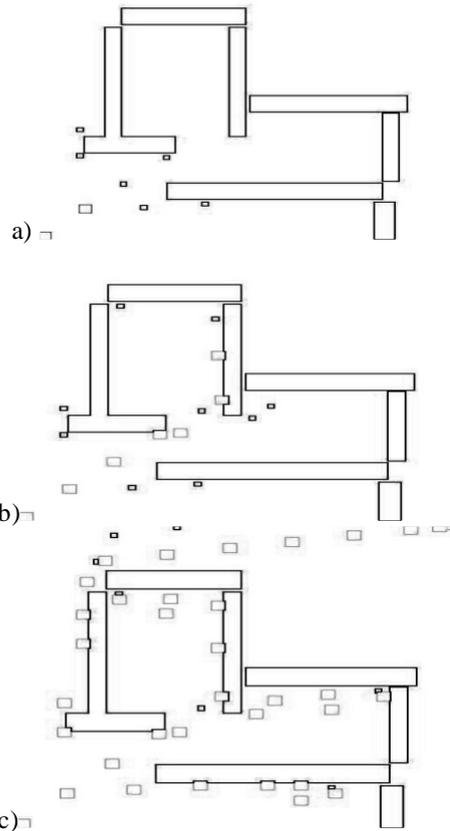


Figure 5: A simulation test; a. 1 merge paths and search optimal path b. exit from blind alley c. successful navigation through obstacle block

Figures 6 and 7 show two simulation experiments. The experiments were designed for an autonomous moving agent to navigate through complex obstacle environments with different motion characteristics. The simulations were quite successful and the results are satisfactory. Figure 8 shows a screen capture of real-time aircraft simulation in a virtual environment, in which the aircraft acted as an autonomous agent who used visual information to detect obstacles in order to find a path to conduct an obstacle-free navigation.

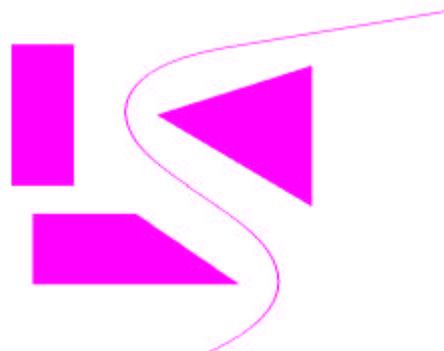


Figure 6 Navigation Simulation Case 1.



aircraft model has been developed, which is addressing the manoeuvring capabilities of the moving agents, while the moving agents' accelerations and velocities are always continuous and bounded. The generated motion path is constituted smoothly and has continuous curvature in the whole state space of the motion thus satisfying the major requirements for the implementation of such strategies in real-time navigation. The clothoid curve has been chosen as the basis for the motion trajectory generation. A 3D aircraft simulation has been conducted and the result is quite promising. The simulation result is quite satisfactory. The next step for our research is to refine the algorithm and look at path planning with more complex group behaviours in simulated environments.

## 11 REFERENCES

- [Ari01] Okan Arikan and Stephen Chenney and {D. A.} Forsyth, "Efficient Multi-Agent Path Planning", Proceedings of the 2001 Eurographics Workshop on Animation and Simulation, Sep, 2001
- [Bem96] A. Bemporad, A. De Luca, G. Oriolo, "Local incremental planning for a car-like robot navigating among obstacles" in Proc. of the 1996 IEEE Int. Conf. on Robotics and Automation, Minneapolis, USA, 1996
- [Ber03] Tomas Berglund, Hakan Jonsson and Inge Soderkvist, "An Obstacle-Avoiding Minimum Variation B-Spline Problem," International Conference on Geometric Modelling and Graphic July 16-18, 2003.
- [Dub57] L. E. Dubins, "On Curves of Minimal Length with a Constraint on Average Curvature and with Prescribed Initial and Terminal Position and Tangents," American Journal of mathematics, vol. 79, pp.497-516, 1957.
- [Esv01] C. Esveld, "Modern Railway Track", Second Edition, Published by MRT-Productions, a subsidiary of ECS, ISBN 90-800324-3-3, 2001.
- [Fra01] Th. Fraichard and J. M. Ahuactzin, "Smooth Path Planning for Cars," IEEE Int. Conf. On Robotics and Automation May 21-26, 2001.
- [Hol92] P. D. Holmes and E.R.A. Jungert, "Symbolic and geometric connectivity graph methods for route planning in digitized maps", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol, 14, no.5, 1992, pp549-565.
- [Jos97] F. M. Jossen, "An optimal pathfinder for vehicles in real-world digital terrain maps", the Royal institute of Science, School of Engineering Physics, Stockholm, Sweden. MSc thesis, 1997.
- [Kan86] Y. J. Kanayama and N. Miyake, "Trajectory Generation for mobile Robots," Robotic Research, vol. 3, Cambridge, MA: MIT Press, 1986, pp.333-340.
- [Kan89] Kanayama, Y and Hartman, B. I, "Smooth local path planning for autonomous vehicles," Robotics and Automation, 1989, vol. 3, pp. 1265-1270.
- [Lat91] Latombe, J.-C., "Robot Motion Planning", Kluwer Academic Publishers, 1991, ISBN 0792391292.
- [Lum90] V. J. Lumelsky, S. Mukhopadhyay, and K. Sun. Dynamic path planning in sensor-based terrain acquisition. IEEE Transactions on Robotics and Automation, Vol 6, No 4, 1990.
- [Mon87] M. Montgomery et al., "Navigation algorithm for a nested hierarchical system of robot path planning among polyhedral obstacles", Proceedings IEEE International conference on Robotics and Automation, pp. 1616-1622, 1987.
- [Nag01] Bryan Nagy and Alonzo Kelly, "Trajectory Generation for Car-Like Robots Using Cubic Curvature Polynomials," in Field and Service Robots 2001, Helsinki, Finland June 11, 2001.
- [Oom87] B. J. Oommen, S. S. Iyengar, N. S. V. Rao, and R. L. Kashyap. Robot navigation in unknown terrain using learned visibility graphs. Part i: The disjoint convex obstacle case. IEEE Journal of Robotics and Automation, Vol RA-3 No.6 December, 1987.
- [Pad00] D. Padmanabhan, "Optimal 2-D Path Planning", AME 598C Project Report, Spring 2000.
- [Sch96] A. Scheuer and Th. Fraichard, "Planning Continuous-Curvature Paths for car-Like Vehicles," IEEE-RSJ Int. Conf. On Intelligent Robots and Systems, November 4-8, 1996. vol. 3, pp. 1304-1311.
- [Ste95] Anthony Stentz and martial Hebert, "A Complete Navigation System for Goal Acquisition in Unknown Environment", In Autonomous Robots, Volume, Number 2, August 1995.
- [Tat91] S. R. Tate. "Arithmetic Circuit Complexity and Motion Planning", Ph. D. Dissertation, Duke University, 1991.
- [Woo97] S. M. Woodcock (editor). "Artificial Intelligence in Games", 1997,
- [Yah98] Alex Yahja, Anthony Stentz, Sanjiv Singh, and Barry L. Brumitt, "Framed-Quadtree Path Planning for Mobile Robots Operating in Sparse Environments", In Proceedings, IEEE Conference on Robotics and Automation, (ICRA), Leuven, Belgium, May 1998.
- [Yam99] M. Yamamoto, M. Iwamura, and A. Mohri, "Quasic-Time-Optimal Motion Planning of Mobile Platforms in the Presence of Obstacles," Int. Conf. on Robotics and Automation, pp. 739-744, 1999.

# Construction of Implicit Complexes: A Case Study

E. Kartasheva<sup>α</sup>, V. Adzhiev<sup>β</sup>, P. Comninos<sup>β</sup>, A. Pasko<sup>γ</sup>, B. Schmitt<sup>δ</sup>

<sup>α</sup> Institute for Mathematical Modeling, Russian Academy of Science, Moscow, Russia, [ekart@imamod.ru](mailto:ekart@imamod.ru)

<sup>β</sup> The National Centre for Computer Animation, Bournemouth University, Poole, UK,  
[{vadzhiiev&peterc}@bournemouth.ac.uk](mailto:{vadzhiiev&peterc}@bournemouth.ac.uk)

<sup>γ</sup> Faculty of Computer and Information Sciences, Hosei University, Tokyo, Japan, [pasko@k.hosei.ac.jp](mailto:pasko@k.hosei.ac.jp)

<sup>δ</sup> Computer Graphics Research Institute, Hosei University (Digital Media Professionals), Tokyo, Japan,  
[schmitt\\_benjamin@yahoo.fr](mailto:schmitt_benjamin@yahoo.fr)

## ABSTRACT

This paper presents a detailed description of a case-study demonstrating a novel method for modelling and rendering of heterogeneous objects containing entities of various dimensionalities within a cellular-functional framework based on the implicit complex notion. Implicit complexes make it possible to combine a cellular representation and a constructive function representation. We briefly describe a formal framework for such a hybrid representation as well as a general structure for implicit complexes. Then, using a representative example, we show how an implicit complex can be constructed geometrically and topologically. We also consider the main rendering issues specific to implicit complexes and describe some implementation problems.

## Keywords

Function representation, cellular representation, implicit complex, polygonization, ray-tracing.

## 1. INTRODUCTION

Heterogeneous object modelling is becoming an important research topic in different application areas, such as volume modelling and rendering, modelling of objects with multiple and varying materials in CAD and rapid prototyping, representing results of physical simulations, geological and medical modelling. Such objects are heterogeneous from two points of view: their internal structure and dimensionality. Varying materials and other attributes of an arbitrary nature constitute a heterogeneous internal structure. A dimensionally heterogeneous object in 3D space can include elements of different dimensions (points, curves, surfaces and solids) combined into a single entity from the geometric point of view (i.e., a point set) and the topological point of view (i.e., a cellular complex).

A model of objects with fixed dimensionality and heterogeneous internal structure (multidimensional

point sets with multiple attributes or so-called *constructive hypervolumes*) was proposed in [Pas01]. This model uses real functions of point coordinates (scalar fields) to represent both the object geometry and its attributes. The hybrid cellular-functional model [Adz02] allows for representing a heterogeneous object as a cellular complex with both explicit and implicit cells (cellular domains) of different dimension. Such an object is called an *implicit complex (IC)*, which is defined as the union of properly joined cellular domains. Explicit cells can be represented as point lists, parametric curves and surfaces. Implicit cells can be implicit surfaces and their patches, intersection curves of implicit surfaces, or functionally represented (FRep) solids [Pas95].

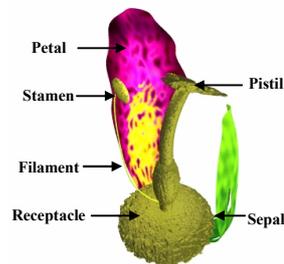


Figure 1. Components of a flower

In this paper, we present a case-study which allows us to demonstrate a novel technology for modelling and rendering of heterogeneous objects using implicit

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 2005 conference proceedings, ISBN 80-903100-7-9  
WSCG'2005, January 31-February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency – Science Press

complexes. This case-study inspired by [Kun03] is based on a model of a flower (Figure 1) which has the following components of different dimensionalities: a 3D receptacle, 3D stamens, 3D pistil, 2D petals and sepals, and 1D filaments. All the components have a colour that can be expressed as an attribute. Of course, such an object can be modelled and rendered using more traditional means than those based on the cellular-functional framework; however we believe this case-study allows us to show all the conceptual phases of modelling and rendering within the cellular-functional framework as well as outline some implementation issues. We pay particular attention to the theoretical and practical issues of the implicit complex construction.

## 2. RELATED WORKS

Here, we briefly discuss approaches for modelling dimensionally heterogeneous objects using various cellular representations and previously published work on modelling objects with varying distribution of material and other attributes.

A typical technique for describing heterogeneous objects is to represent them as collections of homogeneous components. To describe complex topology, different spatial subdivisions, topological stratifications [Mid00], and complexes [Ohm01, Pao93] are used.

Topological complexes and their construction methods are discussed in a number of publications on shape modelling and solid modelling. Multidimensional simplicial complexes are used in [Pao93] for dimension-independent geometric modelling for various applications. A Selective Geometric Complex (SGC) [Ros90] is a non-regularised non-homogeneous point set, represented through enumeration as the union of mutually disjoint connected open subsets of the real algebraic variety. A procedure for designing cellular models based on CW-complexes with the emphasis on the topological validity of the resulting shapes is considered in [Kun99, Ohm01].

To specify non-geometric properties of objects, spatial subdivisions are used in computer graphics and in finite element analysis (FEA) as the underlying structures for piecewise analytical descriptions of attribute functions. Usually a basic topological subdivision is selected, which can be described by a topological stratification [Ros90], a cell complex [Cut02], or a voxel model. Different types of functions can be used to describe attributes [Jac99, Par01].

Another approach to modelling heterogeneous objects is based on using real functions of point

coordinates. For example, the constructive hypervolume model [Pas01] supports uniform constructive modelling of point set geometry and attributes using such functions. Then, a theoretical framework combining a cellular representation and a constructive function representation was proposed in [Adz02]. An independent cellular and functional representation of the same object is useful, but not sufficient in certain applications. For example, in the above mentioned flower model, each dimensionally homogeneous component (3D receptacle, 2D petals, 1D filaments) can be functionally represented. However, without additional information one cannot separate individual functions for the components from the single function describing the entire object. This additional information about objects components, their dimensions, and attachments to each other are used in applications such as finite element mesh generation, animation, and rendering. The above was the motivation for introducing in [Adz02] a hybrid cellular functional model based on the notion of an implicit complex, which allows for the flexible combination of cellular and functional object geometry models and attribute models. In the current paper, we examine in more detail the construction of implicit complexes.

## 3. THEORETICAL FRAMEWORK

Here we provide a brief description of a theoretical framework for the representation of implicit complexes. A more formal and detailed consideration can be found elsewhere [Adz02]. In this paper we present novel material concerned with theoretical and practical matters of the description and construction of ICs.

### A Hybrid Representation of Geometry

The hybrid geometrical model of heterogeneous objects presented in [Adz02] combines a cellular representation and a constructive representation using real-valued functions. Formally, the hybrid representation for a geometric object  $D \subseteq \Omega$  is defined as follows:

$$M_H : D = \{X \mid X \in \Omega, \Omega \subseteq E^n, X \in |K^p|\}$$

where  $\Omega \subseteq E^n$  is a modelling space and  $K^p$  is an implicit  $p$ -dimensional complex. The definition of an implicit complex is based on the concepts of cellular spaces and CW-complexes [Fom97, Mas67]. A CW-complex provides a general representation for different topological complexes including polyhedral and cellular complexes.

