The 16-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision

in co-operation with

EUROGRAPHICS

W S C G ' 2008

Full Papers

University of West Bohemia
Plzen
Czech Republic

February 4 - 7, 2008

Honourary Chair
Wolfgang Strasser, University of Tuebingen, Germany

Co-Chairs
Steve Cunningham, California State University, USA
Vaclav Skala, University of West Bohemia, Czech Republic

Edited by
Steve Cunningham
Vaclav Skala

WSCG'2008 Full Papers Proceedings

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

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

Printed at the University of West Bohemia

Hardcopy: *ISBN 978-80-86943-15-2*

WSCG 2008

International Programme Committee

Baranoski,G. (Canada)
Bartz,D. (Germany)
Benes,B. (United States)
Biri,V. (France)
Bouatouch,K. (France)
Coquillart,S. (France)
Csebfalvi,B. (Hungary)
Cunningham,S. (United States)
Davis,L. (United States)
Debelov,V. (Russia)
Deussen,O. (Germany)
Ferguson,S. (United Kingdom)
Goebel,M. (Germany)
Groeller,E. (Austria)
Chen,M. (United Kingdom)
Chrysanthou,Y. (Cyprus)
Jansen,F. (The Netherlands)
Jorge,J. (Portugal)
Klosowski,J. (United States)
Lee,T.-Y. (Taiwan)
Magnor,M. (Germany)
Myszkowski,K. (Germany)
Pasko,A. (United Kingdom)
Peroche,B. (France)
Puppo,E. (Italy)
Purgathofer,W. (Austria)
Rokita,P. (Poland)
Rosenhahn,B. (Germany)
Rossignac,J. (United States)
Rudomin,I. (Mexico)
Sbert,M. (Spain)
Shamir,A. (Israel)
Schumann,H. (Germany)
Skala,V. (Czech Republic)
Teschner,M. (Germany)
Theoharis,T. (Greece)
Triantafyllidis,G. (Greece)
Veltkamp,R. (Netherlands)
Wu,S. (Brazil)
Zara,J. (Czech Republic)
Zemcik,P. (Czech Republic)

WSCG 2008

Board of Reviewers

Adzhiev,V. (United Kingdom)
Albuquerque,G. (Germany)
Andreadis,I. (Greece)
Andujar,C. (Spain)
Aspragathos,N. (Greece)
Aveneau,L. (France)
Bair,A. (United States)
Bamidis,P. (Greece)
Baranoski,G. (Canada)
Battiato,S. (Italy)
Bellon,O. (Brazil)
Benes,B. (United States)
Bernhard,M. (Austria)
Biri,V. (France)
Bittner,J. (Czech Republic)
Borchani,M. (France)
Bottino,A. (Italy)
Bouatouch,K. (France)
Bouville,C. (France)
Bruni,V. (Italy)
Buehler,K. (Austria)
Carmen Juan-Lizandra,M. (Spain)
Casciola,G. (Italy)
Coleman,S. (United Kingdom)
Comba,J. (Brazil)
Coquillart,S. (France)
Csebfalvi,B. (Hungary)
Daniel,M. (France)
Davis,L. (United States)
de Geus,K. (Brazil)
Debelov,V. (Russia)
Delrieux,C. (Argentina)
Deussen,O. (Germany)
Dingliana,J. (Ireland)
Doleisch,H. (Austria)
Duce,D. (United Kingdom)
Durikovic,R. (Slovakia)
Egges,A. (Netherlands)
Eisemann,M. (Germany)
Erbacher,R. (United States)
Feito,F. (Spain)
Felkel,P. (Czech Republic)
Ferguson,S. (United Kingdom)
Fernandes Marcos,A. (Portugal)
Fernandes,A. (Portugal)
Flaquer,J. (Spain)
Flórez,J. (United States)
Forest,V. (France)
Franke,T. (Germany)
Fuchs,R. (Austria)
Galin,E. (France)
Gallo,G. (Italy)
Galo,M. (Brazil)
Galvez,A. (Spain)
Garcia-Alonso,A. (Spain)
Giannini,F. (Italy)
Gonzalez,P. (Spain)
Groeller,E. (Austria)
Gudukbay,U. (Turkey)
Gumbau,J. (Spain)
Gupta,A. (United States)
Gutierrez,D. (Spain)
Habel,R. (Austria)
Hanak,I. (Czech Republic)
Haro,A. (United States)
Hasler,N. (Germany)
Havemann,S. (Austria)
Havran,V. (Czech Republic)
Hernandez,B. (Mexico)
Horain,P. (France)
House,D. (United States)
Huang,C. (Taiwan)
Chaudhuri,D. (India)
Chen,M. (United Kingdom)
Chen,T.F. (Canada)
Chin,S. (Korea)
Chover,M. (Spain)
Chrysanthou,Y. (Cyprus)
Iglesias,A. (Spain)
Iwanowski,M. (Poland)
Janda,M. (Czech Republic)
Jansen,F. (Netherlands)
Joan-Arinyo,R. (Spain)
Jorge,J. (Portugal)
Kavan,L. (Ireland)
Keller,K. (United States)
Kimmel,B. (Canada)
Kjelldahl,L. (Sweden)
Klosowski,J. (United States)
Knight,M. (United Kingdom)
Kohout,J. (Czech Republic)
Kolcun,A. (Czech Republic)

Krishnaswamy,A. (Canada)
Lee,T. (Taiwan)
Lewis,J. (United States)
Lin,W. (Taiwan)
Lintu,A. (Germany)
Liu,S. (Taiwan)
Maciel,A. (Brazil)
Magnor,M. (Germany)
Maierhofer,S. (Austria)
Mandl,T. (Germany)
Matkovic,K. (Austria)
Mattausch,O. (Austria)
McMenemy,K. (United Kingdom)
Michoud,B. (France)
Mokhtari,M. (Canada)
Mollá Vayá,R. (Spain)
Montrucchio,B. (Italy)
Mould,D. (Canada)
Muigg,P. (Austria)
Muller,H. (Germany)
Myszkowski,K. (Germany)
Nedel,L. (Brazil)
Netto,M. (Brazil)
Odone,F. (Italy)
Ogayar,C. (Spain)
Pan,Z. (China)
Papadakis,P. (Greece)
Pasko,A. (United Kingdom)
Pasko,G. (Japan)
Patera,J. (Czech Republic)
Patil,S. (India)
Paulin,M. (France)
Pedrini,H. (Brazil)
Pelikan,J. (Czech Republic)
Peroche,B. (France)
Pili,P. (Italy)
Platis,N. (Greece)
Plemenos,D. (France)
Podgorelec,D. (Slovenia)
Pratikakis,I. (Greece)
Puig,A. (Spain)
Puppo,E. (Italy)
Purgathofer,W. (Austria)
Ramires Fernandes,A. (Portugal)
Redon,S. (France)
Reichinger,A. (Austria)
Renaud,c. (France)
Ripolles,O. (Spain)
Ritschel,T. (Germany)
Rojas-Sola,J. (Spain)
Rokita,P. (Poland)
Rosenhahn,B. (Germany)
Rossignac,J. (United States)
Rudomin,I. (Mexico)
Sanna,A. (Italy)
Sbert,M. (Spain)
Segundo,M. (0)
Segura,R. (Spain)
Sellent,A. (Germany)
Semwal,S. (United States)
Shamir,A. (Israel)
Schmaltz,C. (Germany)
Schmidt,T. (Germany)
Schneider,B. (United States)
Schumann,H. (Germany)
Singh,M. (United States)
Sips,M. (United States)
Sirakov,N. (United States)
Stanchak,R. (United States)
Stroud,I. (Switzerland)
Suarez Rivero,J. (Spain)
Svoboda,T. (Czech Republic)
Sýkora,D. (Czech Republic)
Teschner,M. (Germany)
Theoharis,T. (Greece)
Theußl,T. (Austria)
Tokuta,A. (United States)
Torrens,F. (Spain)
Triantafyllidis,G. (Greece)
Tytkowski,K. (Poland)
Van Basten,B. (Netherlands)
Van Laerhoven,T. (Belgium)
Vanecek,P. (Czech Republic)
Vasa,L. (Czech Republic)
Veeraraghavan,A. (United States)
Veiga,L. (Portugal)
Vergeest,J. (Netherlands)
Verlinden,J. (Netherlands)
Vitulano,D. (Italy)
Wagner,S. (Germany)
Wan,T. (United Kingdom)
Weiss,G. (Germany)
Wu,S. (Brazil)
Wuethrich,C. (Germany)
You,S. (United States)
Zach,C. (Austria)
Zachmann,G. (Germany)
Zalik,B. (Slovenia)
Zambal,S. (Austria)
Zara,J. (Czech Republic)
Zemcik,P. (Czech Republic)
Zhu,Y. (United States)

FOREWORD

This book contains the abstract proceedings of **WSCG'2008**, the 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2008.

Since its starting point in 1992 with about 20 participants from the former Czechoslovakia, it has become an important event on the international Computer Graphics conference calendar.

Submissions to WSCG'2008 came from five continents, including 16 countries in Europe. From these 186 submissions, 159 papers were sent to reviewers, each to be reviewed by at least three experts from the large International Program Committee or external members of the Board of Reviewers. Of these, 47 papers have been accepted to be presented as full papers at WSCG'08, representing a 29,6 % acceptance ratio.
The best 20 of these papers have been selected to be published in a special issue of the Journal of WSCG (http://wscg.zcu.cz/jwscg).

There were many very good papers reporting on projects in progress submitted and they were accepted as Communication or poster papers as they present research ideas with high research potential.

All the papers accepted will be available through the conference Digital Library at http://wscg.zcu.cz/DL/wscg_DL.htm. The published papers are also listed on the major international scientific indexes.

WSCG'2008 features three keynote lectures:

- *The Visual Computing of Projector-Camera Systems*, Oliver Bimber, University of Weimar, Germany
- *Perception-Based Integration of Vision and Touch in Virtual Environments*, Anatole Lecuyer, INRIA, Rennes, France
- *Virtual Iraq: Initial Results from a VR Exposure Therapy Application for Combat-Related Post Traumatic Stress Disorder*, Albert Rizzo et al., University of Southern California, USA

Finally, we would like to express our thanks to all the authors who submitted technical papers to WSCG'2008, and to the reviewers whose work and dedication made it possible to put together a very exciting program of high technical quality.

The Program Co-Chairs

Steve Cunningham, California State University Stanislaus, Turlock, California, USA
Vaclav Skala, University of West Bohemia, Plzen, Czech Republic

Plzen, January 20, 2007

WSCG 2008 Full Papers

Contents

Title	Page
Rizzo,A., Pair,J., Graap,K., Rothbaum,B.O., Difede,J., Reger,G., Parsons,T., Mclay,R.N., Perlman,K.: Virtual Iraq: Initial Results from a VR Exposure Therapy Application for Combat-Related Post Traumatic Stress Disorder	i
Bimber,O.: The Visual Computing of Projector-Camera Systems	i
Lécuyer,A.: Perception-Based Integration of Vision and Touch in Virtual Environments	ii
Khan,M., Ohno,Y.: Compression of Temporal Video Data by Catmull-Rom Spline and Quadratic Bezier Curve Fitting	1
Park,A., Mukovskiy,A., Omlor,L., Giese,M.: Synthesis of Character Behavior by Dynamic Interaction of Synergies Learned from Motion Capture Data	9
Ropinski,T., Kasten,J., Hinrichs,K.: Efficient Shadows for GPU-based Volume Raycasting	17
Klimenko,S., Mestetskiy,L., Semenov,A.: Handwritten Fonts Modeling Based on Fat Lines of Variable Width	25
Schreck,T., Schuessler,M., Zeilfelder,F., Worm,K.: Butterfly Plots for Visual Analysis of Large Point Cloud Data	33
Hulusic,V., Aranha,M., Chalmers,A.: The Influence of Cross-modal Interaction on Perceived Rendering Quality Thresholds	41
Feichtinger,R., Yang,H., Jüttler,B.: Particle-based T-Spline Level Set Evolution with Range and Volume Constraints	49
Bayar,A., Sami,K.: An Optimal Way to Encode the Outlines of Variable Sized Arabic Letters in a PostScript Font	57
Schnabel,R., Wessel,R., Wahl,R., Klein,R.: Shape Recognition in 3D Point-Clouds	65
Wessel,R., Bluemel,I., Klein,R.: The Room Connectivity Graph: Shape Retrieval in the Architectural Domain	73
Schoor,W., Bollenbeck,F., Hofmann,M., Mecke,R., Seiffert,U., Preim,B.: Automatic Zoom and Pseudo Haptics to Support Semiautomatic Segmentation Tasks	81
Hernandez,J.T., Calderon,J.M.: Automatic Axis Extraction from `Tubular` Structures in CT Images - An Application in Virtual Colonoscopy	89
Lorenz,H., Doellner,J.: Dynamic Mesh Refinement on GPU using Geometry Shaders	97

Redmond,N., Dingliana,J.: A Hybrid Technique for Creating Meaningful Abstractions of Dynamic 3D Scenes in Real-Time 105

Luboschik,M., Schumann,H.: Discovering the Covered: Ghost Views in Information Visualization 113

Rianto,S., Li,L., Squelch,A. Hartley,B.M.: Fluid Dynamic Simulation for Multilayer Cutting in Virtual Environment 119

Pignatel,A., Farella,E., Brevi,F., Benini,L., Gaiani,M.: On the use of a Palm Computer for Design Review Interaction in a Virtual Room 127

Wang,H., Zhang,T., Li,P.: From Bag of Categories to Tree of Object Recognition 135

Sampaio,A.Z., Henriques,P.G.: Visual Simulation of Civil Engineering Activities: Didactic Virtual Models 143

Dodo,M., Andriamanampsisoa,F., Torguet,P., Jessel,J.P.: New Method to Optimize Force-directed Layouts of Large Graphs 151

Barandiaran,I., Cottez,Ch., Paloc,C., Grana,M.: Comparative Evaluation of Random Forest and Ferns Classifiers for Real-Time Feature Matching 159

Bayer,A.O., Sevinc,A.M., Can,T.: Human Skeletal and Muscle Deformation Animation Using Motion Capture Data 167

Lucena,M.J., Fuertes,J.M., Pérez,N.: Spatially Constrained Model for Mean Shift 175

Blasco,J., Escriva,M., Abad,F., Quiros,R., Camahort,E., Vivo,R.: A Generalized Light-Field API and Management System 183

Wolfe,T., Semwal,S.K.: Acousting Modeling of Reverberation using Smooth Particle Hydrodynamics 191

Langerak,T.R.: Geometric Feature Deletion Through Freeform Feature Recognition 199

Dumazet,S., Callet,P., Genty,A.: Spectral Modeling and Virtual Restoration on a Polychrome Medieval Sculpture 207

Virtual Iraq: Initial Results from a VR Exposure Therapy Application for Combat-Related Post Traumatic Stress Disorder

Rizzo,A., Pair,J., Graap,K., Rothbaum,B.O., Difede,J., Reger,G., Parsons,T., Mclay,R.N., Perlman,K.

Abstract

The 'Virtual Iraq' VR environment is designed to be an immersive tool foruse as an Exposure Therapy treatment tool for combat related PTSD. The application consists of a series of virtual scenarios designed to represent relevant contexts for VR exposure therapy, including city and desert road environments. In addition to the visual stimuli presented in the VR HMD, directional 3D audio, vibrotactile and olfactory stimuli of relevance can be delivered. Stimulus presentation is controlled by the clinician via a separate 'wizard of oz'; interface, with the clinician in full audio contact with the patient. The presentation at the conference will detail the results of our research and clinical treatment protocols as they stand at that time. Presently, an open clinical trial to evaluate our system's efficacy for PTSD treatment with military personnel is being conducted at the Naval Medical Center San Diego and at Ft. Lewis, Washington, and a randomized controlled trial comparing VR alone and VRcycloserine is in progress at Emory University. Ten other test sites are now on line between now and the conference addressing a variety of research questions involving assessment of PTSD, physiological markers of the disorder, impact of multiple trauma events, and an fMRI study. Thus far, eight male and female treatment completers (out of 11) at two of the treatment sites have shown clinically significant improvements at post-treatment, with these patients now no longer meeting PTSD criteria. Due to the challenges for treatment of this disorder, we are encouraged by these early results.

The Visual Computing of Projector-Camera Systems

Bimber,O.

Abstract

Their increasing capabilities and declining cost make video projectors widespread and established presentation tools. Being able to generate images that are larger than the actual display device virtually anywhere is an interesting feature for many applications that cannot be provided by desktop screens. Several research groups discover this potential by applying projectors in unconventional ways to develop new and innovative information displays that go beyond simple screen presentations. Todays projectors are able to modulate the displayed images spatially and temporally. Synchronized camera feedback is analyzed to support a real-time image correction that enables projections on complex everyday surfaces that are not bound to projector-optimized canvases or dedicated screen configurations. In this talk I will give an overview over our projector-camera-based image correction techniques for geometric warping, radiometric compensation, reduction of global illumination (such as inter-reflections) or view-dependent effects (such as specular reflections), increasing focal depth, and embedding imperceptible codes with a single or with multiple projection units. Thereby, GPU-based real-time rendering and computer vision on graphics hardware are tightly coupled. Such techniques have proved to be useful tools for many real-world applications. Examples include ad-hoc stereoscopic VR/AR visualizations within everyday environments, quality improvements for (semi-)immersive VR projection displays, on-site architectural simulations, augmentations of museum artifacts, video installations in cultural heritage sites, projections onto stage settings during live performances, presentations using mobile (pocket) projectors, outdoor advertisement displays, digital illumination and projections in modern television studios, computer games, and more.

Perception-Based Integration of Vision and Touch in Virtual Environments

Lécuyer,A.

Abstract

The technological progresses that were achieved concerning Virtual Reality displays for the eyes of the user (head mounted displays, large projection screens, workbenches, CAVE, etc) as well as for his/her hands (force-feedback arms or joysticks, exoskeleton gloves, tactile matrices, etc) allow today to simulate visual and haptic sensations that are very close from sensations generated by the visualization or interaction with real objects, in real situations. However, even though the realism of each one of these stimuli taken separately improves regularly and spectacularly, the coupling and the combination of multiple stimulations on different sensory channels raise numerous novel and complex questions for virtual reality applications. For instance, how to share and spread the sensory information of the virtual world on the multiple sensory channels at disposal? How to synchronize the separated sources of sensory information spatially and temporally? Are there any preferential combinations of sensory information? More generally, how do humans perceive and integrate information provided by different sensory channels? What are the perceptual and cognitive mechanisms that are involved? Can we take advantage from them when designing multi-sensory virtual environments? In addition to these numerous questions, a classical perceptual phenomenon can also strike our attention: the sensory illusions. Sensory illusions illustrate the plasticity of our brain and reveal its surprising and regular mistakes of interpretation of sensory information. The question of sensory illusions is a central question in virtual reality-which can be seen as a vast operation aiming at generating the illusion of another reality, of another tangible universe.

During the past decade, we have developed and proposed a novel concept of haptic feedback in virtual reality called 'pseudo-haptic feedback'. Pseudo-haptic feedback is meant to simulate haptic sensations in virtual environments using properties of the human perception. Pseudo-haptic feedback uses visual feedback and borders on sensory illusion. It has been already used to simulate various haptic properties such as: stiffness or mass of virtual objects, texture of images, etc. It was also successfully implemented in various applications, such as within a virtual environment for the vocational training to milling machines, or a medical simulator for the training to regional anaesthesia procedures.

In this talk, we will first depict related work in the field of human perception, integration of vision and touch, and haptic illusions. Then we will report on experiments and on haptic properties that have been simulated to date using pseudo-haptic feedback. We will assess the potential applications of pseudo-haptic feedback in virtual reality and we will draw lessons from our past experience in implementing and evaluating visuo-haptic virtual environments.

Compression of Temporal Video Data by Catmull-Rom Spline and Quadratic Bézier Curve Fitting

Murtaza Khan
Graduate School of Science & Technology
Keio University, Japan
murtaza@on.ics.keio.ac.jp

Yoshio Ohno
Faculty of Science & Technology
Keio University, Japan
ohno@on.ics.keio.ac.jp

ABSTRACT

This paper presents a new method for lossy compression of temporal data of both naturally recorded and synthetically created videos by Catmull-Rom spline and quadratic Bézier curve fitting. The proposed method approximates the luminance or color variations in a sequence of frames by spline fitting in Euclidean space. Precise control of accuracy at pixel level is achieved by a specified tolerance of error. A break and fit criterion is employed to minimize the number of curve segments required to fit the data. Experimental results show that the described method yields very good results, both in terms of objective and subjective quality measurement, i.e., bit-rate/PSNR and human visual acceptance, without causing any blocking artifacts.

Keywords: Video data, sequence of images, approximation, compression, fitting, Catmull-Rom spline, Bézier curve.

1 INTRODUCTION

Digital video data consists of sequence of frames (images). Each frame consists of rectangular 2-D array of pixels. 3-D *RGB* or 1-D *luminance* values in a sequence of frames are associated with each pixel. *RGB* or *luminance* value(s) of a pixel can be considered as a point in Euclidean space R^3 or R^1 respectively. Let a video consists of a sequence of n frames. Frame width and height are W and H respectively. Then for each spatial location (x_i, y_j), $1 \leq i \leq W$, $1 \leq j \leq H$, we have temporal video data in n frames, $\{p_1, p_2, \ldots, p_n\}$, i.e., $p_j = (R_j, G_j, B_j)$ for *RGB* or $p_j = I_j$ for *luminance*, where $1 \leq j \leq n$. Figure 1 shows *RGB* variation of a pixel in 80 frames of a video. Video data contains temporal and spatial correlation. In our proposed method focus is on temporal compression of video data by approximating it using quadratic Bézier curve and Catmull-Rom spline fitting.

Splines and curves are widely used in computer-aided design and computer graphics because of the simplicity of their construction, accuracy of evaluation, and their capability to approximate complex shapes through curve fitting and interactive curve design [BBB95]. Spline can compress the data by approximating large number of data points with far less number of control points. Control points can be encoded by some appropriate encoding technique. During the decoding process approximated data points are generated by spline interpolation of control points.

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'2008, February 4 – February 7, 2008
Plzen, Czech Republic .
Copyright UNION Agency – Science Press

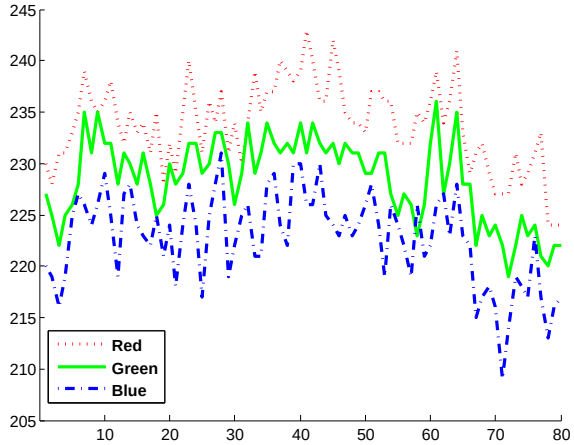


Figure 1: *RGB* temporal variation of a pixel in 80 frames of a video.

In our method, we considered temporal variations of color or luminance values of pixels in a sequence of frames as input data and approximated it with far less number of control points (output data). An important factor in spline approximation of data is finding least number of control points. We achieved this goal by selecting optimal set of control points.

Approximation and compression of data using parametric curves particularly cubic splines are explored by many authors [LCT07, UM00, IO93] et al. A method of dynamic mesh compression of animated sequences is described by [LV07]. The approach used by [LV07] is based on EdgeBreaker and Principal Component Analysis (PCA) and it exploits both spatial and temporal coherence. [LCT07] presented a medical image compression algorithm using cubic spline. Contour data compression method using Curvature Scale Space technique and Hermite curves is proposed by [UM00]. Proposed method of [IO93] uses cubic Bézier curves and is suitable for compressed representation of outline of

 ISBN 978-86943-15-2

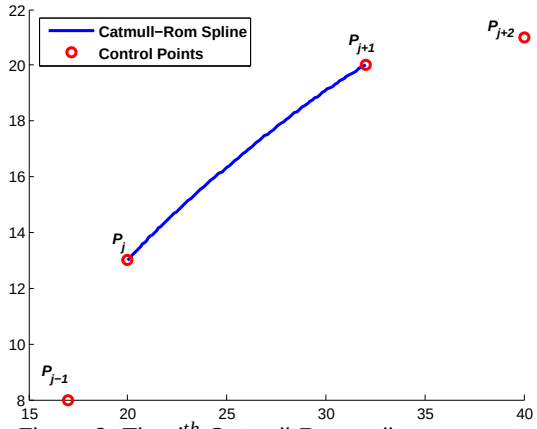


Figure 2: The jth Catmull-Rom spline segment.

fonts. Our method uses quadratic Bézier curve and Catmull-Rom spline that are computationally more efficient than cubic splines. Cubic splines are more appropriate for image compression but less feasible for video data compression. Non-spline based temporal correlation reductions methods are based on motion estimation via translating block matching algorithms (BMAs) [CP02, SD02, KIH+81]. In a typical BMA, a frame is divided into rectangular blocks of pixels. Then the current (predicted) block is matched against blocks in the previous frame, for a maximum motion displacement of w pixels. The best match on the basis of a *mean absolute error* (*MAE*) criterion yields displacement relative to current block called motion vector. Predicted frame is approximated by blocks in reference frame and corresponding motion vectors [Gha03, Thy05, Say05]. In contrast to BMAs, we do not find matching pixel or matching block. We adopts different approach of fitting i.e., approximating the change in color or luminance values of each pixel in a sequence of frames, at the fixed spatial location (without translation of block/pixel), by quadratic Bézier curve and Catmull-Rom spline fitting. BMA works at block level and may cause blocking artifacts [WOZ01, IM00]. Due to pixel level fitting by proposed method, precise control of accuracy and immunity from blocking artifacts is achieved. Due to large size of video data it is also desirable that fitting process is automated. In our proposed scheme, the user has just to initialize a few parameters, then the rest of the steps i.e., fitting, encoding/decoding are fully automated.

2 CATMULL-ROM SPLINE AND QUADRATIC BÉZIER CURVE

The following two subsections i.e., 2.1 and 2.2 describe the theory and mathematical models of Catmull-Rom spline and quadratic Bézier curve, respectively.

2.1 Catmull-Rom Splines (CRS)

Catmull-Rom spline (CRS) [CR74] is a piecewise C^1 continuous curve with specified endpoint tangents. A

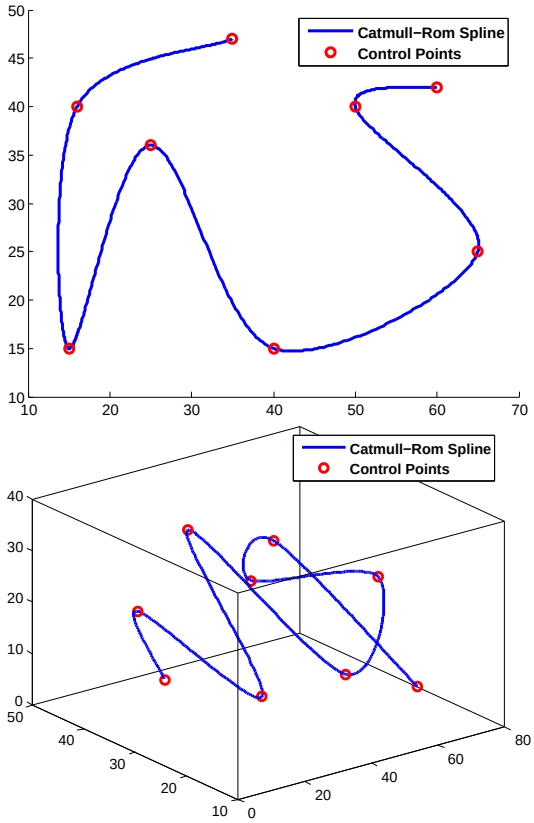


Figure 3: Multi-segment Catmull-Rom splines in 2-D and 3-D space.

CRS segment is defined by four control points, i.e., P_{j-1}, P_j, P_{j+1} and P_{j+2}, as shown in Figure 2. The jth segment of CRS interpolates between two *middle control points*, i.e., P_j and P_{j+1}. The *end control points*, i.e., P_{j-1} and P_{j+2} are used to calculate the tangents of P_j and P_{j+1}. Equations for boundary conditions of jth segment are written as:

$$P'_j = \frac{1}{2}\left(P_{j+1} - P_{j-1}\right), \quad (1)$$

$$P'_{j+1} = \frac{1}{2}\left(P_{j+2} - P_j\right). \quad (2)$$

For l joined segments, there are $2l$ conditions for continuity of functions and $2l$ conditions for continuity of slopes. Finally the equation of CRS for j^{th} segment is written as follows:

$$\begin{aligned} Q_j(t_i) = & \frac{1}{2}[(-t_i^3 + 2t_i^2 - t_i)P_{j-1} \\ & + [3t_i^3 - 5t_i^2 + 2]P_j \\ & + [-3t_i^3 + 4t_i^2 + t_i]P_{j+1} \\ & + (-t_i^3 - t_i^2)P_{j+2}], \end{aligned} \quad (3)$$

where t_i is parameter of interpolation , $0 \leq t_i \leq 1$. In order to generate n points between P_j and P_{j+1} inclusive, the parameter t_i is divided into $(n-1)$ intervals

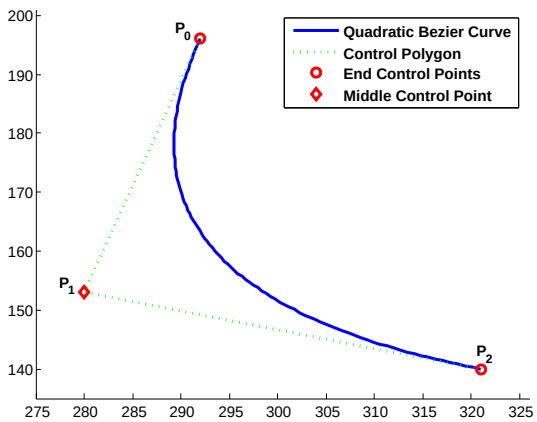


Figure 4: A quadratic Bézier curve segment.

between 0 and 1 inclusive, and $Q_j(t_i)$ is evaluated at n values of t_i. Since the CRS passes through its *middle control points*, therefore $Q_j(0) = P_j$ and $Q_j(1) = P_{j+1}$. Figure 3 shows multi-segment Catmull-Rom splines.

2.2 Quadratic Bézier Curve (QBC)

Quadratic Bézier curve (QBC) is a C^0 continuous curve. A QBC segment, is defined by three control points, i.e., P_0, P_1, and P_2, as shown in Figure 4. P_0 and P_2 are called *end control points* (*ECP*), while P_1 called a *middle control point* (*MCP*). QBC passes through its *end control points*, while a *middle control point* is used to control the shape of curve. To generate continuous quadratic Bézier curves that interpolate $k+1$ points k curve segments are used. Equation of a QBC segment can be written as follows:

$$Q(t_i) = (1-t_i)^2 P_0 + 2t_i(1-t_i)P_1 + t_i^2 P_2, \qquad (4)$$

where t_i is a parameter of interpolation , $0 \le t_i \le 1$. In order to generate n points between P_0 and P_2 inclusive, the parameter t_i is divided into $(n-1)$ intervals between 0 and 1 inclusive, and $Q(t_i)$ is evaluated at n values of t_i. Since the QBC passes through its first and last control points, therefore $Q(0) = P_0$ and $Q(1) = P_2$. Figure 5 shows multi-segment quadratic Bézier curves.

3 FITTING STRATEGY

This section describes the strategy of fitting Catmull-Rom spline and quadratic Bézier curve to video data. Fitting process is applied to temporal data of each spatial location individually. Number of spatial locations are $W \times H$, where W and H are width and height of a frame respectively. Let color or luminance value of a spatial location (x,y), $1 \le x \le W$, $1 \le y \le H$, at frame i is p_i, where $0 \le p_i \le 255$ and $1 \le i \le n$. We have to approximate the n values of each spatial location by quadratic Bézier curve and Catmull-Rom spline fitting. Now we describe the fitting process of an arbitrary spatial location (x_i, y_i). The temporal data of (x_i, y_i) in

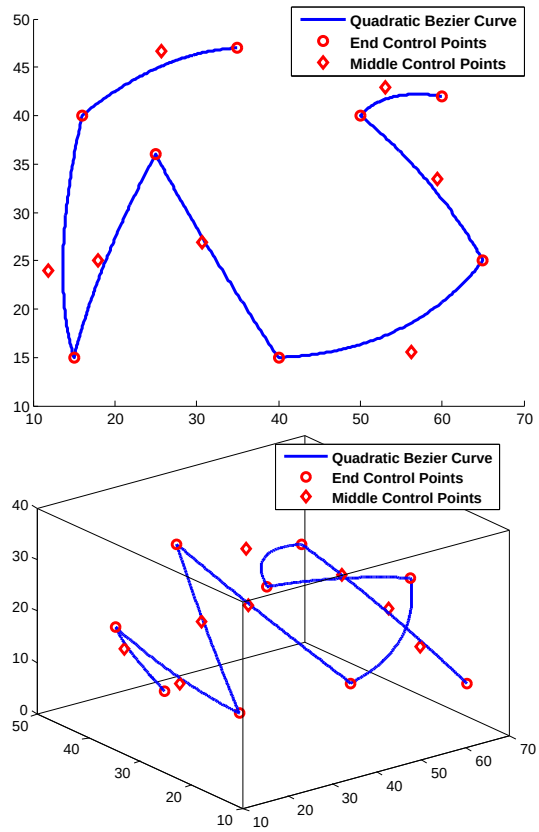


Figure 5: Multi-segment quadratic Bézier curves in 2-D and 3-D space.

n frames is $O = \{p_1, p_2, \ldots, p_n\}$. As an input to algorithm the user specifies two parameters: (1) *upper limit of error* ξ^{lmt}, i.e., maximum allowed squared distance between original and fitted data, e.g., $\xi^{lmt} = 100$, and (2) *initial breakpoint interval* δ, i.e., color or luminance value of pixel after every δ^{th} frames is taken as a breakpoint (control point), e.g., $\delta = 12$ sets the initial breakpoints as $BP = \{p_1, p_{13}, p_{25}, p_{37}, \ldots, p_n\}$ (color or luminance value of pixel in last frame is always taken as a breakpoint). The fitting process divides the data into segments based on breakpoints, i.e., $S = \{S_1, S_2, \ldots, S_{u-1}\}$. A segment is a set of all points (color or luminance values) between two consecutive breakpoints, e.g., $S_1 = \{p_1, p_2, \ldots, p_{13}\}$, $S_2 = \{p_{13}, p_{14}, \ldots, p_{25}\}$.

In addition to control points of the current segment, Catmull-Rom spline needs control points of previous and next segments. Therefore we used breakpoints of previous, current and next segments as control points of a current Catmull-Rom spline segment and obtained the approximated data of current segment using Eq. (3). An interesting question is how to obtain the breakpoints of first and last segments because they do not have previous and next segments respectively. Unlike [KB84] taking arbitrary breakpoints at both ends, we opted to take $P_{j-1} = P_j$, for the first segment and $P_{j+2} = P_{j+1}$, for the last segment. This way we are able to get the control

points of all segments automatically. For a quadratic Bézier curve breakpoints of current segment are taken as *end control points* (*ECP*), while *middle control point* (*MCP*) i.e., P_1 is obtained by least square method. Least square method gives the *best* value of *middle control point* that minimizes the squared distance between the original and the fitted data. If there are m data points in a segment, and O_i and $Q(t_i)$ are values of original and approximated points respectively then we can write the least square equation as follows:

$$U = \sum_{i=1}^{m} [O_i - Q(t_i)]^2. \tag{5}$$

Substituting the value of $Q(t_i)$ from Eq. (4) in Eq. (5) yields:

$$U = \sum_{i=1}^{m} [p_i - (1-t_i)^2 P_0 + 2t_i(1-t_i)P_1 + t_i^2 P_2]^2. \tag{6}$$

To find value of P_1 differentiating Eq. (6) partially with respect to P_1 yields:

$$\frac{\partial U}{\partial P_1} = 0. \tag{7}$$

Solving Eq. (7) for P_1 gives:

$$P_1 = \frac{\sum_{i=1}^{m} \left[p_i - (1-t_i)^2 P_0 - t_i^2 P_2\right]}{\sum_{i=1}^{n} 2t_i(1-t_i)}. \tag{8}$$

Once all three control points are in hand then approximated data of current Bézier segment is obtained using Eq. (4).