$D$  is defined as a closed cellular space (*domain*) and can be represented as a carrier of a CW-complex  $K$ , such that  $D=|K|$ . The hybrid representation scheme

can be defined in the form of the pair  $\langle D, K \rangle$  represented through a union operation as  $\langle D, K \rangle = \bigcup_{i=1}^N \langle D_i, K_i \rangle$ . We can use different representation schemes for various  $\langle D_i, K_i \rangle$  pairs. Suppose that for some subdomains we use the cellular representation. Such subdomains and their subdivisions are called *explicit* and are denoted by  $DE_i$  and  $KE_i$ . Then for the other domains, denoted by  $DI_j$ , we use an FRep; so their cellular representation denoted by  $KI_j$  is not necessarily known but can, in principle, be built using some known method. We call such domains as well as their subcomplexes *implicit* ones. Then the point set  $D$  is represented as the union  $D = (\bigcup_{i=1}^N DE_i) \cup (\bigcup_{j=1}^M DI_j)$ . The

complex  $K$  is also represented as the union of the corresponding explicit subcomplexes  $KE_i$  and the implicit subcomplexes  $KI_j$ . Thus,

$$K = (\bigcup_{i=1}^N KE_i) \cup (\bigcup_{j=1}^M KI_j).$$

Note that if the intersection of two cells of an IC is not empty, then it consists of a collection of cells of this complex. The same is true concerning the boundaries of cells. It is necessary to impose constraints on the domain boundaries similar to those for subdomains.

### Definition of Implicit Complexes

In the general case, a  $p$ -dimensional implicit complex  $K^p$  is expressed as  $K^p = \{ \{ e_i^q \}_{i=1, \dots, n_q}^{q=0, \dots, p}, \{ t_j^s \}_{j=1, \dots, m_j}^{s=0, \dots, p} \}$ ,

where  $e_i^q$  are cells of all the explicit subcomplexes of  $K^p$  and  $t_j^s$  are implicit cells (such that each  $t_j^s$  is the point set coinciding with the carrier of an implicit subcomplex of  $K^p$ ). Thus, for any  $DI_j$  there exist  $t_j^s \in K^p$  such that  $t_j^s = DI_j$ .

Explicit cells  $e_i^q$  are defined with respect to the definition of the CW-complex. The shape of each explicit cell  $e_i^q$  is defined by a characteristic mapping, and its boundary is mapped onto a subcomplex  $M^r \subset K^p$  with dimensionality  $r < q$ .

Each implicit cell  $t_j^s$  is a closed point set defined by an FRep. The boundary of the implicit cell  $t_j^s$  is not necessarily mapped onto any subcomplex of  $K^p$  and can contain both explicit and implicit cells. Explicit cells are indivisible elements of a subdomain subdivision containing no other cells. Some cells of an implicit complex can lie inside implicit cells of the complex. Note that an  $r$ -dimensional  $K^r$  implicit

complex can be reduced to a cellular one. We assume that each implicit cell  $t_j^s \in K^r$  (where  $0 < s \leq r$ ) can be discretized. The corresponding methods were described in detail in [Kar03].

### Implicit Complex Description

IC topology is described by relations between its elements. The general structure of a 3D IC is illustrated by Fig. 2. By definition, a 3D IC consists of 0D, 1D, 2D and 3D cells. Let  $G^p$  be a set of  $p$ -dimensional cells  $g_i^p$ . Such a set contains both explicit and implicit cells. There are two main types of relations that establish connections between cells of different dimensionalities: the boundary relation and the "to contain" relation.

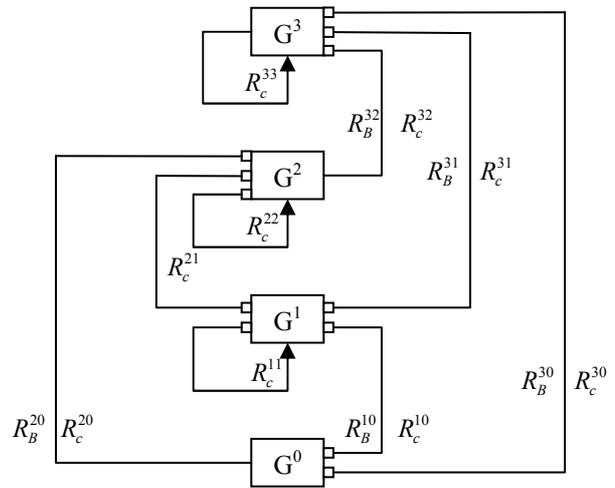


Figure 2. The general structure of a 3D IC

We denote by  $Rb^{ps}$  the *boundary relation* between  $p$ -dimensional and  $s$ -dimensional cells,  $Rb^{ps} \subset G^p \times G^s$ ,  $s < p$ . The pair  $(g_i^p, g_j^s)$  belongs to  $Rb^{ps}$  if  $g_j^s$  belongs to the boundary of  $g_i^p$ . The relation "to contain" is denoted by  $Rc^{ps}$ ,  $Rc^{ps} \subset G^p \times G^s$ ,  $s \leq p$ . The pair  $(g_i^p, g_j^s)$  belongs to  $Rc^{ps}$  if  $g_j^s \in g_i^p$  and  $g_j^s \cap \partial g_i^p \neq g_j^s$ . The entire structure of 3D IC is defined by six different boundary relations and nine different "to contain" relations. Other relations are the *co-boundary*, the "to be contained", the *incidence* and the *adjacency* relations. These can be derived from the boundary and the "to contain" ones using various operations on relations.

The geometry of ICs is described as follows. An explicit cell  $e_i^0$  is described by its coordinates. An implicit cell  $t_j^0$  is defined in FRep with an inequality of the form  $F(X) \geq 0$ . In  $E^3$  space, the function  $F(X)$  for 0D cells (points) can be described using

functions representing the intersection of three surfaces, the intersection of a curve and a surface, or directly as the formulation  $F(X) = -d(X - X_0)$ , where  $d$  is the distance from the given point  $X_0$ . An implicit cell  $t_j^o$  can also be described as an image of a point functionally defined in 2D space.

An explicit 1D cell  $e_j^1$  is defined by a characteristic mapping  $\chi_j^1: X = \varphi_j^1(u)$ , which maps the segment  $[u_0, u_1]$  in the space of the real parameter into a curvilinear and perhaps a closed segment in the  $E^3$  space. An implicit cell  $t_j^1$  can be described in two ways. It can be defined as an FRep object in  $E^3$  by an inequality of the form  $F(X) \geq 0$ . In such a case, the 1D cell takes the form of an arbitrary curve defined as the intersection of two surfaces in  $E^3$ . Alternatively, the cell  $t_j^1$  can be represented as an image of an FRep curve described in 2D space by an inequality of the form  $f_j^1(x) \geq 0$ , where  $x \in E^2$ . This mapping is given by a function of the form  $h_j^1: E^2 \rightarrow E^3$ .

Explicit 2D cells are represented as images of triangles and quadrilaterals resulting from characteristic mapping. For each cell  $e_i^2$ , we define the characteristic mapping  $\chi_i^2: X = \varphi_i^2(u, v)$ , which takes the rectangle  $u_0 \leq u \leq u_1, v_0 \leq v \leq v_1$  in the parameter space and maps it onto a surface patch in  $E^3$ . An implicit 2D cell  $t_j^2$  can also be described in two ways. It can be defined with FRep in  $E^3$  by an inequality of the form  $F(X) \geq 0$ . Alternatively, one can use a functional description of the form  $f_j^2(x) \geq 0$  in  $E^2$  with the subsequent mapping of the form  $h_j^2: E^2 \rightarrow E^3$ .

To represent an explicit 3D cell, a variety of maps of the form  $\chi_i^3: X = \varphi_i^3(u, v, w)$  can be used to describe the shape of curvilinear polyhedrons. Maps of this kind have been extensively used in describing finite element sets. Such maps can be used to describe the cells and attach them to the boundary cells of the complex by obeying certain boundary conditions. Once again, an FRep of the form  $F(X) \geq 0$  is used to describe the implicit cell  $t_j^3$ .

And finally, to describe the non-geometric properties of a heterogeneous object, represented by an IC, we use the cellular-function model of the attributes introduced in [Adz02]. Each attribute  $A_i$  is defined by a function of the form  $S_i: E^3 \rightarrow N_i^{mi}$ , where  $N_i^{mi}$

is a set of attribute values (which can be a vector or tensor space).  $N_i^{mi}$  is embedded into a real-valued space of a proper dimension  $mi$ . Thus,  $N_i^{mi} \subseteq \mathfrak{R}^{mi}$ . Attributes assigned to an implicit complex  $K$  are described by functions  $S_i$  in a piece-wise manner, i.e.  $S_i = \{(S_i)_j^s \mid (S_i)_j^s: g_j^s \rightarrow N_i^{mi}, g_j^s \in K\}$ .

## Implicit Complex Construction

To create an IC, it is necessary to describe the shapes of all its elements and, to specify the entire boundary and the “to contain” relations between its elements. The *attachment operation* is introduced allowing the creation of the IC in a component-wise manner, in order of increasing dimensionality of its components. This process is constructive and iterative.

We start with an empty complex  $K_o = \emptyset$ , and then at each construction step  $i$  we attach a new component  $L_i$  to the already formed subcomplex  $K_{i-1}$ , thus creating a new complex  $K_i = K_{i-1} \cup L_i$  which is a subcomplex of the target complex  $K$ . We introduce two types of the attachment operation based on the one defined for CW complexes [Kun99, Ohm01].

The *cell attachment* operation assumes that at each step  $i$  of the process another cell  $g_i^r$  is attached to the complex  $K_{i-1}$ . First we define the shape of this cell using one of the methods described above. Then, we have to modify the relations. So for all implicit and explicit cells  $g_j^s \in K_{i-1}$  lying on the boundary of  $g_i^r$  we add the pairs  $(g_i^r, g_j^s)$  to the boundary relations  $Rb^{rs}$  (where  $s < r$ ). Then, for each implicit cell  $g_l^p \in K_{i-1}$  (where  $p \geq r$ ) that contains  $g_i^r$ , we have to add the pair  $(g_l^p, g_i^r)$  to the “to contain” relation  $Rc^{pr}$  (where  $p \geq r$ ). Finally, and only if  $g_i^r$  is an implicit cell, for all implicit and explicit cells  $g_m^q \in K_{i-1}$  (where  $0 \leq q \leq r$ ) lying inside  $g_i^r$  we add the pairs  $(g_i^r, g_m^q)$  to the “to contain” relation  $Rc^{rq}$  (where  $q \leq r$ ).

The *complex attachment* operation deals with the *procedure* of attaching the complex  $L$  to the complex  $K_{i-1}$ . Assume that  $L$  is *properly joined* to the complex  $K_{i-1}$  that is  $L \cap K_{i-1} = C$  (where  $C$  is a subcomplex of both  $L$  and  $K_{i-1}$ ). Thus we have to create a complex  $K_i = K_{i-1} \cup L$ ,  $K_i = \bigcup_{p=0}^{p \leq r} G_i^p$ . First, we define an attachment map  $\psi$  that relates the equivalent cells of the initial complexes. Then, we obtain the sets  $G_i^p$  of the complex  $K_i$  as quotient sets

of the union of the corresponding sets of the initial complexes by the quotient map  $\psi$  as follows:  $G^p(K_i) = G^p(K_{i-1}) \cup G^p(L) / \sim_\psi$ , ( $0 \leq p \leq r$ ). Finally, we define the boundary and the "to contain" relations of the complex  $K_i$ .

#### 4. THE FLOWER CASE STUDY

Here we present a systematic description of how the cellular-functional model of a flower that is considered as an example of an heterogeneous object can be constructed based on the theory presented above.

##### The Components

To create a model of such a composite object as a flower, we start from modelling its separate components (see Fig. 1).

- The 3D receptacle is modelled using an FRep, and is defined as a half-ellipsoid combined with a solid noise function (algebraic sum with Gardner's noise function). The corresponding FRep function is denoted by  $F_r$ .
- The 3D pistil is also defined by an FRep, and the corresponding constructive tree is composed of ellipsoids in the leaves and blending union in the nodes. The corresponding function is  $F_p$ .
- The 3D stamens are defined as an algebraic sum of an ellipsoid and Gardner's solid noise function, corresponding to the function  $F_s$ .

- The 2D petals and sepals are specified in two steps. First, an object is described by an FRep in 2D space (Fig. 4a). The function  $F_{pr}$  describing this object on the  $U \times V$  plane is defined as the difference between a large 2D solid ellipse and two smaller ones (representing the holes). Then, tapering and general space mapping deformations are applied to the object. The corresponding mapping function  $h_{pr}$  is defined using a technique similar to FFDs and the resulting object is a surface patch in 3D space (Figure 4b).

- The 1D filaments are explicitly defined by spline curve segments defined in 3D space. These curve segments are defined as  $\bar{r}(u) = \varphi_f(u)$ ,  $u \in [u_0, u_1]$ .

We consider the flower as an heterogeneous object which has a colour property. This is represented by an RGB colour attribute which is described by the function  $S = E^3 \rightarrow \mathfrak{R}^3$  in a piece-wise manner, so that  $S = \{S_i\}$ , where each  $S_i$  maps the corresponding subset of  $E^3$  into the RGB space.

The listed components differ in dimensionality and representation. Let us show how an accurate definition of a composite object can be made.

##### The Implicit Complex Structure

Altogether, the complex  $K$  describing the entire heterogeneous model of the flower fragment consisting of the receptacle, the pistil, a petal and a stamen, and contains cells corresponding to different flower components (Fig. 3). The receptacle is described by the 3D cell  $t_1^3$ , the pistil by the 3D cell  $t_2^3$ , the stamen by the 3D cell  $t_0^3$ , the filament by the 1D cell  $e_0^1$ , and the petal by the 2D cell  $t_0^2$ . To assemble the listed components together we have to add auxiliary cells describing their interconnections.