Same procedure is repeated for each segment for both Catmull-Rom spline and quadratic Bézier curve. This yields n values of approximated data, $Q = \{q_1, q_2, \ldots, q_n\}$. Then we compute error of fitting, i.e., squared distance of each point between the original data and the approximated data, $d_i^2 = |p_i - q_i|^2, \quad 0 \le i \le n$. From the maximum d_i^2 we compute maximum squared distance, $\xi^{max} = Max\left(d_1^2, d_2^2, \ldots, d_n^2\right)$. If maximum squared distance of any j^{th} segment is greater than ξ^{lmt} then this segment is splitted (replaced) with two new segments, j_1^{th} and j_2^{th}, i.e., $S = \left[\{S\} - \{S_j\}\right] \bigcup \{S_{j_1}, S_{j_2}\}$. A new breakpoint bp_{new} from original data is added in the set of breakpoints where the error is maximum, i.e., $BP = \{BP\} \bigcup \{bp_{new}\}$. For example, if segment S_1 splits at p_6 then a new breakpoint $bp_{new} = p_6$ is inserted between breakpoints p_1 and p_{13} ($BP = \left\{ p_1, \overbrace{p_6}^{bp_{new}}, p_{13}, p_{25}, p_{37}, \ldots, p_n \right\}$) and two new segments $\{p_1, p_2, \ldots, p_6\}$ and $\{p_6, p_7, \ldots, p_{13}\}$ replace S_1. The fitting process is repeated with new set of breakpoints until mean square error of each segment is equal to or less than ξ^{lmt}.

The above described fitting process is applied to color or luminance variations in temporal dimension of each spatial location separately. It is obvious that each spatial locations have different number of breakpoints (control points) selected from different frames, in other words rectangular shape of video frames is no longer intact.

Section 4 describes the proposed algorithm formally for a single spatial location. In order to have a better comparative look of fitting process between Catmull-Rom spline (CRS) and quadratic Bézier curve (QBC), we marked steps in curly braces where CRS and QBC differs in fitting. Figure 6 and Figure 7 show fitting of luminance values of in 80 frames of a video by Catmull-Rom spline and quadratic Bézier curve, respectively.

4 ALGORITHM

procedure *Fitting algorithm for luminance variations of single spatial location.*
spatial location (x,y), $1 \le x \le W$, $1 \le y \le H$.
Luminance of (x,y) in frame 1 to n is $\{p_1, p_2, \ldots, p_n\}$

Require: :
Max. allowed squared distance i.e., ξ^{lmt},
Breakpoints interval i.e., δ,
Points of original data: $O = \{p_1, p_2, \ldots, p_n\}$,
Breakpoints: $BP = \{p_1, p_{1+\delta}, , p_{1+2\delta}, \ldots, p_n\}$,

1: Get indices of BP: $V = \{1, 1+\delta, 1+2\delta, \ldots, n\}$= $\{v_1, v_2, \ldots, v_u\}$,
2: Divide O into segments $S = \{S_1, S_2, \ldots, S_{u-1}\}$, $S_i = \left\{p_{v_i}, \ldots, p_{v_{i+1}}\right\}$.
3: $\begin{cases} \text{Not required} & \text{CRS} \\ \text{Find } MCP \text{ of each segment} & \text{QBC} \\ MCP = \left\{P_1^{S_1}, , P_1^{S_2}, \ldots, P_1^{S_{u-1}}\right\} & \text{QBC} \end{cases}$
4: Fit a spline to each segment, i.e., Find $Q = \{q_1, q_2, \ldots, q_n\}$
5: $d_i^2 = |p_i - q_i|^2, \quad 0 \le i \le n$
6: $\xi^{max} = Max\left(d_1^2, d_2^2, \ldots, d_n^2\right)$, $\xi^{max} \in k^{th}$ frame, $k \in j^{th}$ segment
7: **while** $\xi^{max} > \xi^{lmt}$ **do**
8: $\quad bp_{new} = p_k$
9: $\quad V = \{V\} \bigcup \{k\}$
10: $\quad$ Split j^{th} segment into j_1^{th} and j_2^{th} segments
11: $\quad BP = \{BP\} \bigcup \{bp_{new}\}$
12: $\quad S = \left[\{S\} - \{S_j\}\right] \bigcup \{S_{j_1}, S_{j_2}\}$
13: $\quad \begin{cases} \text{Not required} & \text{CRS} \\ \text{Update } MCP = & \\ \{MCP\} \bigcup \left\{P_1^{j_1}, P_1^{j_2}\right\} & \text{QBC} \end{cases}$
14: $\quad$ Using updated BP fit spline/curve to: $\begin{cases} j-1,\ j_1,\ j_2,\ j+1 \text{ segments} & \text{CRS} \\ j_1,\ j_2 \text{ segments} & \text{QBC} \end{cases}$
15: $\quad$ Find d_i^2 of: $\begin{cases} j-1,\ j_1,\ j_2,\ j+1 \text{ segments} & \text{CRS} \\ j_1,\ j_2 \text{ segments} & \text{QBC} \end{cases}$

16: Update ξ^{max}=

$$\begin{cases} Max\left(\xi^{max}, d^2_{j-1}, d^2_{j_1}, d^2_{j_2}, d^2_{j+1}\right) & \text{CRS} \\ Max\left(\xi^{max}, d^2_{j_1}, d^2_{j_2}\right) & \text{QBC} \end{cases}$$

$\xi^{max} \in k^{th}$ frame, $k \in j^{th}$ segment

17: Find count of interpolating points (C) from $V = \{v_1, v_2, \ldots, v_u\}$, $u \lll n$:
$C = \{c_1, c_2, \ldots, c_{u-1}\}$, $c_i = v_{(i+1)} - v_i + 1$

18: $\begin{cases} \text{Encode } BP \text{ and } C & \text{CRS} \\ \text{Encode } BP, MCP \text{ and } C & \text{QBC} \end{cases}$

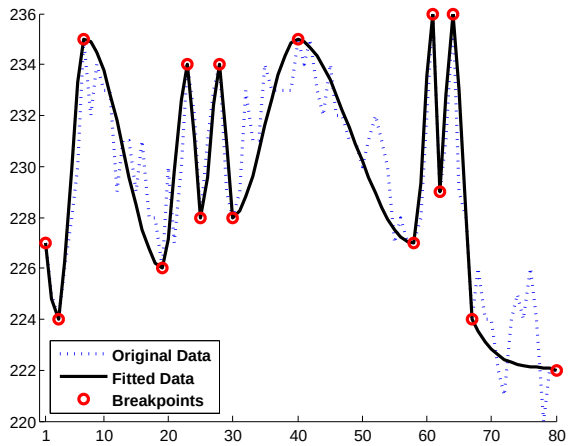


Figure 6: Catmull-Rom spline fitting to luminance values in 80 frames of a video, $\xi^{lmt} = 21$, $\delta = 79$, PSNR=43.845-dB.

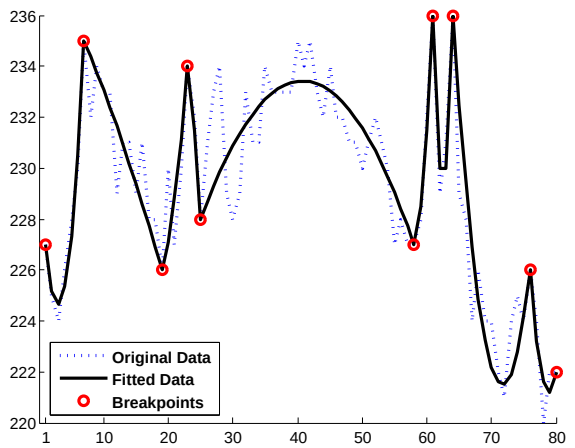


Figure 7: Quadratic Bézier curve fitting to luminance values in 80 frames of a video, $\xi^{lmt} = 21$, $\delta = 79$, PSNR=45.115-dB.

5 EXPERIMENTS AND RESULTS

We have applied the proposed algorithm on various naturally recorded and synthetically created video sequences. Output data is entropy encoded. The performance of proposed method is evaluated in terms of *bit-rate*, measured in bits per pixel (bpp) and PSNR, measured in decibel (dB). Table 1 gives the details of selected input video sequences whose results are presented in this paper. *Salesman* and *Foreman* are naturally recorded luminance videos at 8-bpp, while *Darius* is a synthetically created RGB video at 24-bpp. Table 2 compares the performance of CRS and QBC with Three Step Search method (TSS) [KIH+81], a well known and widely used block matching algorithm for temporal compression of video data. For TSS every alternative frame is predicted from previous frame (reference frame), macro block size is 8×8, and range of search window is ± 8 in both horizontal and vertical directions. Reference frames are first differentially then entropy coded along with motion-vectors (MVs). Predicted frames are not coded because they are approximated from reference frames. The fitting methods achieved better compression performance than TSS, because the fitting work of our algorithm is at pixel level and they approximate the variations of luminance with good level of accuracy. We did not compare the performance for RGB video, because block matching algorithm convert the RGB data to YC_bC_r data, and block matching is performed for only luminance data (Y), while chrominance data (C_bC_r) is sub-sampled (4:2:0), and predicted chrominance frames are obtained from MVs of luminance data. Therefore comparison of CRS and QBC with TSS for RGB video is not appropriate.

Figures 8, 9 and 10 show R-D curves of CRS and QBC for *Salesman*, *Foreman* and *Darius* sequences respectively. Note that these statistics are only for temporal compression of video data; neither spatial compression nor quantization is used. The statistics show that the proposed method yields good objective performance for naturally recorded videos (*Salesman* and *Foreman*) and very good performance for synthetically created video (*Darius*). For the *Salesman* sequence, around 35-dB PSNR is achieved at bit-rate 0.59-bpp and 0.64-bpp by CRS and QBC respectively. For the *Foreman* sequence, around 35-dB PSNR is achieved at bit-rate 1.98-bpp and 2.1-bpp by CRS and QBC respectively. *Darius* is a color sequence and bit-rate of original *Darius* sequence is 24-bpp. We achieved around 35-dB PSNR at bit-rate as low as 1.5-bpp by both CRS and QBC.

Now look at the comparative performance of CRS and QBC. For the *Salesman* sequences at low bit-rate CRS performs better but at bit-rate>0.82-bpp QBC leads CRS. For the *Foreman* sequences CRS performs better than QBC at all values of bit-rate. For the *Darius* sequence the curve is quite interesting, at bit-rate<1.6-bpp CRS performs better, 1.6-bpp<bit-rate<3.5-bpp QBC performs better, and bit-rate<3.5-bpp, again CRS takes leads over QBC.

Figures 11, 12 and 13 show 30th frames of CRS and QBC encoded videos for *Salesman*, *Foreman* and *Darius* video sequences respectively. It can be seen from these figures that subjective performance i.e., *human vi-*

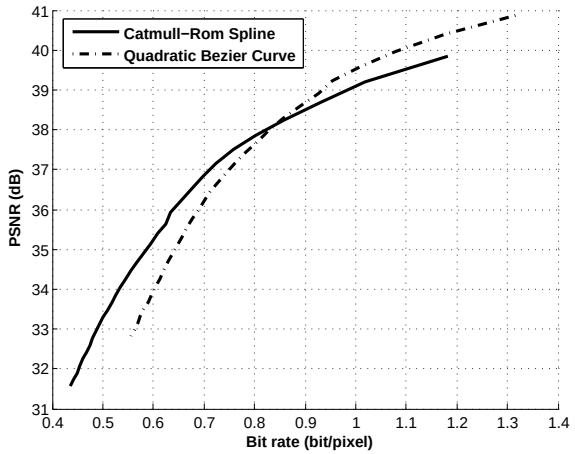


Figure 8: Rate-distortion curves of Catmull-Rom spline & Quadratic Bézier curve, *Salesman* sequence.

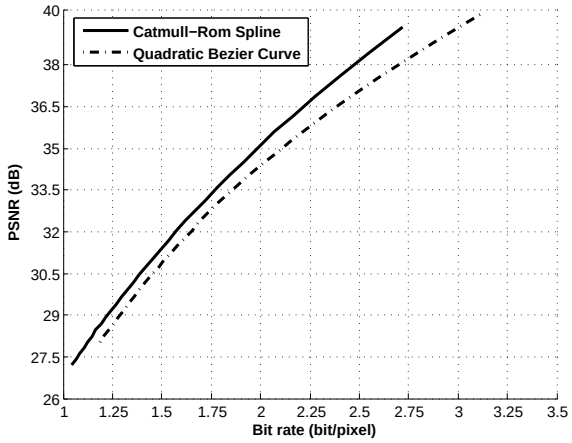


Figure 9: Rate-distortion curves of Catmull-Rom spline & Quadratic Bézier curve, *Foreman* sequence.

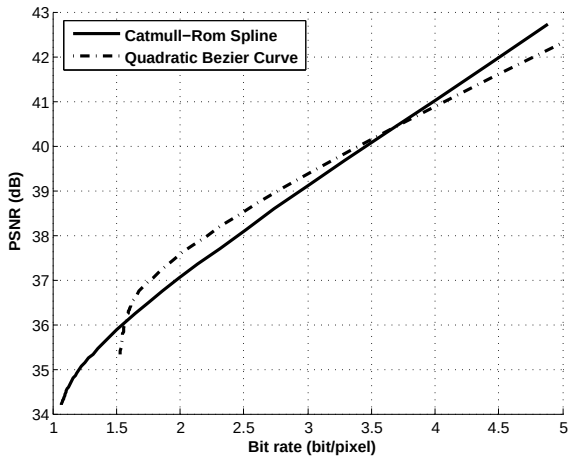


Figure 10: Rate-distortion curves of Catmull-Rom spline & Quadratic Bézier curve, *Darius* sequence.

sual acceptance of CRS and QBC approximated video frames is very good and no blocking artifacts are produced.

6 DISCUSSION

***Split of a segment*:** Lines 8 to 16 are important steps of our fitting algorithm. For the correct and efficient implementation of our algorithm it is very important to find which other segments are affected by splitting of a segment. Let us assume that an arbitrary segment j splits into two segments j_1 and j_2. For the quadratic Bézier curve we have to recompute *middle control points*, approximating points and squared distances of j_1 and j_2 segments only. But for Catmull-Rom spline the situation is not so simple. For three consecutive segments $j-1$, j and $j+1$, the last three control points of $(j-1)^{th}$ segment i.e., P_{j-1}, P_j and P_{j+1} are same to first three control point of jth segment and first control point of $(j-1)^{th}$ segment i.e P_{j-2} is not shared with j^{th} segment. Similarly the first three control points of $(j+1)^{th}$ segment i.e., P_j, P_{j+1} and P_{j+2} are same to last three control points of jth segment and last control point of $(j+1)^{th}$ segment i.e P_{j+3} is not shared with j^{th} segment. This means that a segment shares control points with its previous and next segments. Consequently adding a new breakpoint (splitting) at j^{th} segment requires to recompute approximating values and squared distances for two new segments, i.e., j_1 and j_2, obtained from splitting of jth segment, in addition to that approximating values and squared distances of previous and next segments of jth segments, i.e., $(j-1)^{th}$ and $(j+1)^{th}$ segments are also need to be recomputed. Fortunately Catmull-Rom spline saves some computation by not requiring least square solution as required by quadratic Bézier curve to find the value of its *middle control point*. Lines 3 and 13 of the fitting algorithm describe these recomputation steps for quadratic Bézier curve and Catmull-Rom spline.

Video Name	Format	Number of Frames	Bit-rate
Salesman (luminance)	CIF 352×288	45	8-bpp
Foreman (luminance)	SIF 352×288	44	8-bpp
Darius (RGB)	SIF 352×288	44	24-bpp

Table 1: Details of input video sequences.

Method	*Salesman*		*Foreman*	
Name	PSNR	Bit-rate	PSNR	Bit-rate
TSS	38.132	1.7768	35.875	2.7509
CRS	38.244	0.8574	35.589	2.0687
QBC	38.291	0.8578	35.812	2.2516

Table 2: Performance comparison of TSS [KIH+81], CRS and QBC.

***Rate Control*:** Rate of the output data is controlled by varying the value of ξ^{lmt}. Increasing the value of ξ^{lmt} decreases the bit-rate. The default value of ξ^{lmt} is 100.

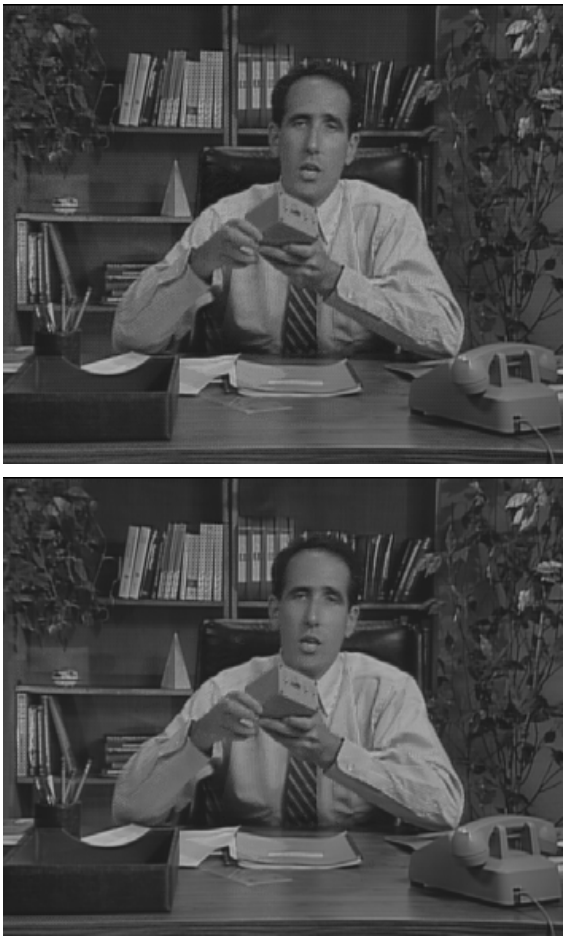


Figure 11: 30th frame of *Salesman* video sequence, $\xi^{lmt} = 100$. Top: CRS approximated frame, 38.68-dB, 0.92991-bpp. Bottom: QBC approximated frame, 39.968-dB, 1.0796-bpp.

Figure 12: 30th frame of *Foreman* video sequence, $\xi^{lmt} = 100$. Top: CRS approximated frame, 37.591-dB, 2.401-bpp. Bottom: QBC approximated frame, 38.191-dB, 2.7374-bpp.

***Output data encoding requirement*:** For CRS we need to encode (1) breakpoints (*BP*) and (2) count of interpolating points (*C*). For QBC, in addition to (1) and (2), we also need to encode *middle control points* (*MCP*). If video data is fitted using *m* segments then for CRS, $m+1$ values of *BP* and *m* values of *C* are need to be encoded. For QBC with equal number of segments, $m+1$ values of *BP*, $m+1$ values of *MCP* and *m* values of *C* are need to be encoded. Apparently CRS has less output data to be encoded. But it also depends on how much splitting of segments occurs. Usually splitting of a segment by CRS fitting is more often than splitting of a segment by QBC fitting.

***Reasons to choose quadratic Bézier curve and Catmull-Rom spline*:** (1) Both QBC and CRS are computationally efficient than other curves e.g., Natural cubic spline, B-spline, cubic Bézier curve etc. In fact we also used Natural cubic spline and cubic Bézier curve and found them less feasible than QBC and CRS. (2) Breaking of a segment into two segments keeps the computation cost within acceptable limit for both QBC and CRS.

Computational cost: QBC is computationally more efficient than CRS, because, (1) QBC is a quadratic function, while CRS is cubic a function. (2) Due to the least square fitting, QBC approximates longer segments, which means it causes lesser splitting and needs lesser computation than CRS.

***Naturally recorded vs synthetically created videos*:** The latter has less noise and more uniform distribution of luminance/color. Therefore, both CRS and QBC perform extremely well for synthetically created videos.

***Choosing between CRS and QBC*:** CRS and QBC behave quite similar, but if higher compression is desired then CRS is slightly better than QBC. If computational efficiency is of more importance then QBC is more appropriate choice.

***Limitations*:** The performance of fitting process degrades for spatial locations where changes in luminance in temporal dimension is very sharp. Because such spa-

Figure 13: 30th frame of *Darius* video sequence, $\xi^{lmt} = 100$. Top: CRS approximated frame, 41.838-dB, 4.4234-bpp. Bottom: QBC approximated frame, 42.272-dB, 4.9813-bpp.

tial locations cause lot of splitting of segments. Due to non-rectangle shape of output data, combining spatial compression with temporal compression needs special care and shape-adaptive wavelet transform can be used for spatial compression.

7 CONCLUSION

We presented a new method for compression of temporal video data by Catmull-Rom spline and quadratic Bézier curve fitting. Detail of fitting strategy and pseudo code of the algorithm are presented. We tested the proposed method using luminance and color (RGB) temporal data of naturally recorded and synthetically created video sequences. Experimental results show that the proposed method yields very good results both in terms of objective and subjective quality measurement parameters, i.e. bit-rate/PSNR and *human visual acceptance*, without causing any blocking artifacts. The method is suitable for compression of both naturally recorded and synthetically created videos such as animations and cartoons.

8 FUTURE WORK

Compression of spatial video data by spline/curve fitting is under investigation.

REFERENCES

[BBB95] Richard H. Bartels, John C. Beatty, , and Brian A. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling.* Morgan Kaufmann, 1995.

[CP02] Chun-Ho Cheung and Lai-Man Po. A novel cross-diamond search algorithm for fast block motion estimation. *IEEE Trans. Circuits And Systems For Video Technology*, 12(12):1168–1177, December 2002.

[CR74] Edwin Catmull and Raphael Rom. *A Class of Local Interpolating Splines, Computer Aided Geometric Design.* Academic Press, San Francisco, 1974.

[Gha03] Mohammed Ghanbari. *Standard Codecs: Image Compression to Advanced Video Coding.* Institution Electrical Engineers, new edition, 2003.

[IM00] Prakash Ishwar and Pierre Moulin. On spatial adaptation of motion-field smoothness in video coding. *IEEE Trans. Circuits And Systems For Video Technology*, 10:980–989, September 2000.

[IO93] Koichi Itoh and Yoshio Ohno. A curve fitting algorithm for character fonts. *Electronic Publishing*, 6(3):195–198, 1993.

[KB84] D. H. U. Kochanek and R. H. Bartels. Interpolating splines with local tension, continuity, and bias control. *Computer Graphics*, 18(3):33–41, 1984.

[KIH+81] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro. Motion compensated interframe coding for video conferencing. *Proc. Nat. Telecommun. Conf., New Orleans, LA*, pages G5.3.1–G5.3.5, December 1981.

[LCT07] Tsung-Ching Lin, Shi-Huang Chen, and Trieu-Kien Truong. Medical image compression using cubic spline interpolation with bit-plane compensation. *Proceedings of the SPIE, Medical Imaging 2007: PACS and Imaging Informatics, San Diego, California, USA*, 6516:65160D, February 2007.

[LV07] Váša Libor and Skala Václav. Coddyac: Connectivity driven dynamic mesh compression. In *3DTV Conference Proceedings*, 2007.

[Say05] Khalid Sayood. *Introduction to Data Compression.* Morgan Kaufmann, third edition, 2005.

[SD02] Somphob Soongsathitanon and Satnam S. Dlay. A new orthogonal logarithmic search algorithm for fixed block-based motion estimation for video coding. *Proceedings of Third International Symposium on Communication Systems Networks and Digital Signal Processing, Sheffield Hallam Univ. Press Learning Centre*, pages 256–256, 2002.

[Thy05] KS Thyagarajan. *Digital Image Processing with Application to Digital Cinema.* Focal Press, 2005.

[UM00] Yoke Khim Ung and Farzin Mokhtarian. Multi-scale spline-based contour data compression and reconstruction through curvature scale space. *Proceedings ICASSP '00, IEEE International Conference on Acoustics, Speech, and Signal Processing*, 6:2123–2126 vol.4, June 2000.

[WOZ01] Yao Wang, Jôrn Ostermann, and Ya-Qin Zhang. *Video Processing and Communications.* Prentice Hall, first edition, 2001.

Synthesis of Character Behaviour by Dynamic Interaction of Synergies Learned from Motion Capture Data

Aee-Ni Park[1], Albert Mukovskiy[1], Lars Omlor[1], Martin A. Giese[1,2]

[1]Lab. for Action Representation and Learning, Dept. of Cognitive Neurology,
Hertie Institute for Clinical Brain Research, Tübingen, Germany

[2]School of Psychology, University of Bangor, UK

Email: aee-ni.park@medizin.uni-tuebingen.de, martin.giese@uni-tuebingen.de

ABSTRACT

Character animation ideally combines the competing requirements of high realism and flexible automatic generation of behaviour. A method for real-time human character animation is presented, which self-organizes character behaviour with high degrees of realism by dynamic coupling of 'synergies' that are learned from motion capture data. Based on a new algorithm for blind source separation that considers time delays, highly compact generative models of body movements are learned from motion capture data. The learned components are mapped onto stable solutions of dynamical systems applying kernel methods, resulting in a coupled network of dynamic pattern generators whose state can be updated in real-time. This new framework is applied for crowd animation and the automatic generation of interactive behaviour for multiple characters.

Keywords: character animation, example-based motion synthesis, synergies, blind source separation, dynamical systems, motion blending.

1 INTRODUCTION

A core problem of modern computer animation is to accomplish high degrees of realism and flexibility at the same time. Animation based on motion capture data offers high levels of realism, but requires tedious editing of the recorded movements to adapt them to relevant constraints of the animation. On the other side, physics-based animation offers the possibility to build real-time capable animation systems that are able to self-organize character behaviour in a reactive manner, taking into account the behaviour of the user and other agents in the scene. However, such reactive systems for animation often produce simplified movements, lacking the rich details of realistic human body movement. It is an important challenge to develop systems that combine the advantages of these two approaches, exploiting motion capture data to simulate highly realistic human movements in a framework that is capable of the self-organization of interactive behaviour.

Motion Capture is presently the standard approach for the generation of highly realistic human movements. The recorded trajectories are typically retargeted to kinematical or physical models [e.g. Gle98a] and different methods for the editing and blending of motion capture data have been proposed [e.g. Wit95a; Ros96a]. Some recent algorithms concatenate segments of motion capture data automatically from large data bases, generating long animation sequences that fulfil specific boundary conditions [e.g. Gle03b; Ari03a; Saf04a]. Few recent studies have started to embed motion capture data into real-time capable animation systems [Hsu05a; Cha05a].

Another set of approaches in computer animation generates reactive behaviour of characters based on physical or dynamical models [e.g. Grz98a; Sha05a]. Many studies have focused on the simulation of scenes with many agents or crowds [e.g. Ulic02a], navigating autonomously or showing collective behaviours. The underlying character models are often strongly simplified, resulting in a manageable complexity of the system dynamics and dynamics simulation, but lacking subtle details of realistic human body movements. Some approaches have managed to simulate highly realistic human behaviour using dynamic models using sophisticated hierarchical control architectures [Sha05a; Hod95a]. However, the design of such systems is complex and the adjustment of their parameters requires much expertise of the animator. It seems thus interesting to develop simpler dynamical architectures that, however, can simulate complex human movements by integrating information learned by motion capture without the necessity of a detailed simulation of the human body dynamics.

We present a new approach that combines accurate simulation of human movements exploiting motion capture data with real-time simulation of reactive movements. By unsupervised learning, we construct highly compact trajectory models from motion capture data that depend only on a small number of hidden

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

source signals. Behaviour is self-organized by low-dimensional nonlinear dynamical systems, whose state space is mapped onto the source signals exploiting kernel methods from machine learning.

This proposed approach is biologically inspired: In motor control it has been a classical concept that complex motor behaviour might be modelled by the superposition of simpler components ('synergies'), encompassing only a subset of available degrees of freedom and forming the basic units of control [Ber67a; Fla05a]. Recent studies in motor control have successfully identified such components applying unsupervised learning methods to trajectories or EMG signals [d'Av05a; Iva04a]. Opposed to standard methods, like PCA or ICA that require often 8-12 components to accomplish highly accurate approximations of human body movements [Saf04a], the proposed novel algorithm achieves the same accuracy with less components. Each learned component is then assigned to a dynamical system (similar to a 'central pattern generator') that generates its dynamics in real time. By coupling of the dynamical systems that drive the different components coordinated behaviour between different sets of degrees of freedom is accomplished. In addition, the dynamical systems are modulated by external forces, induced either by other agents in the scene or by user specifications.

The contributions of this paper are as follows: (1) Presentation of a new algorithm for the unsupervised learning of highly compact models from sets of trajectory data that are based on synergies; (2) development of a kernel-based method for the mapping of the extracted synergies onto nonlinear dynamical systems with defined stability properties; (3) a demonstration of this novel approach for a number of example scenarios from character animation.

In the following, we briefly describe the algorithm for the unsupervised learning of synergies from motion capture data (Section 2). We then discuss how such trajectory models can be linked to nonlinear dynamical systems, suitable for the simulation of trajectories in real-time (Section 3). We demonstrate how meaningful interactions between multiple avatars can be implemented by coupling such dynamical systems, and how autonomous navigation of the characters can be realized within this framework (Section 4). Finally, we discuss applications that illustrate the capabilities of this novel framework in Section 5.

2 SYNERGY-BASED TRAJECTORY MODEL

2.1 Motion capture data

Motion data were recorded using a VICON 612 Motion Capture System with 8 cameras and 41 reflecting markers, with a sample frequency of 120 Hz. The animations presented in this paper are based on two data sets of gaits. The first contains straight walking with neutral and different emotional styles (sad, happy, angry, fearful). The second data set includes neutral and emotional straight walking and neutral walking along a circular paths (with rotations of 45 deg left or right per double step).

The recorded position trajectories were fitted with a hierarchical kinematical model (skeleton) consisting of 17 joints. Rotations between adjacent segments were parameterized as quaternions. The angles given in an axis angle representation served as basis for the modeling by unsupervised learning.

2.2 Blind Source Separation

Joint angle trajectories $x_i(t)$, after substracting the means, were approximated by a weighted mixture of source signals. As analyzed in detail in [Oml07a], a very compact model for gait trajectories is obtained by fitting an overdetermined anechoic mixture model, e.g. a model with more data points than sources. Such a model is given by the equation:

$$x_i(t) = \sum_j w_{ij} s_j (t - \tau_{ij}) \tag{1}$$

The functions $s_j(t)$ denote hidden source signals and the parameters w_{ij} are the mixing weights. Contrasting with common blind source separation techniques, e.g. PCA or ICA, this mixing model allows for time shifts τ_{ij} of the sources before linear superposition. Time shifts (delays), source signals and mixing weights are determined by the blind source separation algorithm described below. Detailed comparisons for periodic and non periodic trajectory data shows that the model (1) results in more compact trajectory approximations for human movement data, requiring less source terms than models based on standard PCA, ICA or Fourier series for the same level o accuracy [Oml06b].

Classical applications in acoustics for anechoic demixing assume frequently an under-determined mixing model of the form (1). A new algorithm for the solution of over-determined problems can be derived based on the Wigner-Ville transform [Mat03a]. The Wigner-Ville spectrum (WVS) of a random process x is defined by the partial Fourier transform of the symmetric autocorrelation function of x:

$$W_x(t,\omega) := \int E\left\{x(t+\frac{\tau}{2})\overline{x(t-\frac{\tau}{2})}\right\} e^{-2\pi i\omega\tau} d\tau \tag{2}$$

Applying this integral transform to equation (1) results in the equation

$$W_x(t,\omega) := \sum_j {w_{ij}}^2 W_{sj}(t-\tau_{ij},\omega) \tag{3}$$

under the assumption that the sources are statistically independent. As two dimensional representation of one dimensional signals, this equation is redundant and can be solved by computing a set of projections onto lower dimensional spaces that specify the same information as the original problem. See [Oml06a] for further details.

For the described data set the estimated time delays vary only significantly between the joints, but not as a function of the emotional style. We thus constrained the joint angles to small intervals around the mean angle for each joint to remove ambiguities in the estimations of the delays. Also the obtained source functions were highly similar between different emotions and movement directions. We thus estimated a common set of source functions for the whole data set. Three source functions were sufficient to obtain a highly accurate approximation of the joint trajectories (accuracy > 96 % of the variance). Since source functions and delays were joint-specific, but identical over the whole data set, all information about emotion style and walking direction was encoded in the mixing weights w_{ij}.

3 DYNAMICS FOR REAL-TIME ANIMATION

The proposed trajectory model parameterizes high-dimensional trajectories in terms of a small number of basis signals. However, this representation is not suitable for real-time animation, e.g. for computer games. To make the system real-time capable, we established a mapping between the solutions of simple dynamical systems and source signals of the trajectory representation. The complete trajectory is then generated by a set of such simple dynamical systems, which are coupled to ensure temporal coordination of the different sources. In an abstract sense, the resulting system is similar to a set of coupled 'central pattern generators' in a biological system. Such architectures have been proposed as model for the generation of gaits and other motor patterns [Ijs02a; Sch03a].

In the following, we first introduce the attractor dynamics and discuss how the mapping between its solutions and the source signals of the trajectory representation is learned. Finally, we demonstrate how dynamic couplings can be introduced that stabilize the coordination within single characters, and which are suitable for the simulation of coordinated behaviour of multiple avatars.

3.1 Attractor dynamics

Often, behaviour can be mapped onto stable solutions of dynamical systems [Kel95a]. This has been successfully demonstrated in a variety of applications in robotics [e.g. Sch95a; Sch03a]. While our approach generalizes for non-periodic movements, for the experiments in this paper we simulated only periodic gaits. It seems a natural to choose limit cycle oscillators as dynamics for the generation of such periodic behaviours [e.g. Ijs02a; Sch03a]. Because of its well-studied dynamics, we use a Van der Pol oscillator as basis element of our architecture. This oscillator can be understood as a harmonic oscillator with amplitude-dependent damping. Its dynamics is given by the differential equation:

$$\ddot{y}(t) + \lambda \left(y(t)^2 - k \right) \dot{y}(t) + \omega_0^2 \, y(t) = 0 \qquad (4)$$

The parameter ω_0 determines the eigenfrequency of the oscillator, and the positive parameter k the amplitude of the stable limit cycle with a speed that depend on the positive parameter λ. After perturbations the state will thus return to this attractor. In addition, the structure of the dynamics does not fundamentally change for moderate couplings with other system components (structural stability). For more details see [And87a]. For each of the three source signals in the model (1) we introduce a separate Van der Pol oscillator.

3.2 Mapping of phase space onto source signals

The mapping between the phase space of the limit cycle oscillator and the values of the source signals is learned by Support Vector Regression (SVR) [Vap98a]. To avoid additional delays in the implementation, we introduce new signals for the individual sources with different delays according to the relationship:

$$\tilde{s}_{ij}(t) = s_j(t - \tau_{ij}) \qquad (5)$$

For each of these modified source signals we construct a separate mapping from the phase space of the Van der Pol oscillator that corresponds to the source s_j, which is given by the variable pair $\mathbf{y}_j(t) = [y_j(t), \dot{y}_j(t)]$ and the values of the variables $\tilde{s}_{ij}(t)$. This mapping is given by the nonlinear functions:

$$\tilde{s}_{ij}(t) = f_{ij}\left(\mathbf{y}_j(t)\right) \qquad (6)$$

The nonlinear function f_{ij} is learned by SVR from M training data pairs $\left\{\mathbf{y}_j(t_l), \tilde{s}_{ij}(t_l)\right\}_{1 \le l \le M}$ that are derived by sampling one cycle of the stationary solution of the oscillator and the modified source signal equidistantly over time. After the functions f_{ij} have been learned, the dynamics corresponds to three limit cycle oscillators with nonlinear instantaneous observers that map the state of each oscillator onto the corresponding set of delayed source signals. These signals then are linearly combined according to (7). The complete reconstruction of the joint angle trajectories requires the addition of the average joint angles m_i, which have subtracted from the data before the blind source separation (Figure 1):

$$x_i(t) = m_i + \sum_j w_{ij} \tilde{s}_{ij}(t) \qquad (7)$$

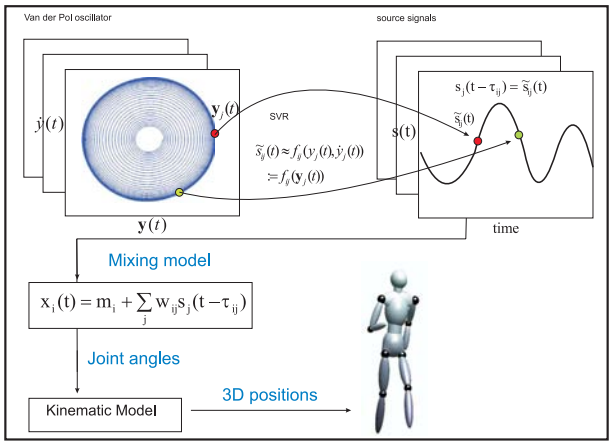


Figure 1: Illustration of the dynamics for real-time animation. The phase space of the limit cycle oscillator is mapped onto the time-shifted source signals using Support Vector Regression. Joint angles are synthesized by combining the signals linearly according to the learned mixture model (1). A kinematic model converts the joint angles into 3-D positions for animation.

3.3 Dynamic coupling

In order to stabilize the timing-relationship between the different sources ('synergies') we introduced dynamic couplings between the three oscillators that drive the source signals of the same avatar.

It has been shown applying contraction theory that stable behaviour of such oscillator networks can be accomplished by introduction velocity couplings [Wan05a]. The couplings within the same avatar were given in the form (cf. Figure 2):

$$\begin{aligned} \ddot{y}_1 + \lambda\left(y_1^2 - k\right)\dot{y}_1 + \omega_0^2\, y_1 &= \\ \alpha\left(\dot{y}_2 - \dot{y}_1\right) + \alpha\left(\dot{y}_3 - \dot{y}_1\right) \\ \ddot{y}_2 + \lambda\left(y_2^2 - k\right)\dot{y}_2 + \omega_0^2\, y_2 &= \\ \alpha\left(\dot{y}_1 - \dot{y}_2\right) + \alpha\left(\dot{y}_3 - \dot{y}_2\right) \\ \ddot{y}_3 + \lambda\left(y_3^2 - k\right)\dot{y}_3 + \omega_0^2\, y_3 &= \\ \alpha\left(\dot{y}_1 - \dot{y}_3\right) + \alpha\left(\dot{y}_2 - \dot{y}_3\right) \end{aligned} \tag{8}$$

Such dynamic couplings can also be exploited for simulating coordinated behaviour of multiple characters and crowds, for example, to enforce that multiple avatars walk in synchrony (e.g. soldiers in lock-step). To implement such couplings we only connected the oscillators assigned to the source with the lowest frequencies (Oc1). Introducing unidirectional couplings it is possible to make multiple characters following one, who is the leader [Wan05a].

3.4 Interactive change of motion style

The proposed model for the real-time generation of trajectories permits style morphing in a straight forward way. To interpolate, e.g., between neutral (style a) and

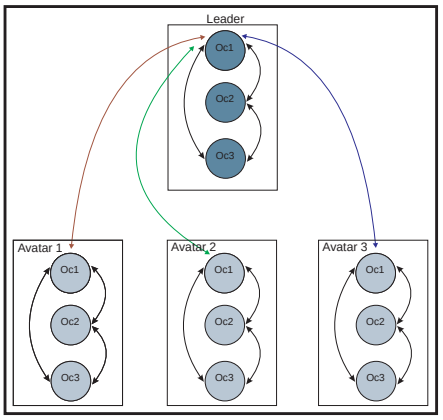


Figure 2: Coupling of multiple avatars, each of them comprising three coupled oscillators.

emotional walking (style b), the mixing weights and the mean joint angles are linearly interpolated:

$$\begin{aligned} w_{ij}(t) &= \mu(t)\, w_{ij}^a + (1-\mu(t))\, w_{ij}^b \\ m_i(t) &= \mu(t)\, m_i^a + (1-\mu(t))\, m_i^b \end{aligned} \tag{9}$$

The time-dependent morphing parameter $\mu(t)$ specifies movement style. In addition, the gait speed is adjusted by interpolating the eigenfrequencies of the oscillators:

$$\omega_0(t) = \mu(t)\, \omega_0^a + (1-\mu(t))\, \omega_0^b \tag{10}$$

The same type of morphing was also applied to implement direction changes of the avatars for navigation, morphing between straight and curved gait steps. The change of the morphing parameter can be made dependent on the behaviour of other avatars in the scene, e.g. to influence the emotional style dependent on the distance d of the avatar from another 'dangerous' colleague, by introducing a distance-dependent morphing weight. $\mu(t) = g(d(t))$. Similarly, the eigenfrequency can be made dependent on the distance from other agents. In this way, the walking speed of a 'follower' can be adjusted to that of a leader.

4 NAVIGATION

The model so far is capable of synthesizing joint angle trajectories with high degrees of realism in real time. Yet, the characters need to change position and to propagate in the simulated scene. We compute the translated motion in the horizontal plane and the rotation of the hip around the vertical axis in order to ensure the floor contact constraints of the feet. In this way, we generate the propagation of the avatars indirectly from the joint angle movements. By interpolating between straight and curved walking we simulate walking with differently curved walking paths. Addition of a navigation dynamics makes it possible to simulate avatars that move towards specific targets in space, or to avoid obstacles and other avatars.

4.1 Character propagation

The pelvis forms the root of the kinematic chain of our avatar model. The core of the algorithm for computing the avatar's propagation is to adjust the horizontal pelvis translation and rotation in order to fulfil the constraint that the feet that make contact to the ground do not translate. For this purpose, we first detect whether the feet make ground contact using a simple threshold criterion for the vertical position of the foot centers. This criterion works well when the original motion capture data did not contain foot slipping. The correction was based on the foot with the lowest vertical coordinate.

We add a differential translation and horizontal rotation to the pelvis coordinate system in order to minimize foot slipping. In addition, the vertical position of the pelvis is adjusted by assuming that the foot forms the lowest point of the figure.

4.2 Navigation dynamics

Reactive behaviour of navigating robots and also humans has been successfully modeled using nonlinear dynamical systems [Sch95a; War06a]. We applied a simple dynamic navigation model, originally developed in robotics. The present implementation does not modify the propagation speed of the characters, while this could be easily added. The heading direction of character i is modelled by the dynamic variable $\varphi_i(t)$. The change of the heading direction is determined by three components of the vector field of the heading dynamics:

$$d\varphi_i/dt = h^{\text{goal}}\left(\varphi_i, \mathbf{p}_i, \mathbf{p}_i^{\text{goal}}\right) + \sum_j h^{\text{avoid}}\left(\varphi_i, \mathbf{p}_i, \mathbf{p}_j\right) + \sum_j h^{\text{pcoll}}\left(\varphi_i, \varphi_j, \mathbf{p}_i, \mathbf{p}_j\right) \tag{11}$$

The two-dimensional vectors $\mathbf{p}_i$ and $\mathbf{p}_j$ signify the (2D) positions of the characters i and j in the scene, and the variables $\varphi_i(t)$ and $\varphi_j(t)$ their heading directions, which can be estimated from the momentary velocities. The first term in the vector field permits to define a goal position $\mathbf{p}_i^{\text{goal}}$ that the character tries to approach. It is defined by the function $h^{\text{goal}}\left(\varphi_i, \mathbf{p}_i, \mathbf{p}_i^{\text{goal}}\right) = \sin(\varphi_i^{\text{goal}} - \varphi_i)$, where φ^i_{goal} is the direction angle of the goal relative to the position $\mathbf{p}_i$ in external coordinates.

Following [Sch95a], the second term that is responsible for obstacle avoidance is defined by:

$$h^{\text{avoid}}\left(\varphi_i, \mathbf{p}_i, \mathbf{p}_j\right) = \sin\left(\Delta\varphi_{ij}\right) \cdot \exp\left(-\frac{\Delta\varphi_{ij}^2}{2\sigma_\varphi^2}\right) \cdot \exp\left(-\frac{d_{ij}^2}{2\sigma_d^2}\right) \tag{12}$$

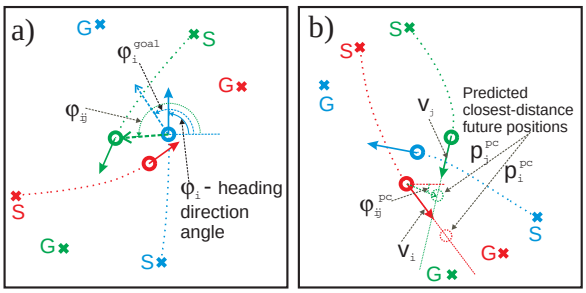


Figure 3: Avoidance scenarios illustrating the notations used in the text: (a) Angles $\varphi_i, \varphi_i^{\text{goal}}, \varphi_{ij}$ shown for one character (blue). (b) Angle φ_{ij}^{pc} shown for one character (red). Crosses mark "Start" and "Goal" positions for the characters

The quantity $\Delta\varphi_{ij} = \varphi_i - \varphi_{ij}$ is the relative direction of character j seem from character i in global coordinates. Introducing the vector $\mathbf{d}_{ij} = \mathbf{p}_j - \mathbf{p}_i$ one obtains $\varphi_{ij} = \arctan((\mathbf{d}_{ij})_y/(\mathbf{d}_{ij})_x)$ and $d_{ij} = |\mathbf{d}_{ij}|$, the 2D distance between the two characters (Figure 3 a). The values of the constants were $\sigma_\varphi = \pi/6$ and $\sigma_d = 1.5\ldots2$ m. In order to remove far field interactions, we set h^{avoid} to zero for $|\Delta\varphi_{ij}| \geq \pi/2$ and for $d_{ij} > 4$ m. The last term in (11) implements an avoidance strategy that takes into account predicted possible future collision positions. Inclusion of this term provided the most naturally looking avoidance behaviour. The implemented force depends on the future positions of the characters i and j, computed from their current positions and momentary velocities $\mathbf{v}_i$, $\mathbf{v}_j$ (Figure 3 b). We define $\Delta\varphi_{ij}^{\text{pc}} = \varphi_i - \varphi_{ij}^{\text{pc}}$ the direction to the character j at the expected moment of minimal distance. With $\mathbf{v}_{ij} = \mathbf{v}_j - \mathbf{v}_i$ and $\mathbf{d}_{ij} = \mathbf{p}_j - \mathbf{p}_i$ the characters approach each other only for ${\mathbf{v}_{ij}}^T\mathbf{d}_{ij} < 0$. In this case, the closest relative position in the future occurs after the time $\tau^{\text{pc}} = {\mathbf{v}_{ij}}^T\mathbf{d}_{ij}/({\mathbf{v}_{ij}}^T\mathbf{v}_{ij})$. At this point in time the predicted position and heading direction of character j are given by $\mathbf{p}_j^{\text{pc}} = \mathbf{p}_j + \tau^{\text{pc}}\mathbf{v}_j$ and $\varphi_{ij}^{\text{pc}} = \arctan((\mathbf{p}_j^{\text{pc}} - \mathbf{p}_i)_y/(\mathbf{p}_j^{\text{pc}} - \mathbf{p}_i)_x)$, $\mathbf{d}_{ij}^{pc} = \mathbf{p}_j^{pc} - \mathbf{p}_i$. Analogous to (12), the avoidance dynamics is

$$h^{\text{pcoll}}\left(\varphi_i, \varphi_j, \mathbf{p}_i, \mathbf{p}_j\right) = \sin\left(\Delta\varphi_{ij}^{\text{pc}}\right) \cdot \exp\left(-\frac{(\Delta\varphi_{ij}^{\text{pc}})^2}{2\sigma_\varphi^2}\right) \cdot \exp\left(-\frac{\left(d_{ij}^{\text{pc}}\right)^2}{2\sigma_d^2}\right). \tag{13}$$

where the constants σ_φ and σ_d were chosen as before, and where h^{pcoll} was set to zero for $\left|\Delta\varphi_{ij}^{\text{pc}}\right| \geq \pi/2$ and $d_{ij}^{\text{pc}} > 8$ m.

4.3 Control of walking direction

The walking direction of the characters is changed by interpolation between straight walking and walking

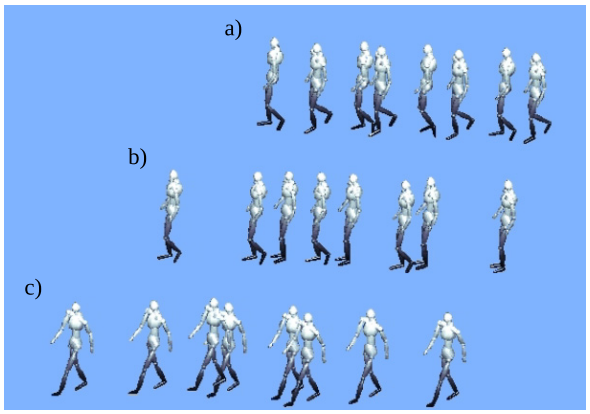


Figure 4: Synchronization of the gaits within a crowd. (a) Avatars start with self-paced walks that are out of phase. After a transitory period (b), the gaits of the characters become completely synchronized (c).

along curved paths to the left (for $d\varphi_i/dt > 0$ or to the right ($d\varphi_i/dt \leq 0$) using equation (9). The morph parameter μ was taken proportional to $|d\varphi_i/dt|$, normalizing in a way that ensures $\mu = 1$ for the maximum possible value of this derivative. The heading direction $\varphi_i(t)$ generated by the navigation dynamics was low-pass filtered with a time constant equal to one step cycle to improve the smoothness of the navigation behaviour.

5 RESULTS

The proposed technique for motion capture-based self-organized simulation of character behaviour has a broad application spectrum, which could only be partially be explored in this paper. In the following, we present a number of example applications that highlight possibilities of the approach. In general the method provide very accurate approximations of the original joint trajectories, as shown in the attached *Demos*[1].

The first example shows the self-organization of coordinated behaviour of many avatars. A crowd of characters is first locomoting individually without fixed phase relationship. An instruction to move synchronously (in lock-step) is introduced, modelled by introducing couplings between the oscillator triples of the individual avatars, taking one oscillator as a leader (Section 3.3). For appropriate coupling, the transition between uncoordinated and coordinated crowd behaviour is quite short (less than three steps) and looks quite natural. After the transition, the characters adapt walking behaviour that looks highly realistic [*Demo*[2]].

The second example is shown in Figure 5. One character is following the second at a fixed distance.

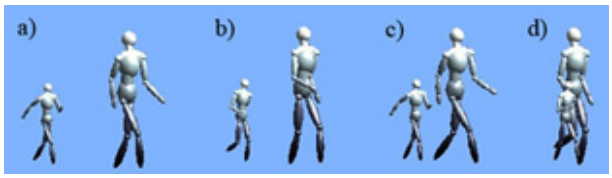


Figure 5: Following behaviour realized by two coupled avatars. (a) With the distance exceeding a certain threshold the smaller avatar accelerates (b-c) to catch up with the leader (d).

The small character has to change his eigenfrequency to catch up with the big avatar, who takes the role of the leader. This behaviour was simulated by making the eigenfrequencies of the oscillators of the follower dependent on the distance to the leader [cf. *Demo*[3]].

The third example is illustrated in Figure 6. A group of avatars that meets in the center of the scene changes their affect upon the contact with the other characters. This behaviour was implemented by making the affect of each avatar dependent on the distance from the others. In addition, the avatars avoid each other, due to the navigation dynamics described in Section 4. In this simulation, navigation and changes of emotional styles were combined, based on only three prototypical gaits: neutral walking with rotation right or left and emotional straight walking. The *Demos*[4] show examples for navigation with emotional changes from neutral to happy, neutral to sad, and sad to happy.

In order to produce the morphs between straight emotional gaits and neutral curved walking (Left and Right), we first created an intermediate balanced mixture by interpolating the mixing weights according to the relationship:

$$w_{ij} = \frac{3}{4} w_{ij}^{\text{emotional}} + \frac{1}{8}\left((1+\beta_{LR}) w_{ij}^{\text{Left}} + (1-\beta_{LR}) w_{ij}^{\text{Right}}\right) \quad (14)$$

Where the parameter β_{LR}, with $0 < \beta_{LR} < 1$, was adjusted for different emotional styles in order to balance left-right declinations from the straight line. Corresponding with Section 4, morphing was done in a piecewise linear manner dependent on the sign of the change of the heading direction.

Another simulation based on a similar implementation is shown in Figure 7. Here two groups of avatars cross each other, avoiding collision. When they meet each other their emotions switch to another affect. The simulation shows that the proposed framework integrates style morphing and autonomous navigation

[1] www.uni-tuebingen.de/uni/knv/arl/avi/compareA.avi
www.uni-tuebingen.de/uni/knv/arl/avi/compareS.avi
[2] www.uni-tuebingen.de/uni/knv/arl/avi/synchronization.avi
[3] www.uni-tuebingen.de/uni/knv/arl/avi/following.avi
[4] www.uni-tuebingen.de/uni/knv/arl/avi/avoidance3NH.avi
www.uni-tuebingen.de/uni/knv/arl/avi/avoidance3NS.avi
www.uni-tuebingen.de/uni/knv/arl/avi/avoidance3SH.avi

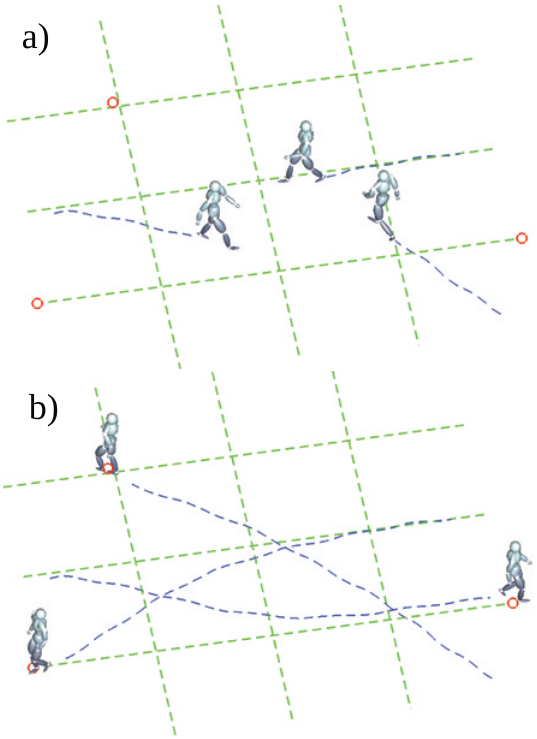


Figure 6: Avoidance behaviour and change of emotional style. a) Avatars starting from different positions with a sad emotion are heading towards their goals (red circles). b) At the meeting point the emotional styles change to happy. In addition, the characters avoid each other.

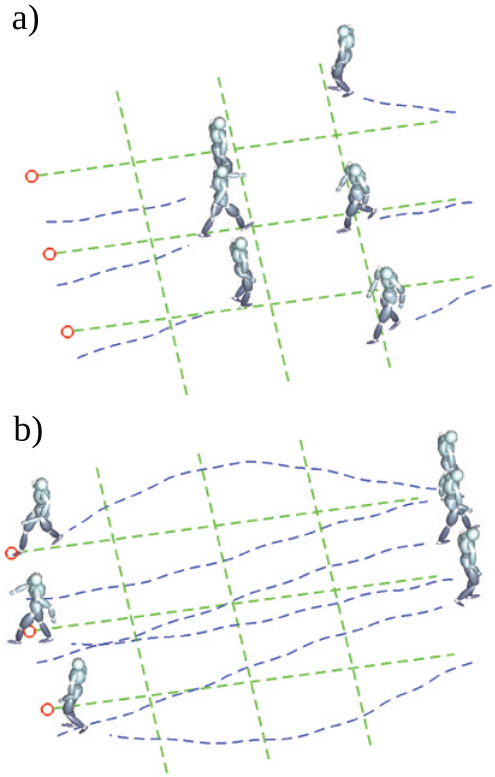


Figure 7: Avoidance behaviour and change of emotional style. Three avatars, starting from the left side, change their emotion from happy to sad while proceeding to their goals. A second group of avatars starting from the right side change their emotions from sad to happy while avoiding the opposing group.

of characters. Few examples are demonstrated *Demos*[5], including transition between neutral and emotional gaits and different emotional gaits.

6 CONCLUSION

We have presented a new framework that links the synthesis of highly realistic human movements based on motion capture data with the self-organization of behaviour using nonlinear dynamical systems. The proposed method exploits biologically inspired concepts and is based on the learning of highly compact models for human movement trajectories by a superposition of learned 'synergies', which are controlled by nonlinear attractors dynamics. By introducing of appropriate dynamic couplings complex realistically looking behaviour, reproducing the fine structure of human movements, could be synthesized.

The present paper has only given a fist set of demonstrations, and substantial additional work will be needed to explore the limits of this approach. Specifically, it needs to be demonstrated and quantified what the benefits of a synergy-based approach over other classical approaches are. The present implementation has not been optimized for computation speed and efficiency. While we tested simulations up to 20 characters, it remains to be explored how the computational limits of the proposed methods are for an appropriately chosen optimized hardware in comparison with other approaches.

Future work will extend this approach for non-periodic movements and will exploit more the synergy concept, trying to learn components that encompass only limited sets of degrees of freedom, potentially resulting in more flexible control of motion styles. An additional very interesting application field for such methods is facial animation. Here, synergies might correspond to sets of statistically optimized 'facial action units' [Ekm78a] that are adapted to the relevant animation data. In addition, such extracted statistical basis units could be compared to the classical concept of facial action units in psychology and to EMG recordings providing signatures of real physiological synergies of facial actions.

[5] www.uni-tuebingen.de/uni/knv/arl/avi/avoid6.avi
www.uni-tuebingen.de/uni/knv/arl/avi/avoid6N.avi
www.uni-tuebingen.de/uni/knv/arl/avi/avoid6NHS.avi

ACKNOWLEDGEMENTS

Supported by DFG Forschergruppe 'Perceptual Graphics', Volkswagenstiftung, HFSP and EC FP6 project COBOL. Additional support was provided by the Hermann and Lilly-Schilling-Stiftung. We are grateful to T. Flash and J.J. Slotine for many interesting discussions and W. Strasser for his support.

REFERENCES

[And87a] Andronov, A.A., Vitt, A.A. and Khaikin, S.E., Theory of oscillators, Dover Publ. Inc., New York, 1987.

[Ari03a] Arikan, O., Forsyth, D.A. and O'Brien, J.F., Motion synthesis from annotations, ACM Trans. on Graph. SIGGRAPH, Vol.22, No.3, pp. 402-408, 2003.

[Ber67a] Bernstein, N.A., The coordination and regulation of movements, Pergamon Press, Oxford, 1967.

[Cha05a] Chai, J. and Hodgins, J.K., Performance Animation from Low-dimensional Control Signals, ACM Trans. on Graph. SIGGRAPH, Vol.24, No.3, pp. 686-696, 2005.

[d'Av05a] d'Avella, A. and Bizzi, E., Shared and specific Muscle synergies in neural motor behaviors, Proc. Nat. Acad. Sci. USA, Vol.102, No.8, pp. 3076-3081, 2005.

[Ekm78a] Ekman, P. and Friesen, W., Facial Action Coding System: A Technique for the Measurement of Facial Movement, Consulting Psychologists Press, Palo Alto, 1978.

[Fla05a] Flash, T. and Hochner, B., Motor primitives in vertebrates and invertebrates, Curr. Opin. Neurobiol., Vol.15, No.6, pp. 660-666, 2005.

[Gle98a] Gleicher, M., Retargetting Motion to New Characters, Proc. ACM SIGGRAPH, Int. Conf. on Comp. Graph. and Interactive Techniques, pp. 33-42, 1998.

[Gle03b] Gleicher, M., Shin, H. J., Kovar, L. and Jepsen, A., Snap-together motion: Assembling runtime animation, ACM Trans. on Graph. SIGGRAPH, Vol.22, No.3, pp. 702-702, 2003.

[Grz98a] Grzeszczuk, R., Terzopoulos, D. and Hinton, G., NeuroAnimator: Fast neural network emulation and control of physics based models, Proc. ACM SIGGRAPH, Int. Conf. on Comp. Graph. and Interactive Techniques, pp. 9-20, 1998.

[Hod95a] Hodgins, J.K., Wooten, W.L., Brogan, D.C. and O'Brien, J.F., Animating human athletics, Proc. ACM SIGGRAPH, pp. 71-78, 1995.

[Hsu05a] Hsu, E., Pulli, K. and Popović, J., Style Translation for Human Motion, ACM Trans. on Graph. SIGGRAPH, Vol.24, No.3, pp. 1082-1089, 2005.

[Ijs02a] Ijspeert, A., Nakanishi, J. and Schaal, S., Learning Attractor Landscapes for Learning Motor Primitives, In S. Becker et al eds., Adv. in Neural Inf. Process. Systems, Vol.15, pp. 1547-1554, 2002.

[Iva04a] Ivanenko, Y., Poppele, R. and Lacquaniti, F., Five basic muscle activation patterns account for muscle activity during human locomotion, Vol.556, pp. 267-282, 2004.

[Kel95a] Kelso, J.A.S., Dynamic Patterns: The Self-Organization of Brain and Behavior, MIT Press, Cambridge, 1995.

[Mat03a] Matz, G. and Hlawatsch, F., Wigner distributions (nearly) everywhere: Timefrequency analysis of signals, systems, random processes, signal spaces, and frames, Signal Processing, Vol.83, No.7, pp. 1355-1378, 2003.

[Oml07a] Omlor, L. and Giese, M.A., Extraction of spatio-temporal primitives of emotional body expressions, Neurocomp., Vol.70, pp. 10-12, 2007.

[Oml06b] Omlor, L. and Giese, M.A., Blind source separation for over-determined delayed mixtures, Adv. in Neural Inf. Process. Syst., Vol.19, pp. 1049-1056, 2006.

[Ros96a] Rose, C., Guenter, B., Bodeheimer, B. and Cohen, M.F., Efficient Generation of Motion Transitions Using Spacetime Constraints, Proc. ACM SIGGRAPH, Int. Conf. on Comp. Graph. and Interactive Techniques, Vol.30, pp. 147-154, 1996.

[Saf04a] Safanova, A., Hodgins J.K. and Pollard, N.S., Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces, Proc. ACM SIGGRAPH, Vol.23, No.3, pp. 514-521, 2004.

[Sch03a] Schaal, S., Ijspeert and A., Billard, A., Computational approaches to motor learning by imitation, Philos Trans of the R Soc of London, Biol. Sci., Vol.358, No.1431, pp. 537-547, 2003.

[Sch95a] Schöner, G., Dose, M. and Engels C., Dynamics of behavior: Theory and applications for autonomous robot architectures, Robotics and Autonomous Sys., Vol.16, No.2-4, pp. 213-245, 1995.

[Sha05a] Shao W. and Terzopoulos, D., Artificial intelligence for animation: Autonomous pedestrians, Proc. ACM SIGGRAPH, Vol.69, No.5-6, pp. 19-28, 2005.

[Ulic02a] Ulicny, B. and Thalmann, D., Towards Interactive Real-Time Crowd Behavior Simulation, Comp. Graph. Forum, Vol.21, No.4, pp. 767-775, 2002.

[Vap98a] Vapnik, V., Statistical Learning Theory, Wiley-Interscience, New York, 1998.

[Wan05a] Wang, W. and Slotine, J.J.E., On Partial Contraction Analysis for Coupled Nonlinear Oscillators, Biological Cybernetics, Vol.92, No.1, pp. 38-53, 2005.

[War06a] Warren W.H., The Dynamics of Perception and Action, Psychological Rev., Vol.113, No.2, pp. 358-389, 2006.

[Wit95a] Witkin, A. and Popović, Z., Motion warping, Proc. ACM SIGGRAPH, Annual Conf. Series, Vol.29, pp. 105-108, 1995.

Efficient Shadows for GPU-based Volume Raycasting

Timo Ropinski Jens Kasten Klaus Hinrichs
Department of Computer Science
University of Münster
Einsteinstraße 62
48149 Münster, Germany
{ropinski, j_kast03, khh}@math.uni-muenster.de

ABSTRACT

GPU-based raycasting has emerged as the defacto standard for interactive volume rendering on off-the-shelf graphics hardware. Even though in theory this technique can be easily extended by shadow feelers in order to support shadows, this obvious approach has a major impact on the rendering performance. In this paper we will investigate shadowing extensions for GPU-based volume raycasting and compare them with respect to their quality and their performance. In particular, we will consider shadow rays, shadow mapping and deep shadow maps. For these techniques we will address their implementations using current graphics boards, and we will compare their visual results as well as their runtime behavior.

Keywords
Volume Rendering, Shadow Generation, Shadow Mapping, Deep Shadow Maps.

1 INTRODUCTION

Volume rendering has evolved into a mature area of computer graphics. Many domains such as medical acquisition or seismic sonography techniques produce volumetric data sets which have to be explored interactively. Besides an immediate visual feedback also good perception of the spatial relationships is important. It has been shown that the used lighting model has a major impact on the spatial comprehension [LB00]. Besides diffuse interreflections also shadows serve as an important depth cue [SSMK05]. While volume rendering techniques developed in recent years mainly had the goal to achieve interactive frame rates, the objective of realistic renderings has been neglected. Now that interactive volume rendering is possible on standard graphics hardware [KW03] it is time to strive for producing more realistic images in order to support better spatial comprehension. In this paper we focus on shadow algorithms which can be integrated into volume rendering in order to provide additional depth cues (see Figure 1). We consider in particular those algorithms and implementations which have the potential to permit interactive shadowing. However, for efficient shadow generation it is important to adapt the used techniques to the volume rendering techniques exploited for image generation. Although different algorithms have been proposed for slice-based volume rendering [KPHE02], we will concentrate on techniques suitable to integrate shadows in GPU-based volume raycasters [KW03], since GPU-based volume raycasting has emerged as the state-of-the-art volume rendering technique which is found in most recent volume rendering frameworks. While the benefits of GPU-based raycasting include an easy as well as flexible implementation, it is also often stated that GPU-based raycasting allows an easy integration of shadowing effects. However, while the brute force approach for shadows can be truly considered as easy to integrate, it is not efficient at all. In fact, it is just an adaptation of the shadow feeler rays known from conventional raytracers, with all their downsides. To provide the reader with a more detailed knowledge about shadows in volume rendering, this paper reviews different shadow algorithms and their implementations in the context of GPU-based volume raycasting. In particular we will compare the brute force approach, i.e., casting of shadow rays, with a shadow mapping technique for GPU-based raycasting as well as deep shadow maps.

The remainder of this paper is structured as follows. In Section 2 we will discuss related work regarding shadow generation by specifically addressing algorithms suitable for interactive volume rendering. In Section 3 our implementations of the shadow algorithms are discussed, and they are compared in Section 4. The paper concludes in Section 5 by giving some overview over 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.

Copyright UNION Agency – Science Press, Plzen, Czech Republic

2 RELATED WORK

The underlying GPU-based volume raycasting technique, in the following called GPU-based raycasting, has been introduced by Roettger et al. [RGWE03].

Figure 1: The engine data set ($256 \times 256 \times 128$ voxel) rendered using phong shading in combination with shadow mapping.

Krüger and Westermann have proposed extensions to better exploit the capabilities of current graphics hardware [KW03]. GPU-based raycasting enables very efficient volume rendering on commodity graphics hardware by casting rays through the volume data set, which is represented as a 3D texture. To determine the entry and exit points for the rays the bounding box of the volume data set is rendered in a color coded manner, such that the values in the RGB color channels can be interpreted as volume texture coordinates.

Various shadow algorithms have been developed in the past. Crow has proposed a shadow volume technique for generating shadows for scenes containing polygonal data [Cro77]. To compute shadows of an object, its silhouette is extracted and extruded in direction of the light rays in order to generate shadow polygons which form the shadow volume of an object. During rendering each object is tested whether it lies inside or outside a shadow volume, and thus it can be determined whether the object is shadowed. Due to the polygonal nature of this algorithm, it is not suited for volume rendering. Another common approach for generating shadows when rendering polygonal data is shadow mapping which has been presented in 1978 by Williams [Wil78]. Shadow mapping is an image-based approach which exploits an additional rendering pass in which the scene is rendered from the light source's point of view in order to determine the structures closest to the light source. With this knowledge, a fragment-based shadow test can be introduced in the main rendering pass, i.e., each fragment is tested whether it is further away from the light source than the corresponding texel in the shadow map. While shadow mapping allows very efficient shadows on a fragment basis, it does not support semi-transparent occluders which can be found in volume rendering. In order to address semi-transparent structures opacity shadow maps serve as a stack of shadow maps, which store alpha values instead of depth values in each shadow map [KN01].

A more compact representation for semi-transparent occluders are deep shadow maps [LV00]. The used data structure consists also of a stack of textures, but in contrast to the opacity shadow maps, an approximation to the shadowfunction is stored in these textures. Thus it is possible to approximate shadows by using fewer hardware resources. Deep shadow mapping has already been applied to volume rendering [HKSB06]. In this paper we will propose an alternative implementation.

Although work has been done regarding shadows in slice-based volume rendering approaches [BR98], to our knowledge the integration of shadows into GPU-based raycasting, which is considered the state-of-the-art volume rendering technique, has not been addressed so far. However, due to the similar ray paradigm, it should be noted that shadows have been integrated in volume ray-tracing systems [WKB+02, WFMS05]. An overview of these techniques can be found in [MFS06].

While the previously cited work as well as this paper focus on volume rendering, it should also be mentioned that some researchers address the combination of polygonal and volumetric data when considering light interactions [ZXC05]. Also soft-shadowing techniques [CD03, ZC03, HLHS03] are not covered in this paper, but it could be investigated how they can be extended to volume rendering.

3 IMPLEMENTATION OF SHADOW TECHNIQUES

In this section we address the implemented shadowing techniques, which we will compare in Section 4. We review the concept and implementation details of shadow rays (see Subsection 3.1), describe our implementation of shadow maps to exploit them efficiently in GPU-based raycasting (see Subsection 3.2) and will describe our improved deep shadow map implementation (see Subsection 3.3).

3.1 Shadow Rays

Shadow rays are the most obvious way to integrate shadows into a GPU-based raycaster. For each sample along the viewing ray a shadow feeler is sent to the light source in order to determine whether the current sample is shadowed.