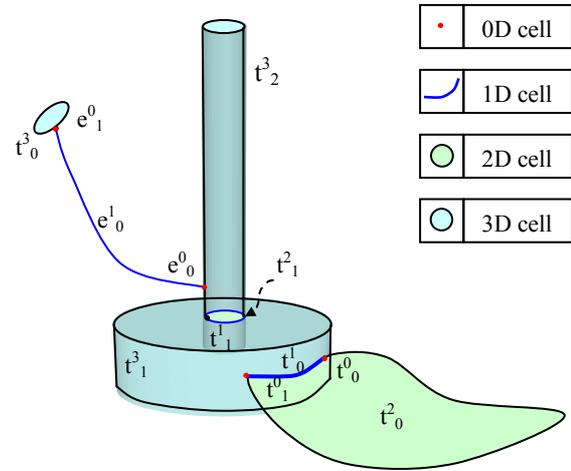


Figure 3. IC structure for 'Flower' model.

Let  $G^i$  be a set of cells of dimension  $i$ . As each cell is associated with the assigned attribute functions being specified in brackets, we have:

$$G^0 = \{e_0^0(S_2), e_1^0(S_1), t_0^0(S_3), t_1^0(S_3)\}$$

$$G^1 = \{e_0^1(S_1), t_0^1(S_3), t_1^1(S_2)\}$$

$$G^2 = \{t_0^2(S_3), t_1^2(S_2)\}$$

$$G^3 = \{t_0^3(S_1), t_1^3(S_2), t_2^3(S_1)\}$$

The boundary relations  $Rb$  establishing connections between cells include the following pairs:

$$Rb^{10} = \{(e_0^1, e_0^0), (e_0^1, e_1^0)(t_0^1, t_0^0), (t_0^1, t_1^0)\}$$

$$Rb^{21} = \{(t_0^2, t_0^1), (t_1^2, t_1^1)\}; Rb^{20} = \{(t_0^2, t_0^0), (t_1^2, t_1^0)\}$$

$$Rb^{32} = \{(t_2^3, t_1^2), (t_1^3, t_1^2)\}; Rb^{31} = \{(t_1^3, t_0^1), (t_1^3, t_1^1), (t_2^3, t_1^1)\};$$

$$Rb^{30} = \{(t_0^3, e_1^0), (t_1^3, t_0^0), (t_1^3, t_1^0), (t_2^3, e_0^0)\}.$$

All the "to contain" relations are empty for this complex.

##### A Detailed Mathematical Description

Let us introduce the following mathematical entities:

- $FrepCell(xdim,F)$  represents an implicit cell defined by a real-valued function  $F(x)$  and depends on the space dimension  $xdim$ .

- $MappedFCCell(xdim,F,h)$  represents a mapped implicit cell defined by a real value function  $F(x)$  and a mapping  $h:E^2 \rightarrow E^3$ .

- $ExplicitCell(bnd,\chi)$  represents explicit cells defined by their boundaries  $bnd$  and characteristic mapping  $\chi$ . Note that explicit 0D cells are described by just their coordinates.

Now, we can define all the cells in step-by-step manner.

1. The stamen is described by the 3D implicit cell  $t_0^3 = FrepCell(xdim=3,F=F_s)$ .

2. The receptacle is described by the 3D implicit cell  $t_1^3 = FrepCell(xdim=3,F=F_r)$  and the pistil – by the 3D implicit cell  $t_2^3 = FrepCell(xdim=3,F = F_p \setminus_0 F_r)$  where  $\setminus_0$  denotes an R-function defining set-theoretic subtraction. Note that such a definition insures that the cells  $t_1^3$ ,  $t_2^3$  have no common internal points. Initially the pistil and the receptacle were described independently so they may overlap in 3D space. To specify the intersection of the cells  $t_1^3$ ,  $t_2^3$  we introduce a 2D implicit cell  $t_1^2 = FrepCell(xdim=3,F = -(F_r)^2 \Lambda_0 F_p)$  which represents a surface segment, here  $\Lambda_0$  denotes an R-function defining a set-theoretic intersection. The boundary of the cell  $t_1^2$  intersects the boundaries of the cells  $t_1^3$ ,  $t_2^3$ , so we have to add a 1D cell  $t_1^1 = FrepCell(xdim=3,F = -(-(F_r)^2 \Lambda_0 F_p)^2)$  describing this intersection.

3. The stamen and the pistil are connected to each other by the filament described by the 1D explicit cell  $e_0^1$ . We set  $e_0^1 = ExplicitCell( bnd=\{\varphi_f(u_0),\varphi_f(u_1)\}, \chi=\varphi_f(u), u \in [u_0, u_1] )$ . We assume that the end points of the curve segment lie exactly on the boundaries of the cells  $t_2^3$  and  $t_0^3$ , and the segment has no other common points with  $t_2^3$  and  $t_0^3$ . So we can define the intersection of the filament  $e_0^1$  with the pistil  $t_2^3$  and the stamen  $t_0^3$  explicitly by 0D cells  $e_0^0$  and  $e_1^0$ . These cells are specified by their Cartesian coordinates:  $e_0^0 = \varphi_f(u_0)$ ,  $e_1^0 = \varphi_f(u_1)$ .

4. The petal is described by the mapped implicit cell  $t_0^2$ . Initially the petal was defined in the same manner by the pair  $(F_{pt}, h_{pt})$ . But this definition

does not take into account the adjacent component, namely the receptacle. Then, one can formulate the following constraints for the trimmed petal which intersects the receptacle only along the boundary:

$$\begin{cases} F_r(h_{ptx}(u,v), h_{pty}(u,v), h_{ptz}(u,v)) \leq 0 \\ F_{pt}(u,v) \geq 0 \end{cases}$$

Here we assume that the mapping function

$h_{pt} : E^2 \rightarrow E^3$  is defined as

$h_{pt}(u,v) = (h_{ptx}(u,v), h_{pty}(u,v), h_{ptz}(u,v))$ , where  $(u,v)$  is a point in  $E^2$ . Thi is equivalent to the description of the cell  $t_0^2 = MappedFCCell(xdim=2,$

$$F = -F_r(h_{pt}(u,v)) \Lambda_0 F_{pt}(u,v), h = h_{pt}(u,v).$$

5. The petal  $t_0^2$  is a 2D cell in 3D space, and the receptacle  $t_1^3$  is a 3D cell. Their intersection is defined by a curve segment represented by the implicit cell  $t_0^1$ . The constraints for this cell (which has to belong to both the surface of the receptacle and the boundary of the petal) can be expressed as the following system:

$$\begin{cases} F_r(h_{ptx}(u,v), h_{pty}(u,v), h_{ptz}(u,v)) = 0 \\ F_{pt}(u,v) \geq 0 \end{cases}$$

Therefore,  $t_0^1 = MappedFCCell(xdim=2,$

$$F = -(F_r(h_{pt}(u,v)))^2 \Lambda_0 F_{pt}(u,v), h_0 = h_{pt}(u,v).$$

Finally, the start and end points  $t_0^0$  and  $t_1^0$  of the 1D cell  $t_0^2$  are defined in a similar manner taking into account some relevant constraints (omitted here because of shortage of space).

## The Implementation Model

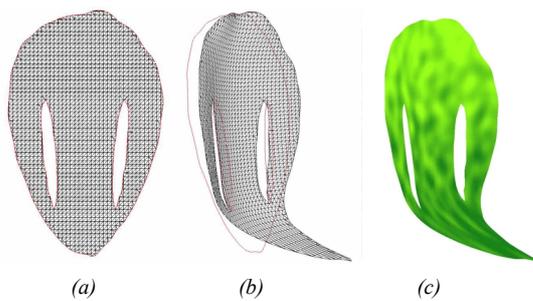
We have implemented cellular-functional modelling of heterogeneous objects within an object-orientated framework. Let us outline the principal classes which are directly derived from the presented theoretical description. The basic *IComplex* class represents an implicit complex data structure (Fig.2 is an illustration). Its attributes represent six boundary relations and nine “to contain” relations as well as cells of various dimensionalities. It is assumed that all the cells are enumerated. Each relation is described by the object of the *Relation* class which contains all the pairs of numbers of related cells. The operations of the *Relation* class allow us to get the indices of all the related cells as well as to add and delete pairs of cells. Accordingly, the *IComplex* class includes operations for adding, deleting and modifying relations, as well as inquiry operations on relations.

The implicit complex geometry within the *IComplex* class is specified using objects of classes inherited from the abstract  $G<dim>$  class parameterized by the

cell dimensionality  $dim$ . Objects of the *ExplicitCell* classes represent explicitly specified parametric curves, surfaces and solid objects. As to implicit cells, they are represented by objects of either *FRepCell* or *MappedFCell* classes. They are all built on the basis of an abstract *FRep* class that is also parameterized by the dimensionality of the coordinate space  $xdim$ . This basic class contains virtual operations for defining the point membership with respect to some FRep object as well for rendering and discretization. All the classes describing FRep primitives and operations are inherited from the basic abstract *FRep* $\langle xdim \rangle$  class. *ParamCurve*, *ParamSurf* and *ParamSolid* classes allow the definition of corresponding parametric entities. They also contain virtual operations for rendering and discretization. Our software tools used for implementation of the flower case-study are built using this object-oriented model.

## 5. RENDERING

In this section we describe how an implicit complex can be rendered.



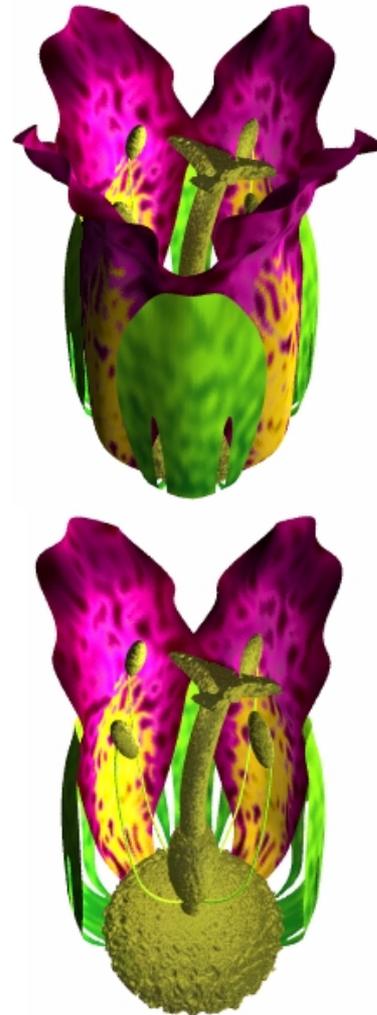
**Figure 4. (a) planar 2D surface; (b) polygonized 2D petal in 3D space; (c) textured sepal**

An implicit complex includes implicit and explicit cells of different dimensions. Let us first consider the application of existing rendering techniques. Explicit cells, such as points, lines and triangles, can be rendered using traditional techniques such as a standard library (OpenGL or DirectX) and graphics hardware. Implicit cells require more advanced techniques, such as raytracing [Blo97] or polygonization algorithms for 3D implicit cells [Pas88]. These techniques have been extended in [Sch04] by rendering implicit cells of lower dimensions, defined as trimmed implicit surfaces and curves. Here, we assume that each cell of the implicit complex can be rendered individually using one of the above-mentioned techniques.

To render the flower model, one can choose different polygonal based and ray-trace based rendering techniques. If one wishes to visualize the implicit complex using a polygonal representation, one can easily convert each implicit cell to an explicit representation using an ad-hoc polygonization

algorithm ([Pas88, Sch04]), and then render the implicit complex with traditional graphics hardware. Figure 4 shows the polygonized sepal of a flower. To generate Fig. 4c a colour attribute implemented through procedural texturing was used.

To directly ray-trace an implicit complex, one needs to combine the existing ray-tracing techniques for explicit and implicit cells using a common Z-buffer.



**Figure 5. Rendering a flower modelled as an IC. The filaments are defined as 1D explicit cells, the petals (yellow and magenta) and sepals (green) are defined as 2D implicit cells, and the remaining components (stamens, receptacle and pistil) are defined as implicit 3D cells.**

The main problem is to render 1D cells as they can not be ray-traced directly since they are defined as curves and line segments. Therefore, 1D explicit and implicit cells have to be first rendered by techniques other than ray-tracing. For instance, one can retrieve the data stored in a frame buffer and a Z-buffer after using a polygonization routine and graphics hardware for rendering, or directly use an existing line drawing

algorithm. Then, the remaining cells of higher dimensions can be ray-traced all together. For each ray, the intersection points with the explicit cells are directly computed, and the intersection points for the implicit cells are procedurally evaluated.

Fig. 5 shows rendered images of the modelled flower. The 2D and 3D cells have attributes representing the colour based on the constructive hypervolume model [Pas01]. Note that for visualization purposes, we replace 1D cells with thin cylinders to be able to shade them. The petals and sepals have first been polygonized, as they are defined as 2D shapes, and then deformed by a forward mapping. First, 1D and 2D cells were rendered by ray-tracing resulting in a frame buffer and a z-buffer. Next, 3D implicit cells were ray traced and combined with the image already stored in the frame buffer, depending on the comparison of the depth of the current ray intersection with the implicit cell and the depth stored in the z-buffer.

## 6. CONCLUSION

Implicit complexes make it possible to combine a cellular representation and a constructive function representation into a uniform model. In this paper, we have described the theoretical framework and the implementation techniques for the construction and rendering of such models using a simple but representative case-study. We have paid particular attention to the practical problems of construction of a cellular-function model.

This relatively simple case-study has allowed us to show the benefits of the approach which are invaluable for complex real assemblies. Such benefits include: preserving the initial representation of all the components (however different they may be) and guaranteeing topologically correct definitions for all parts and relationships (in particular for boundaries). This approach also allows us to handle conformity between the object's geometry and its attributes which represent its non-geometric properties.

Future work directions include the development of specific operations applicable to entire implicit complexes, an extension of the model to time-dependent implicit complexes; further development of the multidimensional version of the model and its applications, and the implementation of a specialized modelling and animation language which uses this novel modelling technique.

## 7. REFERENCES

- [Adz02] Adzhiev V., Kartasheva E., Kunii T., Pasko A., Schmitt B.: Hybrid cellular-functional modelling of heterogeneous objects, *Journal of Computing and Information Science in Engineering, Transactions of the ASME* 2, 4 (2002), 312-322.

- [Blo97] Bloomenthal J. et al.: *Introduction to Implicit Surfaces*. Morgan Kaufmann, 1997.
- [Cut02] Cutler B., Dorsey J., McMillan L., Mueller M., Jagnow R.: A procedural approach to authoring solid models. *In Proc. SIGGRAPH'02, ACM TOG* 21, 3 (2002), 302-311.
- [Fom97] Fomenko A.T., Kunii T.L.: *Topological modeling for visualization*, Springer-Verlag, Tokyo, 1997.
- [Jac99] Jackson T., Liu H., Patrikalakis N., Sachs E., Cima, M.: Modeling and designing functionally graded material components for fabrication with local composition. *Control, Materials and Design* 20, 2/3 (1999), 63-75.
- [Kar03] Kartasheva E., Adzhiev V., Pasko, A., Fryazhinov O., Gasilov V.: Surface and volume discretization of functionally-based heterogeneous objects. *Journal of Computing and Information Science in Engineering, Transactions of the ASME* 3, 4 (2003), 285-294.
- [Kun99] Kunii T.: Valid computational shape modeling: design and implementation. *International Journal of Shape Modeling* 5, 2, (1999), 123-133.
- [Kun03] Kunii T.L., Ibusuki M., Pasko G., Pasko A., Terasaki D., Hanaizumi H.: Modeling of conceptual multiresolution analysis by an incrementally modular abstraction hierarchy. *IEICE Transactions on Information and Systems*, vol. E86-D, No. 7, 2003, pp. 1181-1190.
- [Mas67] Massey W.S.: *Algebraic topology: An introduction*. Harcourt, Brace&World, Inc, 1967.
- [Mid00] Middleditch A., Reade C., Gomes A.: Point-sets and cell structures relevant to computer aided design. *Int. Journal of Shape Modeling* 6, 2, (2000), 175-205.
- [Ohm01] Ohmori K., Kunii T.: Shape modeling using homotopy. *In Proc. Int. Conf. on Shape Modeling and Applications*, IEEE Computer Society, 2001, 126-133.
- [Par01] Park S. M., Crawford R., Beaman J.: Volumetric multi-texturing for functionally gradient material representation. *In Proc. Sixth ACM Symposium on Solid Modeling and Applications*, ACM Press, 2001, 216-224.
- [Pas88] Pasko A., Pilyugin V., Pokrovskiy V.: Geometric modelling in the analysis of trivariate functions. *Computers and Graphics* 12, 3/4 (1988), 457-465.
- [Pas95] Pasko A., Adzhiev V., Sourin A., Savchenko V.: Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer* 11, 8 (1995), 429-446.
- [Pas01] Pasko A., Adzhiev V., Schmitt B., Schlick C.: Constructive hypervolume modeling. *Graphical Models* 63, 6 (2001), 413-442.
- [Pao93] Paoluzzi A., Bernardini F., Cattani C., Ferrucci V.: Dimension-independent modeling with simplicial complexes. *ACM TOG* 12, 1 (1993), 56-102.
- [Ros90] Rossignac J., O'Connor M. SGC: A dimension independent model for pointsets with internal structures and incomplete boundaries. *In Geometric modeling for product engineering*, ed. by M. Wozny, J. Turner, K. Preiss, 1990.
- [Sch04] B. Schmitt, A. Pasko, G. Pasko, T. Kunii: Rendering trimmed implicit surfaces and curves. *In Proc. Afrigraph 2004*, South Africa, ACM SIGGRAPH publication, 2004, pp. 7-13.

# Constructing Smooth Non-Manifold Meshes of Multi-Labeled Volumetric Datasets

Bernhard Reitinger, Alexander Bornik, Reinhard Beichel

Institute for Computer Graphics and Vision  
Graz University of Technology  
Inffeldgasse 16/II  
A-8010 Graz, Austria  
contact: breiting@icg.tu-graz.ac.at

## ABSTRACT

This paper presents a method for constructing consistent non-manifold meshes of multi-labeled volumetric datasets. This approach is different to traditional surface reconstruction algorithms which often only support extracting 2-manifold surfaces based on a binary voxel classification. However, in some – especially medical – applications, multi-labeled datasets, where up to eight differently labeled voxels can be adjacent, are subject to visualization resulting in non-manifold meshes. In addition to an efficient surface reconstruction method, a constrained geometric filter is developed which can be applied to these non-manifold meshes without producing ridges at mesh junctions.

## Keywords

surface reconstruction, mesh generation, multi-labeled volume, constrained smoothing

## 1 INTRODUCTION

Surface reconstruction for volumetric datasets is an important method for exploring important features especially in the medical field. By segmenting stacks of 2D gray-valued images (e.g. CT, MR images), 2-manifold meshes in form of iso-surfaces can be extracted and visualized. The traditional method is the *Marching Cubes* (MC) algorithm proposed by Lorensen and Cline in [Loren87] or some of its variations like [Lewin03, Labsi02, Lopes03] generating triangular models. All of these methods are concerned about 2-manifold meshes (homeomorphic to a sphere) which only allow mesh interfaces between two different materials (one below and one above a certain threshold).

However, in some applications multiple coherent surfaces should be reconstructed consistently based on a non-binary classification resulted from a previous image segmentation or labeling like – for example – a full segmentation of the Visible Human Project [Acker98] where each organ is tagged with a certain label and connected to other organs. All interfaces between adjacent organs must be extracted consistently. A naive solution for this problem would be to apply the MC iteratively on each labeled region while masking out all other regions. Although all interfaces would be extracted, a lot of inconsistencies are expected even between two adjacent regions because two surfaces of one interface will be generated which might not fit together. This gets even worse if more than two materials are meeting at one *cell* (a *cell* is defined by its eight adjacent voxels). Different to the traditional *MC* which generates a 2-manifold, our algorithm inevitably produces non-manifolds if more than two materials meet at one *cell*.

This paper proposes an efficient method for extracting such non-manifold surfaces based on labeled volumetric datasets in a consistent way. The resulting mesh can either be used for visualization or as input for a volumetric mesh generator. Our algorithm neither generates holes nor cracks. Depending on the resolution of the input dataset, staircase artifacts can occur which are smoothed

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 80-903100-7-9  
WSCG'2005, January 31–February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency Science Press

by applying customized topology-invariant filters enhancing the overall mesh quality. For this reason, we extended two existing filters. After surface relaxation, triangle simplification may be applied using the quadric error metric [Garla97].

This work was motivated by a medical project which is concerned about virtual liver surgery planning [Borni03]. The presented algorithm can be used for two different tasks within this project. At first, a visualization of liver segments by their interface boundaries is provided which allows surgeons to examine the location and size of each single liver segment (see Figure 10). Secondly, the output mesh can be used as input for a consequent volumetric mesh generator by applying the algorithm proposed in [Si02]. An example for one liver dataset is shown in Figure 1. For both cases, the liver is labeled a priori using the segment classification algorithm proposed in [Beich04] where each liver voxel is tagged with a certain material value indicating one of eight different liver segments (see Figure 1(a)). Due to the liver’s anatomy, more than two different segments can be adjacent. Therefore, the presented reconstruction method is necessary in order to handle multiple labels within one *cell*. The generated mesh can be classified in two different kinds of surfaces; one domain boundary surface covering the object itself and multiple interfaces which build the interior structure (surfaces) of the object.

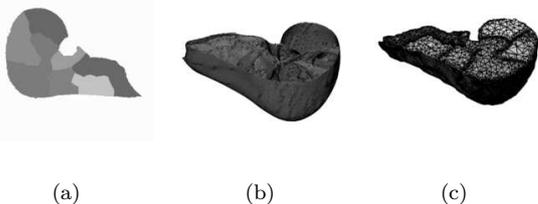


Figure 1: Visualization of a liver dataset containing different liver segments. (a) A slice of the labeled input volume, (b) non-manifold extracted using the proposed algorithm, (c) a volumetric mesh based on the mesh in (b) using the TetGen library [Si02].

The rest of this paper is organized as follows: Section 2 discusses related work for existing surface reconstruction methods. Section 3 presents our method for multi-labeled surface generation. Section 4 outlines an extension to existing geometric filters which are applied on the generated surface. Section 5 presents a simplification method for decreasing the number of generated triangles. Results and screenshots are shown in Section 6 and a conclusion closes this report.

## 2 RELATED WORK

The traditional method for extracting iso-surfaces based on a certain threshold is the *Marching Cubes* algorithm [Loren87] which distinguishes between two different domains (above and below a certain iso-value). Given a binary classification only  $2^8$  (256) different configurations can occur in a *cell* which can be implemented using look-up tables. Recently, Lewiner et al. presented an extension avoiding topological errors which can occur due to uncertainties [Lewin03].

Bloomenthal and Ferguson described in [Bloom95] one of the first approaches for generating surfaces from non-binary classifications which rely on implicit surface modelling and computational solid geometry. By subdividing cubic cells into tetrahedra, a triangulation is constructed algorithmically. This approach generates a large amount of triangles which can also be degenerate and not well-shaped.

Hege et al. presented a different approach for extracting non-manifold surfaces based on a non-binary classification [Hege97]. By using probabilities assigned to each voxel, cells are subdivided producing a lot of intermediate triangles. Therefore, a post-processing patch generation must be initiated in order to reduce the large number of faces. In their approach, a look-up table was implemented supporting up to three different materials meeting at one *cell*.

Another algorithm for generating surfaces based on a non-binary classification was presented by Wu and Sullivan [Wu03]. In their work, an extension for the traditional *Marching Cubes* was developed supporting 2D and 3D datasets.

Different to these related algorithms, we present an efficient yet simple approach which supports up to eight different materials meeting at one *cell*. In the following we will describe our method in detail.

## 3 MESH GENERATION

The input for our algorithm is a rectilinear labeled volumetric dataset where a distinct label (material) is assigned to each voxel  $\mathbf{V}$  at position  $(x, y, z)$ . A zero label indicates background (outside of the domain) and all other labels unequal to zero assign material. For a binary classification,  $2^8$  (256) different cases can occur at one *cell*. Therefore, look-up tables exist for the *Marching Cubes* algorithm to gain performance. If the dataset con-

sists of multiple different labels (non-binary),  $8^8$  (16777216) cases are possible within one *cell* and a look-up table to cover all these configurations is not feasible.

### 3.1 The Idea in 2D

The main idea of our algorithm is based on a cell subdivision strategy and will first be explained for the 2D case. Each non-homogeneous *cell* having two or more different materials is subdivided by inserting a *cell* mid-point. Additionally, segment mid-points are generated if the labels of two adjacent voxels (nodes) are different. For each generated segment mid-point a new segment (line) to the *cell* mid-point is generated. Figure 2 shows the four possible configurations for the 2D case. By permutating these cases, all  $4^4$  can be captured. If using the traditional *Marching Cubes* scheme for a *cell* with three different labels, a triangular void would be generated (see Figure 3). This generated void is invalid and cannot be assigned to a material.

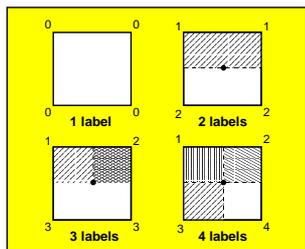


Figure 2: Four possible configurations in 2D.

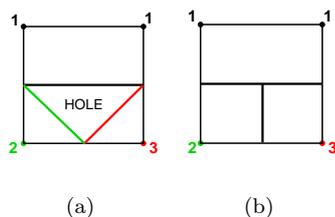


Figure 3: Three different materials meeting at one *cell* (quad). (a) Traditional *Marching Cubes* generates a “hole”, (b) correct partitioning with three different materials.

As we can observe in Figure 2, our algorithm does produce valid T-junctions at *cell* mid-points and is therefore a conforming triangulation.

Considering a 2D *cell* with two different materials where two equal materials are opposite, the topology is preserved due to the insertion of a *cell* mid-point. However, this configuration produces singularities which must be considered for filtering in order to prevent spikes (see later).

### 3.2 3D Algorithm

As shown in the previous section, the 2D algorithm inserts additional *cell* mid-points in order to reconstruct interfaces correctly. The same strategy is used for the 3D case. Before going into more details of the algorithm we need to define some constraints. At first, our generated mesh is a non-manifold represented by a simplicial complex of 2-simplices. By considering this constraint, we guarantee a conforming mesh without invalid triangle intersections.

Having a rectilinear volumetric grid, a *cell*  $\mathbf{C}$  at location  $(x, y, z)$  is defined by

$$\mathbf{C}_{x,y,z} = [\mathbf{V}_{x,y,z}; \mathbf{V}_{x+1,y,z}; \mathbf{V}_{x+1,y+1,z}; \mathbf{V}_{x,y+1,z}; \mathbf{V}_{x,y,z+1}; \mathbf{V}_{x+1,y,z+1}; \mathbf{V}_{x+1,y+1,z+1}; \mathbf{V}_{x,y+1,z+1}].$$

The generation algorithm is initiated by processing each single *cell*  $\mathbf{C}_{x,y,z}$  of the rectilinear volume (see Algorithm 1). If a *cell* is not homogeneous (having more than one material), it is enqueued in a list  $Q$ .

---

#### Algorithm 1 Pre-processing

---

**Input:** rectilinear labeled 3D volume  $V$

**Ensure:**  $\mathbf{V}_{x,y,z} = \{l : l \in \mathbb{N}\}$

```

for all  $\mathbf{C}_{x,y,z}$  do
  if  $\mathbf{C}_{x,y,z}$  is !homogeneous then
     $Q.add(\mathbf{C}_{x,y,z})$ 
  end if
end for

```

**Output:**  $Q$

---

In the second step, only enqueued *cells* are processed. Depending on the homogeneity of the input, the processing list can be very small. The second sweep is also called *simplex generation* because for a given node configuration, faces (triangles) are generated using a small look-up table. All generated faces and vertices are stored in a compact data structure where each vertex is unique and duplicates are avoided. This is necessary to guarantee a correct post-processing (i.e. smoothing or simplification). Algorithm 2 describes the complete second sweep and its details are explained in the following sections. The output of the *simplex generation* algorithm is an indexed face set  $I$  storing the triangulation of the whole domain and its according vertex list  $VL$ .

#### 3.2.1 Face Generation

Our face generation strategy is very efficient. In any case of a non-homogeneous *cell*, a *cell* mid-point is generated which subdivides this *cell* into 8 sub-cells. Additionally, face mid-points are gen-

---

**Algorithm 2** Simplex generation

---

**Input:**  $Q$ 

```
for all  $C_{x,y,z}$  in  $Q$  do
  subdivide  $C_{x,y,z}$  by creating a cell mid-point
  if  $C_{x,y,z}$  has one zero label then
     $C_{x,y,z} = \text{EXTERNAL\_NODE}$ 
  else
     $C_{x,y,z} = \text{INTERNAL\_NODE}$ 
  end if
  generate vertices and faces using a LUT
  classify cell nodes of  $C_{x,y,z}$ 
  for all generated faces  $f_i$  do
     $I.add(f_i)$ 
     $VL.add(\text{vertices of } f_i)$ 
  end for
end for
```

**Output:**  $I, VL$ 

---

erated if a face of  $C_{x,y,z}$  has non-homogeneous nodes. All interfaces between two sub-cells are investigated if two adjacent sub-cells have different materials. If a difference is found, two triangles are spanned between these two sub-cells covering the interface. For efficiency, we use a look-up (LUT) table to get the adjacencies between sub-cells. For each *cell* we need 12 look-ups to generate adjacent faces within one *cell*. Figure 4 shows some face generation examples for two, three or more materials meeting at one *cell*.