We have implemented three different variations of this shadowing technique. To support opaque occluders only, each shadow feeler is cast until a sample is reached where the intensity exceeds the shadow threshold. If such a sample lies along the shadow feeler, the current sample is assumed to lie in shadow. However, due to the fact that only the first occurrence of a sample exceeding the shadow threshold is taken into account, shadowing becomes a binary effect, i.e., a sample is either shadowed or unshadowed. To achieve different

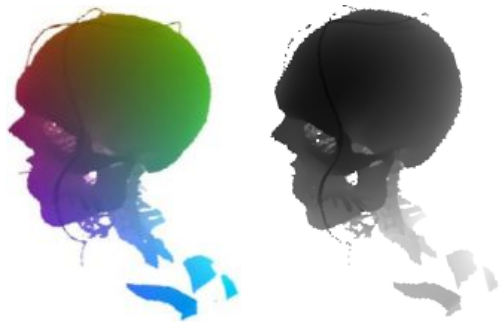


Figure 2: The first hit points in color-coded volume coordinates as seen from the light source (left) and the resulting depth map (right). Both maps have a resolution of 512×512 texel and are generated during rendering the visible human head data set (see Figure 8).

visual results, the shadowing threshold can be changed through the GUI.

The straight-forward extension to support semi-transparent occluders incorporates the intensities of the samples along the shadow feelers. When casting the shadow feelers, the rendering integral is evaluated and for each sample the intensity is considered as the alpha value describing the degree of absorption. Since there is no early ray-termination as when using opaque occluders only, this technique has a recognizable impact on the performance.

For both shadow ray techniques, a shadow feeler needs to be cast for each sample along a viewing ray. Thus the runtime is directly proportional to the screen resolution and the sampling rate. Especially when high quality images are desired this may prevent interactive frame rates. Therefore we have combined the shadow ray technique allowing opaque occluders with a render to 3D texture functionality to be able to cache the results of the shadow feelers: When casting the shadow feeler for a sample, the resulting shadow value is stored within a shadow lookup volume. By using this caching the shadow feelers need only be casted when either the light source or the occluders are changed.

3.2 Shadow Mapping

Shadow mapping can be used as an alternative approach to the shadow rays supporting opaque occluders. In this section we describe our shadow mapping implementation adapted for GPU-based raycasting. In analogy to the general shadow mapping approach (see Subsection 2), we generate a depth map in order to store light source visibility. Therefore we compute the first hit positions as seen from the light source and use them to compute the light source distance (see Figure 2). When rendering polygonal models a depth value is properly defined. In contrast, volume rendering does not provide depth values in general. Therefore we again introduce a shadow threshold value, which can be controlled using the GUI. Thus the intensity values representing opaque geometry can be changed, and with it the content of the generated shadow map. The example shadow map shown in Figure 2 has the same resolution as the viewport, i.e., 512×512 pixel, which allows to generate shadows without introducing aliasing artifacts (see Figure 8).

In comparison to shadow rays, shadow mapping is expected to run faster, since the light source visibility does not need to be recalculated for each sample, but for each light ray only. This results in the fact that no semi-transparent occluders can be supported. However, the benefit of shadow mapping is that soft shadows can be approximated by exploiting percentage closer filtering [RSC87]. This is demonstrated in Figure 3, where the visible human torso data set is rendered with both, hard shadows (left) and fake soft shadows (right). Furthermore, when using shadow mapping the combination with polygonal models can be supported much more easily, since the geometry can also be represented within the shadow map.

3.3 Deep Shadow Maps

As mentioned above, deep shadow maps allow to capture shadows of semi-transparent occluders. For demonstration purposes we have applied our implementation to a simple synthetic scene consisting of $128 \times 128 \times 128$ voxel (see Figure 4). The blue ball, which is set to be semi-transparent by modifying the transfer function, casts a *transparent* shadow on the back wall. The difference to the shadow of the opaque purple box can be noticed, especially when looking at the shadow borders.

In our implementation the deep shadow map consists of eight layers, which are shown on the right in Figure 4. Similar to the approach proposed in [HKSB06], we exploit current graphics hardware in order to generate the shadow layers on-the-fly by using multiple render targets. Although current graphics hardware allows to address up to eight rendering targets in a sin-

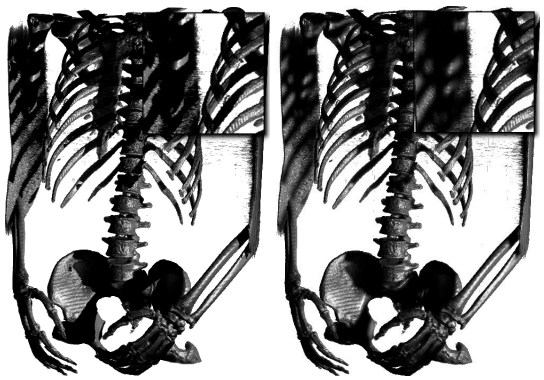


Figure 3: The visible human torso data set ($256 \times 256 \times 512$ voxel) rendered with hard shadows (left) and with fake soft shadows by using shadow mapping exploiting percentage closer filtering (right).

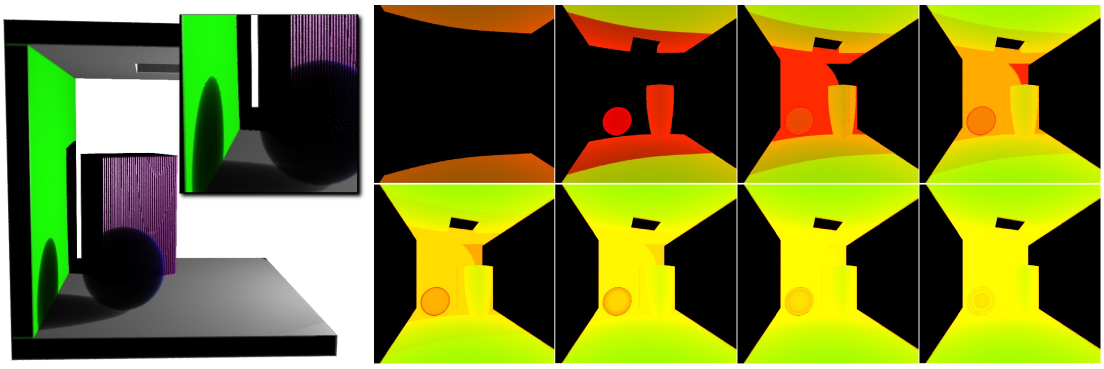


Figure 4: A synthetic scene ($128 \times 128 \times 128$ voxel) rendered using deep shadow mapping. The shadow of the semi-transparent blue ball is correctly captured. The images on the right show the successive layers of the deep shadow map.

gle pass, we only use four rendering targets and create eight deep shadow layers by performing simple channel splitting. While the original deep shadow map approach [LV00] stores the overall light intensity in each layer, we store the absorption given by the accumulated alpha value in analogy to the volume rendering integral. For each shadow ray, we analyze the alpha function, i.e., the function describing the absorption, and approximate it by using linear functions. Unlike the implementation described in [HKSB06] we restrict the depth interval approximated by a linear function to a maximum number of samples, in our case 255 samples. This is necessary in order to have a termination criterion for the fragment shader which performs the processing. However, when it is determined that the currently analyzed voxels cannot be approximated sufficiently by a linear function, smaller depth intervals are considered. Thus, the approximation works as follows. Initially, the first hit point for each shadow ray is computed, similar as done for the shadow mapping described above. Next, the distance to the light source of the first hit point and the alpha value for this position are stored within the first layer of the deep shadow map. Since we are at the first hit position, the alpha value usually equals zero when the shadow threshold is set accordingly. Starting from this first hit point, we traverse each shadow ray and check iteratively wether the samples encountered so far can be approximated by a linear function. In case this cannot be done, the distance of the previous sample to the light source as well as the accumulated alpha value at the previous sample are stored in the next layer of the deep shadow map. This is repeated until all eight layers of the deep shadow map have been created.

The light source distance and the alpha value are stored in two successive channels, i.e., R and G as well as B and A. Thus, we can represent two shadow layers by using only one $RGBA$ texture. However, for illustration purposes, we wrote these values into the R and G channels when generating the pictures shown in Figure 4.

To determine wether the currently analyzed samples can be approximated by a linear function an error value is introduced. In analogy to the original deep shadow mapping technique [LV00], this error value constrains the variance of the approximation. This is done by adding (resp. subtracting) the error value at each sample's position. When the alpha function does not lie anymore within the range given by the error value, a new segment to be approximated by a linear function is started. The effect of choosing a too small error value is shown in Figure 5. As it can be seen, a small error value results in a too close approximation, and the eight layers are not sufficient anymore to represent shadow rays having a higher depth complexity. Thus especially in regions, were the two occluders both intersect a shadow ray, shadow artifacts appear. Obviously this drawback can be avoided by introducing additional shadow layers. This would allow a more precise approximation of the shadow function, but would also result in decreased rendering performance since additional rendering passes are required.

Similar to the previous techniques, we have combined our deep shadow map algorithm with the generation of a shadow volume. Thus, when shadows do not change, the shadow values can be directly fetched from a 3D texture for each sample.

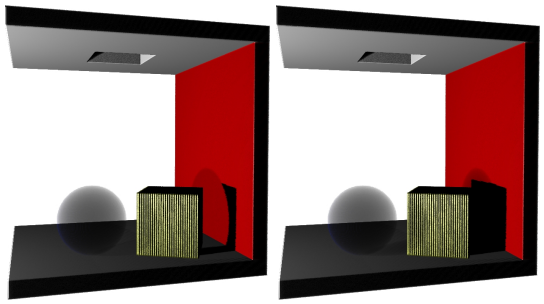


Figure 5: Different error values for deep shadow mapping: 0.00005 (left), 0.01 (right). Artifacts appear when using too small error values.

 ISBN 978-86943-15-2

Shadow Mode	RC	without RC
shadow rays (B)	7.98	7.98
shadow rays (A)	6.70	6.70
shadow rays (B + PP)	4.25	14.90
shadow mapping (B)	12.50	15.00
deep shadow map (A)	10.00	14.90
deep shadow map (A + PP)	8.85	14.90

Table 1: Average frame rates for the different techniques as captured using a GeForce 8800GTX graphics board without optimized graphics settings.

Shadow Mode	RC	without RC
shadow rays (B)	10.03	10.03
shadow rays (A)	10.00	10.00
shadow rays (B + PP)	5.59	46.00
shadow mapping (B)	29.08	45.50
deep shadow map (A)	15.76	34.50
deep shadow map (A + PP)	13.06	45.20

Table 2: Average frame rates for the different techniques as captured using a GeForce 8800GTX graphics board with optimized graphics settings.

4 PERFORMANCE AND RESULTS

To compare not only the capabilities and limitations of each discussed shadowing technique, we have also conducted performance tests. We have scripted an animation which lets the virtual camera rotate around the engine data set shown in Figure 1. The data set has a resolution of $256 \times 256 \times 128$ voxel and has been rendered on a standard desktop computer, having an Intel Core2 CPU 6600 running at 2.40 GHz, 2 GB of main memory, and an nVidia GeForce 8800GTX graphics board.

The results of our performance tests are shown in Table 1 and Table 2; Table 1 shows the frame rates when the graphics drivers are tuned to achieve maximum quality, and Table 2 shows the results when setting the driver to maximum performance. Since this option allows accelerated texture fetches, a vast performance gain can be achieved, though major display artifacts could not be noticed (see Table 2).

The different implementations compared in the tables are three versions of shadow rays, shadow mapping as well as two versions of deep shadow maps. The identifier *A* denotes that semi-transparent occluders with an alpha value unequal to 1 have been supported, while *B* means only opaque occluders are supported. *PP* means that the shadow values are only computed for the first frame and then stored in a 3D texture, which is later on fetched during rendering of successive frames. This *caching* mechanism can be used for rendering successive frames, when only the camera position is changed. *RC* means that shadow computation is done for every frame, i.e., also when neither the light source nor an object has been modified. Thus, the frame rates shown in the column *RC* are the ones to be expected, when light source and/or objects are changed, while the column *without RC* shows the frame rates when for instance only the camera is modified and shadows do not change. This has only an impact on those techniques in which a data structure is used to store the *shadowing state*, but not for the shadow ray traversal in which no knowledge of prior frames can be exploited during rendering.

The results indicate that confining shadow rays to support only opaque occluders does not have a major impact on the frame rates. However, using render to 3D texture functionality in order to generate a 3D shadow lookup texture has a major influence on the performance. When recomputing this 3D shadow texture for each frame, it slows down the system significantly. This is due to the fact that the render to 3D texture functionality only allows to render into one slice of the 3D texture after another. However, when no frequent recomputation is required, the frame rates can be improved when using *PP*. Therefore it should be investigated in how far the shadow recomputation might be predictable for certain cases in order to decide whether the 3D shadow texture should be generated or not. As shown in the tables, shadow mapping allows frame rates which are superior to those achieved by shadow rays. This is due to the fact that a 3D texture fetch is an expensive operation on current graphics boards. Thus, when only opaque occluders are supported, shadow mapping is definitely the technique of choice. When semi-transparent occluders should be taken into account, deep shadow maps also allow a performance gain. However, generating 3D shadow textures does not seem to be as beneficial as when using shadow rays. The reason for this is probably the fact that shadow rays have to operate on 3D textures anyway, while our deep shadow mapping implementation uses 2D textures only when *PP* is disabled.

A comparison of shadow mapping and deep shadow maps with both opaque and semi-transparent occluders is shown in Figure 6. In the Figures 7-10 a visual comparison of the discussed shadowing techniques is shown side by side for different data sets. Compared to the unshadowed images, using shadows results in a higher degree of realism and objects appear less flat. While the shadow rays and the deep shadow maps allow to capture shadows of semi-transparent occluders, shadow mapping supports opaque occluders only. As it can be seen especially in Figure 8 and in Figure 9, deep shadow maps introduce artifacts when thin occluder structures are present. The shadows of these structures show transparency effects although the objects are opaque. This results from the fact that an approximation of the alpha function is exploited. Espe-

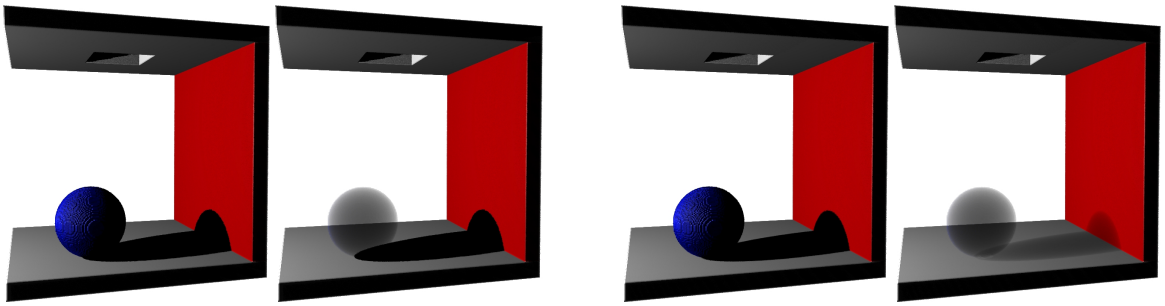


Figure 6: Volume rendering of a synthetic scene ($256 \times 256 \times 256$ voxel) with shadow mapping (left) and deep shadow mapping (right). The transfer function has been changed for both techniques in order to change the ball from being opaque to semi-transparent. Shadow mapping is not sufficient to correctly mimic shadows of semi-transparent shadow casters.

cially thin structures may diminish when approximating over too long depth intervals.

5 CONCLUSIONS AND FUTURE WORK

In this paper we have compared different techniques for combining shadows with GPU-based raycasting. As expected, we were able to show that the most obvious way for generating shadows, i.e., casting shadow rays as done in ray tracing systems, is by far not the optimal technique. However, besides deep shadow mapping it is the only technique which is capable of plausibly capturing shadows cast by semi-transparent objects. We have proposed an efficient implementation of deep shadow maps, which is sufficient to generate the deep shadow map layers in a single rendering pass. Since deep shadow mapping depends on the chosen error values, which are data set dependent, it also can not be considered as the best solution for the general case. Especially when fine structures should cast shadows, deep shadow mapping requires additional effort to provide sufficient results. However, when using the performance improvements proposed in this paper and carefully choosing the error value, deep shadow mapping can produce convincing results.

Because of the identified drawbacks, further research on interactive shadowing techniques is needed in the future. It should be examined in how far deep shadow mapping can be extended to support rather fine structures casting shadows without introducing visual artifacts. Furthermore, it should be investigated in how far a combination of the existing shadowing techniques might help in order to simulate shadows for a wider range of applications. A rather simple extension to the proposed deep shadow mapping technique would be the integration of colored shadows.

Since a lot of research has targeted shadow algorithms for polygonal data sets, it should be examined in how far these concepts can be transferred to volume data sets. For instance, rendering of soft shadows for volumetric data sets has not yet been addressed.

ACKNOWLEDGMENTS

This work was partly supported by grants from the Deutsche Forschungsgemeinschaft (DFG), SFB 656 MoBil Münster, Germany (project Z1). The presented concepts have been integrated into the Voreen volume rendering engine (`www.voreen.org`). The Visible Human is an anatomical data set licensed from the National Library of Medicine and the NCAT data set has been generated using the 4D NCAT phantom.

REFERENCES

[BR98] Uwe Behrens and Ralf Ratering. Adding shadows to a texture-based volume renderer. In *VVS '98: Proceedings of the 1998 IEEE symposium on Volume visualization*, pages 39–46. ACM Press, 1998.

[CD03] Eric Chan and Frédo Durand. Rendering fake soft shadows with smoothies. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 208–218, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[Cro77] Franklin C. Crow. Shadow algorithms for computer graphics. In *SIGGRAPH '77: Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, pages 242–248. ACM Press, 1977.

[HKSB06] Markus Hadwiger, Andrea Kratz, Christian Sigg, and Katja Bühler. Gpu-accelerated deep shadow maps for direct volume rendering. In *GH '06: Proceedings of the 21st ACM SIGGRAPH/Eurographics symposium on Graphics hardware*, pages 49–52, New York, NY, USA, 2006. ACM Press.

[HLHS03] Jean-Marc Hasenfratz, Marc Lapierre, Nicolas Holzschuch, and François Sillion. A survey of real-time soft shadows al-

gorithms. *Computer Graphics Forum*, 22(4):753–774, dec 2003.

[KN01] Tae-Yong Kim and Ulrich Neumann. Opacity shadow maps. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 177–182, London, UK, 2001. Springer-Verlag.

[KPHE02] Joe Kniss, Simon Premoze, Charles Hansen, and David Ebert. Interactive translucent volume rendering and procedural modeling. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 109–116. IEEE Computer Society, 2002.

[KW03] J. Krüger and R. Westermann. Acceleration techniques for GPU-based volume rendering. In *Proceedings IEEE Visualization 2003*, 2003.

[LB00] Michael S. Langer and Heinrich H. Bülthoff. Depth discrimination from shading under diffuse lighting. *Perception*, 29(6):649–660, 2000.

[LV00] Tom Lokovic and Eric Veach. Deep shadow maps. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 385–392, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[MFS06] Gerd Marmitt, Heiko Friedrich, and Philipp Slusallek. Interactive Volume Rendering with Ray Tracing. In *Eurographics State of the Art Reports*, 2006.

[RGWE03] S. Roettger, S. Guthe, D. Weiskopf, and T. Ertl. Smart hardware-accelerated volume rendering. In *Procceedings of EG/IEEE TCVG Symposium on Visualization VisSym '03*, pages 231–238, 2003.

[RSC87] William T. Reeves, David H. Salesin, and Robert L. Cook. Rendering antialiased shadows with depth maps. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 283–291. ACM Press, 1987.

[SSMK05] Mirko Sattler, Ralf Sarlette, Thomas Mücken, and Reinhard Klein. Exploitation of human shadow perception for fast shadow rendering. In *APGV '05: Proceedings of the 2nd symposium on Applied perception in graphics and visualization*, pages 131–134. ACM Press, 2005.

[WFMS05] Ingo Wald, Heiko Friedrich, Gerd Marmitt, and Hans-Peter Seidel. Faster isosurface ray tracing using implicit kd-trees. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):562–572, 2005. Member-Philipp Slusallek.

[Wil78] Lance Williams. 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, 1978.

[WKB+02] Ingo Wald, Thomas Kollig, Carsten Benthin, Alexander Keller, and Philipp Slusallek. Interactive global illumination using fast ray tracing. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 15–24, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[ZC03] Caixia Zhang and Roger Crawfis. Shadows and soft shadows with participating media using splatting. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):139–149, 2003.

[ZXC05] C. Zhang, D. Xue, and R. Crawfis. Light propagation for mixed polygonal and volumetric data. In *CGI '05: Proceedings of the Computer Graphics International 2005*, pages 249–256, Washington, DC, USA, 2005. IEEE Computer Society.

Figure 7: The NCAT phantom data set ($128 \times 128 \times 128$ voxel) rendered without shadows, with shadow rays (A), with shadow mapping and with deep shadow maps (from left to right).

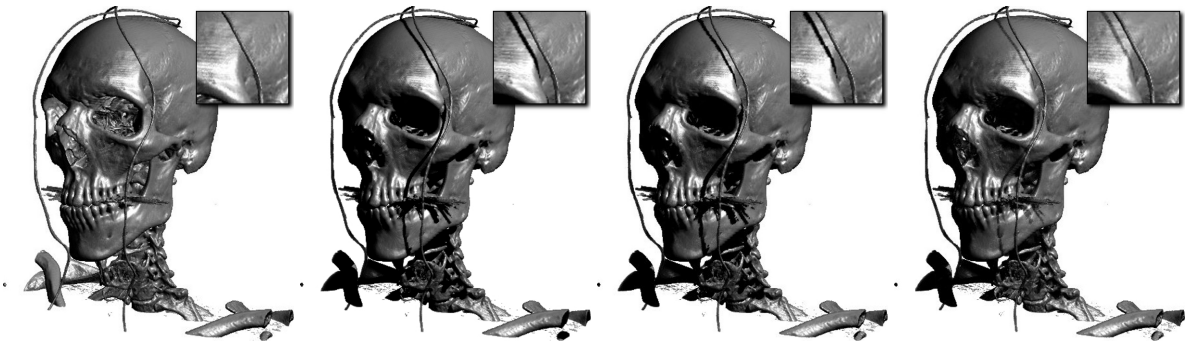


Figure 8: The visible human head data set ($512 \times 512 \times 294$ voxel) rendered without shadows, with shadow rays (A), with shadow mapping and with deep shadow maps (from left to right). The generated shadow map is shown in Figure 2.

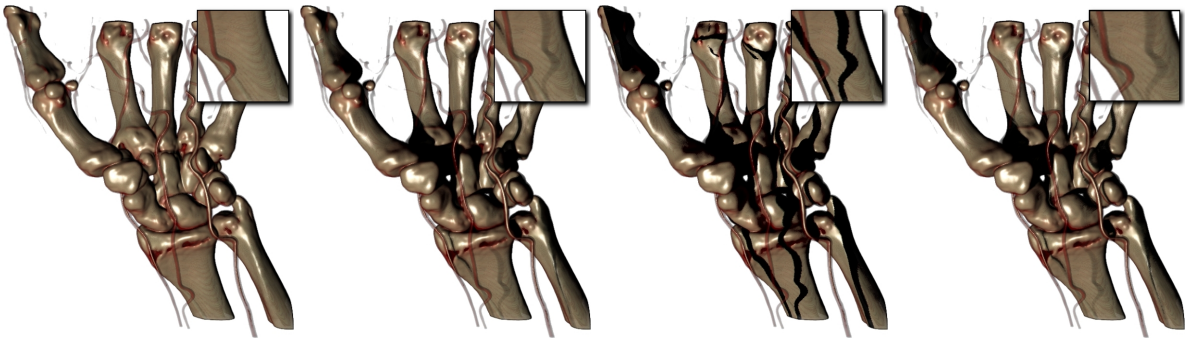


Figure 9: The hand data set ($244 \times 124 \times 257$ voxel) rendered without shadows, with shadow rays (A), with shadow mapping and with deep shadow maps (from left to right). Semi-transparent shadows become visible when using shadow rays or deep shadow maps.

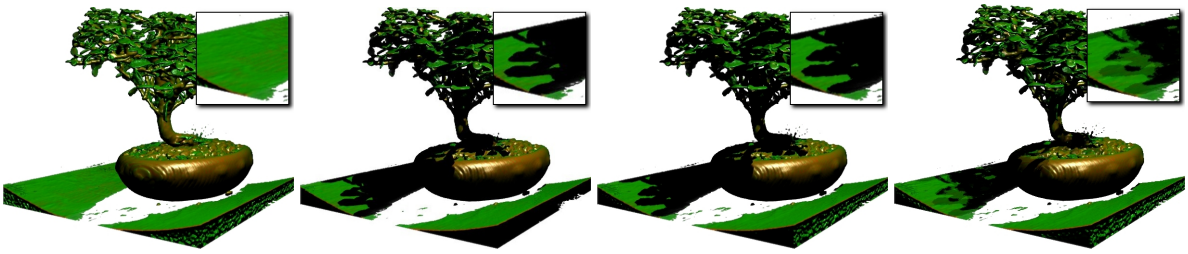


Figure 10: The bonsai data set ($256 \times 256 \times 256$ voxel) rendered without shadows, with shadow rays (A), with shadow mapping and with deep shadow maps (from left to right).

Handwritten Fonts Modeling Based on Fat Lines of Variable Width

Stanislav Klimenko
Moscow Physicotechnical Institute
Moscow, Russia
klimenko@sim.ol.ru

Leonid Mestetskiy
Moscow State University
Moscow, Russia
l.mest@ru.net

Andrey Semenov
Tver State University
Tver, Russia
semenov@tversu.ru

ABSTRACT

In this paper we consider a new geometrical primitive - a curve of variable width (a fat curve). The fat curve is a trace of circle with variable radius moving along a smooth trajectory of the finite length called as an axis. For the mathematical description of such primitives cubic B-splines are used. By means of them it is possible to specify both an axis and a radius value (a thickness) of a fat curve. We solve a problem of representation of hand-written characters given as raster binary images in the form of association of fat curves. We demonstrate a possibility of modeling of "real" handwriting by entering small deformations into the form of fat curves. We consider a method of automatic kerning (visual alignment of letter spacing) of hand-written characters given as curves of variable width. We describe the way of output of arbitrary text presented by series of fat curves on raster-scan devices (printed output, displayed output). A prototype of a font editor of fat curves was developed by the authors of this paper. The presented approach can be used as embedded module (plug-in) in font editors of hand-written characters.

Keywords

B-spline fat curve, skeleton of binary bitmap, circular representation, handwritten font, personal vector cursive, automatic kerning.

1. INTRODUCTION

Attempts to use a body of mathematical tool for description of forms of alpha characters have been undertaken as long ago as in 15 century. The predecessor of all existing typographical fonts is exactly the cursive. There are three types of fonts which are used in a varying degree in personal computer: bitmapped, vector and outline. The last type is the most widespread and usable at the present time. Contours of characters are formed as straight-line segments and sections of Bézier curves of the second (TrueType) and the third (PostScript) orders. In fig. 1 the hand-written character (the letter «a») made of segments of Bézier curves of the third order is shown. Now it becomes very popular to have your personal computer cursive in a collection of fonts. Such font allows singling out the author's letter among the others and individualizing it. There are commercial firms (FontShop - Germany, ParaType - Russia, Signature Software - the USA), which offer their services in designing of full-fledged computer fonts on basis of a sample of penscript (handwriting) of any person [Kar01], in the market of software products. Such service is labor-consuming for the type designer and so it is expensive for the customer.

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

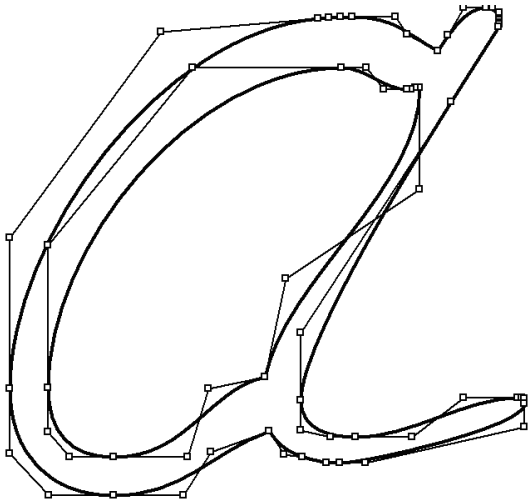


Figure 1. The hand-written character (the letter «a»), described by Bézier curves.

The important feature of a cursive is the requirement of its "revival" in the process of outputting on display devices (display or printer). Revival is understood as small variations in typeface of particular characters, inherent to real handwriting of any person. These variations are expressed in little changes of width and form of strokes for the same characters. From the point of view of such "revival" existing solutions of designing of outline handwritten fonts possess a number of disadvantages. First of all, the outline description of a font doesn't allow implementing variations of stroke width and form of pen. The outline description requires solving an intricate problem of synchronous variation of both borders of a stroke.

Secondly, during the process of stroke variations it is necessary to provide preservation of contour smoothness where it is required. Automatic maintaining of smoothness of compound Bézier curves is also nontrivial problem. Representation of cursive using a model of trace of a pen «with width» (figure 2) potentially gives more possibilities for creation of personal «computer handwriting».

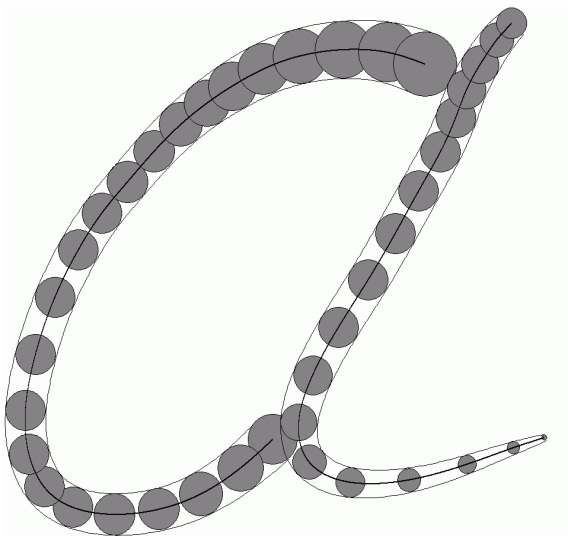


Figure 2. The hand-written character (the letter «a»), described by fat curves.

Such model is used in various graphic editors of hieroglyphs [Kla01], and also in Knut's system *METAFONT* [Knu79,Knu01]. This model makes possible to vary trajectory and pen width, i.e. it is much better adapted for modeling real variations of sample font during writing. However practical implementation of such approach for creation of cursives faces a problem of representation of font characters in the form of curves with width on the initial scanned sample of text. The method presented in this work describes solution of this problem. The source data are the handwritten text given on a paper, for example, the alphabet (fig. 3). And the result of the solution is the description of characters in the form of curves with width.

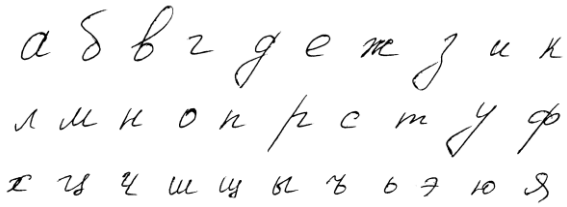


Figure 3. The hand-written Russian alphabet.

On basis of presented method is the idea of representation of the initial raster hand-written text in the form of set of continuous graphic primitives - fat curves [Yao91] (figure 4). A fat curve describes as a trace of circle with variable radius moving along a smooth trajectory called as an axis of this fat curve. B-splines of the third order are used as a mathematical apparatus for description of fat curves.

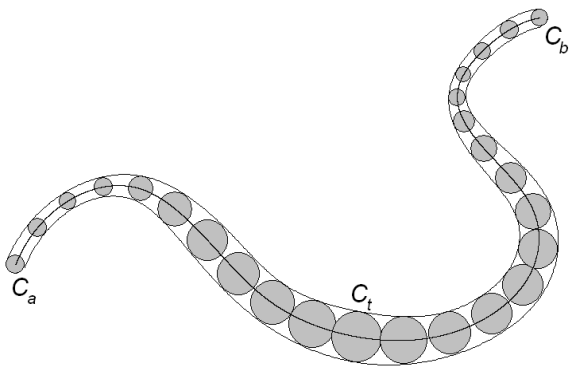


Figure 4. Fat curve.

2. FAT CURVES

Elementary B-spline fat curve of the third order [Mes00] is specified by the following vector equation

$$C(t) = (x(t), y(t), r(t)) = \frac{1}{6}\sum_{i=0}^{3} B_i(t) H_i ,$$

where $t \in [0,1]$ - a curve parameter, $H_i = \{H_{ix}, H_{iy}, H_{ir}\}$ - a set of control circles centered in points (H_{ix}, H_{iy}), with radiuses H_{ir}, $B_i(t)$ - base functions of the third order B-spline satisfying the following conditions [Boo72]:

$$B_0(t) = (1-t)^3$$

$$B_1(t) = 3t^3 - 6t^2 + 4$$

$$B_2(t) = -3t^3 + 3t^2 + 3t + 1$$

$$B_3(t) = t^3$$

Each control circle H_i can be considered as three-dimensional vector. The example of elementary B-spline fat curve with control circles $\{H_0,..., H_3\}$ is shown in fig. 5. The fat curve (as a trace of moving circle) is presented by grey color, the axis of the fat curve - by black color, control circles - by dotted line.

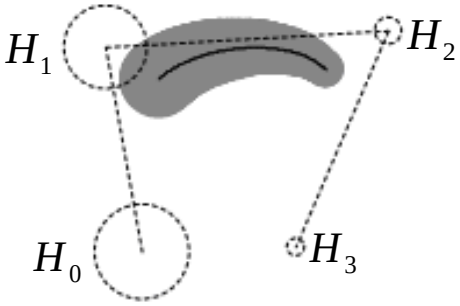


Figure 5. The elementary B-spline fat curve.

Elementary curves can unite into more complex graphic primitives - compound fat curves. Let a curve γ, presented as an association of elementary cubic B-spline fat curves $\gamma^{(1)},..., \gamma^{(m-2)}$, be a compound cubic B-spline curve with common set of control circles $H = (H_0,..., H_m)$, where $H_{k-1}, H_k, H_{k+1}, H_{k+2}$ - control circles of curve $\gamma^{(k)}$. There is the compound B-spline fat curve constructed of 6 elementary sections and having 9 control circles $H_0,..., H_8$ is presented in fig. 6.

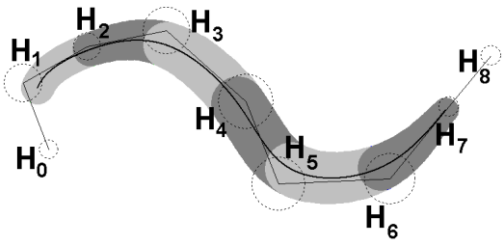


Figure 6. The compound B-spline fat curve.

3. METHOD DESCRIPTION

The presented method to create computer cursive (obtaining of fat-linear representation) consists of the following steps:

- an author of handwriting writes by hand on a paper separate characters, for example, the alphabet;
- an image is entered into computer using scanner;
- a designer selected characters in the scanned image in form of binary bitmaps (black characters on white background);
- for each character the continuous skeleton (as a set of middle axes) [Lee82,Mes98] is constructed;
- each branch of the skeletal graph is approximated by compound B-spline fat curve;
- fat curves unite in strokes representing a character.

Let's call an association of finite number of fat curves as a circular shape. A representation (an approximation) of raster images of characters by circular figures lays on basis of a method. The problem of approximation of the raster sample of handwriting by a circular shape can be solved in several ways of different parts of manual work. Completely manual way assumes direct input and editing of fat curves in the form of sequence of control circles by the designer. Automatic construction of a circular shape is provided using a method described in [Mes00]. The idea of this approach consists in construction of a continuous skeleton and approximation of its branches by fat curves. A problem of primary adjustment is the most labor-consuming. But after it is solved it is necessary to receive qualitative representation of a font where the manual technique is fundamental. However, ample opportunities of editing of fat curves allow developing a special graphic editor for construction of letter descriptions, supporting operations of alignment, editing of axes and width, "interchange" of difficult crossings and overlapping of strokes.