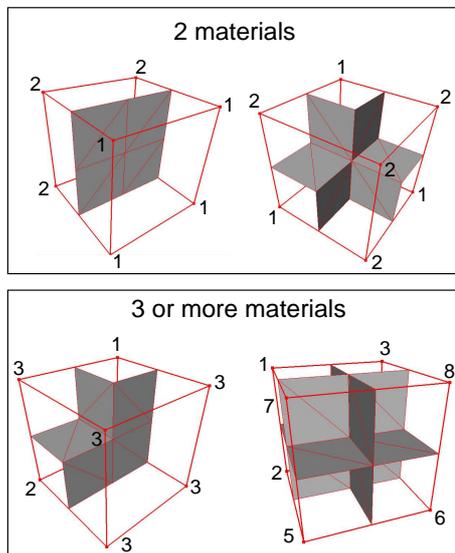


Figure 4: Examples of face generations for the 3D case

By using this generation method, adjacent *cells* are connected correctly. As the input has no quantity information but only labels, we cannot apply any interpolation scheme ad hoc. However, as the

resulting mesh should be visual appealing, we need to apply a filter for smoothing the surface to reduce the staircase characteristics produced by our algorithm. However, in order to guarantee a correct smoothing of these non-manifold meshes, we have to classify the nodes in a *cell* due to a certain configuration.

### 3.2.2 Node Classification

Each node (or vertex in the manner of surfaces) must be classified according to its location. This guarantees a correct smoothing of non-manifold meshes and avoids ridges between the domain boundary and interface boundaries. Figure 5 shows a *cell* with its possible vertex locations. At first, the possible 27 vertices are named by their locations in the *cell*. *Corner-points* are the original voxels of the dataset each of them storing a certain label. Face mid-points are called *2D center-points* and are generated if a *cell's* face is not homogeneous. *Border-points* divide a segment between two adjacent *corner-points*. And finally, the *cell* mid-point is classified as *3D center-point*. Additionally, we define certain types of nodes as the following:

**Definition 1: Domain Node ( $D$ )**

A node (vertex) is classified as *domain node*, if the node is part of the enclosing domain surface if one exists.

**Definition 2: Interface Node ( $I$ )**

A node (vertex) is classified as *interface node*, if the node is part of an interior surface which separates two different interior materials.

**Definition 3: Junction Node ( $J$ )**

A node (vertex) is classified as *junction node*, if the node is part of an interior surface and separates more than two different interior materials.

**Definition 4: Domain Junction Node ( $DJ$ )**

A node (vertex) is classified as *domain junction node*, if the node is part of an exterior surface and separates at least two interior materials but also one exterior material.

Table 1 shows the mapping between nodes and possible types. A *corner-point* will not be classified at all because the surface will never pass

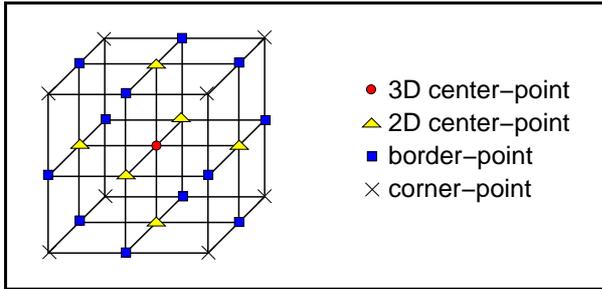


Figure 5: Classification of nodes in a *cell*.

|                 | D | I | J | DJ |
|-----------------|---|---|---|----|
| 3D center-point | x | x | x | x  |
| 2D center-point | x | x | x | x  |
| border-point    | x | x |   |    |
| corner-point    |   |   |   |    |

Table 1: Mapping of nodes to node types.

through it. A *border-point* is a *domain node* if one of its adjacent *corner-points* has a zero material, else it will be assigned as *interface node*. A *2D center-point* is assigned as *junction node* if its corresponding face has no zero material and material count (number of different materials per face)  $\geq 3$ . If a face has at least one zero material, it is assigned as *domain junction node*. If the material count for one face is  $< 3$  and the face has no zero material it is assigned as *interface node*, else *domain node*. *3D center-points* are classified according to a *cell's* classification. If  $\mathbf{C}_{x,y,z}$  is assigned `EXTERNALNODE` and material count is 2, then we classify it as *domain node*, else *domain junction node*. If  $\mathbf{C}_{x,y,z}$  is an `INTERNALNODE` and material count is 2, the *3D center-point* is assigned as *interface node*, else, *junction node*.

If this type mapping is applied, ridges are avoided and self-intersections are prevented if smoothing the surface. Figure 6 shows a comparison of two examples one without applying this classification scheme and one with these considerations.

## 4 SURFACE FILTERING

Surface filtering is necessary to reduce the staircase artifacts generated by our face generation method. We use surface filters which only affect geometry and do not alter topology. Beside smoothing, geometric filters are also used for improving the overall quality of the mesh. Different geometric filters exist in literature which can be applied for smoothing. The most sim-

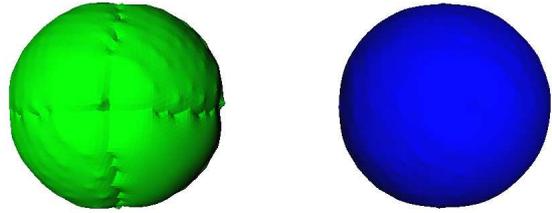


Figure 6: If node classification is not assigned, ridges occur (see left image). If the classification is applied, no ridges are generated.

ple but effective smoothing filter is the *Laplacian* filter [Field88]. Each vertex at position  $x_i$  is smoothed iteratively by using the following formula:

$$x_{i+1} = x_i + \lambda \Delta_i \quad (1)$$

where  $\Delta_i$  is defined as:

$$\Delta_i = \sum_{j=1}^N w_{ij} (x_j - x_i) \quad (2)$$

where  $w_{ij}$  specifies the weight between  $x_i$  and its neighboring  $x_j$  and  $\sum w_{ij} = 1$ . A good choice for  $w_{ij}$  is  $\frac{1}{N}$ , where  $N$  is the number of adjacent vertices. The scale factor  $\lambda$  ( $0 < \lambda < 1$ ) influences the degree of smoothness and is defined equally for all vertices. Intrinsically, this filter also improves the overall mesh quality. The big advantage of this filter is its simplicity, however, it produces shrinkage if applied many times.

Therefore, an extension to the *Laplacian* filter was presented by Taubin [Taubi95] which avoids shrinkage. The main idea is to apply two consecutive *Laplacian* steps: at first, using a positive scale factor  $\lambda$  and then using a negative scale factor  $\mu$ , greater in magnitude than  $\lambda$  ( $0 < \lambda < -\mu$ ).

Applying the standard filter for our produced mesh, ridges on the domain boundary can occur as shown in Figure 7(a). This is because vertices on the domain boundary which are adjacent to interface nodes are attracted by these nodes and produce valleys. Therefore, we have performed the node classification which will be considered now for the filter. The  $(x_j - x_i)$  expression in Equation 2 will only be calculated under certain constraints. If the vertex at location  $x_i$  is assigned as *domain node* or *interface node*, all adjacent vertices at  $x_j$  are used for calculating  $\Delta_i$ . However, if a vertex at location  $x_i$  is a *domain junction node*,

only vertices at  $x_j$  with classification *domain junction node* and *domain node* are considered. Similarly, if vertices at  $x_i$  is a *junction node*, only vertices at  $x_j$  of type *junction node* or *domain junction node* are used. The result can be seen in Figure 7(b).

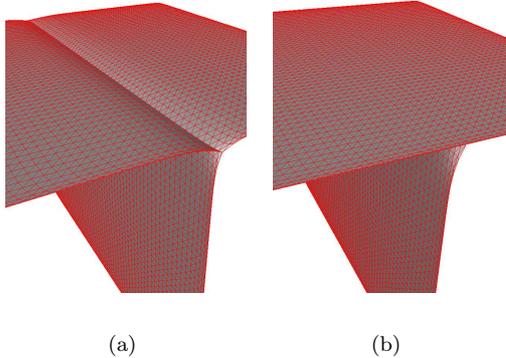


Figure 7: By applying the constrained *Laplacian* (Taubin) filter, ridges are avoided successfully as shown in this figure.

We observed that singularities can occur if voxels are only 26-connected for 3D datasets. This gets a problem if the surface is filtered. Two different material regions are only connected through one vertex and filtering would generate unwanted spikes. In the current implementation we assign a freeze label to these vertices which means that they are not moved during smoothing and avoids therefore these spikes.

## 5 MESH SIMPLIFICATION

In order to reduce the triangle count a simplification algorithm is applied which is invariant to the surface's topology. We are using the quadric error metric which was introduced by Garland and Heckbert [Garla97]. The algorithm works with an iterative contraction of vertex pairs to simplify models and maintains surface error approximations using quadric matrices. The advantage of this algorithm is that it supports non-manifold meshes. Therefore, it can be applied to our data structure without any modifications.

## 6 RESULTS

After explaining all different components of our mesh generation method, some results are presented. The test input datasets are either gen-

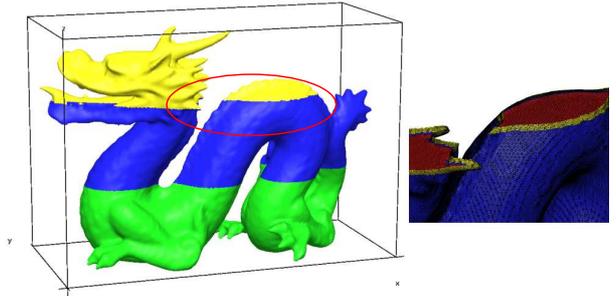


Figure 8: Example of the dragon model which was generated using our method using the *Laplacian* filter with  $\lambda = 0.8$  and 30 iterations. The right image shows a zoom-in where the boundary interface can be seen clearly.

erated by a prior segmentation and labeling of a CT scan, or artificially generated by using the hardware-accelerated voxelization presented in [Reiti03]. Table 2 shows a summary of the input datasets with different resolution (size). The  $\#vtx$  and  $\#f$  indicate the generated vertices and triangles after the *simplex generation*. If the input resolution is high, Algorithm 1 is the dominant factor in the measured time. If a lot of non-homogeneous *cells* are found, Algorithm 2 takes more time. We have measured, that – on average – Algorithm 1 takes 83% and Algorithm 2 17% of time for the tested datasets.

|        | size                        | $\#vtx$ | $\#f$   | time |
|--------|-----------------------------|---------|---------|------|
| Dragon | $256^3$                     | 453921  | 910400  | 25.7 |
| Horse  | $128^3$                     | 63147   | 127960  | 2.6  |
| Liver  | $256 \times 256 \times 174$ | 632272  | 1272088 | 18.5 |
| Lung   | $128 \times 128 \times 148$ | 331934  | 668624  | 6.1  |

Table 2: Table showing results of different datasets. The time is measured in seconds.

Figure 8 shows the dragon model where the dataset is divided into multiple layers simulating interface boundaries. The right image shows a zoom-in showing three bounded materials (yellow, blue, red). Similarly, we also sub-divided the horse dataset into different labels. Figure 9(a) shows the raw output before smoothing, and Figure 9(b) displays the smoothed result using the constrained *Laplacian* filter.

We have also applied our algorithm on medical datasets which are generated using a segmentation and classification as described in [Beich04]. Figure 10 shows a trajectory of different smoothing levels performed on the liver dataset, starting from no smoothing up to 30 iterations. Similar to

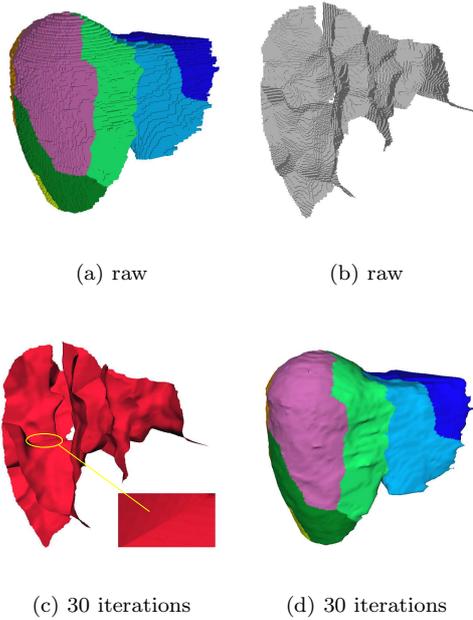


Figure 10: Construction of liver segment boundaries. Left image visualizes the liver with domain boundaries without smoothing. Image (b) shows the interior structure. Images (c) and (d) display a smoothed liver with  $\lambda = 0.88$  and 30 iterations. In (c) the ridge-free interface between multi-material regions can be seen.

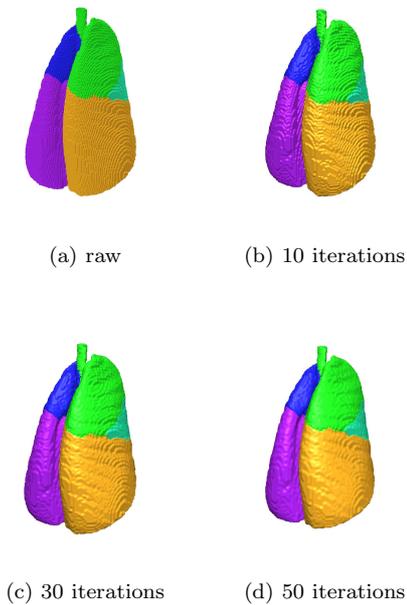


Figure 11: The sheep dataset with different levels of filtering. Left image is the raw output of the *simplex generation*. The consecutive images show the result of smoothing with the Taubin filter using  $\lambda = 0.88$  and  $\mu = -0.9$ .

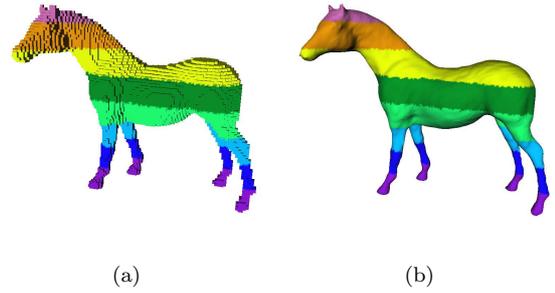


Figure 9: Example of the horse model. Left image shows the raw output of the surface generation. The right image is the result of a *Laplacian* smoothing using  $\lambda = 0.7$  and 20 iterations.

the human liver, we also applied our algorithm on a sheep lung which is shown in Figure 11. For this dataset we used the Taubin filter with different levels of smoothness.

As the resulting triangle counts are quite large for interactive usage, we applied a the simplification algorithm as explained in Section 5. Figure 12 shows two different datasets where the left model was simplified using the quadric error metric to a target face count of 70000 faces which decreases the number of triangles to about 10% of the original surface.

## 7 CONCLUSION

This paper presented an algorithm for generating non-manifold meshes based on a non-binary classification of volumetric datasets. The generation method consists of two main components, one pre-processing step and a consecutive *simplex generation*. As the raw output produces staircase artifacts constrained surface filters based on a vertex labeling scheme are applied. This prevents from generating ridges at interface boundaries and provides a high-quality mesh.

As the output surface is a conforming mesh, and therefore guarantees consistency, it can be used for a further volumetric mesh generation as presented in [Si02].

## ACKNOWLEDGMENTS

This work was supported by the Austrian Science Foundation (FWF) under grant P17066-N04.

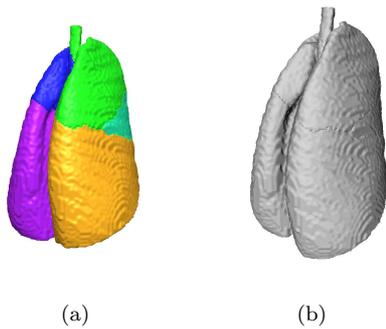


Figure 12: The sheep lung before and after simplification using the quadric error metric. Target face size is 70000.

## References

- [Acker98] M.J. Ackerman. The visible human project. *Proc. of the IEEE*, 86(3):504–511, 1998.
- [Beich04] R. Beichel, T. Pock, Ch. Janko, R. Zotter, B. Reitinger, A. Bornik, K. Palagyi, E. Sorantin, G. Werkgartner, H. Bischof, and M. Sonka. Liver segment approximation in CT data for surgical resection planning. In *In SPIE Medical Imaging '04*, San Diego, 2004. in print.
- [Bloom95] J. Bloomenthal and K. Ferguson. Polygonization of non-manifold implicit surfaces. In *Proc. of SIGGRAPH '95*, pages 309–316, 1995.
- [Borni03] A. Bornik, R. Beichel, B. Reitinger, G. Gotschuli, E. Sorantin, F. Leberl, and M. Sonka. Computer aided liver surgery planning: An augmented reality approach. In R.L. Galloway, editor, *Medical Imaging 2003, Proceedings of SPIE*, volume 5029. SPIE Press, May 2003.
- [Field88] D.A. Field. Laplacian smoothing and delaunay triangulations. *Communications in Applied Numerical Methods*, 4:709–712, 1988.
- [Garla97] M. Garland and P.S. Heckbert. Surface simplification using quadric error metrics. In *Proc. of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997.
- [Hege97] H.-C. Hege, M. Seebaß, D. Stalling, and M. Zöckler. A generalized marching cubes algorithm based on non-binary classifications. Technical report, Konrad-Zuse-Zentrum (ZIB), 1997. SC 97-05.
- [Labsi02] U. Labsik, K. Hormann, M. Meister, and G. Greiner. Hierarchical iso-surface extraction. *Journal of Computing and Information Science in Engineering*, 2(4):323–329, December 2002.
- [Lewin03] T. Lewiner, H. Lopes, A. Wilson Vieira, and G. Tavares. Efficient implementation of marching cubes: Cases with topological guarantees. *Journal of Graphics Tools*, 8(2):1–15, 2003.
- [Lopes03] A. Lopes and K. Brodlie. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):16–29, 2003.
- [Loren87] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.
- [Reiti03] B. Reitinger, A. Bornik, and R. Beichel. Efficient volume measurement using voxelization. In *Proc. of the Spring Conference on Computer Graphics 2003*, pages 57–64, Budmerice, April 2003. Comenius University, Bratislava.
- [Si02] H. Si. *TetGen: A 3D Delaunay Tetrahedral Mesh Generator, Version 1.2 User Manual*. Weierstrass Institute for Apply Analysis and Stochastics, 2002. No. 4.
- [Taubi95] G. Taubin. Curve and surface smoothing without shrinkage. In *Proc. of the Fifth International Conference on Computer Vision*, pages 852–857. IEEE Computer Society, 1995.
- [Wu03] J. Wu and J.M. Sullivan. Multiple material marching cubes algorithm. *Journal for Numerical Methods in Engineering*, 2003.

# Feature Preserving Volumetric Data Simplification for Application in Medical Imaging

C. Jin T. Fevens S. Li S. P. Mudur  
Computer Science and Software Engineering Department  
Concordia University  
1455 de Maisonneuve Blvd. W.  
Canada, H3G 1M8, Montreal, QC  
{chao\_jin, fevens, shuo\_li, mudur}@cs.concordia.ca

## ABSTRACT

In this paper, we propose a new simplification algorithm to reduce the large amount of redundancy in 3D medical image datasets and generate a new representation in tetrahedral meshes with considerably lower storage requirements. In the proposed algorithm, we first apply level set segmentation to partition the volume data into several homogenous sub-regions. We consider the interior boundaries between sub-regions as contributing more to the significant visible features. Next we convert the regular grid data into a tetrahedral representation and simplify the irregular volume representation by iteratively removing tetrahedra without significantly altering the exterior boundary or interior field distribution features. Within each sub-region, field gradients, tetrahedral aspect ratio changes and variances of interior region values are further used so as to maintain features of the original dataset in regional interiors. We tested our algorithm on several 3D medical datasets. The promising results show that we reduce redundancy and yet preserve important features and structures present in the original data set for decimation rates up to 50%.

## Keywords

Medical Imagery, Mesh Simplification, Irregular Tetrahedral Meshes, Level of Detail

## 1. INTRODUCTION

In the past few decades, 3D medical image analysis has become one of the most active research areas supporting computer aided diagnosis. Medical scanning devices, such as those generating Computed Tomograph (CT) or Magnetic Resonance Imaging (MRI) scans, are continuously increasing in their resolution capabilities. The resulting volumetric data sets are thus getting larger with increasing demands on time and storage resources for tasks such as archiving, loading, rendering, transmission, etc.

A more efficient volumetric representation, which

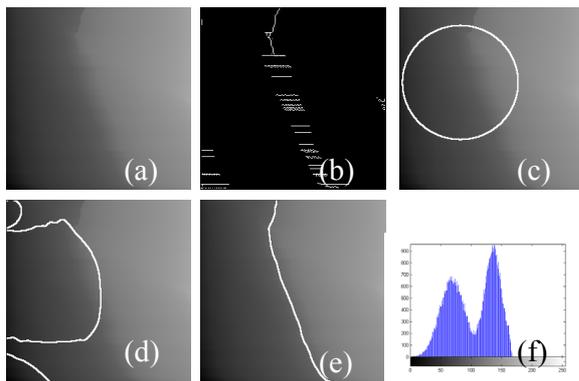
maintains the same features but uses less physical storage space, is necessary for further visualization or analysis [Kau91a, Kau93a, Nie00a]. However, due to the specific characteristics of regular grids, it is very hard to achieve accurate simplification. For example, to simply replace clusters of voxels (a volume element in a 3D regular grid) with ‘supervoxels’ would introduce too much noise and blur significant features. Therefore, we use another representation of volume data, a tetrahedral mesh, to perform the simplification. The tetrahedral mesh has attracted much attention over the last decade or so since it provides greater flexibility and other representations can be converted into tetrahedral meshes relatively easily. The basic representational primitives, tetrahedra, are easy to deform and to merge or subdivide. It is convenient to assign properties and functions to the vertices and to tetrahedral cells. Computational steps such as interpolation, integration, and differentiation are fast and often can be done in similar forms. For example, finite element analysis is conveniently performed on tetrahedral meshes. Also, the triangles that are generated by the faces of tetrahedra may be rendered using hardware acceleration [Yao00a].

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 80-903100-7-9  
WSCG'2005, January 31-February 4, 2005  
Plzen, Czech Republic.  
Copyright UNION Agency – Science Press*

Simplification of tetrahedral meshes has been studied in the past decade [Chi03a, Cho02a, Cig00a, Gel99a, Hong03a, Sta98a, Tro98a, Tro99a]. In 1998 and 1999, Trotts et al. [Tro98a, Tro99a] presented a method for simplification by extending a polygonal geometry deduction technique with a trivariate spline function associated with each tetrahedron to detect the features. In 1998, Staadt and Gross [Sta98a] extended the work of Hoppe [Hop96a] on progressive triangular meshes to tetrahedral meshes to generate an incrementally refined progressive tetrahedralization based on edge collapse. In 2000, Cignoni et al. [Cig00a] gave a framework for incremental 3D mesh simplification also based on edge collapse. In 2002, Chopra and Meyer [Cho02a] introduced a fast tetrahedral decimation algorithm called TetFusion. It preserves all cells, with large gradients. Recently, two groups of researchers, Hong and Kaufman [Hong03a], and Chiang and Lu [Chi03a] used Morse theory to detect topological critical points of original data, with the aim to preserve the topological structure of isosurfaces during simplification.

However, existing algorithms are not well suited for 3D medical data sets because the data are naturally divided into different regions, which need to be treated individually, and the data obtained from scanning devices are usually noisy. Traditional volumetric simplification algorithms deal with features based on local measurement, such as gradient or aspect ratio [Cho02a, Cig00a, Sta98a, Tro98a, Tro99a]. But local measurements sometimes lead to the misidentification of features. For example, in Fig. 1, we can see that the data is composed of two regions. However, by the local feature measurement method, based on the gradients, it cannot partition the two regions. Instead these methods misidentify the boundary feature as a set of horizontal lines.



**Figure 1:** a) a 2D dataset which includes the two Gaussian distributions shown in f). b) the incorrect features detected by gradient method. c) and d) two intermediate stages in level set segmentation process. e) feature detection result by using level set method.

The drawback in the method of maintaining the topological structure of isosurfaces [Chi03a, Hong03a], is that it is too sensitive to noise, which is very likely to be present in 3D medical data [Web03a]. Thus, they cannot achieve high decimation rates.

To meet the requirement of simplifying tetrahedral medical images while preserving visible features, our new method first applies level set segmentation, which is robust to noise, to partition the volumetric medical data into several homogenous regions. Then, it simplifies each region individually. The final result of our method is a tetrahedral mesh, which maintains the fidelity of features as present in the original data while using much less physical storage space.

This paper is organized as follows. In section 2, we introduce our methodology. Section 3 shows our experimental results, and the final section 4 contains our conclusions.

## 2. METHODOLOGY

The first step of our algorithm is to apply level set segmentation in a preprocessing phase to partition the data into several homogenous regions. Then, the regular grid data is converted into an irregular mesh by simple tetrahedralization. The boundaries between sub-regions represent different features and should be preserved during the simplification phase. Finally, the tetrahedral cell decimation is applied to each sub-region individually. Within each sub-region, the algorithm will always preserve features detected by local feature measurement. Thus, by this approach, we preserve more of the significant features than other existing algorithms. Also by simplifying within regions without worrying about topology, we have greater flexibility during the simplification phase while at the same time not losing out on any visible features.

### Level set segmentation

#### 2.1.1 Level Set Function

In 1988, Osher and Sethian [Osh88a] proposed a new segmentation approach based on the class of deformable models, referred as “level set” or “geodesic active contours or surfaces”. The approach, based on an evolving curve naturally dividing the image, represented as the domain  $\Omega \in \mathbb{R}^2$ , into two parts. The method has become popular because of its ability to capture the topology of shapes in 3D datasets. The curve  $C$  is represented implicitly via a *Lipschitz Function*, which is also referred to as the level set function  $\phi$ , where  $C = \{(x, y) | \phi(x, y) = 0\}$ . The curve divides the image into a region where  $\phi(x, y)$  is positive valued and a complementary region where  $\phi(x, y)$  is negative

valued. The evolution of the curve is given by the zero-level curve at time  $t$ , and the curve  $C$  evolves according to direction and speed of dictated by force  $F$ , as described in Equation (1).

$$\begin{aligned} \frac{\partial \phi}{\partial t} &= |\nabla \phi| F, \\ \phi(0, x, y) &= \phi_0(x, y) \end{aligned} \quad (1)$$

Equation (1) is level set function, where the set  $C = \{(x, y) | \phi_0(x, y) = 0\}$  defines the initial contour.

A particular case is the motion by mean curvature. This is given by

$$F = \text{div} \left( \frac{\nabla \phi}{|\nabla \phi|} \right)$$

as the curvature of  $\phi$  passing through  $(x, y)$ , which when substituted into Equation (1) gives us:

$$\begin{cases} \frac{\partial \phi}{\partial t} = |\nabla \phi| \text{div} \left( \frac{\nabla \phi}{|\nabla \phi|} \right), & t \in (0, \infty), x \in \mathbb{R}^2 \\ \phi(0, x, y) = \phi_0(x, y), & x \in \mathbb{R}^2 \end{cases} \quad (2)$$

where  $\phi_0$  is initial level set function.