There are stages of transformation of the character given on a raster to the set of fat curves below.

Figure 7. The transformation of the raster character to the set of fat curves.

Some parts of characters (letters) can contain noise which will be visually expressed in not so smooth border and axis of approximating fat curves, as is obvious from figure 7. This noise can be an effect of errors arising during scanning and binarization. Deliverance of a fat curve from such artifacts we will name cleanup. Editing of B-spline fat curve is made by changing positions of centers and radiuses of its control circles. A designer can easily correct the form of a fat curve by moving of control circles and changing their radiuses in a manual mode or he can try to do this operation automatically. For each control circle H_i, $i = 1,..., n-2$, despite of the first one and the last one, we will recount value of its radius using the following formula: $H'_{ir} = k_1 H_{i-1r} + k_2 H_{ir} + k_3 H_{i+1r}$, where H'_{ir} - the new value of radius of circle H_i, k_i - weighting coefficients, satisfying the condition $k_1 + k_2 + k_3 = 1$. The given procedure represents an operation of discrete convolution for a one-dimensional signal with a kernel (k_1, k_2, k_3). A choice of convolution kernel (k_1, k_2, k_3) will characterize a degree of smoothing of control-circle radiuses in a fat curve. An operation of smoothing of control-circle radiuses with the convolution kernel $(1/3, 1/3, 1/3)$ is demonstrated in fig. 8. There is the initial fat curve on the left and the smoothed fat curve on the right. Corresponding circles before and after smoothing operation are presented by polylines.

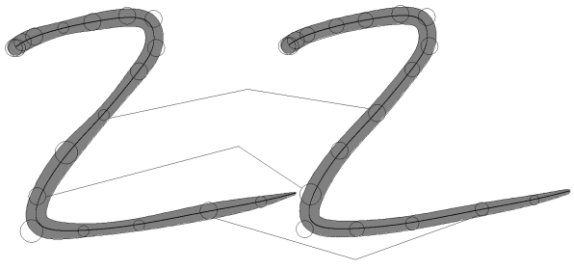


Figure 8. A smoothing of control-circle radiuses.

Similarly, such step can be done for an axis of fat curve. In this case a formula of smoothing operation is the following:

$$H'_{ix} = k_1 H_{i-1x} + k_2 H_{ix} + k_3 H_{i+1x},$$
$$H'_{iy} = k_1 H_{i-1y} + k_2 H_{iy} + k_3 H_{i+1y},$$
$$i = 1,..., n-2.$$

An example of smoothing of axis function of fat curve with the convolution kernel $(1/3, 1/3, 1/3)$ is presented in fig. 9. There is the initial fat curve on the left and the smoothed one on the right.

Figure 9. A smoothing of axis function of fat curve.

It is very convenient to use smoothing operation for radiuses of control circles and an axis of fat curve simultaneously.

The letter breaks into separate fat curves, and their number coincides with number of branches of the corresponding skeletal graph, as is evident from figure 7. It occurs because an automatic adjustment of a circular shape to a raster image gives as a set of fat curves only noncrossing fragments of trajectories. These fragments meet in endpoints, in threes or more branches in a point. Therefore there is a necessity of association of two pairs of branches in one common trajectory for more adequate modeling of pen movement. We will call an operation of joining of one fat line to another as sewing. Sewing procedure is the operation that demands work of a designer. This sewing procedure will allow reducing a number of fat curves, of which a character consists, and making strokes smoother. Solution of a problem of reducing of fat curve from the given set of circles lays on basis of the sewing method. Let $G_0,..., G_{n-1}$ be a consequence of circles, each of them is described as $G_j = (G_{jx}, G_{jy}, G_{jr})$, where (G_{jx}, G_{jy}) - coordinates of the center of the circle and G_{jr} - its radius. Let's call this set of circles as basic. It is required to construct a compound B-spline fat curve $C(t) = (u(t), v(t), r(t))$, that exactly passes all circles G_j (an interpolation problem) or sufficiently near from circles G_j (an approximation problem). Constructing of the required fat curve consists in finding its control circles. Formally, an interpolation problem consists in choice of such description $C(t)$ that the following condition is satisfied:

$$(\exists\, t = t_i : C(t_i) = G_i,\ i = 0,..., n-1).$$

A in a an approximation problem – the condition

$$(\exists\, t = t_i : \mu(C(t_i), G_i) < \varepsilon,\ i = 0,...,n-1),$$

where ε is a preassigned accuracy.

As stated above, both these problems can be solved using maximal empty circles of skeletal graph as a set of basic circles. From the algorithmic point of view sewing procedure consists in the following. Sets of basic circles of skeletal branches unite in one sequence for which an approximation of the derived set of circles is made using the method described in [Mes00]. In fig. 10 the letter after "sewing" procedure is shown. After each step of "sewing" it is possible to make smoothing and editing procedure of radiuses and centers of control circles of a new fat curve. This procedure allows adequate simulating of strokes and also reducing a dimension of shape description and gives the convenient tool to change character forms.

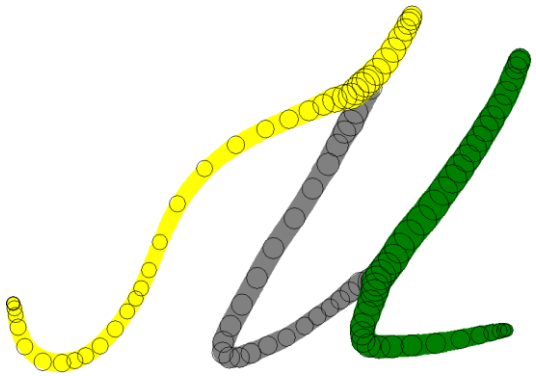


Figure 10. A step of "sewing" fat curves.

It is required to consider a case when two different strokes are partially or completely overlapped by each other during a process of writing for correct modeling of letter characters by fat curves. Visible form of a stroke in the scanned image is an association of two or more trajectories of pen movement. Therefore it is necessary to select them from this association for the adequate description of these trajectories. It is difficult to automatically consider this feature of human writing during approximation of such characters by fat curves. The case when two pen trajectories look like one is shown in fig. 11

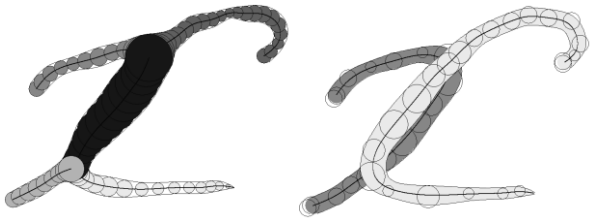


Figure 11. A step of cloning and "sewing" fat curves.

Participation of a designer is required to solve such situations. The presented decision consists in copying (cloning) of that fat curve which has united in itself two strokes. Each of two clones is integrated ("sewed") in a corresponding trajectory. The further step consists in sewing, smoothing and editing derived fat curves. The result of this operation is demonstrated in fig. 11. Such character "structure" consisting of fat curves is closer to a real writing of letter by person.

The next step to create of a personal cursive is a reduction of all letters (characters) to a base line (a line of writing). A designer should do this step for each character himself. Five characters reduced to a base line of a font are shown in fig. 12. A base line is easily entered into description of a circular shape. It is possible to enter it both manually and automatically, for example, using a requirement that the sample of handwriting is presented in a form with a base line

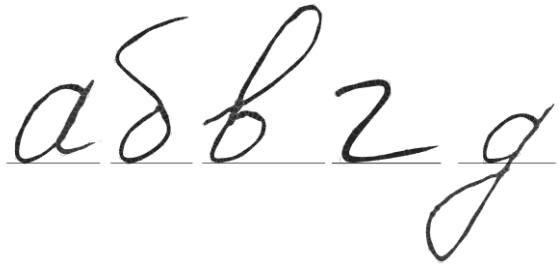


Figure 12. A reduction of all letters to a base line.

4. METHOD OF AUTOMATIC KERNING

A modern computer font consists of a set of character images, and also of a collection of instructions and rules for correct display of letters during rasterization. One of examples of such rules is a visual alignment of letter spacing (kerning) in process of text on raster devices [Kar01]. There can be such combinations of signs and characters which will form visual clearances and clottings, that introduce a certain discomfort in a rhythm of text reading, practically in any text. This problem becomes especially serious when a font size is large. A feature of a kerning problem is in squared relationship of number of formed kerning pairs from a general number of characters in a font. Though actually kerning pairs which practically can't be found in texts exist. Nevertheless, processing of such volume of information in manual mode is obviously a labor-consuming problem. Therefore development of an automatic method of visual alignment of letter spacing is a rather actual problem. It is necessary to notice, that for a cursive the kerning problem has more importance than for typographical type-setting

fonts. The example of hand-written words without the kerning mechanism and with it is presented below (fig. 13). For clearness the minimum rectangles of each character are also shown.

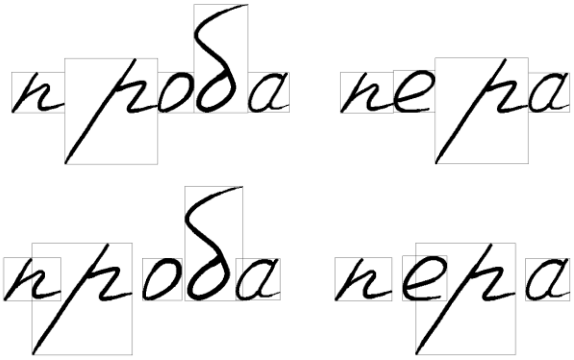


Figure 13. A hand-written words without/with the kerning.

The kerning operation is applied to achieve optical balance of the text. Because of features of the form of some character pairs there is an illusion that the distance between them is more or less than real. Kerning can be manual and automatic. In manual kerning subjective feeling of a designer is the criterion of optical balance of a text. The presented method of automatic kerning of hand-written characters includes the following steps:

1. rasterization of kerning pair of characters on a common base line of raster;
2. scanning by a horizontal line "from top to bottom" with step of 1 pixel;
3. the most east and the most west cross points are defined for left and right character of kerning pair respectively at each position of a scanning line;
4. position of a scanning line, at which distance between the points found on the previous step is minimum, is defined;
5. the right character of kerning pair is displaced horizontally (to the right/to the left) in such a way that the minimum distance between the characters was equaled to preassigned constant (fig. 14).

This value will characterize the minimum permissible horizontal distance between two characters of kerning pair in rasterization. The more is this value, the more will be visual “clearance” between characters in the text. Set of all character pairs form a so-called kerning table. Value of an element of kerning table, located on crossing of line i and column j, specifies on how many pixels to the right/to the left it is necessary to move the right character of kerning pair in process of printed output.

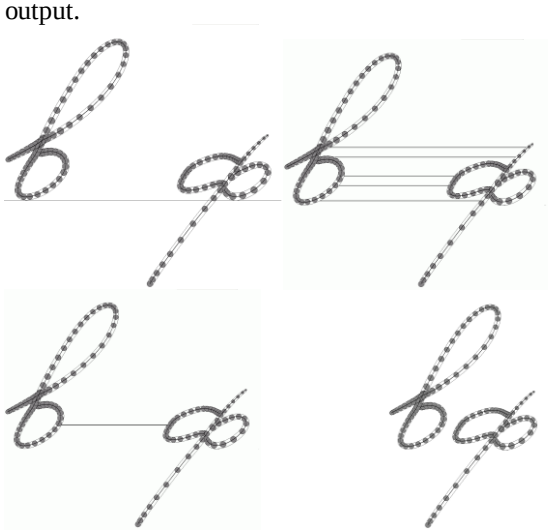
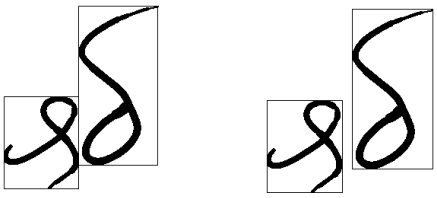


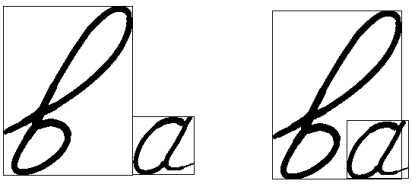
Figure 14. A automatic kerning of hand-written characters.

Positive value of an element of kerning table means that it is necessary to “move aside” (to displace to the right) the right character. Negative value signed that it is necessary to “move up” (to displace to the left) the right character of kerning pair to the left character (fig. 15).

	а	**б**	**в**	**..**	**ю**	**я**
а	19	15	8	..	11	19
б	-13	-17	-5	..	-29	-21
в	**-72**	1	-62	..	-88	-80
..	..	..	..	..	..	..
ю	16	12	6	..	2	12
я	19	**15**	11	..	19	19



KERNING[*я*][*б*] = 15

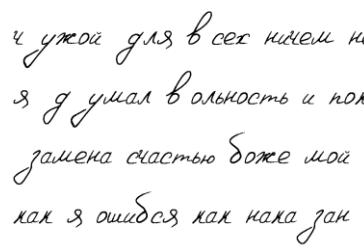


KERNING[*в*][a] = -72

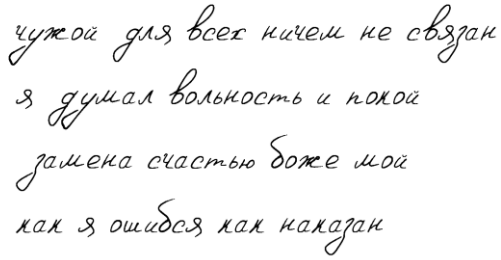
Figure 15. Examples of kerning pair.

It is necessary to notice, that minimum permissible distance between kerning pairs is a value which can be operated. The same hand-written text with different values of this parameter is shown below.

(a)



(b)



(c)

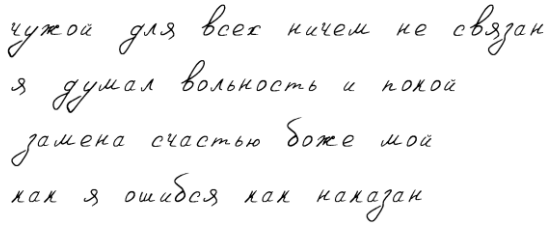


Figure 16. Examples of hand-written text with various kerning
(16a – no kerning, 16b – kerning = 20,
16c – kerning = 40).

5. APPLICATIONS

The mathematical apparatus of letter description using fat curves allows realizing modeling of "real" handwriting by entering of small dynamic changes into the letter forms. The idea of such revival of handwriting consists in the following: having printed the same text using such font several times, we will receive distinct results. There is a software product ParaNoise belonging to Russian firm ParaType which allows changing the form of characters by the given rule. The basic idea of this program is a deformation of a contour (shift, rotation, scaling) using certain algorithm. In terms of fat curves the effect of "randomness" for character forms can be implemented much more variously and easier like this. In process of rasterization (an output on a raster device) of each particular character of all radiuses of control circles, we will add a certain random variate, namely $H_{ir} = H_{ir} + \xi_i$, where H_{ir} – a radius of control circle H_i, ξ_i – a random variate, evenly distributed on a given segment $[-a, a]$. The more is the value of a, the more random and "dissimilar" the corresponding characters will be. There is a letter with modeling of random form (segment $[-a, a] = [-5,5]$) in fig. 17. Here we use a display pixel as a unit of measurement.

Figure 17. Modeling of "randomness" forms.

Besides, it is possible to use different types (normal, thin, wide) of pen in process of text output on raster devices. The construction of fat curve allows changing pen width easily using proportional change of radiuses of all control circles. There is the initial symbol in fig. 18a, in fig. 18b radiuses are reduced by 30 %, in fig. 18c - radiuses are increased by 50 %.

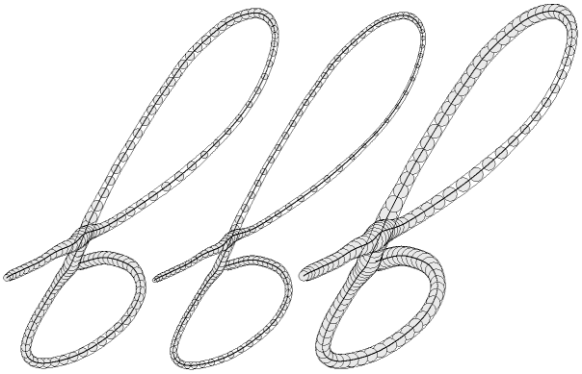


Figure 18. Modeling of "thickness".

The way of character description using the model of a pen trace allows making converting of the received font into standard formats (TrueType and PostScript) in the form of the outline description (fig. 19). It is

possible to make by means of construction of border of a fat curve as envelope curve for smooth sets of circles [Mes00] and approximation of these curves by Bézier curves of the second and the third order [Mes06]. The possibility of approximation of characters with controllable accuracy in the form of outlined description by Bézier curves of the third order is implemented by authors.

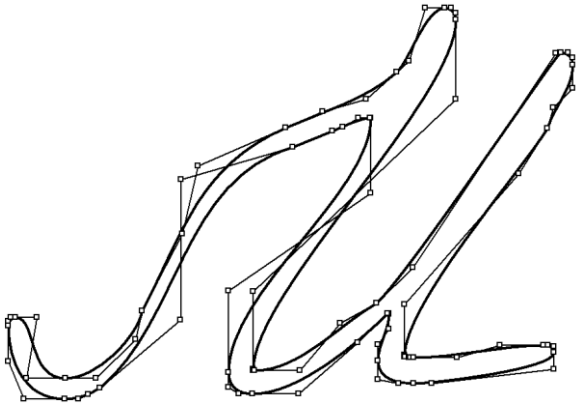


Figure 19. The outline description.

6. CONCLUSIONS

The method presented in this work to describe hand-written characters using curves of variable width was software implemented in the form of prototype of the font editor. All illustrative examples demonstrated in the paper, are received using the developed program. The conducted research has shown that the method of fat curves is potentially convenient and efficient tool in process of modeling of form of hand-written characters. The main advantage of the presented approach consists in more adequate description of a cursive using «a pen with width». The presented approach can be used as embedded module (plug-in) in font editors of hand-written characters.

7. ACKNOWLEDGMENTS

Authors are grateful to the Russian Foundation of Basic Researches for the support of this work (grant № 05-01-00542).

8. REFERENCES

[Boo72] de Boor C. On calculation with B-splines. J. Approx. Theory, Vol. 6, 1972.

[Lee82] Lee D.T. Medial axes transform of planar shape. IEEE Trans. Patt. Anal. Mach. Intell. PAMI-4 (1982), 363-369.

[Kar01] Karov P. Font technologies. The description and toolkit, 2001 (In Russian).

[Kla01] Klassen V. Variable width splines: a possible font representation? Electronic publishing, vol.6 (3), 1993.

[Knu03] Knuth D. All about METAFONT. Publishing house VILJAMS, 2003 (In Russian).

[Knu79] METAFONT, A system for alphabet design, apart 3 of TeX and METAFONT: New directions in typesetting, American Mathematical Society and Digital Press, Bedford, Mass, 1979).

[Mes00] Mestetskiy L. Fat curves and representation of planar figures. *Computers & Graphics*, vol.24, No. 1,2000, pp. 9-21.

[Mes06] Mestetskiy L. Skeleton of a multiply connected polygonal figure. International conference of computer graphics "Graphicon", Novosibirsk, 2006, pp. 299-314. (In Russian).

[Mes98] Mestetskiy L. Continuous skeleton of binary raster bitmap. In Graphicon'98, International Conference on computer graphics, Moscow, 1998 (in Russian).

[Yao91] Yao C., Rokne J. Fat curves. Computer graphics forum, 10, 1991.

.

Butterfly Plots for Visual Analysis of Large Point Cloud Data

Tobias Schreck
Technische Universität Darmstadt
tobias.schreck@gris.informatik.tu-darmstadt.de

Michael Schüßler
Siemens ElectroCom
Postautomation GmbH, Berlin
michael.schuessler@siemens.com

Katja Worm
Siemens ElectroCom
Postautomation GmbH, Berlin
katja.worm@siemens.com

Frank Zeilfelder
Technische Universität
Darmstadt
frank.zeilfelder@gris.informatik.tu-darmstadt.de

ABSTRACT

Visualization of 2D point clouds is one of the most basic yet one of the most important problems in many visual data analysis tasks. Point clouds arise in many contexts including scatter plot analysis, or the visualization of high-dimensional or geo-spatial data. Typical analysis tasks in point cloud data include assessing the overall structure and distribution of the data, assessing spatial relationships between data elements, and identification of clusters and outliers. Standard point-based visualization methods do not scale well with respect to the data set size. Specifically, as the number of data points and data classes increases, the display quickly gets crowded, making it difficult to effectively analyze the point clouds.
We propose to abstract large sets of point clouds to compact shapes, facilitating the scalability of point cloud visualization with respect to data set size. We introduce a novel algorithm for constructing compact shapes that enclose all members of a given point cloud, providing good perceptional properties and supporting visual analysis of large data sets of many overlapping point clouds. We apply the algorithm in two different applications, demonstrating the effectiveness of the technique for large point cloud data. We also present an evaluation of key shape metrics, showing the efficiency of the solution as compared to standard approaches.

Keywords: Visual Analytics, Point Clouds, Visual Aggregation, Shape Construction, Shape Refinement.

1 INTRODUCTION

Visualization of 2D point clouds is one of the most basic yet one of the most important task in many data analysis scenarios. Point clouds are an ubiquitous type of data arising in many data analysis tasks. E.g., point clouds may be obtained by plotting pairs of selected attributes of a multivariate data set against each other, obtaining scatter plots which are useful for analysis of correlations, clusters, and outliers. As another example, high-dimensional data sets can be visually inspected by obtaining projections to low-dimensional display space, e.g., by using Principal Components Analysis or Multidimensional Scaling techniques. As a third example, the visualization of geo-spatial data often has to deal with sets of points representing certain locations. Typical analysis tasks in point cloud data include assessing the overall structure and distribution of the data, assessing spatial relationships between data elements, and identification of clusters and outliers.

Standard point-based visualization methods do not scale well with respect to the data set size. More specifically, as the number of data points and data classes increases, the display usually gets crowded quickly. Then, it is very difficult for the user to distinguish different point clouds from each other, or to correctly perceive their shape. Both effects harm the effective visual analysis in standard point cloud visualization approaches.

A typical example where the user is confronted with large sets of point clouds is projection-based visual analysis of high-dimensional data, e.g., in a database exploration application. Assume a database consisting of thousands of multidimensional records in many different groups. Projection of the records to 2D display space, which is desirable for visual analysis of database content, will most likely lead to cluttered and crowded displays of many overlapping point clouds.

The scalability problem may be addressed by aggregation or reduction on either the data level, the visualization level, or both. In this paper, we propose to abstract point clouds by compact shapes, forming so-called Butterfly plots. We demonstrate the suitability of these shapes for the effective visualization of large point cloud data. We develop an algorithm for constructing shapes that (a) enclose all members of a point cloud, that (b) are compact in terms of covered space, and that (c) provide good perceptional properties. We apply the algorithm on two large data sets, demonstrating the effectiveness of the approach. A systematic evaluation of the shape construction algorithm is given, including practical parameter setting recommendations.

2 RELATED WORK

The need to visualize point cloud data arises in many important application areas. Point-based data is obtained e.g., by projecting high-dimensional input data to display space for visual analysis. Principal Compo-

nents Analysis (PCA) [10] or Multidimensional Scaling (MDS) are two popular techniques for projecting high-dimensional input data to low-dimensional display space. Projection-based applications include e.g., the analysis of high-dimensional financial data [3], or multimedia database exploration [13]. Other applications often relying on point cloud data visualization include the analysis of geo-spatial and bivariate data. Multivariate data can be visualized by so-called scatter plot matrices.

Many of these applications require the appropriate visualization of point data. The Bag Plot [14] is a statistical technique that visualizes core and peripheral areas of 2D point data. The over-plotting problem which may occur in large geo-spatial data sets is addressed in [12, 7] by introducing certain geometric distortions and rearrangement of points within their neighborhood. In [8], the authors consider the efficiency of rendering very large sets of point data using improved data structures and rendering algorithms.

A promising approach to visualize large point cloud data is based on visual abstraction of point clouds by shape. The reconstruction of surface from unorganized 3D point sets is a challenge currently addressed in point-based Computer Graphics. E.g., in Computer Aided Design, the reconstruction of shape from CAD models given in point-based representation is addressed [2, 11]. In Image Processing [5], an important problem is the segmentation of shapes in digital images, which is addressed e.g., by means of morphological operations.

In [15], we proposed to abstract large point-cloud data by forming convex hulls over *thinned* point clouds. However, in many cases thinning (data reduction) is not an option, as the whole data set needs to be analyzed and outliers may play an important role. In this paper, we therefore develop an effective point cloud visualization not requiring data reduction.

3 SHAPE ABSTRACTION

In this section, we discuss basic requirements for abstracting point clouds by shapes, and introduce an algorithm for construction of compact shapes completely covering clouds of points.

3.1 Requirements analysis

Traditionally, point clouds are visualized by representing each data item by a symbol. Symbol position represents attribute values, while the symbol itself encodes the class a point belongs to. Usually, form and color is used to indicate class labels. This, however, is not expected to scale with the number of classes. Regarding color, there is evidence that human perception is limited to discrimination only about ten colors simultaneously [16]. The situation is less clear regarding usage of symbol shape, but it can be expected that also, perceptional limits exist. Figure 3 (bottom-left) gives an example

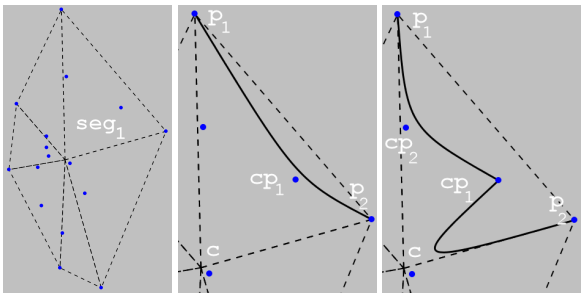


Figure 1: Left: Convex hull enclosing a set of points. The middle and right images show the initial and the first sub-refinement steps respectively, applied on the hull boundary line $\mathbf{p_1 p_2}$.

symbol plot of a medium size data set. The number of data items and different classes (45) makes it extremely difficult to perceive even the most essential characteristics of the data distribution.

Abstraction of point clouds by suitable *shapes* is a promising approach to improve a point cloud display in case of many different clouds overlapping each other. An appropriate shape abstraction replaces sets of points by a number of shapes, reducing plot complexity and at the same time, visualizing the essence of the data characteristics.

Shape-based abstraction of large sets of point clouds should provide that the shapes (a) are as *compact* as possible, and (b) are as *predictable* as possible. When visualizing many shapes simultaneously, more compact shapes usually reduce the degree of overlap among the different shapes, thereby supporting shape discrimination. On the other hand, overlap often cannot be avoided on large data sets, even for the most compact shapes. Then, it should be possible for the user to mentally reconstruct as much of the occluded shapes as possible based on the clues given by the non-occluded parts of the shapes. We call this property shape predictability. A tradeoff exists between the two criteria: More compact shapes typically involve boundaries of higher complexity, thereby harming predictability.

Many ways exists to abstract point clouds to shapes. Shapes can be formed to cover the point clouds completely, or just partially. The shapes can be as simple as a bounding disc or rectangle, or be of higher complexity. The shapes may be allowed to contain holes, or not. In [15], we previously proposed to form convex hulls over point clouds thinned for outlier points at the periphery of the point clouds. The convex hulls were found to be more effective that bounding discs and boxes for visual analysis in large point cloud data.

We here are concerned with forming shapes for *complete* coverage of the point clouds. This is desirable as in many applications, it is interesting to analyze the characteristics of outlier points with respect to the whole point distribution. Without outlier removal, the

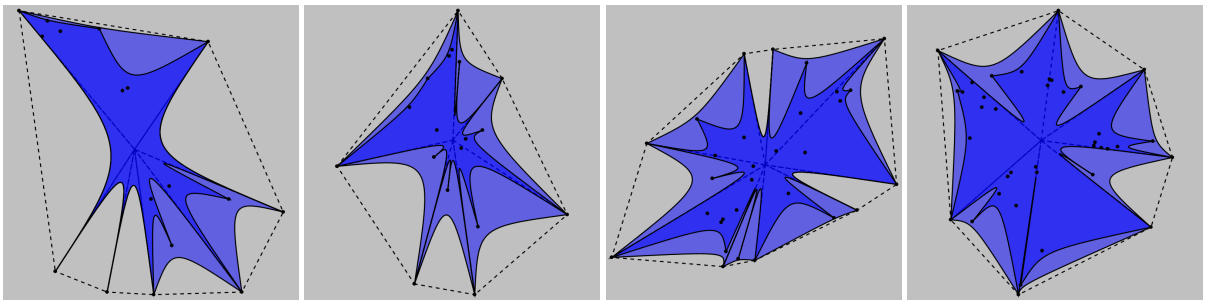


Figure 2: Butterfly plots resulting after the initial (outer shapes) and first sub-refinement steps (inner shapes) have taken place on the convex hulls of several point clouds. Significant reduction of the convex hull area between 56% and 20%, and between 74% and 46% are observed as a result of the first and second refinement steps.

compactness of the convex hull abstraction suffers, as its area is sensitive with respect to outliers. In the next section, we develop a shape abstraction covering *all* points of a given point cloud, which is both compact and offers good perceptional properties.

3.2 Butterfly plot construction

The convex hull [6] is the most compact convex enclosure of a number of points. This makes it a good starting point for a refinement process aimed at reducing the shape area. Our basic idea is to iteratively replace the straight convex hull boundary lines by curves more tightly fitting to the interior points of the point cloud.

We describe our algorithm by means of a constructive example. Consider a set of points $P \in \Re^2, |P| > 2$ with associated convex hull consisting of n boundary lines. Partition the convex hull into n triangular segments by connecting the center of gravity $\mathbf{c}$ of the point cloud with the end points of each convex hull boundary line. Figure 1 (left) illustrates such a partitioning, and Figure 1 (middle) closes up on segment $seg_1 = \triangle(\mathbf{c}, \mathbf{p_1}, \mathbf{p_2})$. Refinement of the convex hull takes place by two steps: Mandatory *initial refinement* of each convex hull boundary line by exactly one curve segment, followed by optional *recursive sub-refinement* of the output of the initial refinement. The optional sub-refinement is controlled by an area reduction-based threshold test, and a global limit of the maximum allowed recursion depth.

The initial refinement stage works by replacing each convex hull boundary line by a curve connecting the line end points and fitting to a control point $\mathbf{cp}$. This control point is found as the point closest to the line to be refined and located inside the respective convex hull segment. In case the given segment does not contain any data points, we take the center of gravity $\mathbf{c}$ as the respective control point. Figure 1 (middle) illustrates the initial refinement of boundary line $\mathbf{p_1p_2}$ by a curve from $\mathbf{p_1}$ to $\mathbf{p_2}$ controlled by $\mathbf{cp_1}$, which is the data point closest to line $\mathbf{p_1p_2}$ and contained in seg_1.

After the initial refinement step has taken place, we optionally continue to refine the found curve segment. To this end, we partition each convex hull segment into two triangular sub-segments formed each by the center of gravity, the segment's initial control point, and either one of the segment line end points. Refinement may continue recursively on the sub-segments until either all data points are exhausted, or a termination criterion is met. Figure 1 (right) illustrates the first sub-refinement step applied on seg_1. The respective sub-segments are given by triangles $seg_{1,1} = \triangle(\mathbf{c}, \mathbf{p_1}, \mathbf{cp_1})$ and $seg_{1,2} = \triangle(\mathbf{c}, \mathbf{cp_1}, \mathbf{p_2})$. We refine $seg_{1,1}$ by a curve between $\mathbf{p_1}$ and $\mathbf{cp_1}$ controlled by $\mathbf{cp_2}$, and $seg_{1,2}$ by a curve between $\mathbf{cp_1}$ and $\mathbf{p_2}$ controlled by $\mathbf{c}$.

The refinement of convex hulls by this scheme is expected to show two effects: A recovery of area from the initial convex hull, and an increase in boundary circumference and complexity. We reflect this trade-off by evaluating a test prior to executing each candidate sub-refinement step. Specifically, we execute a candidate sub-refinement step only if

(a) its recursion depth is within a limit ρ, and

(b) it recovers at least a fraction τ of convex hull area.

ρ and τ are useful for balancing area reduction and shape complexity. Note that we evaluate this test only on recursive sub-refinement steps, and not on the initial refinement step which is executed mandatorily. The rationale is to avoid shapes mixing unrefined segment lines with curves, a combination which according to our experiments showed undesirable perceptional effects. Also, the *type of curve* used for refinement needs to be specified. We experimented with different curve types, and found the Bézier Cubic Spline [4], obtained by doubling the control point, giving good results. This curve tightly fits its control point, and runs strictly within the triangle spanned by the given start, end, and control point. The latter property in conjunction with our convex hull segmentation scheme provides that the obtained shape completely encloses the given point cloud.

Algorithm 1 gives our convex hull refinement algorithm. It consists of the main procedure `butterfly` taking as input a point cloud P, an area reduction thresh-

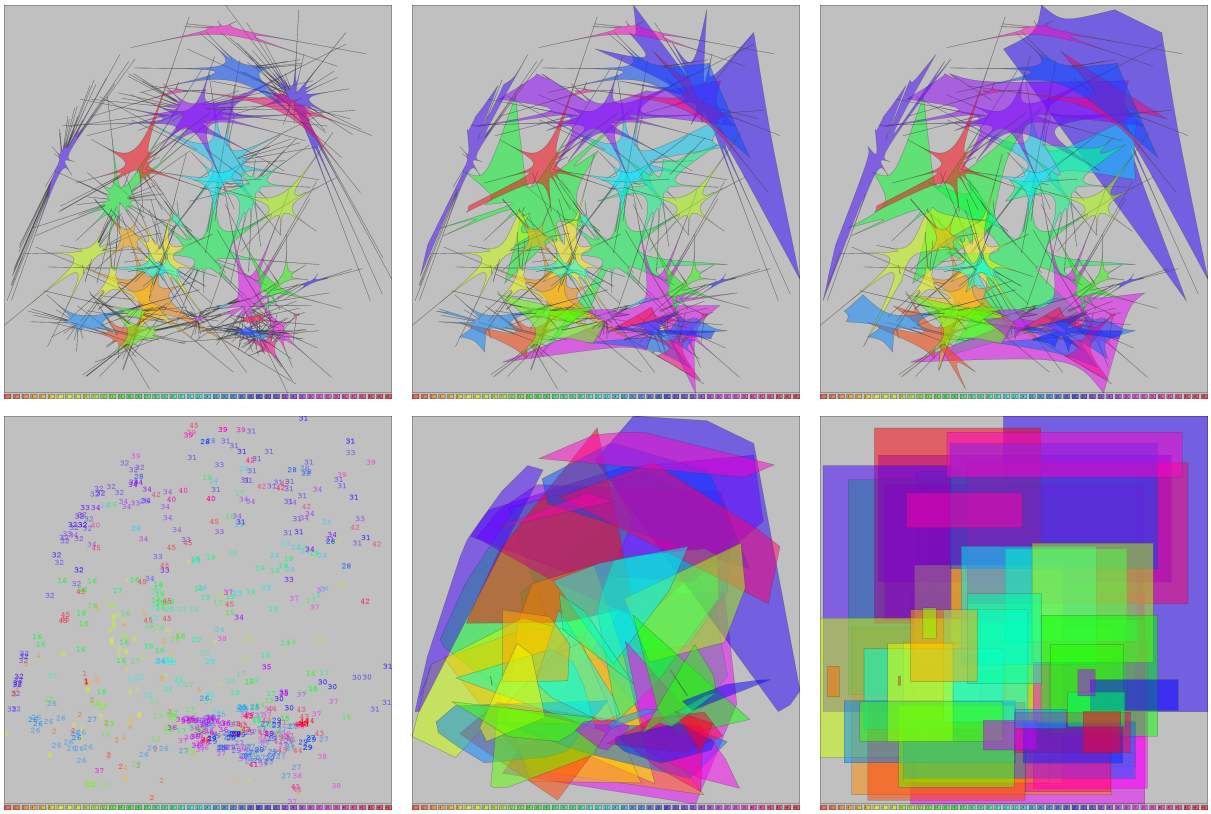


Figure 3: Butterfly plots obtained using refinement thresholds $\tau = \{0.00,\ 0.02,\ 0.04\}$ and unlimited recursion depth, for a data set consisting of 45 classes (top row, left to right). The bottom row shows plots using labeled point clouds, and enclosing convex hulls and minimum bounding boxes. The Butterfly plot approach improves over these standard point cloud visualization methods, allowing effective visual point cloud analysis.

old τ, and a recursion depth limit ρ. Note that in procedure `refine` the term $|l \cup curve(l.p1, cp, l.p2)|$ denotes the area enclosed between a (segment) line l and a refinement curve connecting l's end points. It corresponds to the area reduction achieved by applying a given refinement step. Figure 2 shows exemplary results obtained for several point clouds from a data set to be further discussed in Section 4.2. The shapes were obtained by setting $\tau = 0$ and $\rho = \{0, 1\}$, respectively. We observe the shapes are compact and possess a smooth, predictable boundary of limited complexity. We decided to call the obtained shapes Butterfly plots, as one can recognize certain similarities between the shapes and the insect, on an abstract level.