### 2.1.2 Level Set Segmentation

In 2000, Samson *et al.* [Sam00a] proposed a supervised classification model to find a partition composed of homogeneous regions, assuming that the number of classes and their attribute value properties are known. Therefore, for segmentation into  $K$  regions, the proposed method uses  $K$  level set functions  $\phi_i$  ( $i \in [1, K]$ ) to represent each of region as shown in Equation (3), which demonstrates the energy function. It consists of three terms: minimal variance energy  $E_{\text{minv}}$ , minimal length energy  $E_{\text{minl}}$ , and non-overlap energy  $E_{\text{nonover}}$ .

$$\begin{aligned} E(\phi_1, \dots, \phi_k) &= E_{\text{minv}}(\phi_1, \dots, \phi_k) + E_{\text{minl}}(\phi_1, \dots, \phi_k) \\ &\quad + E_{\text{nonover}}(\phi_1, \dots, \phi_k) \end{aligned} \quad (3)$$

where

$$E_{\text{minv}}(\phi_1, \dots, \phi_k) = \sum_{i=1}^k e_i \int_{\Omega} H_{\alpha}(\phi_i)(1 - c_i)^2 dx dy$$

$$E_{\text{minl}}(\phi_1, \dots, \phi_k) = \sum_{i=1}^k \gamma_i \int_{\Omega} \delta_{\alpha}(\phi_i) |\Delta \phi_i| dx dy$$

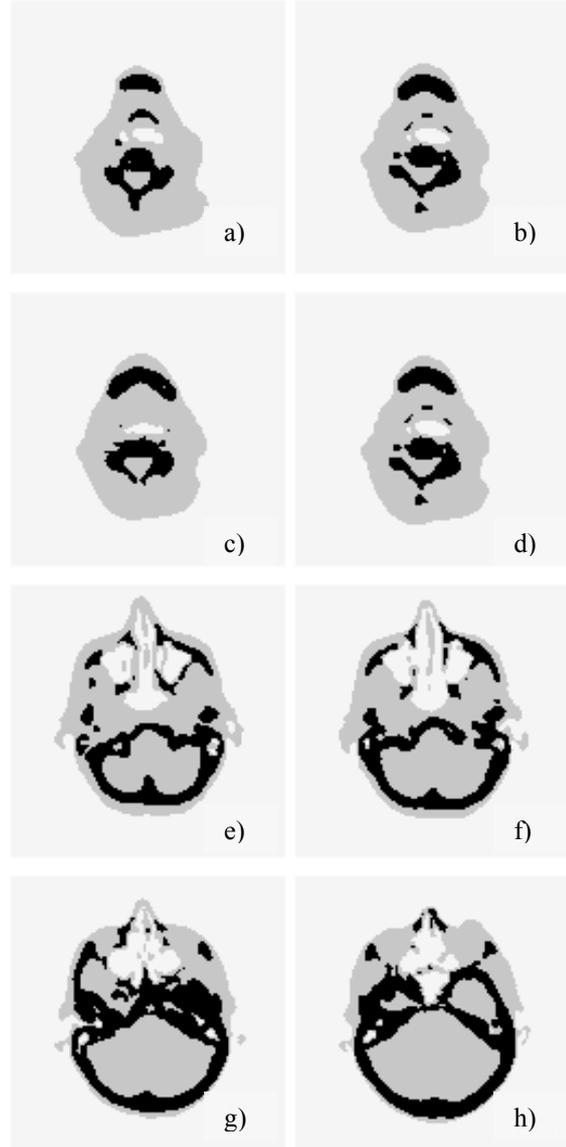
$$E_{\text{nonover}}(\phi_1, \dots, \phi_k) = \frac{\lambda}{2} \int_{\Omega} (\sum_{i=1}^k H_{\alpha}(\phi_i) - 1)^2 dx dy$$

$$\forall i, e_i, \gamma_i, \lambda \in \mathbb{R},$$

The evolution of the level function  $\phi_i$  is shown in Equation (4).

$$\begin{aligned} \phi_i^{t+1} - \phi_i^t &= \Delta t \cdot \delta_{\alpha}(\phi_i^t) \cdot \\ &\quad \left[ v_i \text{div} \left( \frac{\Delta \phi_i}{|\Delta \phi_i|} \right) - e_i (I - c_i)^2 - \right. \\ &\quad \left. \beta \left( \sum_{i=1}^k H_{\alpha}(\phi_i) - 1 \right)^2 \right] \end{aligned} \quad (4)$$

In Equation (4), the  $H_{\alpha}(\cdot)$  is the *Heaviside Function*,  $c_i$  are the means of positive areas in level set function  $\phi_k$ , and  $\delta_{\alpha}(\cdot)$  is the Dirac delta function and  $v_i$  and  $\beta$  are constants.



**Figure 2: Region partition results. The segmentation phase divides the data into three homogenous regions, which are shown in white, gray and black. a) to d) are segmentation result for the CT-neck data; and e) to h) are for the CT-head data, shown in slices for easier identification.**

After performing the level set segmentation, we divide the data into different homogeneous regions (see Fig. 2), and label all vertices located on the interior boundaries of homogeneous regions as feature vertices; and all vertices located inside homogeneous regions as normal vertices (see Fig. 3).

## Region Based Feature Preserving Simplification Algorithms

### 2.2.1 Definitions

Before we begin the discussion of the feature preserving simplification algorithm, we introduce some definitions which we will need later.

- **Neighbor Sets**  
For a tetrahedral cell  $\tau$ ,  $A(\tau)$  defines a set of cells which share one and only one vertex with  $\tau$ ;  $D(\tau)$  defines a set of cells which share at least two vertices with  $\tau$ . We denote the union of  $A(\tau)$  and  $D(\tau)$  as  $Neighbor(\tau)$ , the neighbor set of cell  $\tau$ .
- **Normal Cell**  
If all vertices of a cell  $\tau$  are normal vertices and they are all located in same sub-region, we denote the cell  $\tau$  as a normal cell.
- **Feature Cell**  
If a cell contains one or more feature vertex, we label it as a feature cell.
- **Cross-Region Cell**  
If a cell has vertices in more than one sub-region, but it is not a feature cell, we label it as a cross region cell.
- **Cross-Region Neighbor Cell**  
If a cell is a normal cell, and it has a neighbor cell which is a cross region cell, we label it as a cross region neighbor cell.
- **Boundary Cell**  
If a cell contains at least one vertex on the external boundary, we label it as a boundary cell.

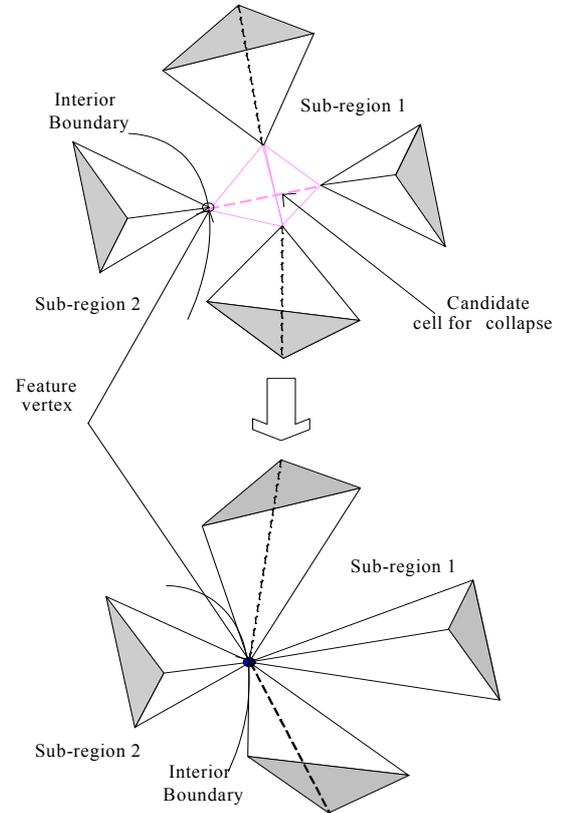
### 2.2.2 Feature Preserving Rules

After feature detection, a number of rules are defined to preserve both interior and exterior features.

1. **Normal Cell:** The normal cells may be collapsed freely. Here we define the collapse operation for normal cells as  $\tau$  to  $v_c$ , where  $v_c$  is the centroid of the cell. By choosing the centroid, the volume of the deleted cell is distributed evenly amongst the remaining local neighboring cells.
2. **Feature Cells:** If any feature cell  $\tau$  has more than one feature vertex, we do not perform the

collapse operation on it. If it has exactly one feature vertex  $v_f$ , we define the collapse operation as collapsing  $\tau$  to  $v_f$ .

3. **Cross-region Cell:** For any cross-region cell  $\tau$ , to achieve the decimation of different sub-regions individually, we do not perform the collapse operation on it.
4. **Cross-region Neighbor Cells:** For any cell  $\tau$ , if its  $A(\tau)$  or  $D(\tau)$  contains only one cross-region cell, doing a collapse operation to the centroid will cause the change of the structure of the sub-region. To avoid this problem, we define the collapse operation for such a cell as the collapse of  $\tau$  to the vertex  $v_n$  which is the common vertex with the cross-region cell. If its  $A(\tau)$  or  $D(\tau)$  contains more than one feature cell and more than one feature vertex, we do not collapse the cell. Cells in  $D(\tau)$  may contain any number of feature points.



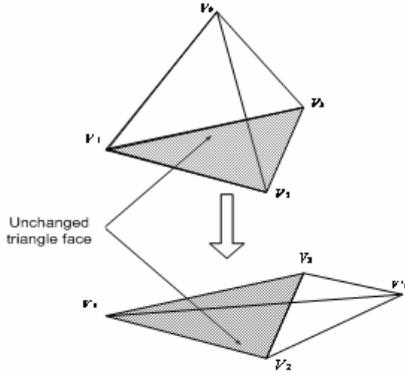
**Figure 3: The operation  $\tau$  to  $v_f$ . The candidate cell, which is in center, is a feature cell. Its feature vertex is labeled by  $v_f$ . As rule 1 describes, we collapse the cell  $\tau$  into its feature vertex  $v_f$ .**

The above tests are defined so as to ensure that there is no large change in interior boundaries of sub-regions. In some cases, the geometry of the exterior boundary should also be preserved. Therefore, we define a boundary cell check as follows:

5. **Boundary Cells:** If the boundary cell  $\tau$  has exactly one vertex on the boundary, we define the decimation operation as a collapse of  $\tau$  to  $v_b$  (the vertex on the exterior boundary). We denote the operation as  $\tau$  to  $v_b$ . If it has more than one vertex on the boundary, we do not perform the collapse operation on it.

We also define a flipping check to ensure no flipping occurs during the simplification process.

6. **Flipping Check:** For any cell inside the  $A(\tau)$ , where  $v_o$  moves to  $v'_o$ ,  $v'_o$  should stay on the same side of its facing triangle, see Fig. 4. We test all the cells in  $A(\tau)$  and check whether its signed volume has the same sign (positive or negative) before and after the collapse operation. If any cell changes the sign of its volume, we do not allow the collapse operation and return the dataset to the previous state before the collapse.



**Figure 4: The flipping problem. After the collapse operation, it is possible for some cell that the moving vertex  $v_o$  goes to the other side of the unchanged triangle face  $v_1v_2v_3$ .**

### 2.2.3 Error Prediction

We use a combination of regional and local error prediction functions to choose the cell to be collapsed next.

#### 2.2.3.1 Regional Error Prediction Function

Because each sub-region has homogenous field value distribution within it, we expect our error prediction function to reflect the modification of distribution affected by decimation operation. Therefore, we forbid any large change of the homogenous distribution. We suppose the mean value of the attributes in each homogenous sub-region will not change. The cell, whose distribution is close to the mean of the region, has more priority to be chosen as the collapse candidate.

We define the regional error prediction function as shown in Equation (5):

$$\varepsilon_r = \frac{\delta_c}{\delta_N}, \quad (5)$$

$$\delta_N = \frac{1}{N} \sum_{i=0}^N (s_i - \bar{s})^2, \quad \delta_c = \frac{1}{4} \sum_{i=0}^3 (s_i - \bar{s})^2$$

Where  $s_i$  is the attributes value of original vertices of collapse cell,  $\bar{s}$  is the mean of distribution of region  $N$ ,  $\delta_N$  is the variance of distribution of region  $N$ , and  $\delta_c$  is the variance of cell  $\tau$ .

#### 2.2.3.2 Local Error Prediction Function

A large gradient change means there is a possible feature [Cho02a, Chi03a, Kin98a, Kau93a, Sta98a]. We define the gradient error prediction function as in Equation (6):

$$\varepsilon_g = \frac{1}{4} \sum_{i=0}^3 |s_i - s'_n| \quad (6)$$

Here  $s_i$  is the attribute values of the original vertices of the collapse cell, and  $s'_n$  is the attribute value of new vertex which is created after the collapse operation.

Also, we do not want the decimation operation to change the attributes distribution of candidate's neighbor sets very much. Therefore, we define three measurements for the aspect ratio of cell as the form in Equation (7), where  $v_i$  is the original vertex of cell  $\tau$ ;  $v_n$  is the new vertex generated after collapse operation;  $a$ ,  $b$  and  $c$  represent the three edges of cell  $\tau$  which share one common vertex:

$$\varepsilon_n = \frac{\text{volume}(\tau)}{\sum_{\tau_i \in A(\tau) \cup D(\tau)} \text{volume}(\tau_i)} \quad (7)$$

$$\varepsilon_s = \frac{1}{4} \sum_{i=0}^3 |v_i - v_n|$$

$$\varepsilon_v = \frac{1}{3!} |a \cdot (b \times c)|$$

The final error prediction function is the combination of regional and local functions, shown in Equation (8):

$$\Delta\varepsilon = \omega_r \varepsilon_r + \omega_g \varepsilon_g + \omega_s \varepsilon_s + \omega_v \varepsilon_v + \omega_n \varepsilon_n \quad (8)$$

#### 2.2.4 Greedy Decimation Based Simplification

To achieve fast processing speed, we choose the greedy method to implement our algorithm.

1. Compute the predicted error based on the error prediction function shown in Equation (8) for all tetrahedral cells, and arrange them into a priority queue, ordered by predicted error.
2. While there is still at least one cell remaining in the priority queue, with the predicted error less

than the user specified tolerance, pick the first cell  $\tau$ , and do following:

- a. Delete  $\tau$  from the priority queue.
- b. Determine  $A(\tau)$  and  $D(\tau)$  of the cell  $\tau$ .
- c. Check the type of the current cell  $\tau$ : feature cell, cross-region cell, cross-region neighbor cell, boundary cell or normal cell, as described in Section (2.1.2).
- d. If the cell is a cross-region cell, or has two or more feature vertices, skip step 2e, 2f and 2g. Otherwise, perform the collapse operation based on feature preservation rules 1, 2, 3 described in Section (2.1.2).
- e. Carry out the flipping check (feature preservation rule 6) among the cells in  $A(\tau)$ . If the collapse operation causes any flipping problem, then recover the original vertices of  $\tau$  and skip steps 2f and 2g.
- f. Delete all cells in  $D(\tau)$  from the priority queue, and update cell-vertex index.
- g. Remove all cells in  $A(\tau)$  from the priority queue, and label it to forbid any further selection of collapse candidates.