4 APPLICATION

We next present two Butterfly plot applications, demonstrating the effectiveness of the technique.

4.1 Database exploration data set

The first application is in *database exploration*. Visual database exploration methods often provide graphical representations of the database objects and their interrelationships. In [13], the visual exploration of a database of more than 850 3D CAD models [9] was demonstrated by means of PCA projection of corresponding object feature vectors. We visualize that database to support exploration of the relationship between different object classes under a given feature vector representation. We generate Butterfly plots of the database as follows. We first map the database to 2D by PCA analysis of the database feature vectors (specifically, *complex-SH* 3D features [1] were used). Together with a classification associated to the database [9], 45 2D point clouds are obtained, for which we generate a Butterfly plot each. We sort the resulting shapes decreasingly by size, and give each a distinct color sampled equally from the rainbow palette. An overall Butterfly plot for the full database is obtained by rendering all individual Butterfly plots in order, using the associated color at medium transparency. Note that color here supports visual class discrimination but does not carry additional information.

The top row in Figure 3 shows Butterfly plots obtained for $\rho = \infty$ and $\tau = \{0.00,\ 0.02,\ 0.04\}$. The leftmost plot represents the maximum refinement possible for a Butterfly plot, yielding a skeleton-like abstraction of the point clouds allowing effective visual perception of two salient data characteristics: Location of point

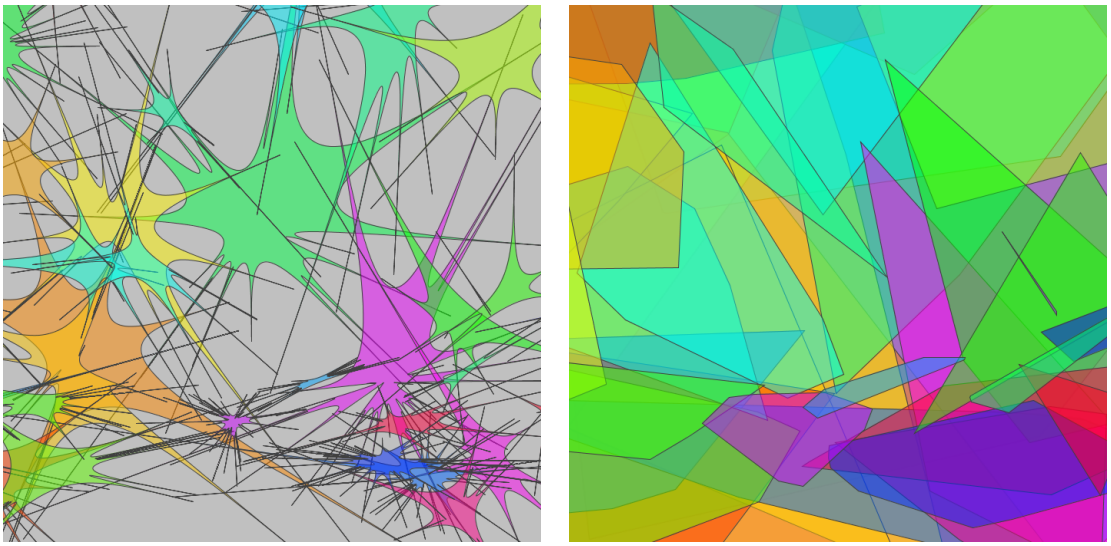


Figure 4: Zooming into a dense area of the Butterfly ($\tau = 0$) and convex hull plots from Figure 3.

cloud centers, and distribution of cloud member points. The center is easily perceivable by the shapes' main area, which is also pointed to by the various curve segments. By construction, we know that each corner point of the shape contains at least one data point. Thereby, we can follow the elongated attachments to analyze the relation between point cloud center and member points. Butterfly plots for $\tau = 2\%$ and $\tau = 4\%$ are shown in Figure 3 (middle and right image in top row). As expected, less refinement takes place, resulting in less compact shapes which emphasize to a larger extent the degree of outlier distribution: Usually, for more scattered point clouds, Butterfly plots of larger area result. The user can be allowed to adjust parameters ρ and τ interactively to analyze both data characteristics.

For comparison, Figure 3 shows plots of the data using colored labels (bottom left), and enclosing convex hulls (bottom middle) and minimum bounding rectangles (bottom right). Neither of these plots is effective for analyzing the given data set. In the colored label plot, it is extremely difficult to discriminate the 45 different class labels or to asses the class distributions. In the other two plots, due to the size of the shapes, too much overlap prevents effective usage (cf. also 4 for a closeup.) The Butterfly plots, on the other hand, manage to effectively visualize the most important data characteristics: The number of classes, the distribution of class member points, and the relationship between the classes can easily be analyzed.

4.2 Pattern recognition data set

The second application is in the field of *pattern recognition*. The task under consideration is recognition and identification of postal stamps in images of mail pieces. Automatic stamp recognition is needed in postal sorting machines, e.g., for checking the correct franking value. The system described here is part of automatic mail sorting systems operating in several countries.

Part of the stamp recognition process is extraction of color features from possible stamp regions in the mail piece image, and matching these features to a set of known prototypes from a stamp image database. The color value of each pixel in a region is entered into a color histogram, then relative frequencies in each histogram bin are used as features. Therefore, the feature vectors have the same size as the histogram, which is typically a few hundred. For purposes of stamp image analysis and recognition performance tuning it is desirable to visualize these feature vectors with their respective class memberships. Besides the high dimensionality of the respective feature space, the large number of stamp classes poses additional visualization problems. Typically, the stamp image database contains some 100 to 500 stamp types (classes), each represented by up to 10 specimen.

The feature vectors of the samples of a given class should, in theory, be all identical. In practice, however, variations in illumination, image background and stamp region detection lead to significant deviations within the sample feature vectors. During performance tuning of the stamp recognition system, often the question arises why some sample images of a given class could not be recognized correctly but were rejected by the classifier. An analysis of such cases is often tedious and involves investigating feature and classifier properties.

To support this process, we designed a Butterfly plot-based visualization for analysis of the deviations between class prototypes and rejected samples. Figure 5 (top) displays the PCA-projected prototypes of 109 stamp classes in a Butterfly plot. Four different stamp classes of which samples were rejected by the classifier are highlighted by increased Butterfly color opacity. The rejected samples are marked by their true class la-

Input: 2D point set P, refinement threshold τ, recursion limit ρ
Output: Closed sequence of curve segments S

```
procedure butterfly(P, τ, ρ):
  S ← {}
  point c ← c(P) /* find center of gravity */
  polygon CH ← CH(P) /* find convex hull */
  /* calculate minimum area reduction required */
  area a ← τ * | CH |
  /* loop convex hull boundary lines and refine */
  for each line l ∈ CH boundary do
      S ← S + refine(P, l, c, 0, ρ, a)
  return S
```

Input: Point set P, line l, point c, recursion level r, recursion limit ρ, area a
Output: Set of connected curve segments S

```
procedure refine(P, l, c, r, ρ, a):
  control point cp ← getControlPoint(P, l, c)
  /* evaluate if area reduction threshold is met */
  boolean area_accept ← (|l ∪ curve(l.p1, cp, l.p2)| ≥ a)
  if ( r > 0 ∧ (¬area_accept ∨ r > ρ) ) then
      /* discard refinement step: area reduction of non-initial refine-
      ment step below threshold, or recursion limit exceeded */
      return {}
  if (cp = c) then
      return curve(l.p1, c, l.p2) /* P exhausted for this segment */
  /* recurse */
  S1 ← refine(P, (l.p1, cp), c, r+1, ρ, a)
  S2 ← refine(P, (cp, l.p2), c, r+1, ρ, a)
  if ( S1 = {} || S2 = {} ) then
      /* at least one sub-refinement failed */
      return curve(l.p1, cp, l.p2)
  return S1 + S2 /* sub-refinement successful */
```

Input: Set of points P, line l, point c
Output: Control point cp
$d()$: Distance between a point and a line

```
procedure getControlPoint(P, l, c)
  point set P_cand ← P ∩ △(l.p1, l.p2, c)
  /* remove line end points, keep center of gravity */
  P_cand ← P_cand \ (l.p1, l.p2)
  /* find nearest neighbor to l from P_cand */
  cp ← p ∈ P_cand | ∄q : q ∈ P_cand ∧ d(q, l) < d(p, l)
  return cp
```

Algorithm 1: Butterfly plot construction.

bels, and connected to their respective class centers using straight lines. The display allows the visual analysis of the problematic samples in context of the whole classifier data.

Interestingly, in this example data set we recognize that many of the displayed rejected samples have similar deviations from their class center. A view focusing on the rejected classes is shown in Figure 5 (bottom left). We learn that similar deviation patterns are most dominant in classes 22 and 88, while for classes 6 and 77, more diverse deviation patterns occur. We suspect systematic differences between the stamp prototypes used by the classifier, and the stamps seen on the rejected sample mail pieces. The reasons for these differences may be various, but the fact that there are systematic differences is very helpful and motivates further analysis. We have implemented interaction functionality which enables the user to perform drill-down analysis. E.g., we may choose to further inspect the lower bundle of rejected class 77 samples. To this end, the system can identify and highlight those prototype classes which (in the PCA-projection) interfere most with the mass of rejected samples. This is done for class 77 in Figure 5 (bottom right). Such drill-down analysis of the relationship between rejected samples and interfering neighboring classes in projected feature space is expected to be a useful tool in the process of classifier tuning.

5 EVALUATION AND DISCUSSION

Generation of Butterfly plots requires specification of the area reduction threshold τ and the recursion depth limit ρ. In experimenting with the data discussed in Section 4.1 useful parameters were found quickly, by testing just a small number of settings. Specifically, we found that $\rho \in \{0,1\}$ and $\tau \in [2\%,\ldots,10\%]$ gave useful results in that application. Generally, the choice of parameters will depend both on data and given task. For dense, mixed class distributions, lower area recovery thresholds and larger recursion limits produce leaner shapes, avoiding excessive overlap. For less crowded data sets, one can use higher thresholds.

In experiments we observed important shape metrics at different parameter settings. A series of plots was generated for the CAD database discussed in Section 4.1 by varying τ while allowing unlimited recursion depth. Figure 6 (top) shows the average size and overlap[1] of the Butterfly plots, relative to the corresponding convex hull plot. At $\tau = 0.00$, the Butterfly plot shows just about 7% (30%) of the area (overlap) of the corresponding convex hull plot. Increasing τ to 0.15 drives up relative area (overlap) of the Butterfly plot to 47% (55%). After that, the metrics increase slower, converging to 54% (63%) relative area (relative overlap) at $\tau = 0.30$. After that the metrics remain stable, as practically all candidate sub-refinement steps are rejected and only initial refinement steps are performed. We also observed the average number of curve segments, relative to the average number of convex hull segment lines (Figure 6 (bottom)). This metric can be interpreted as a measure of shape complexity. At $\tau = 0$, the Butterfly plots on average consist of 2.4 times the number of segments of their corresponding convex hull plots. Increasing τ quickly leads to a reduction in Butterfly plot complexity.

Similar results were obtained for the data set from Section 4.2. The analysis of these shape characteristics may serve for automatic determination of good parameter settings. Assume that shape complexity would be

[1] Size is measured by the number of pixels covered by a shape, and overlap is measured as the average number of shapes covering each non-empty pixel. Measures taken on a $1200 * 1200$ pixel display.

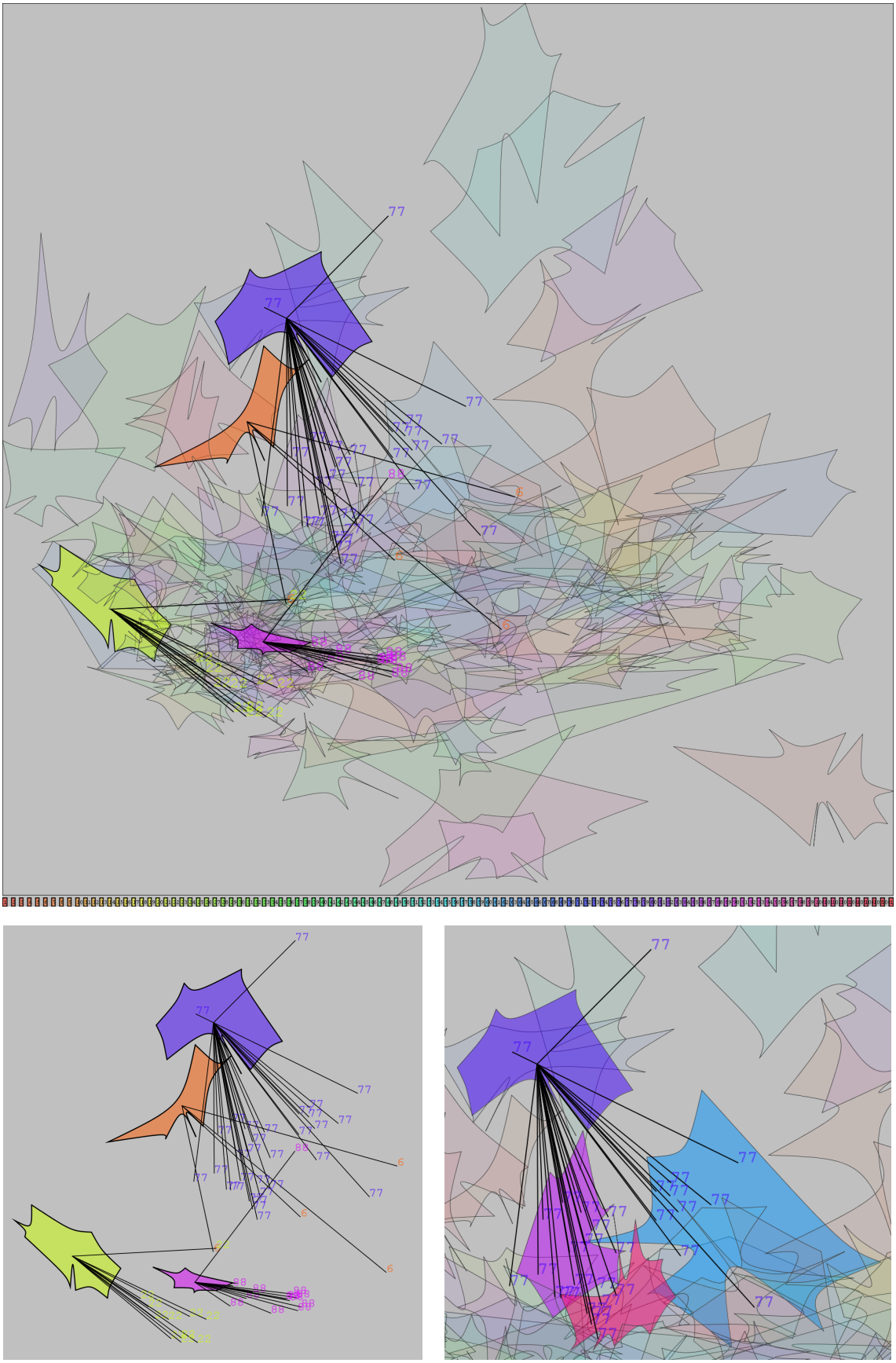


Figure 5: Visualizations based on the Butterfly plot, applied to a pattern recognition application.

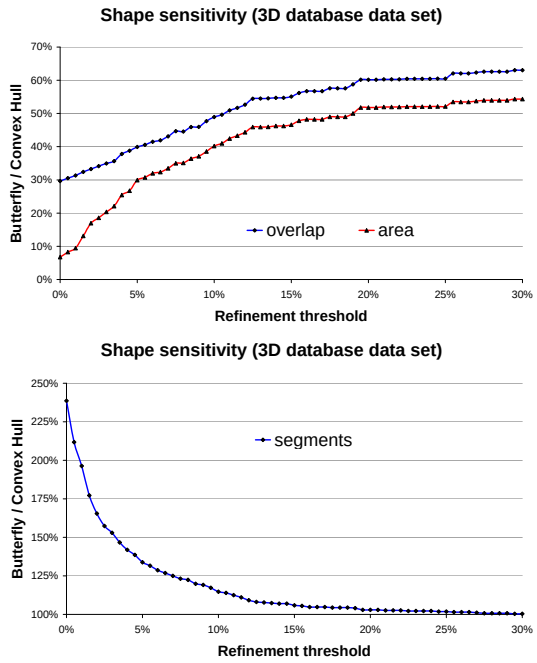


Figure 6: Evaluation of key Butterfly plot metrics.

the most important aspect to control in a given application. Then, by observing the shape complexity behavior, the system could apply the well-known elbow criterion to find $\tau = 0.05$ as a good parameter selection. It is also possible that the user specifies an overlap or shape complexity target which the system in turn uses to determine τ and ρ.

6 CONCLUSION

We addressed the problem of visualizing large point cloud data by abstraction to compact enclosing shapes. We introduced an algorithm that recursively refines the convex hull of a point cloud by curve segments, adapting to the given data. Two parameters allow to adjust the shape construction process to data and user requirements. By application and evaluation, it was shown that the generated shapes are an effective tool for visual analysis of large point cloud data.

One of the greatest challenges in the visualization of large point cloud data lies in optimizing the perception of shape discrimination, in presence of high degrees of overlap between the point clouds. In future work, usage of color for shape discrimination could be optimized, considering the perceptional and spatial proximity of shapes and color. Also, additional shape construction methods can be thought of, and their compactness and perceptional properties should be studied. We currently work on employing appropriate distance field representations for point cloud visualization in conjunction with advantageous spline operators, and already obtained promising first results.

A widely unsolved problem is the visual analysis support for extremely large point data sets. In certain classification applications, tens of thousands of point pairs may arise, each forming a separate class. As such data distributions pose challenges to any point-based visualization technique in general, we presume that in this case suitable data preprocessing is required. Also in this case, computational efficiency concerns may arise, a point which we did not address in this work. Finally, it would be a good idea to do user studies on the perceptional effects of different shapes in different user tasks. Results thereof could guide future work in constructing shapes for visual analysis of large point cloud data.

ACKNOWLEDGMENTS

We thank Tatiana Tekušová of Fraunhofer IGD for valuable suggestions and for discussion. We also thank the anonymous reviewers for their comments which helped to improve this work.

REFERENCES

[1] B. Bustos, D. Keim, D. Saupe, T. Schreck, and D. Vranic. An experimental effectiveness comparison of methods for 3D similarity search. *Int. Journal on Digital Libraries*, 6(1), 2006.

[2] J. Daniels, L. Ha, T. Ochotta, and C. Silva. Robust smooth feature extraction from point clouds. In *Proc. IEEE Shape Modeling International*, 2007.

[3] T. Dwyer and D. Gallagher. Visualising changes in fund manager holdings in two and a half-dimensions. *Information Visualization*, 3(4):227–244, 2004.

[4] Gerald E. Farin. Triangular bernstein-bézier patches. *Computer Aided Geometric Design*, 3(2):83–127, 1986.

[5] R. Gonzalez and R. Woods. *Digital Image Processing*. Prentice Hall, 3rd edition, 2007.

[6] R. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133, 1972.

[7] A. Herrmann and D. Keim. The gridfit algorithm: An efficient and effective approach to visualizing large amounts of spatial data. In *Proc. IEEE Visualization*, 1998.

[8] M. Hopf and T. Ertl. Hierarchical splatting of scattered data. In *Proc. IEEE Visualization*, 2003.

[9] S. Jayanti, Y. Kalyanaraman, N. Iyer, and K. Ramani. Developing an engineering shape benchmark for cad models. *Computer-Aided Design*, 38(9):939–953, 2006.

[10] I. Jolliffe. *Principal Components Analysis*. Springer, 3rd edition, 2002.

[11] I. Kyriazis, I. Fudos, and L. Palios. Detecting features from sliced point clouds. In *Int. Conference on Computer Graphics Theory and Applications*, 2007.

[12] C. Panse, M. Sips, D. Keim, and S. North. Visualization of geospatial point sets via global shape transformation and local pixel placement. *IEEE Transactions on Visualization and Computer Graphics*, 12, September-October 2006.

[13] J. Pu, Y. Kalyanaraman, S. Jayanti, K. Ramani, and Z. Pizlo. Navigation and discovery in 3d cad repositories. *IEEE Computer Graphics and Applications*, 27(4):38–47, 2007.

[14] P. Rousseeuw, I. Ruts, and J. Tukey. The bagplot: A bivariate boxplot. *The American Statistician*, 53(4):382–387, 1999.

[15] T. Schreck and C. Panse. A new metaphor for projection-based visual analysis and data exploration. In *IS&T/SPIE Conference on Visualization and Data Analysis*, 2007.

[16] C. Ware. *Information visualization: Perception for Design*. Morgan Kaufmann, 2000.

The influence of cross-modal interaction on perceived rendering quality thresholds

V.Hulusic
Warwick Digital Lab
University of Warwick
United Kingdom
V.Hulusic@warwick.ac.uk

M.Aranha
University of Bristol
United Kingdom
aranha@cs.bristol.ac.uk

A.Chalmers
Warwick Digital Lab
University of Warwick
United Kingdom
A.G.Chalmers@warwick.ac.uk

ABSTRACT

High-fidelity rendering is computationally demanding and has only recently become achievable at interactive frame rates on high-performance desktop PCs. Research on visual perception has demonstrated that parts of the scene that are not in the focus of viewer's attention may be rendered at much lower quality without this quality difference being perceived. It has also been shown that cross-modal interaction between visual and auditory stimuli can have a significant influence on perception. This paper investigates the limitations of the human visual system and the impact cross-modal interactions has on perceivable rendering thresholds. We show that by exploiting cross-modal interaction, significant savings in rendering quality and hence computational requirements can be achieved, while maintaining the same overall perceptual high quality of the resultant image.

Keywords: High-fidelity rendering, cross-modal, perceptual threshold.

1 INTRODUCTION

High-fidelity rendering is a process of computing accurate physically-based images using rendering algorithms based on real-world physical interaction. By taking into account complexity of the rendering process, e.g. simulation of photons' propagation in ray tracing algorithms, it is easy to infer that this process is very computationally demanding. Despite the improvements in performance of rendering related hardware and rendering algorithms, it is still not possible to render complex high-fidelity scenes at interactive rates.

Research on human vision and visual perception have demonstrated that the human visual system (HVS) has constraints which influence the way we perceive our environment [IKN98]. As a result of these limitations, it is common that high-fidelity rendered scenes have a greater level of details than it is possible to perceive. In order to quantify this so that it can be exploited within computer graphics, two phenomena were investigated: visual attention and inattentional blindness. It has been shown that by using these findings, the rendering process can been optimised to reduce the computational costs without perceivable degradation [CCW03] [YPG01].

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

Another factor when perceiving an environment is the affect of cross-modal interactions between visual and auditory stimuli [Mas06]. It has been demonstrated that this can be taken into consideration for improving the performance of rendering algorithms. Using sound it is possible to render animations at lower frame rates in conjunction with selective rendering techniques without decreasing the perceptible visual quality [MDCT05b] [MDCT05a].

In this paper we investigate the affect of cross-modal interaction on the perceived rendering threshold for high fidelity graphics involving an user study. In the following section, we shall present a short review of related work on perceptually-based rendering and cross-modal interaction. Section 3 talks about the stimuli and the experimental procedure and setup used in the user study. In Section 4 we analyse the results obtained from the experiment, comparing them with the results gained using Visual Difference Predictor (VDP) image comparison. Finally, we present our conclusions and suggest some ideas for future work.

2 RELATED WORK

Many researchers in the field of computer graphics have started to investigate the characteristics of HVS and the application of such features when developing rendering algorithms. Until now, the focus of this research has been visual attention, saliency and the application of these constraints for achieving the high-fidelity rendering in real time.

2.1 Visual Perception and Perceptually-Based Rendering

The way in which humans observe environments depends on mechanisms in the eye as well as visual pro-

cessing in the brain. An important characteristic of the human eye is the angular sensitivity. Only objects located in the centre of the gaze can be perceived in full details, while toward the periphery the ability to perceive detail decreases. This phenomena, labelled as an "internal spotlight", was introduced by James [Jam90] and further commented by Humphreys et al. [HB89]. This idea was further explored by Mack et al. [MR98], who formalised the concept of "Inattentional blindness", in which items unrelated to a task fail to be perceived despite falling under the gaze of the viewer. This was demonstrated within the Computer Graphics field by Cater et al. [CCL02]. These results have been later used for developing perceptually-guided selective rendering systems, using different types of human point-of-gaze prediction [CDS06] [CCW03]. In such a system the scene is rendered so that the viewer is attending to at the highest quality while the remainder of the scene is rendered in a much lower quality, and thus at a substantially reduced computational cost, without the viewer being aware of this quality difference. Additional related work can be found in [FPSG97] [MTAS01] [Mys02] [RPG99] [SFWG04].

Related to this is the work of Itti et al. [IKN98], who developed the model of saliency-based visual attention which uses the saliency map, a two-dimensional map based on colours, intensity and orientations to predict the likelihood of regions drawing attention of the viewer. This can be used in order to decrease rendering computational costs. Similar approaches are used by Yee et al. using error tolerance map (Aleph map) in dynamic environments [YPG01]. Sundstedt et al. combined saliency and task map, creating a new selective guidance system called "Importance map" [SDL+05]. An interesting approach using a "snapshot" image for generating a saliency map in real time was presented in [LDC05]. Luebke et al. [LH01] also presented a novel approach for reducing the model complexity using perceptual criteria.

Lately, some research has investigated perceived rendering thresholds for still images and animations, attempting to determine the limit at which further increase in rendering quality fails to become noticeable. Sundstedt et al. [SDC05] investigated perceived aliasing thresholds by altering the number of rays shot per pixel (rpp) on static scenes and animations. A similar study was conducted using small screen devices [ADCH06]. In both studies it has been demonstrated that it is possible to decrease the rendering quality and computational costs without a perceivable difference to an observer.

2.2 Cross-Modal Interaction

Despite the significant research into the limitation of the HVS within Computer Graphics, little work has been undertaken into the strong cross-modal interactions between visual and auditory stimuli. Research by Mastoropoulou et al. [MDCT05b] [MDCT05a] [Mas06] demonstrated that sound can influence the visual perception using knowledge from the psychology field in combination with selective rendering algorithms. This research inferred that by exploiting the presence of sound it is possible to render animations with lower frame rates without perceived difference in animation smoothness. It was also proven that observers are unable to perceive reductions in quality of animated sequences when selective rendering was used to render sound emitting objects (SEO) in higher quality than the rest of the scene.

Other research by Storms [L.98] and Winkler et al. [SC05] showed that using the high-quality audio stimuli increases the perceptual quality of video observed and that the quality of both of them contribute in the perceived quality, respectively.

There is currently even an EU project, CROSSMOD [CRO07] investigating the perception of audio and visual cross-modal interaction for real time rasterised graphics.

Although there is a growing body of work on the cross-modal effect of graphics and audio, no one, to the best of our knowledge, has investigated how the presence of audio may affect the perception of quality in a ray traced image.

3 EXPERIMENT

In order to quantify the perceived rendering threshold of high-fidelity images when sound was present we conducted a psychophysical study as well as analysing the perceptual quality of the results using the Visual Difference Predictor (VDP) [Dal93]. In our study we used the independent samples design. Our dependent variable was the perceived quality of the rendered images with the actual image quality and the audio background as the independent variables.

For the auditory stimuli three conditions were considered: no sound, related sound and noise. Every sound group consisted of 28 pairs of images, i.e. 7 images for each of 4 scenes. Every pair contained two images shown one after another in a five slide sequence, see Figure 1. All pairs were ordered randomly in order to avoid bias and reduce the affect of the human short-term memory (STM).

Figure 1: Example of the slide sequence from the experiment

Peterson et al., in their psychological studies [PP59] deduced that in presence of distractors, humans have problems with remembering even three elements for

Figure 2: Scenes used for the experiment (from left to right): Checkerboard, Corridor, Kalabsha, Library

more than eighteen seconds. Thus every picture in each pair was presented for 5 seconds, so that the whole sequence (Figure 1) lasted for about sixteen seconds.

3.1 Stimuli

For carrying out the research we rendered several 3D scenes which were used as visual stimuli and created audio samples for the accompanying sounds. To allow our work to be compared to previous work, we chose a sub group of scenes used by Sundstedt et al. [SDC05] and Aranha et al. [ADCH06], see Figure 2. Three scenes represented realistic environments and the fourth, the checkerboard scene, was used as a control scene because of its high spatial frequency.

For rendering we used the modified version of the *Radiance* `rpict` renderer [War94] developed by Debattista [Deb06]. This renderer was developed for selective, progressive and time-constrained rendering. The sampling algorithm is based on the hierarchical low-discrepancy (0,2) sampling sequence [KK02] composed of the Sobol and van der Corput sequences. Image reconstruction is performed using a Gaussian filter. This renderer is an improved version of the one used in the similar experiments used in [ADCH06, SDC05]. All scenes were rendered with 1, 4, 9, 16, 25, 36 and 49 rays per pixel (rpp), at a resolution of 1024 $\times$ 768 pixels. They were later converted to tif format using Radiance's `ra_tiff` command.

Using knowledge from previous studies by Mastoropolouou [Mas06], we decided to use related sound and noise, but not music. Sounds chosen for the experiment are presented in Table 1.

Checker board	Corridor	Kalabsha	Library
Footsteps	Background chatting	Sounds of nature	Office noises

Table 1: Related sounds used for the experiment

3.2 Visual Difference Predictor

In order to predict the probability of perceiving difference between the pair of images observed by a human, Daly developed the Visual Difference Predictor [Dal93]. This algorithm uses two images as input - the mask and a target image and compares them, producing a map of perceivable differences and values of detection probabilities (Figure 3). The VDP has some shortcomings: it can be used only for low dynamic range (LDR) images and it predicts the global level of adaptation to luminance. With significant progress in HDR applications and its general usage, Mantiuk et al. [MMS04] extended the original VDP algorithm to include comparison of HDR images taking into the account the local adaptation of the eye to every segment of the observed scene and the entire luminance spectrum visible to human eye.

Figure 3: VDP comparison. Top: mask image; Middle: target image; Bottom: difference map with probability of detection - green:0-50%; yellow: 50-75%; red:75-95%; pink:95-100%

Using the **HDR VDP**, we analysed an extensive range of 1024 $\times$ 748 images at varying rpp qualities. We chose a suitable gold-standard image beyond which there was no significant difference between images.

 ISBN 978-86943-15-2

According to this method, it was found that at 1024 × 768, 49 rpp was an ideal gold-standard.

3.3 Hardware and rendering time

All images were rendered on a PC with a Intel Core2Duo E6650 CPU at 2.33GHz, 2GB of DDR2 PC6400 RAM memory and GeForce 8800GTS with 640MB graphic card. Rendering times are given in Table 2.

The experiment was conducted using Intel Pentium 4 computer working on 3.00GHz, Compaq 1825 19" monitor with 1280 × 1024 pixels resolution and Sony MDR-V300 sound-insulating Dynamic headphones.

rpp	Checker board	Corridor	Kalabsha	Library
1	13.12	673.99	98.29	130.57
4	51.98	1431.31	170.00	513.40
9	115.81	2471.16	288.62	1153.65
16	206.89	4110.91	409.61	2086.31
25	329.79	5556.64	581.24	3354.27
36	464.08	7807.02	796.40	4627.24
49	812.92	10327.32	1057.14	6277.47

Table 2: Rendering times for all scenes presented in seconds

3.4 Procedure

48 volunteers (33 male and 15 female) aged from 18 to 63, with an average of 25, participated in our user study. Most of them (about 85%) were undergraduate or postgraduate students at the University of Warwick. Other subjects were employees at the same University. All participants reported normal or corrected to normal vision with no hearing impairments. All use computers in everyday work, and reported average familiarity with computer graphics. The subjects were naive about the purpose of the experiment and participated in only one of the randomly selected group (no-sound, related sound, unrelated sound).

Subjects were asked to compare the quality of the images presented in each pair and choose the one they thought contained the higher rendering quality - two alternative forced choice (2AFC). The display of the stimuli was controlled using a program we wrote to provide fixed display time, synchronised audio and collection of the participants decisions. Images were displayed with a black background in a completely dark and noise-isolated room. The observers distance from the monitor was fixed at 60cm with subjects from all three groups asked to put on headphones. The experiment lasted about 9 minutes.

Prior to the experiment, participants were shown the demonstration with two image pairs which were not used in the study, the first of which accompanied with a related sound, and the second without. Images were displayed in pairs of the same scene and auditory stimulus with varying rendering levels. After each pair, a question mark would appear (Figure 1), in order to cue the participant to press either button 1 or 2 on the keyboard relating to which image they believe to contain the higher rendering quality. The keyboard was situated within arms-length distance from the observer on the desk in front, so there was no need to move the position of the body or the hand for pressing the buttons. After an answer was provided, the following pair of images were shown. All results were stored in a text file generated by the program.

After the experiment all subjects were asked to say on which features the differences were most obvious and approximately how many of the pairs they perceived any difference (i.e. they were not guessing).

4 RESULTS

Based on the research of Sundstedt et al. [SDC05], Aranha et al. [ADCH06] and Mastoropolouou [Mas06] we hypothesised that it would be possible to further reduce the number of rays shot per pixel without perceivable degradation when using audio stimuli as a distractor during scene observation.

In order to analyse the findings we used both *statistical analysis* of the psychophysical study as well as *comparison using the VDP*.

4.1 Statistical Analysis of Psychophysical Experiment

The results of our psychophysical study were analysed using *Pearson's Chi-square test* with *Yates' correction* [Yat34]. The Chi-square is a non-parametric test, commonly used to produce the statistical confidence of a hypothesis. This test allowed us to verify whether there was a statistical significant preference for one of the images in a pair. Our null hypothesis for each pair of images is that they should have equal preference. Since we assume no bias, it is expected that the gold-standard self test (49 v 49 rpp) should deliver equal preference. Our computed Chi-square values averaged over all scenes can be seen in Figure 4 with complete results presented in the Appendix.

The results of the Chi-square were analysed with a probability bound greater than 0.05 for significant results, i.e. a less than 5% chance in such an occurrence being observed.

According to this probability measure, a perceived degradation in rendering quality in comparison to our gold standard occurs in most scenes at less than 4 rays-per-pixel for the *no sound* and *related sound* groups. In the third group where we used *noise* as an audio stimulus, we can see that the subjects were unable to perceive any difference even at the 1 rpp comparison for the Corridor and Kalabsha scenes. For the other two scenes the

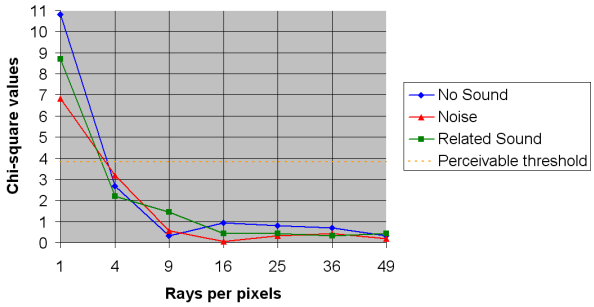


Figure 4: Comparison of chi-square thresholds averaged over all scenes

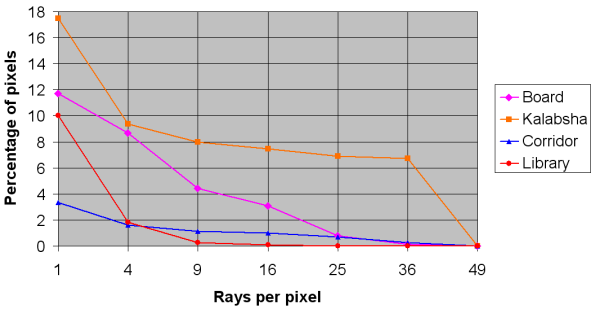


Figure 5: VDP comparison of perceivable differences for all scenes

perceived threshold was the same as in the *no sound* group.

The results were consistent across the majority of the scenes. As expected from the Checkerboard scene, the threshold was slightly greater than for the other scenes because of its high spatial frequency characteristics.

By comparing the results of the Chi-square across each of the three groups (no sound, related sound and noise), it can be seen that the *noise* threshold was lower than for the other conditions which confirmed our hypothesis. However, the *related sound* group contradicted our initial ideas and we actually experienced higher threshold level which may indicate that a related sound during the experiment caused subjects to look at the scene more closely.

It was found that in general the perceivable difference decreased monotonically with the increase in the number of rays-per-pixel.

4.2 Comparison using VDP

As discussed in Section 3.2, VDP can be used to highlight perceived differences as if viewed by the HVS. A constraint of VDP, however, is that it assumes significant viewing time. Our comparison between the psychophysical study and the VDP analysis allowed us to verify any difference which may have resulted from the finite viewing time.

In order to verify the results of our psychophysical study, we compared our gold-standard (49 rpp) with all other images of the same scene. The results of the VDP error measurements are included on Figure 5. These results were significantly different to what we experienced from our statistical analysis.

The results of our VDP comparison of the Kalabsha scene indicated significant variance between pairs of images, however, from our Chi-square results little statistical preference between images for this scene occurred. The differences between images in this scene are likely to be the result of texturing method of the model and the sampling strategy used by the renderer. This difference was picked up by VDP possibly due to HDR VDP's increased conservativeness, as described in the research by Ramanarayanan et al. [RFWB07]. For other scenes the perceptual threshold was significantly higher, in some cases more than ten times. We can also see from the Figure 5 that the difference was more likely to be perceived for the Checkerboard scene, as it was the case in our psychophysical experiment.

From post-experiment questioning of participants, approximately half of the subjects reported noticeable artefacts occurring on the edges of objects. This is verified by the VDP maps produced as seen in Figure 3. These regions are a consequence from the reduced sampling caused by tracing fewer rays in regions with high spatial variation.

5 CONCLUSION AND FUTURE WORK

Our experimental analysis indicates that cross-modal interaction has a significant affect on perceived thresholds. The presence of unrelated sounds, in our case noise, acts as a distractor as expected, producing a reduction in the thresholds. However, it was found that in the case of related sounds, a greater threshold was discovered which may indicate a heightened awareness of differences between rendering qualities.

We can conclude that for scenes with unrelated sounds, we can exploit such accompanying audio by ray tracing a lower number of rays without noticeable degradation resulting in reduced rendering times. Although our research indicates that related sounds may require an increase in rendering quality when compared to a no-sound scenario, we will investigate the manner in which this affects user's attention as a part of our future work. It may be possible that the focus is concentrated at specific regions, in which case it may still be possible to reduce rendering costs by using selective rendering techniques. As with any such statistical study, the validity of the result is dependent upon the sample size.

In the future we will further explore our findings by using a greater granularity of rpp levels around our discovered thresholds and a higher quality of the accompanying audio. We also intend to investigate the affect of cross-modal interaction on animations. The results

of this research will form the basis of a cross-modally aware rendering framework.

ACKNOWLEDGEMENTS

Thanks to Kurt Debattista for the use of his renderer and some general feedback on this work and Cheng-Hung Lo for his advice on experimental design and analysis. The use of the Checkboard, Corridor and Kalabsha was with the permission of the University of Bristol. The Library scene was created by Peter Apian-Bennewitz. We would also like to thank all the volunteers that participated in the experiment.

REFERENCES

[ADCH06] Matt Aranha, Kurt Debattista, Alan Chalmers, and Steve Hill. Perceived rendering thresholds for high-fidelity graphics on small screen devices. In *Theory and Practice of Computer Graphics 2006*, pages 133–140. EG, July 2006.

[CCL02] Kirsten Cater, Alan Chalmers, and Patrick Ledda. Selective quality rendering by exploiting human inattentional blindness: Looking but not seeing. In *Symposium on Virtual Reality Software and Technology 2002*, pages 17–24. ACM, November 2002.

[CCW03] K. Cater, A. Chalmers, and G. Ward. Detail to attention: exploiting visual tasks for selective rendering. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 270–280, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[CDS06] Alan Chalmers, Kurt Debattista, and Luis Paulo Santos. Selective rendering: computing only what you see. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 9–18, New York, NY, USA, 2006. ACM Press.

[CRO07] CROSSMOD. Cross-modal perceptual interaction and rendering. World Wide Web electronic publication, 2007. `http://www-sop.inria.fr/reves/CrossmodPublic/`.

[Dal93] Scott Daly. *The visible differences predictor: an algorithm for the assessment of image fidelity*, pages 179–206. MIT Press, Cambridge, MA, USA, 1993.

[Deb06] Kurt Debattista. *Selective Rendering for High Fidelity Graphics*. PhD in Computer science, University of Bristol, 2006.

[FPSG97] James A. Ferwerda, Sumanta N. Pattanaik, Peter Shirley, and Donald P. Greenberg. A model of visual masking for computer graphics. *Computer Graphics*, 31(Annual Conference Series):143–152, 1997.

[HB89] Glyn W. Humphreys and Vicki Bruce. *Visual Cognition: Computational, Experimental and Neuropsychological Perspectives*. Lawrence Erlbaum Associates Ltd, East Sussex, BN3 2FA, UK, 1989.

[IKN98] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(11):1254–1259, 1998.

[Jam90] William James. *The principles of psychology*. Holt, New York, 1890.

[KK02] T. Kollig and A. Keller. Efficient multidimensional sampling. *Computer Graphics Forum*, 21(3):557–563, September 2002.

[L.98] Storms R. L. *Auditory-Visual Cross-Modal Perception Phenomena*. PhD thesis, Naval Postgraduate School, 1998.

[LDC05] Peter Longhurst, Kurt Debattista, and Alan Chalmers. Snapshot: A rapid technique for driving a selective global illumination renderer. In *WSCG 2005*, February 2005.

[LH01] David P. Luebke and Benjamin Hallen. Perceptually-driven simplification for interactive rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 223–234, London, UK, 2001. Springer-Verlag.

[Mas06] Georgia Mastoropoulou. *The Effect of Audio on the Visual Perception of High-Fidelity Animated 3D Computer Graphics*. PhD in Computer science, University of Bristol, 2006.

[MDCT05a] Georgia Mastoropoulou, Kurt Debattista, Alan Chalmers, and Tom Troscianko. Auditory bias of visual attention for perceptually-guided selective rendering of animations. In *GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 363–369, New York, NY, USA, 2005. ACM Press.

[MDCT05b] Georgia Mastoropoulou, Kurt Debattista, Alan Chalmers, and Tom Troscianko. The influence of sound effects on the perceived smoothness of rendered anima-

tions. In *APGV '05: Proceedings of the 2nd symposium on Applied perception in graphics and visualization*, pages 9–15, New York, NY, USA, 2005. ACM Press.

[MMS04] Rafal Mantiuk, Karol Myszkowski, and Hans-Peter Seidel. Visible difference predictor for high dynamic range images. In Wil Thissen, Peter Wieringa, Maja Pantic, and Marcel Ludema, editors, *2004 IEEE International Conference on Systems, Man & Cybernetics*, volume 3, pages 2763–2769, Hague, The Netherlands, October 2004. IEEE.

[MR98] Arien Mack and Irvin Rock. *Inattentional Blindness*. The MIT Press, 1998.

[MTAS01] Karol Myszkowski, Takehiro Tawara, Hiroyuki Akamine, and Hans-Peter Seidel. Perception-guided global illumination solution for animation rendering. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 221–230. ACM Press / ACM SIGGRAPH, 2001.

[Mys02] Karol Myszkowski. Perception-based global illumination, rendering, and animation techniques. In *SIGGRAPH 2002, Proceedings of the 18th spring conference on Computer graphics*, pages 13–24. ACM Press / ACM SIGGRAPH, 2002.

[PP59] L. R. Peterson and M. J. Peterson. Short-term memory retention of individual items. *Journal of Experimental Psychology*, 58:193–198, 1959.

[RFWB07] Ganesh Ramanarayanan, James Ferwerda, Bruce Walter, and Kavita Bala. Visual equivalence: towards a new standard for image fidelity. *ACM Trans. Graph.*, 26(3):76, 2007.

[RPG99] Mahesh Ramasubramanian, Sumanta N. Pattanaik, and Donald P. Greenberg. A perceptually based physical error metric for realistic image synthesis. In Alyn Rockwood, editor, *Siggraph 1999, Computer Graphics Proceedings*, pages 73–82, Los Angeles, 1999. Addison Wesley Longman.

[SC05] Winkler S. and Faller C. Audiovisual quality evaluation of low-bitrate video. *SPIE/IS&T Human Vision and Electronic Imaging*, 5666:139–148, 2005.

[SDC05] Veronica Sundstedt, Kurt Debattista, and Alan Chalmers. Perceived aliasing thresholds in high-fidelity rendering. In *APGV 2005 - Second Symposium on Applied Perception in Graphics and Visualization (poster)*. ACM, August 2005.

[SDL+05] V. Sundstedt, K. Debattista, P. Longhurst, A. Chalmers, and T. Troscianko. Visual attention for efficient high-fidelity graphics. In *SCCG '05: Proceedings of the 21st spring conference on Computer graphics*, pages 169–175, New York, NY, USA, 2005. ACM Press.

[SFWG04] William A. Stokes, James A. Ferwerda, Bruce Walter, and Donald P. Greenberg. Perceptual illumination components: a new approach to efficient, high quality global illumination rendering. *ACM Trans. Graph.*, 23(3):742–749, 2004.

[War94] Gregory J. Ward. The radiance lighting simulation and rendering system. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 459–472, New York, NY, USA, 1994. ACM Press.

[Yat34] Frank Yates. Contingency table involving small numbers and the x^2 test. In *Journal of the Royal Statistical Society (Supplement) 1*, pages 217–235. Blackwell Publishing, 1934.

[YPG01] Hector Yee, Sumanita Pattanaik, and Donald P. Greenberg. Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. *ACM Trans. Graph.*, 20(1):39–65, 2001.

APPENDIX

	Checkerboard		Corridor		Kalabsha		Library	
No. of Rays	x^2	p	x^2	p	x^2	p	x^2	p
1	**7.562**	**0.0059**	**14.062**	**0.0001**	**7.562**	**0.0059**	**14.062**	**0.0001**
4	**7.562**	**0.0059**	0.062	0.8033	0.062	0.8033	3.062	0.0801
9	0.062	0.8033	0.562	0.4534	0.062	0.8033	0.562	0.4534
16	0.062	0.8033	0.562	0.4534	0.062	0.8033	3.062	0.0801
25	0.562	0.4534	0.562	0.4534	0.562	0.4534	1.562	0.2113
36	1.562	0.2113	0.062	0.8033	0.562	0.4534	0.562	0.4534
49	0.062	0.8033	0.062	0.8033	0.562	0.4534	0.562	0.4534

Table 3: "No sound" group: Chi-Square Analysis (df=1; critical value 3.841 at 0.05 level of significance). Significant results in bold.

	Checkerboard		Corridor		Kalabsha		Library	
No. of Rays	x^2	p	x^2	p	x^2	p	x^2	p
1	**10.562**	**0.0011**	3.062	0.0801	3.062	0.0801	**10.562**	**0.0011**
4	**7.562**	**0.0059**	0.562	0.4534	1.562	0.2113	3.062	0.0801
9	1.562	0.2113	0.562	0.4534	0.062	0.8033	0.062	0.8033
16	0.062	0.8033	0.062	0.8033	0.062	0.8033	0.062	0.8033
25	0.062	0.8033	0.562	0.4534	0.562	0.4534	0.062	0.8033
36	0.062	0.8033	0.062	0.8033	1.562	0.2113	0.062	0.8033
49	0.062	0.8033	0.562	0.4534	0.062	0.8033	0.062	0.8033

Table 4: "Noise" group: Chi-Square Analysis (df=1; critical value 3.841 at 0.05 level of significance). Significant results in bold. Different noise sounds: white, brown and pink noises were used in order to decrease the boredom and to avoid the bias as a consequence of the familiarization with the sound.

	Checkerboard		Corridor		Kalabsha		Library	
No. of Rays	x^2	p	x^2	p	x^2	p	x^2	p
1	**14.062**	**0.0001**	**10.562**	**0.0011**	**5.062**	**0.0244**	**5.062**	**0.0244**
4	1.562	0.2133	0.562	0.4534	1.562	0.2133	**5.062**	**0.0124**
9	**5.062**	**0.0244**	0.062	0.8033	0.562	0.4534	0.062	0.8033
16	1.562	0.2133	0.062	0.8033	0.062	0.8033	0.062	0.8033
25	0.062	0.8033	0.062	0.8033	0.062	0.8033	1.562	0.2133
36	0.062	0.8033	0.562	0.4534	0.062	0.8033	0.562	0.4534
49	0.062	0.8033	1.562	0.2133	0.062	0.8033	0.062	0.8033

Table 5: "Related sound" group: Chi-Square Analysis (df=1; critical value 3.841 at 0.05 level of significance). Significant results in bold. Results show that the subjects were looking more closely, and were able to find more differences.

Particle–based T-Spline Level Set Evolution for 3D object reconstruction with Range and Volume Constraints

Robert Feichtinger
Johannes Kepler University Linz, Austria
robert.feichtinger@jku.at

Huaiping Yang
Johannes Kepler University Linz, Austria
yang.huaiping@jku.at

Bert Jüttler
Johannes Kepler University Linz, Austria
bert.juettler@jku.at

ABSTRACT

We consider an evolution process for implicitly defined surfaces, which are represented as the zero–levels of T-spline functions. The paper presents two novel contributions. First, we will use particles on the evolving surface in order to discretize the evolution equation. In particular we describe criteria for local and global resampling, which are needed in order to maintain a sufficiently uniform distribution of the particles. Second, we discuss volume and range constraints which can be added to the framework. More precisely, it is possible to specify a fixed volume (volume constraint) or to define a region which should or should not be contained in the final object (range constraint). These constraints can also be regarded as a priori knowledge of the data.

Keywords: T-splines, particles, range constraint, volume constraints

1 INTRODUCTION

This paper addresses the problem of evolution of implicitly defined surfaces, with applications to geometry reconstruction from discrete data. The evolution process generates time-dependent families of surfaces converging to the target shape defined by the data, which are guided by an implicit velocity field in the direction of surface normals. For example this kind of evolution is used for segmentation in image processing. Kass et al. [16] proposed "snakes" or active contours for boundary detection, which is based on deforming an initial contour towards the boundary to be detected. Caselles et al. [5] proved that the classic active contour model in 2D is equivalent to finding a geodesic curve in a Riemannian space with a metric derived from the image content. For implicitly defined surfaces, one may formulate evolution processes by using the level set approach of Osher and Sethian [19]. Similar evolution processes have also been used for geometry reconstruction from unorganized data points [18, 28].

Surface reconstruction from scattered data points (possibly noisy and incomplete) is an important problem in geometric modeling. Depending on the type of the application, different representations have been used, such as triangular meshes [7, 17], subdivision surfaces [6, 22], parametric spline surfaces [8], discretized level sets [18], scalar spline functions [14, 23, 28], radial basis functions [4], and point set surfaces [1, 2, 13, 20, 21]. In comparison with parametric representations, implicit representations offer advantages such as the non-existence of the parametrization problem, repairing capabilities of incomplete data, and simple operations of shape editing [29]. As a major advantage in evolution-based approaches, where the topology is not known a priori, the implicit representation intrinsically adapts to topological changes during the evolution.

In many applications of surface reconstruction, there may be a priori knowledge concerning the geometric properties of the object to be reconstructed. In particular, this a priori knowledge could be formulated by certain shape constraints to be combined into the evolution equation of the surface. These additional constraints may help to obtain a desired reconstruction result, especially when the given data contains holes. For example, a suitable volume constraint can be used to stop the evolving surface from entering the holes left by the data. The idea of volume constraint has also been used in shape deformation to achieve better quality and realism, e.g. the swirling-sweepers proposed by Angelidis et al. [3] and the divergence-free vector-field based method by Funck et al. [25, 26].

Our method is different from those in that we use zero contour scalar functions instead of meshes to represent surfaces, where the volume constraint is formulated with the help of the time derivatives of the coefficients. Another constraint to be addressed is the so-called range constraint [9, 10], which allows us to specify regions lying inside or outside of the reconstructed surface.

Particles have been used for different applications in mathematics and computer graphics. One can use them for numerically solving differential equations or to sample and to control implicitly defined surfaces [27].

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'2008, February 4 – February 7, 2008 .
Copyright UNION Agency – Science Press, Plzen, Czech Republic

They have also found attention for simulation of fur, grass and other fuzzy textures, also special effects like fire and explosions are created by particles in this applications collision detection is important [15].

In this paper we use particles in order to efficiently deal with the generation of sample points on evolving surfaces. Special attention is paid to local and global resampling, which is needed in order to maintain a sufficiently uniform distribution of the sample points. In addition we show how to apply constraints during the evolution process. These constraints represent a priori knowledge of the data. The user can specify a fixed volume (volume constraint) or a region which should not lie inside the final object (range constraint).

The paper is organized as follows. In the next section we start with a description of the T-spline level set evolution. Afterwards we go into more detail for the particle generation and handling of particle resampling. In section 4 we treat range and volume constraints. Finally we conclude with some examples.

2 EVOLUTION

First we recall the idea of the evolution of an implicitly defined surface. Such a surface is also denoted as an active surface. In particular we consider implicitly defined surfaces represented as the zero–level set of a T-spline function.

2.1 Problem specification

Throughout the paper we assume that we have some given data, which describes one or more objects in 3D. This data can be given in various forms, e.g. as a triangular mesh, an implicitly or parametrically defined surface or an unorganized point cloud. Here we will focus on point clouds (e.g. data from a 3D scanner). The given data will be referred to as the "target".

In order to detect the topology of the target and to reconstruct it, we consider an evolution process which drives the surface towards the target. More precisely we will use a time dependent family of surfaces $C = C_t$. In our case each surface from this family will be defined as the zero–level set of a scalar field, whose coefficients depend on the time variable τ. During the evolution we will modify these coefficients such that the surface moves towards the given data, see Fig. 1. In particular we want the zero–level set to move in the direction of its normals in dependence of a speed function v.

2.2 Speed functions

As mentioned before the evolution of the active surface will be guided by a speed (or velocity) function v. This function depends on the target, the implicitly defined surface and some geometric properties (curvature κ and normals $\vec{\mathbf{n}}$) of the active surface.

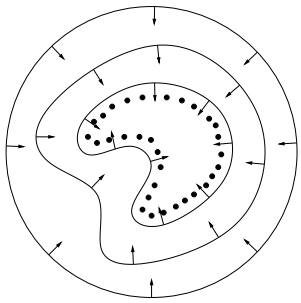


Figure 1: 3 time steps of an active curve moving towards a point cloud data

We use the following speed function for point cloud data

$$v = e(d)\,(\lambda + \kappa) - (1 - e(d))\,(\vec{\mathbf{n}}^T \nabla d), \qquad (1)$$

where e is called the edge detector function,

$$e(d) = 1 - e^{-\eta\, d^2}. \qquad (2)$$

In this function $\vec{\mathbf{n}}$ and κ are geometric quantities derived from the surface, while λ is a constant velocity (also known as the balloon force). η is a pre-described constant which depends on the range of the data. For additional information regarding the choice of these constants we refer to the extended version of [28]. The last variable d is the unsigned distance function which depends on the target.

Both the unsigned distance field and the edge detector functions will be pre–computed. We use graphics hardware acceleration [12] to determine the unsigned distance function $d(\mathbf{x})$. The gradient $\nabla d(\mathbf{x})$ can be efficiently acquired by trilinear interpolation of the neighbouring grid points.

2.3 Evolution of T-spline level sets

We use the zero–level set of a T-spline function (see [24]) to represent the implicitly defined surface. Such a function has the form

$$f(\mathbf{x}, \tau) = \sum_{i=1}^{n} T_i(\mathbf{x})\, c_i(\tau) \;\; \mathbf{x} = (x_1, x_2, x_3) \in \Omega \subset \mathbb{R}^3, \quad (3)$$

where T_i are the trivariate T-spline basis functions and the real coefficients c_i depend on the time variable τ. The axis–aligned bounding box $\Omega = [-1, 1]^3$ contains the region of interest. The given data is scaled such that it lies inside Ω. The basis functions

$$T_i(\mathbf{x}) = \frac{B^3_{r_i}(x_1) B^3_{s_i}(x_2) B^3_{t_i}(x_3)}{\sum_{i=1}^{n} B^3_{r_i}(x_1) B^3_{s_i}(x_2) B^3_{t_i}(x_3)}$$

are rational functions which are defined with cubic B-splines over certain knot vectors $r_i = (r_{i0}, r_{i1}, r_{i2}, r_{i3}, r_{i4})$, s_i and t_i, which are determined with the help of the so–called T-spline

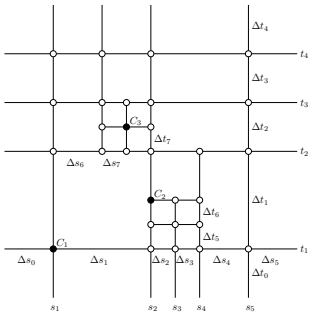


Figure 2: A T-spline grid. We use 4–fold knots at boundaries.

control points	x-knots y-knots
C_1	$[s_1-\Delta s_0, s_1-\Delta s_0, s_1, s_2, s_3]$ $[t_1-\Delta t_0, t_1-\Delta t_0, t_1, t_2, t_3]$
C_2	$[s_1-\Delta s_0, s_1, s_2, s_3, s_4]$ $[t_1, t_1+\Delta t_5, t_1+\Delta t_5+\Delta t_6, t_2, t_2+\Delta t_7]$
C_3	$[s_1, s_1+\Delta s_6, s_1+\Delta s_6+\Delta s_7, s_2, s_5]$ $[t_1, t_2, t_2+\Delta t_7, t_3, t_4]$

Figure 3: The knot vectors for some selected control points.

grid. This is a generalization of the knot vectors of tensor–product splines. See Fig. 2 and Fig. 3 for an illustration where we use a 2D graphic which is more convenient to illustrate the idea of T-splines. See [24] for more information.

As the main advantage of using T-splines, they support T-junctions in the grid. Therefore we can refine the grid locally which is not the case for tensor product B-splines. If there are no T-junctions in the grid, then the T-spline simplifies to a tensor–product spline.

The zero–level set of such a T-spline function f defines a time–dependent surface.

$$\Gamma(f,\tau) = \{\mathbf{x} \in \Omega \subset \mathbb{R}^3 \mid f(\mathbf{x},\tau) = 0\}. \tag{4}$$

To describe the evolution we use the approach presented in [28]. We consider $\Gamma(f,\tau)$ the zero–level set of the T-spline f at a given time τ. It will be subject to the evolution process

$$\frac{\partial \mathbf{x}}{\partial \tau} = v\vec{\mathbf{n}}, \;\; \mathbf{x} \in \Gamma(f,\tau). \tag{5}$$

Here v is the speed function which was introduced in Section 2.2. The value of v depends on the point $\mathbf{x} \in \Gamma$ and on the first and second derivative of the T-spline f at $\mathbf{x}$. $\vec{\mathbf{n}}$ is the unit normal direction given by $\vec{\mathbf{n}} = \nabla f(\mathbf{x},\tau)/|\nabla f(\mathbf{x},\tau)|$. Since

$$f(\mathbf{x},\tau) = 0, \;\; \mathbf{x} \in \Gamma(f,\tau), \tag{6}$$

has to hold during the evolution, it follows that

$$\dot{f} + \nabla f(\mathbf{x},\tau)\,\frac{\partial \mathbf{x}}{\partial \tau} = 0, \;\; \mathbf{x} \in \Gamma(f,\tau), \tag{7}$$

where $\dot{f} = \frac{\partial f(\mathbf{x},\tau)}{\partial \tau}$. Combining (5) and (7) we get the following equation for the evolution of the zero–level set

$$\frac{\partial f(\mathbf{x},\tau)}{\partial \tau} = -v|\nabla f(\mathbf{x},\tau)|, \;\; \mathbf{x} \in \Gamma(f,\tau). \tag{8}$$

To translate this equation into an equation for the time dependent coefficients $c_i(\tau)$ we use a least-squares approach

$$E_0(\dot{\mathbf{c}}) = \int_{\mathbf{x}\in\Gamma(f)} (\dot{f}(\mathbf{x},\tau) + v\,|\nabla f(\mathbf{x},\tau)|)^2 \,\mathrm{d}A \to \min. \tag{9}$$

where A represents the area element of the T-spline level set, and $\mathbf{c} = (c_1,\ldots,c_n)$. In order to solve this equation we use numerical integration to get a discretized version

$$E(\dot{\mathbf{c}}) = \sum_{j=1}^{N_0} (\dot{f}(\mathbf{x}_j,\tau) + v(\mathbf{x}_j,\tau)\,\left|\nabla f(\mathbf{x}_j,\tau)\right|)^2 \to \min. \tag{10}$$

Here $\mathbf{x}_j,\ j = 1\ldots N_0$ are sample points on the active surface. In the present work we will use particles, instead of considering uniformly distributed sample points as in [28]. This sampling method will be described in the following section.

Before we continue with the description of the particle generation we summarize the algorithm used for the evolution process.

Algorithm 1

1. *Initialization:* Pre-compute the evolution speed function and choose the initial position of the implicitly defined surface.
2. *Initial particles:* Compute a set of particles on the initial active surface.
3. *Evolution:* Apply one time step of the evolution to the implicitly defined surface.
4. *Projection:* Project the particles in normal direction onto the new implicitly defined surface.
5. *Consistency check:* Check if a redistribution of the particles is necessary.
6. *Termination:* Check whether the stopping criterion is satisfied. Continue with step 3 (no) or finish (yes).

We choose the initial implicitly defined surface as a ball (Step 1) such that all data points lie inside. We use uniformly distributed points on this ball (Step 2) as a starting set of particles. To generate this set we chose a random point close to the ball and use the method described in section 3.4.

For step 3, we have to solve the Eqn. (10). We first linearize the quadratic problem for the $\dot{c}_i$ and then use an explicit Euler step to compute the new coefficients c_i.

Note that we use the distance field constraint to avoid the reinitialization of the T-spline function, see [28].

3 PARTICLES

For the evolution of an implicit level set surface we need sample points on the current zero–level set. It is necessary to have efficient techniques for the computation and the update of the sample points in each time step.

In our previous work [28], these sample points are uniformly distributed and computed by using a fine grid. So one has to check in which grid cells the zero–level set lies after each step, which is rather time consuming.

We will use particles which do not need to be recomputed each time step. Instead they move with the active surface. This movement is achieved by using a projection along the normal direction. During the evolution it can happen that the distance between two neighbouring particles gets too big. Therefore we need to define a criterion when a redistribution is necessary. In most situations the redistribution is done locally instead of recomputing all particles.

Since we also want to gather information about the number of components in the target, we need a way to generate one set of particles per component. This is achieved by splitting the set of particles according to the current state of the implicitly defined surface. So when the topology of the surface changes during the evolution, this change should be reflected in the particles.

3.1 Particles data structure

The data structure used for the particles is similar to the data structure for a triangular mesh. For each particle we store the coordinates and a list pointing to the neighbouring particles, but we do not use the face information. These particles are then used as sample points for the T-spline level set evolution.

To compute this triangulation we use Hartmann's algorithm for marching triangulation [11]. This algorithm allows us to compute a triangulation either for a whole object or for a bounded domain on a surface.

To apply Hartmann's algorithm we have to provide the possibility to compute or at least to estimate the normal direction at any point of the surface, which is of course possible for the zero–level set of a T-spline function f. The unit normals are given by $\vec{\mathbf{n}} = \nabla f / |\nabla f|$.

As an initial set we take the vertices of an uniformly distributed triangulation of the sphere, since the initial level set is approximately a sphere. The distance between these points is denoted by ρ. We refer to ρ as the feature size.

3.2 Criteria for particle resampling

During the evolution we use two criteria to decide if a resampling is necessary. Afterwards we check if the resampling can be done locally or globally. See section 3.3 and 3.4 for the resampling methods.

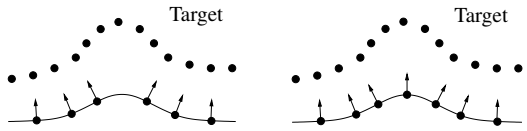


Figure 4: Target with a concave feature and an active curve with particles

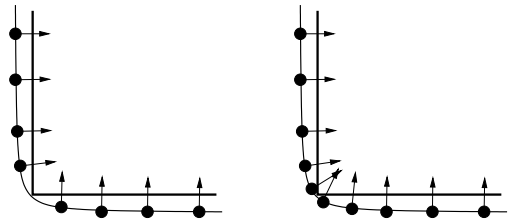


Figure 5: Target with a sharp feature

The marching triangulation can be applied either to a whole object (global resampling) or to a bounded domain (local resampling).

We use the following criteria:

1. The distance between two neighbouring sample points P_i and P_j becomes too large. This means we get areas with too few sample points. As a criterion we use
$$\left|P_i - P_j\right| > 2\rho \tag{11}$$
For example this occurs if the target contains a concave region. We will demonstrate this in a 2D example, see Fig. 4. On the left-hand side, the distance between the two particles in the center is growing. On the right-hand side, the result after resampling is shown. The arrows indicate the normal direction at the particles.

2. The normals of neighbouring points vary too much. Here the criterion can be written as
$$\mathbf{n}_i \cdot \mathbf{n}_j < \varepsilon \tag{12}$$
where $\mathbf{n}_i$ is the unit normal of the implicitly defined surface at the sample point P_i. ε is chosen such that too big angles are detected e.g. to filter out angles greater than $80°$. This criterion helps to produce additional sample points at sharp corners and therefore to increase the accuracy. For an example see Fig. 5. Again the left-hand side shows the particles before resampling and the normal directions. The right-hand side displays the result after resampling with a modified step size.

 ISBN 978-86943-15-2

If one of the two criteria is satisfied we first use a local resampling. In case that criterion 2 holds we additionally modify the step size for the marching triangulation, to ensure that we produce additional sample points in the region around the critical points. This gives us a new set of particles S_1. Afterwards we check if a global resampling is necessary as well. Therefore we take the old particles $\widehat{S}$ and test if they are contained in the new generated set of particles or at least close enough to this set,

$$\forall_{P_i \in \widehat{S}} \exists_{P_j \in S_1} \left| P_i - P_j \right| \leq \rho. \tag{13}$$

The global resampling is used only when the implicitly defined surface has split during the evolution step, where the condition (13) is not fulfilled. Most of the time we will just use the local method. Note that holes which occur during the evolution in the implicitly defined surface are also handled by the local method.

We summarize the algorithm for particle resampling.

Algorithm 2

1. Apply the two criteria to test if resampling is necessary. If none of them is satisfied, then stop.
2. Apply local resampling.
3. Test if the old particles are close enough to the newly generated ones. If one of them is too far away, then apply global resampling; a splitting event may have happened.

3.3 Local resampling

We start with two sample points P_i and P_j, which are neighbours in the old set of particles. These points have been detected by one of the two previously described criteria. First we create a local region Ω. This region contains P_i, P_j and the neighbouring sample points (Note that this information is stored in our data structure). Then we check, if additional sample points lie inside this region. If this is the case add these points. Afterwards we apply the marching triangulation to Ω.

Note that the two criteria (11) and (12) might be satisfied for more than one pair of sample points. If we get more than one pair of sample points we create the local region Ω for one pair. For the remaining pairs we check whether they are separated from already generated regions or not. In the first case we create a new region $\widehat{\Omega}$ in the second case we have to extend the already existing region. These two cases are demonstrated in the following example.

Example 1 We use data taken from a torus. The top left picture in Fig. 6 shows the sample points on the initial surface. The top right picture visualizes the corresponding T-spline grid. The second row shows the first resampling, which involves two disjoint regions.

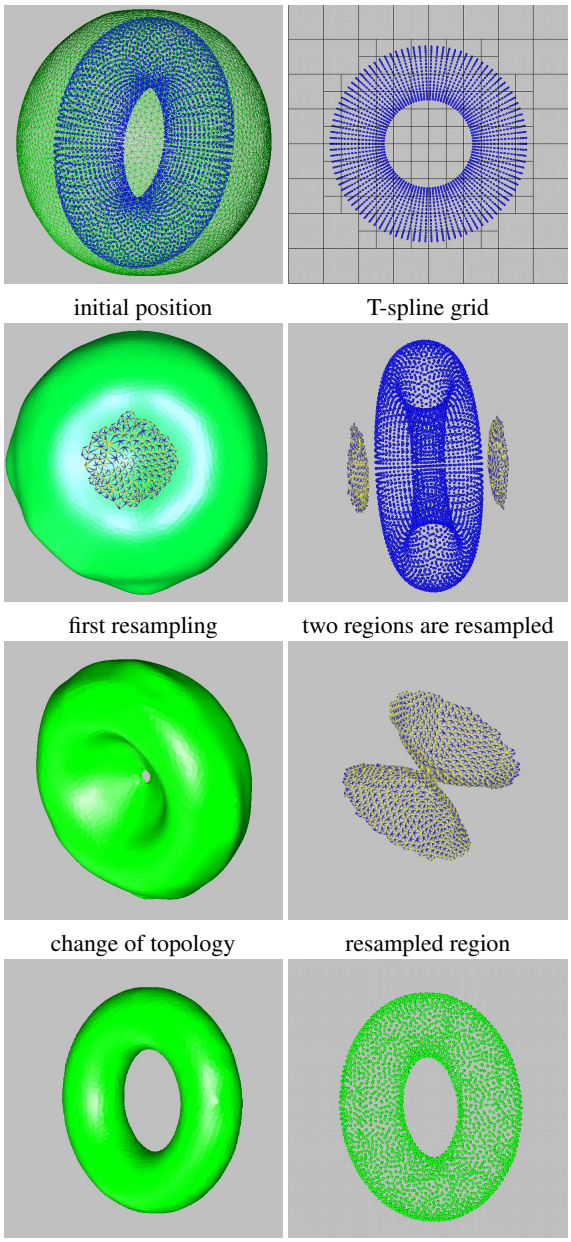


Figure 6: Evolution with local resampling

In the next row a change of topology takes place; a larger region has to be resampled in order to reflect this change. The pictures in the bottom row show the final result (left) and the final set of particles (right).