3. Save the result.

### 3. EXPERIMENTAL RESULTS

We have implemented our algorithm on a Windows Platform with a 2.39GHz Intel Pentium 4 CPU and NVIDIA Quadro4900 XGL adapter with 128 Meg RAM. We use ZSweep [Far00a] as our irregular mesh rendering technique to obtain the final images. We have tested our algorithm on several medical datasets. Below we provide the results for dataset CT-head with the resolution 128x128x53, and CT-neck with the resolution 128x128x13. Results for other datasets show similar performance statistics.

| Data    | Number of cells | Number of vertices | Number of feature vertices |
|---------|-----------------|--------------------|----------------------------|
| CT-head | 4,193,540       | 868,352            | 140,233                    |
| CT-neck | 967,740         | 212,992            | 27,528                     |

Table 1 Detailed information of data sets

| Data    | $\omega_r$ | $\omega_g$ | $\omega_s$ | $\omega_v$ | $\omega_n$ |
|---------|------------|------------|------------|------------|------------|
| CT-head | 10         | 2          | 1          | 1          | 0.5        |
| CT-neck | 10         | 2          | 1          | 1          | 0.8        |

Table 2 Parameters used in simplification algorithm.

Table (1) shows the detailed information of our testing data. Table (2) shows the parameters we have used in error prediction function to perform our simplification algorithm.

Fig 5 is the comparison of our method with the simplification algorithm, TetFusion [Cho02a]. It clearly shows that for the same decimation rate, our method is better than the algorithm that uses only local error measurement. The horizontal line shows the number of cells that has been decimated. The vertical line shows the average difference of color defined, in form of Equation (9) defined on average of difference in image color per pixel, of the rendering result with the original one, where  $n$  is the number of pixel.

$$diff_1 = \frac{\sum_{i=0}^n \sqrt{(r_{i0} - r_{i1})^2 + (g_{i0} - g_{i1})^2 + (b_{i0} - b_{i1})^2}}{n} \quad (9)$$

With increasing decimation numbers, we draw two plots. The upper one is obtained by TetFusion method, which is based on local measurement, and the lower one is obtained by our method, which has a regional based measurement. As it is shown clearly, for the same number of decimation cells, the color difference of our method is less than the one of TetFusion. That shows our method maintains more volume interior information than TetFusion does.

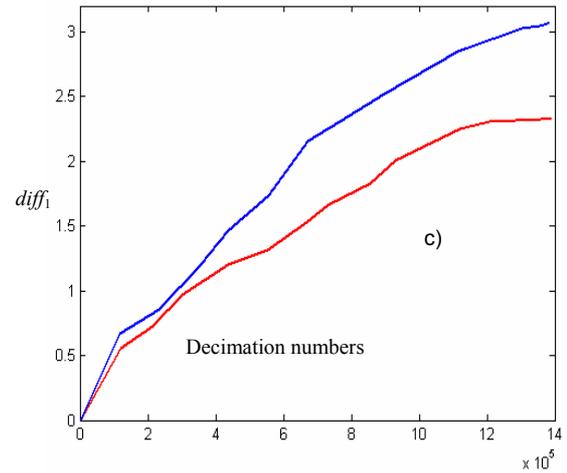
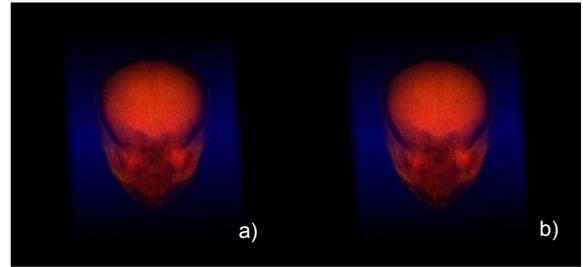
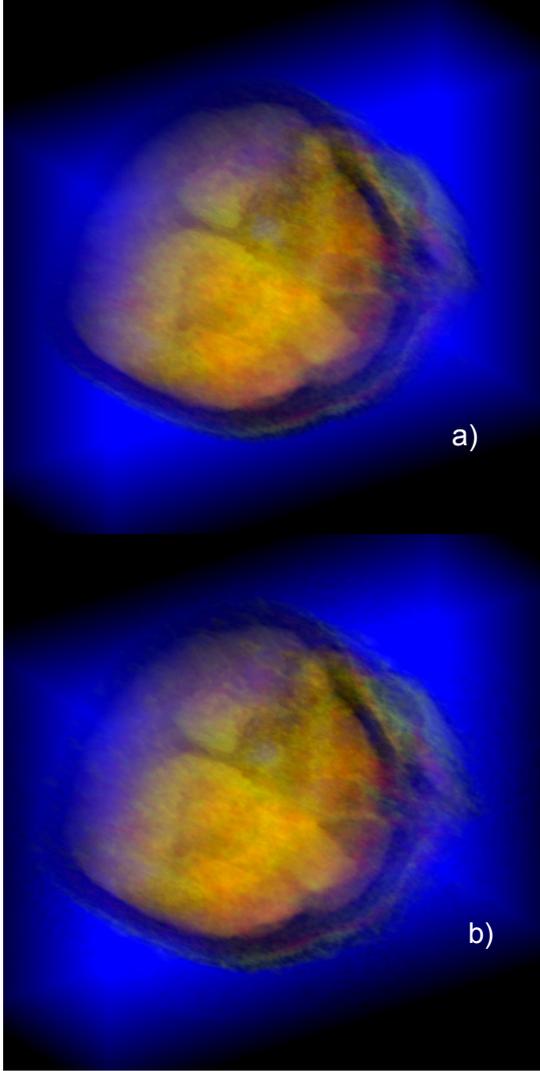


Figure 5: Demonstration that our feature preserving simplification algorithm is an improvement of previous approaches. a) is generated by our method, and b) is generated by TetFusion [Cho02a]. c) demonstrates the average color comparison as the form in Equation (9). The upper plot shows the  $diff_1$  of TetFusion, and the lower one shows our method.

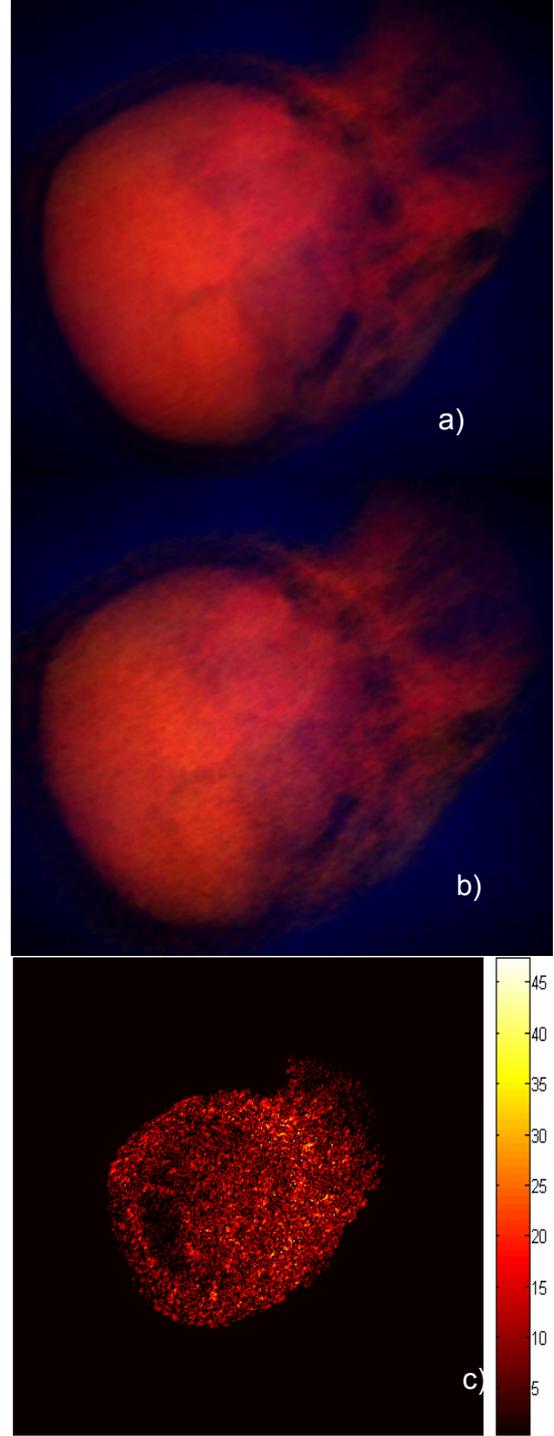


**Figure 6: The rendering result of CT head data. a) The original image; b) The 50% simplified result.**

Fig. 6 shows that after applying our simplification algorithm, the resultant dataset contains half size of original one. However, the direct rendering result is very promising. The half size dataset maintains nearly the same information of the original one. We can clearly identify the structure of original data.

Fig. 7 shows another experimental result with a numerical comparison between the original image with simplified one. The value, shown in Fig. 7-(c), demonstrates the total error ( $diff_2$ ) per pixel between Fig. 7-(a) and (b) as defined by Equation (10), which is defined on difference in image color per pixel. The color bar shows the range of the difference.

$$diff_2 = \sqrt{(r_{i0} - r_{i1})^2 + (g_{i0} - g_{i1})^2 + (b_{i0} - b_{i1})^2} \quad (10)$$



**Figure 7: Demonstration of the simplification results on CT-Head data. a) is the image rendered by original data. b) is the image rendered by 48% simplified data. c) is a comparison image by analyzing the color difference between a) and b) pixelwise. The value, shown in c), is obtained by the Equation 9. The color bar represents the range of the difference.**

## 4. CONCLUSION

In this paper, we have proposed a new simplification algorithm to reduce the large amount of redundancy of 3D medical datasets. A cutting edge technique of image processing, namely level set segmentation, is applied in a preprocess phase to simplify the data to achieve a noise robust region partition of volume data. We convert the regular grid data into a tetrahedral representation and then simplify the data while preserving both regional and local features. Our final result is a simplified tetrahedral mesh, which can be further analyzed, visualized, and animated. The experimental results amply show the advantages of this approach.

## 5. ACKNOWLEDGMENTS

We thank Dr. R. Farias, Computer Science Dept., Mississippi State University, Dr. Joseph S.B. Mitchell, Department of Applied Mathematics & Statistics, and Dr. C. T. Silva, School of Computing, University of Utah, for providing us the ZSweep code [Far00a]. We thank the maintainers of the web page [www.volvis.org](http://www.volvis.org) for the CT head, CT-neck, and other medical datasets used in our testing of the algorithm.

## 6. REFERENCES

- [Chi03a] Chiang, Y. J. and Lu, X., "Progressive Simplification of Tetrahedral Meshes Preserving all Isosurface Topologies", EG'03, conf. proc. Granada, Spain, Springer Press, pp.493-504, 2003
- [Cho02a] Chopra, P. and Meyer, J., "Tetfusion: An Algorithm for Rapid Tetrahedral Mesh Simplification", IEEE VIS'02, conf. proc. Boston, MA, USA, IEEE Computer Society, pp.133-140, 2002
- [Cig00a] Cignoni, P., Costanza, C., Montani, C., Rocchin, C. and Scopigno, R. "Simplification of Tetrahedral Meshes with Accurate Error Evaluation", in IEEE VIS'00 conf. proc. Salt Lake City, UT, USA, IEEE Computer Society, pp. 85-92, 2000
- [Far00a] Farias, R., Mitchell, J. S.B., and Silva, C. T., "Zsweep: An Efficient and Exact Projection Algorithm for Unstructured Volume Rendering", IEEE VolVis'00, symposium, Salt Lake City, UT, USA, IEEE Computer Society, pp. 91-99, 2000.
- [Gel99a] Gelder, A. V., Verma, V. and Wilhelms, J., "Volume Decimation of Irregular Tetrahedral Grids", Journal of Computer Graphics International, pp 222-230, 1999.
- [Ger00a] Gerstner, T., and Pajarola, R., "Topology Preserving and Controlled Topology Simplifying Multiresolution Isosurface Extraction", IEEE VIS'00, conf. proc. Salt Lake City, UT, USA, pp.259-266. IEEE Computer Society Press, 2000.
- [Hong03a] Hong, W. and Kaufman, A. E., "Feature Preserved Volume Simplification". ACM SMA'03, sym. proc. Karlsruhe, Germany, ACM Press, pp. 334-339. 2003.
- [Hop96a] Hoppe, H., "Progressive Meshes". ACM SIGGRAPH'96, conf. proc. New Orleans, LO, USA, pp 99-108. ACM Press, 1996.
- [Kau91a] Kaufman, A. E., "3D Volume Visualization", EG'00, conf. tutorial, Interlaken, Switzerland, Springer Press, pp175-203, 1991.
- [Kau93a] Kaufman, A. E., Cohen, D. and Yagel, R., "Volume Graphics". Journal Computer, 26(7):51-64, 1993.
- [Nie00a] Nielson, G. M., "Volume Modeling", Volume Modeling. In: M. Chen et al. (eds.), Volume Graphics, Springer Press, pp. 29-48, 2000.
- [Osh88a] Osher, S. and Sethian, J. A., "Fronts Propagating with Curvature-dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations", Journal Comput. Phys. 79(1):12-49, 1988.
- [Sam00a] Samson, C., Feraud, L. B., Aubert, G. and Zerubia, J., "A Level Set Model for Image Classification", Journal Computer Vision, Volume 40, Issue 3, pp 187-197, 2000.
- [Sta98a] Staadt, O. G. and Gross, M. H., "Progressive Tetrahedralizations", IEEE VIS'98, conf. proc., Research Triangle Park, NC, USA, IEEE Computer Society Press, pp 397-402, 1998.
- [Tro98a] Trotts, I. J., Hamann, B., Joy, K. I. and Wiley, D. F., "Simplification of Tetrahedral Meshes", IEEE VIS'98, conf. proc., Research Triangle Park, NC, USA, IEEE Computer Society Press, pp. 287-295, 1998.
- [Tro99a] Trotts, I. J., Hamann, B., Joy, K. I. and Wiley, D. F., "Simplification of Tetrahedral Meshes with Error Bounds", Journal IEEE TVCG, vol 5(3):224-237, 1999.
- [Web03a] Weber, G. H., Scheuermann, G. and Hamann, B., "Detecting Critical Regions in Scalar Fields", VisSym'03, Sym. Proc., Grenoble, France, Eurographics Association, pp. 85-94, 2003.
- [Yao00a] Yao, J. H., and Taylor, R. H., "Tetrahedral Mesh Modeling of Density Data for Anatomical Atlases and Intensity-Based Registration", MICCAI'00, conf. proc. Pittsburgh, PA, USA, Springer Press, pp. 531-540, 2000