3.4 Global resampling

In this case we start with all particles $\widehat{S}' \subseteq \widehat{S}$ which do not satisfy the condition (13). We pick one of these points $P \in \widehat{S}'$ as the starting point for a global marching triangulation. This gives us a set S_2. Afterwards we check if the remaining particles in $S' \setminus \{P\}$ lie close to S_2. Otherwise we repeat the procedure. So we end up with one set of particles for each part of the implicitly defined surface at the current stage.

In practice we will not check for all sample points in $\widehat{S}$ instead we randomly select points such that $1/10$th of the points is tested. This is used to speed up the process.

4 CONSTRAINTS

In this section we will present two constraints which can be added to our evolution–based frame work. The first constraint C_R that we will present allows us to define regions where we force the active surface to lie inside or outside this region. This constraint will lead again to a quadratic minimization problem in $\dot{\mathbf{c}}$. So it has the same form as our original problem $E(\dot{\mathbf{c}})$, see Eqn. (10).

With help of the second constraint C_V we can specify a volume which the active surface has to enclose. This gives us a linear constraint which can be solved by using Lagrangian multipliers.

4.1 Range Constraints

This constraint allows us to specify regions such that the surface either contains this region or does not pass trough it. Since the zero–level set of our T-spline function has the property that

$$\begin{array}{lcll} f(\mathbf{x}) & > & 0 & \mathbf{x} \text{ outside } \Gamma \\ f(\mathbf{x}) & = & 0 & \mathbf{x} \in \Gamma \\ f(\mathbf{x}) & < & 0 & \mathbf{x} \text{ inside } \Gamma \end{array} \tag{14}$$

it divides the domain in two parts. (Note that one of these parts could be empty, but that would not make much sense in our framework.) Considering a set of points $\{\mathbf{x}_i\}_{i=1\ldots N_0}$ which should lie inside the zero–level set, we have to ensure that

$$f(\mathbf{x}_i) \leq 0. \tag{15}$$

In the case that $f(\mathbf{x}) > 0$ for some $\mathbf{x} \in \{\mathbf{x}_i\}$ the function value has to be pushed downward at this point. This is achieved by forcing $\dot{f}(\mathbf{x}) < 0$. Therefore we propose to add the following therm to our framework

$$C_R(\dot{\mathbf{c}}) = \sum_{j=1}^{N_0} (\dot{f}(\mathbf{x}_j,\tau) + f(\mathbf{x}_j,\tau) + \delta)^2 \, \alpha_\varepsilon(f(\mathbf{x}_j,\tau)) \tag{16}$$

where δ is an user–defined constant. For our examples we use $\delta = 0.2$. This constant controls how steep the T-spline becomes. The'activator' function α_ε controls the influence of the term $C_R(\dot{\mathbf{c}})$,

$$\alpha_\varepsilon(f) = \begin{cases} 1 & f > -\varepsilon \\ 0 & f < -2\varepsilon \\ C^2 - \text{blend} & \text{in between} \end{cases} \tag{17}$$

where ε is an user–defined positive constant (e.g., the feature size ρ can again be used). The optimization problem

$$\widehat{F}(\dot{\mathbf{c}}) = E(\dot{\mathbf{c}}) + \omega_c r \, C_R(\dot{\mathbf{c}}) \rightarrow \min. \tag{18}$$

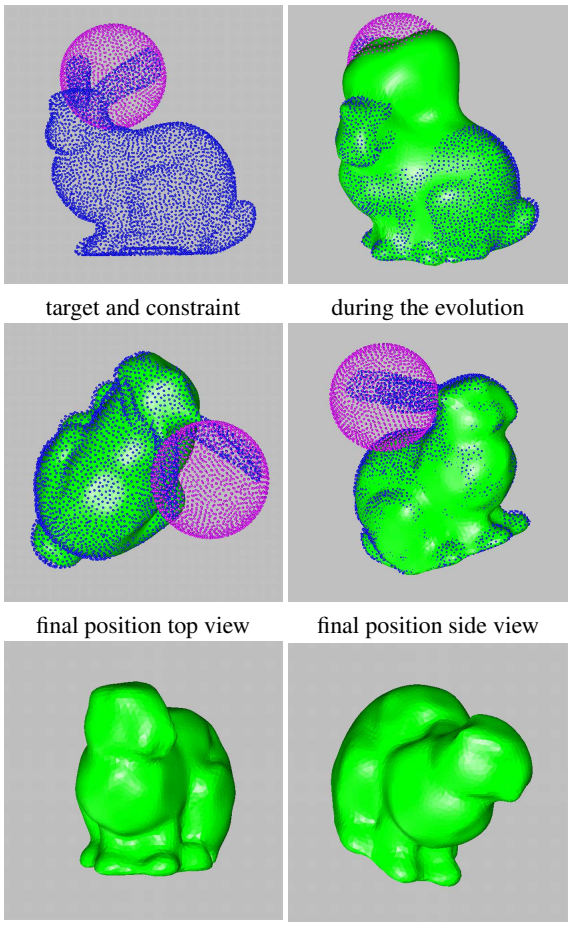


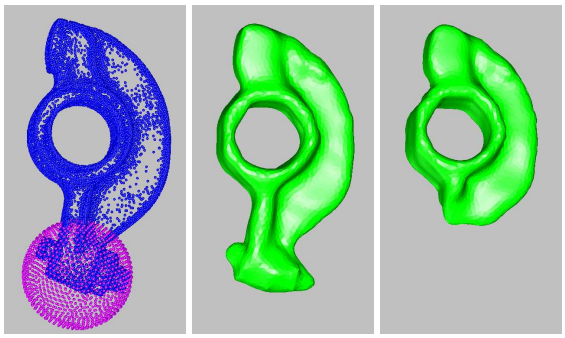
Figure 7: Evolution with range constraints

leads again to a sparse linear system of equations with a symmetric positive definite matrix, which can be dealt with efficiently.

Note that we can treat the case, where a given set of points $\{\mathbf{y}_i\}_{i=1\ldots N_1}$ should lie outside of our active surface, in a similar way.

We can also use this method for producing offsets of the target. Therefore we use all target points as sample points for the constraint. In this case δ defines the offset distance.

Example 2 Range constraints can be used to define regions which should not lie inside the target. In this example we will use a sphere to cut away the ears of the bunny model. The top left picture in Fig. 7 shows the target points and the constraint set. As the initial implicitly defined surface we take a sphere, this is not displayed. The top right picture visualizes the T-spline zero–level set after some time steps. The middle row shows two views on the final position. The ears have been cut off while the rest is recovered. In the bottom row, the final mesh is visualized from two different view points.



target and constraint | result without constraint | result with constraint

Figure 8: Rocker arm reconstruction

Example 3 The left picture in Fig. 8 shows the rocker arm data and a constraint region (ball). The data consists of 10044 points, and we used 1423 T-spline coefficients to reconstruct it. In the middle the result without using the constraint is shown, 4330 particles are used. The right picture visualizes the result using the constraint to cut away the lower part of the rocker arm, therefore 3622 particles are needed.

4.2 Volume Constraints

The volume constraint allows the surface to maintain a specified volume change during the evolution. Suppose $V(\tau)$ is the specified volume function with respect to the time τ, then the volume constraint can be represented as

$$\int_{\Gamma} v_n(\mathbf{x},\tau)dA = \dot{V}(\tau), \tag{19}$$

where A is the area element of the surface Γ, and

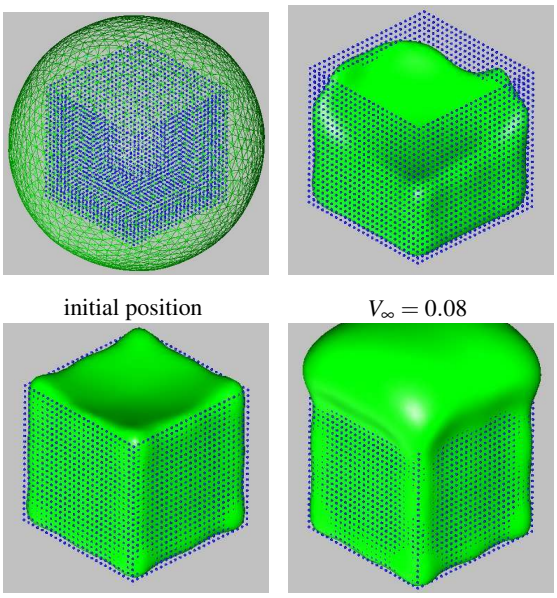
$$v_n(\mathbf{x},\tau) = -\frac{\dot{\mathbf{f}}(\mathbf{x},\tau)}{|\nabla\mathbf{f}(\mathbf{x},\tau)|}$$

is the normal velocity of the T-spline level set.

The volume function V can be an arbitrary function of τ, as long as the time derivative $\dot{V}(\tau)$ is well defined. The volume constraint defined in (19) is a volume-increase, volume-preserving or volume-decrease constraint, when $\dot{V}(\tau) > 0$, $\dot{V}(\tau) = 0$ or $\dot{V}(\tau) < 0$, respectively.

The volume constraint (19) is linear in the time derivatives of the T-spline control coefficients. Again we use numerical integration to obtain a linear constraint, which is to be considered along with the quadratic objective function. This leads to a quadratic optimization problem with linear constraints, which is solved using Lagrangian multipliers.

Example 4 The volume constraint is very helpful when dealing with noisy data containing holes. One may wish to specify the volume of the target object. By defining an appropriate function $V(\tau)$ such that $V(0) = V_0$ and $V(\tau) \to V_\infty$ as $\tau \to \infty$, the volume of the surface (initially equal to V_0) will converge to the desired value V_∞. Fig. 9 shows an example to demonstrate how this constraint works. The given data points are sampled from a cube after removing the top face. The volume of the cube equals 0.5^3. The figure shows the different results by specifying different values of the target volume V_∞. The original shape of the cube is recovered when $V_\infty = 0.125$.



initial position | $V_\infty = 0.08$

$V_\infty = 0.125$ | $V_\infty = 0.2$

Figure 9: Evolution with volume constraints

5 CONCLUDING REMARKS

We presented a particle-based approach to T-spline level set evolution. The set of particles is updated during the evolution, in order to avoid the frequent recomputation in each time step. In addition we can apply range and volume constraints to the evolving surface. The range constraints can be formulated as linear inequalities, which have been dealt with by a penalty method. The volume constraints lead to linear conditions, which can be incorporated directly into the framework.

In order to capture finer details of the target by T-spline level sets, a large number of T-spline coefficients is needed, slowing down the computations. Instead, one should better use a post–processing step, which is based on a displacement map and bilateral filtering, see [29]. That paper also presents computing times for the reconstruction of realistic 3D models via T-spline Level sets.

Acknowledgments.

The financial support of the Austrian Science fund through the Joint Research Program S92 'Industrial Geometry', subproject 2, and by the European Union through the Marie Curie IIF Fellowship for Huaiping Yang (project 22073 ISIS) is gratefully acknowledged.

REFERENCES

[1] M. Alexa and A. Adamson. Interpolatory point set surfaces - Convexity and Hermite data. *ACM Trans. Graph.* accepted.

[2] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. Silva. Point set surfaces. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 21–28, 2001.

[3] A. Angelidis, M.-P. Cani, G. Wyvill, and S. King. Swirling-sweepers: constant volume modeling. *Graphical Models (Special issue on PG'04)*, 68(4):324–332, 2006.

[4] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In *Proc. SIGGRAPH'01*, pages 67–76, 2001.

[5] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *Int. J. of Computer Vision*, 22(1):61–79, 1997.

[6] K.-S. D. Cheng, W. Wang, H. Qin, K.-Y. K. Wong, H. Yang, and Y. Liu. Fitting subdivision surfaces to unorganized point data using SDM. In *Pacific Conference on Computer Graphics and Applications 2004*, pages 16–24, 2004.

[7] T. K. Dey and S. Goswami. Tight cocone: a watertight surface reconstructor. In *Shape Modeling International*, pages 127–134, 2003.

[8] M. Eck and H. Hoppe. Automatic reconstruction of B-spline surfaces of arbitrary topological type. In *Proc. SIGGRAPH'96*, pages 325–334, 1996.

[9] R. Feichtinger, H. Yang, M. Fuchs, B. Jüttler, and O. Scherzer. Dual evolution of planar parametric spline curves and T-spline level sets. *Computer–Aided Design*, 2007. in press.

[10] S. Flöry and M. Hofer. Constrained curve fitting on manifolds. *Computer–Aided Design*, 2007. in press.

[11] E. Hartmann. A marching method for the triangulation of surfaces. *The Visual Computer*, 14(3):95–108, 1998.

[12] K. E. Hoff III, J. Keyser, M. Lin, D. Manocha, and T. Culver. Fast computation of generalized Voronoi diagrams using graphics hardware. In *Proc. SIGGRAPH '99*, pages 277–286, 1999.

[13] P. Jenke, M. Wand, M. Bokeloh, A. Schilling, and W. Strasser. Bayesian point cloud reconstruction. In *Proc. Eurographics '06*, 2006.

[14] B. Jüttler and A. Felis. Least squares fitting of algebraic spline surfaces. *Adv. Comput. Math.*, 17:135–152, 2002.

[15] E.-A. Karabassi, G. Papaioannou, T. Theoharis, and A. Boehm. Intersection test for collision detection in particle systems. *J. Graph. Tools*, 4(1):25–37, 1999.

[16] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. *Int. J. of Computer Vision*, 1(4):321–331, 1988.

[17] Y. Ohtake, A. Belyaev, and H.-P. Seidel. A composite approach to meshing scattered data. *Graph. Models*, 68(3):255–267, 2006.

[18] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer Verlag, New York, 2002.

[19] S. Osher and J.A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations. *J. of Computational Physics*, 79:12–49, 1988.

[20] M. Pauly, R. Keiser, L. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. *Proc. SIGGRAPH'03*, pages 641–650, 2003.

[21] M. Pauly, L. P. Kobbelt, and M. Gross. Point-based multiscale surface representation. *ACM Trans. Graph.*, 25(2):177–193, 2006.

[22] H. Qin, C. Mandal, and B. C. Vemuri. Dynamic catmull-clark subdivision surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 4(3):215–229, 1998.

[23] A. Raviv and G. Elber. Three dimensional freeform sculpting via zero sets of scalar trivariate functions. In *Proc. 5th ACM Symposium on Solid Modeling and Applications*, pages 246–257, 1999.

[24] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCs. In *Proc. SIGGRAPH '03*, pages 477–484, 2003.

[25] W. v. Funck, H. Theisel, and H.-P. Seidel. Vector field based shape deformations. In *Proc. SIGGRAPH'06*, pages 1118–1125, 2006.

[26] W. v. Funck, H. Theisel, and H.-P. Seidel. Explicit control of vector field based shape deformations. In *Pacific Conference on Computer Graphics and Applications 2007*, pages 291–300, 2007.

[27] A. P. Witkin and P. S. Heckbert. Using particles to sample and control implicit surfaces. In *Proc. SIGGRAPH '94*, pages 269–277, 1994.

[28] H. Yang, M. Fuchs, B. Jüttler, and O. Scherzer. Evolution of T-spline level sets with distance field constraints for geometry reconstruction and image segmentation. In *Shape Modeling International*, pages 247–252. IEEE Press, 2006. Extended version available at http://www.ig.jku.at.

[29] H. Yang and B. Jüttler. Meshing non-uniformly sampled and incomplete data based on displaced T-spline level sets. In *Shape Modeling International*, pages 251–260. IEEE Press, 2007.

An Optimial Way to Encode the Outlines of Variable Sized Arabic Letters in a PostScript Font

Abdelouahad BAYAR
Ecole Supérieure de Technologie, Safi
Cadi Ayyad University
Rue Sidi Aissa, P.O.Box 89
46000, Safi, Morocco
bayar@ucam.ac.ma

Khalid SAMI
Faculty of Sciences, Marrakesh
Cadi Ayyad University
12 BD. My Abdellah, P.B 2390
40000, Marrakesh, Morocco
k_sami@ucam.ac.ma

ABSTRACT

The scripts based on the Arabic alphabet are cursive and their characters are dynamically variable. These characteristics of such scripts need changes in the fonts conception. As a consequence of the cursivity, the representation of Arabic characters in outlines may be different of those concerning the Latin ones. The characters dynamic variability can be materialized through the stretchability of some characters in both the vertical and horizontal directions, at the same time, when justifying lines for instance. In order to get texts written with stretchable characters, represented as outlines, some curves intersections, among the curves composing the characters outlines, are to be determined dynamically. The PostScript procedure to produce a dynamic character, in a dynamic font, is repeated whenever the letter is to draw. So, the determination of the curves intersections coefficients in the procedure, applying an iterative algorithm is of a high cost. A new method for developing an optimal font is to find out. In this paper, a method based on the curves comparison is presented. It allows the determination of the characters with eventually overlapping outlines. Then, a way to approximate the curves intersections coefficients is given. This is enough to remove overlapping in outlines.

Keywords

Arabic Calligraphy, Cursivity, Dynamic Stretching, Outlines Writing, Curves Comparison, Bézier Intersection, Approximation.

1. INTRODUCTION

The Arabic calligraphy rules are as important as grammar in handwriting texts in Arabic alphabet based scripts. Stretching letters is used instead of the ordinary insertion of spaces (blanks) between words, to justify lines in such texts. This stretching; the Keshideh, is almost meaningful and mandatory. There are two types of keshideh, between two connected letters and inside the same letter. These two types of stretching are presented with *bounding boxes* of characters in the figure 1 below. In the first line, there are stretchings in different sizes between the two connected letters ظ and ر in the word ظفر. In the following line, the last letter ق of the word حق is stretched in four different sizes.

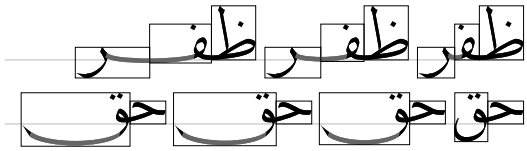


Figure 1. Inter-curvilinear and intra-curvilinear stretchings in Arabic calligraphy.

The keshideh spreads out in both *horizontal* and *vertical* directions and obey to some conditions. The horizontal stretching can go from 0 to 12 diacritic points (the diacritic point is a metric unit in Arabic calligraphy).
Of course, the keshideh is curvilinear and context dependent. A suitable font for typesetting texts in

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

an Arabic alphabet based script would be based on a dynamic language like PostScript [Ado99] or the instructions for encoding characters should be in TrueType fonts [Wei92]. Unfortunately, such support is not offered by Metafont [Knu86] and consequently, (La)TEX [Lam85] extensions like ArabTex [Lag92] don't support the dynamic character needed in Arabic fonts. So, only some discrete finite dynamism can be found. Currently, the system CurExt [Laz03] used in RydArab [Laz01] provides only 256 stretchings in different sizes in a given document.

Danniel Berry developed a type 3 PostScript font for the system dittroff/fforttid [Ber99]. This system offers a good support for dynamic characters. Unfortunately, the stretching goes only in the horizontal direction and the stretching model doesn't take into account the motion of the nib's head. Big stretchings can show some weakness of the calligraphic quality. A way to fix such flaws can be found in [Bayar].

The main goal of this paper is to provide a support for developing a PostScript font that allows typesetting Arabic texts, in characters represented as outlines, according to the basic calligraphic rules. The existing systems don't offer such support. In the first line of the figure 2 (from right to left), the word ظفر is displayed in outlines, in two options : with and without limits between two consecutive characters. Most of the systems allowing writing Arabic script in outlines, such as Microsoft Word, can produce only the first sample. Of course, The cursivity is better observed in the second way of writing on the figure 2.

Figure 2. Possibilities of Outlines writing in Arabic.

Managing stretchability is better shown with fonts that provide characters in outlines. An example of stretched outlines is presented in the figure 3. This characters stretchability needs a mathematical formalization and the stretching model presented in [Bayar] will so be deeply improved. That's the goal of the following sections.

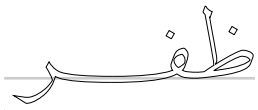
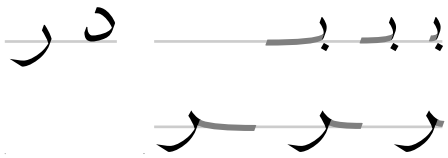


Figure 3. Stretchable outlines in Arabic.

Let us recall that there are *static* letters and *variable sized* ones. Static letters have static shapes as the letter د (standalone DEL) and ر (standalone REH) on the figure 4-a. Variable sized letters are those with stretchable parts. The first line of the figure 4-b displays the letter بـ (BEH in the beginning of a word) in three different curvilinear stretchings whereas the second line presents the letter ـر (REH in the end of a word). Variable sized letters are composed of a static part (in terms of shape; the shape is kept as it is after a shifting transformation) and a dynamic one. On the figure 4-b, the dynamic parts are in gray. For the letter بـ, this part is used to connect the letter to the following letter in the left, since the Arabic writing is a right-to-left. In the same context, the stretchable part in the other letter is the preceding connection in the letter ـر (this is used to connect it to the letter directly in its right).



(a) (b)

Figure 4. a) Examples of static letters, b) Examples of variable sized letters.

In the following, the paper follows the plan : the second section, presents the mathematical formalization of the nib's motion. In the third section, the way to optimize the implementation of letters in PostScript is given. The paper ends with conclusions and perspectives.

2. THE NIB'S HEAD MOTION-RAZING MODELING AND DECOMPOSITION

There are many *styles* in Arabic writing. Among the most popular ones, we can mention : Farisi, Koufi, Maghribi, Naskh, Thuluth, Rouqaa, DywAni... Samples of such styles are presented on the figure 5. For our present purpose, let us consider the Naskh style (The framed one on the figure 5) since it is the most widely used in digital typography.

In this Part, we develop the mathematical tools to be used for the development of a *stretchable* Naskh font. Of course, this font will support writing in outlines. Let's give first some characteristics of the nib's head motion under Arabic Calligraphy rules.

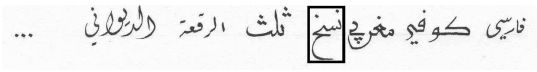


Figure 5. Most popular Arabic writing styles.

Qalam's Motion Modeling

In the Naskh style, the nib's head or simply the qalam (The pen calligraphers use to write with) behaves as a rectangle of width l and thickness $e = \frac{l}{6}$. This rectangle moves with a constant inclination angle of about 70° with regard to the baseline. A nib's head with $l = 12$ mm (and $e = 2$mm) is presented in the figure 6.

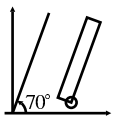


Figure 6. Nib's head in Naskh style with 12 mm width

In the following, let us consider the general case where the pen's head motion can be represented by a polygonal shape with $n+1$ vertices, in translation. Examples are given with rectangles whose sizes and inclinations are different from the Naskh qalam ones for better clarity. Let's represent a pen's head with $n+1$ vertices M_0, M_1, ..., M_n by $\mathcal{B}(M_0, \mathcal{F})$, where $\mathcal{F} = (\overrightarrow{u_i})_{i=0,\ldots,n}$ is a family of $n+1$ vectors such that $\overrightarrow{u_i} = \overrightarrow{M_0 M_i}$. The point M_0 stands for an *origin* for the pen's head motion. Whenever particular values are associated to that point and to the vectors, all the vertices, and therefore positions, of the pen's head can be determined precisely. When the trajectory of the origin vertex is the parametric curve f defined on $[0,1]$, the motion of a pen $\mathcal{B}(M_0, \mathcal{F})$ will be denoted by $\mathcal{M}(\mathcal{B}(M_0, \mathcal{F}), f, [0,1])$. In the figure 7, the motion is characterized by the start position of the pen's head and a Bézier curve [Bez77] B_0 representing the trajectory of the vertex M_{00}. In the case of the qalam, the origin is the vertex surrounded with a circle on the figure 6.

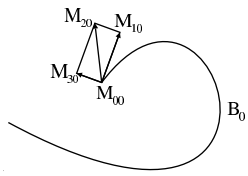


Figure 7. Example of motion

The Motion-Razing Study

The razing of a qalam motion is the set of points darken by the qalam in the plane. Of course, this set is delimited by external curves corresponding to the trajectories of the qalam's vertices and some edges of the qalam. Therefore, we need a mean to help comparing the shifting transforms of curves in order to find out the external curves of a razing. For this, it is necessary to define the notion of *the orthogonal range associated to a parametric function according to a vector.*
Consider :

- the affine space $\mathbb{R}^3$ with the orthonormal basis $R\left(O, \overrightarrow{\imath}, \overrightarrow{\jmath}, \overrightarrow{k}\right)$,
- an affine plane $\mathbb{R}^2$ in $\mathbb{R}^3$ containing the origin O with a direct orthonormal basis $(\overrightarrow{\imath}, \overrightarrow{\jmath})$. Let's denote $\overrightarrow{\mathbb{R}}^2$ the vector plane associated to $\mathbb{R}^2$.
- a parametric function f defined on $[0,1]$ with values in $\mathbb{R}^2$. Suppose that f is continuous on $[0,1]$ and differentiable on $]0,1[$.
- and the vectors $\overrightarrow{u}$, $\overrightarrow{u}_1$, $\overrightarrow{u}_2$ in $\overrightarrow{\mathbb{R}}^2$.

Definition 1 (Direct orthogonal range with respect to a vector)
The Direct orthogonal range associated to f, according to the vector $\overrightarrow{u}$, is the scalar function $\mathcal{R}_{(f,\overrightarrow{u})}$ defined from $[0,1]$ onto $\mathbb{R}$ such that $\mathcal{R}_{(f,\overrightarrow{u})}(t) = \left(\overrightarrow{u} \wedge \overrightarrow{Of(t)}\right) \cdot \overrightarrow{k}$.

We can remark that the derivative verify $\mathcal{R}'_{(f,\overrightarrow{u})}(t) = \mathcal{R}_{(f',\overrightarrow{u})}(t)$
Now, we can define the way to compare the shifting transforms of a curve.

Definition 2 (Comparison of shifted curves)
We have $t_{\overrightarrow{u}_1}(f) \leq t_{\overrightarrow{u}_2}(f)$ on $[0,1]$ (respectively $t_{\overrightarrow{u}_1}(f) \geq t_{\overrightarrow{u}_2}(f)$) if and only if $\mathcal{R}'_{(f,\overrightarrow{u}_1-\overrightarrow{u}_2)}(t) \leq 0$ (respectively $\mathcal{R}'_{(f,\overrightarrow{u}_1-\overrightarrow{u}_2)}(t) \geq 0$).

The comparison between $t_{\overrightarrow{u}_1}(f)$ and $t_{\overrightarrow{u}_2}(f)$ on $[0,1]$ can be done through the study of the monotony of $\mathcal{R}_{(f,\overrightarrow{u}_1-\overrightarrow{u}_2)}$ on $[0,1]$.
Now, we have the tools necessary for studying the nib's head motions and their associated razings. Before doing that, let's present a particular type of motions, the *normal* motions, and their associated razings.
Consider the motion $M = \mathcal{M}(\mathcal{B}(M_{00}, \mathcal{F}), f, [0,1])$ such that $\mathcal{F} = (\overrightarrow{u_i})_{i=0,\ldots,n}$.

Definition 3 (Normal Motion) *M is said to be* normal *on $[0,1]$ if and only if $\exists\, i \in \{0,\ldots,n\}$, $\exists\, j \in \{0,\ldots,n\}$ such that :*
$t_{\overrightarrow{u}_k}(f) \leq t_{\overrightarrow{u}_i}(f)$ and $t_{\overrightarrow{u}_k}(f) \geq t_{\overrightarrow{u}_j}(f)$ on $[0,1]$ $\forall\, k \in \{0,\ldots,n\}$.

 ISBN 978-86943-15-2

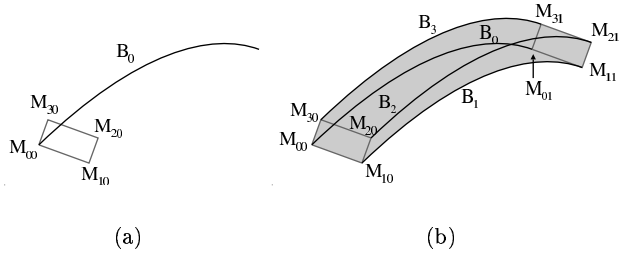


Figure 8. a) Normal motion, b) The corresponding normal razing.

In this definition, $t_{\overrightarrow{u}_i}(f)$ is the *maximal* curve whereas $t_{\overrightarrow{u}_j}(f)$ is the *minimal* one among the set $\left\{t_{\overrightarrow{u}_k}(f),\ \overrightarrow{u}_k \in \mathcal{F}\right\}$. A normal motion and its corresponding razing are presented on the figure 8. The razing is delimited by the two segments $[M_{30}M_{00}]$, $[M_{00}M_{10}]$, the Bézier curve B_1, the two segments $[M_{11}M_{21}]$, $[M_{21}M_{31}]$ and the Bézier curve B_3.

Lemma 1 *If* $\forall i \in \{0,\ldots,n\}$ $\mathcal{R}_{(f,\overrightarrow{u}_{(i+1)mod\ n}-\overrightarrow{u}_i)}$ *is monotone on* $[0,1]$ *then* M *is a normal motion.*

The lemma can be proved through considering the motion on the figure 8. In general, the razing associated to a motion is determined via its decomposition into normal sub-motions. Then, we have the theorem.

Theorem 1 (Decomposition into normal sub-motions)
The motion M *can be decomposed into normal sub-motions on* $[0,1]$ *so that the concatenation of their associated razings constitute exactly the global razing associated to* M.

Proof
Let $\mathcal{F} = (\overrightarrow{u_i})_{i=0,\ldots,n}$ ($\overrightarrow{u_0} = \overrightarrow{0}$),
Through studying the sign of $\mathcal{R}'_{(f,\overrightarrow{u}_{(i+1)\mathrm{mod}\ n}-\overrightarrow{u}_i)}$ on $]0,1[$ for $i \in \{0,\ldots,n-1\}$ we get the set of extrema of $\mathcal{R}_{(f,\overrightarrow{u}_{(i+1)\mathrm{mod}\ n}-\overrightarrow{u}_i)}$ on $]0,1[$.
Let $E_i = \left\{T_{i,1}, T_{i,2}, \ldots, T_{i,\phi(i)}\right\}$ be the set of extrema of $\mathcal{R}_{(f,\overrightarrow{u}_{(i+1)\mathrm{mod}\ n}-\overrightarrow{u}_i)}$ ordered in an increasing order ($\phi(i)$ reverses the cardinal of E_i).
The restrictions of the direct orthogonal range $\mathcal{R}_{(f,\overrightarrow{u}_{(i+1)\mathrm{mod}\ n}-\overrightarrow{u}_i)}$ to the segments $[0,T_{i,1}[$, $[T_{i,1},T_{i,2}[$, $\ldots,[T_{i,\phi(i)-1},T_{i,\phi(i)}[$, $[T_{i,\phi(i)},1]$ are monotone.
Now, let us consider $E = \{T_1, T_2, \ldots, T_m\}$ the union of the sets E_i, ordered in an increasing order.
According to the previous reasoning, $\mathcal{R}_{(f,\overrightarrow{u}_{(i+1)\mathrm{mod}\ n}-\overrightarrow{u}_i)}$ is monotone on $[0,T_1[$, $[T_1,T_2[$, $\ldots,[T_{m-1},T_m[$, $[T_m,1]$ for $i \in \{0,\ldots,n-1\}$.
By the lemma 1, the qalam's motion is normal on $[0,T_1[$, $[T_1,T_2[$, $\ldots,[T_{m-1},T_m[$, $[T_m,1]$.
The restrictions considered together constitute the exact trajectories of the vertices of the qalam's head.
So, the concatenation of the razings gives the exact razing of the pen's head. ■
An example is given in the file attached to the paper.

3. OPTIMAL IMPLEMENTATION OF STRETCHABLE OUTLINES

This part deals with the way to solve the problem of implementing letters in their outlines in Arabic calligraphy, taking into account the stretchability. The concepts in use will be given through considering examples.
In this paragraph, the model of stretching a letter is given. Then, the concept of tolerance and digital equivalence will be highlighted. Finally, the ways to implement a static or a dynamic letter will be presented.

The Stretching Model

In variable sized letters, dynamic parts, i.e. the preceding and following connections, are quadratic Bézier curves. Let's denote a quadratic Bézier curve with control points M_0, M_1 and M_2 by $[M_0, M_1, M_2]$. The preceding connections are curves in the set $\mathcal{B}_1$ and the succeeding ones are in $\mathcal{B}_2$. The set $\mathcal{B}_1$ and $\mathcal{B}_2$ are defined in the following. Let $B = [M_0, M_1, M_2]$, then :

- $B \in \mathcal{B}_1 \Leftrightarrow \overrightarrow{M_0M_1} = \lambda\overrightarrow{\imath}$, $\lambda \in \mathbb{R}_*^-$ and $\left(\widehat{\overrightarrow{M_2M_1}, \overrightarrow{\imath}}\right) \leq \frac{\pi}{2}$ (see figure 9-a) where $(\widehat{\overrightarrow{u}_1, \overrightarrow{u}_2})$ stands for the angle between the two vectors $\overrightarrow{u}_1$ and $\overrightarrow{u}_2$.
- $B \in \mathcal{B}_2 \Leftrightarrow \overrightarrow{M_1M_2} = \lambda\overrightarrow{\imath}$, $\lambda \in \mathbb{R}_*^-$ and $\left(\widehat{-\overrightarrow{\imath}, \overrightarrow{M_0M_1}}\right) \leq \frac{\pi}{2}$ (see figure 9-b).

Now, let's present the way to stretch curves in $\mathcal{B}_1$ and $\mathcal{B}_2$. Let $B_1 = [M_{10}, M_{11}, M_{12}]$ be the quadratic Bézier curve with control points $M_{10} = (x_{10}, y_{10})$, $M_{11} = (x_{11}, y_{11})$ and $M_{12} = (x_{12}, y_{12})$ and $B_2 = [M_{20}, M_{21}, M_{22}]$ a stretched version of B_1 with a horizontal stretching value h and a vertical one v. We have :

- if $B_1 \in \mathcal{B}_1$ then $M_{20} = (x_{10}, y_{10} - v)$, $M_{22} = (x_{12} - h, y_{12})$ and $M_{21} =$

 ISBN 978-86943-15-2

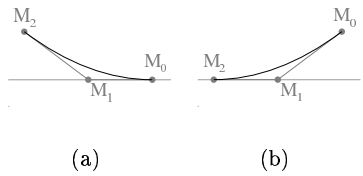


Figure 9. a) Quadratic curve of type 1, b) Quadratic curve of type 2.

$$\left(x_{12} - h + \frac{(y_{10}-y_{12}-v)(x_{11}-x_{12})}{y_{11}-y_{12}}, y_{10} - v\right).$$

- if $B_1 \in \mathcal{B}_2$ then $M_{20} = (x_{10}, y_{10})$, $M_{22} = (x_{12} - h, y_{12} - v)$ and $M_{21} = \left(x_{10} + \frac{(y_{12}-y_{10}-v)(x_{11}-x_{10})}{y_{11}-y_{10}}, y_{12} - v\right)$.

In particular, in the font, the vertical stretching value v depends on h. Let $v = \frac{V}{H}h$. Then, V and H stand for the maximal values of the vertical and horizontal stretchings respectively. We can remark that the type of the curve (type 1 or 2) to stretch is preserved after stretching. An example of stretching for the two types are presented on the figure 10.

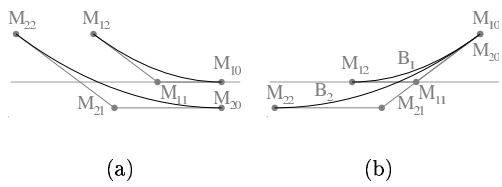


Figure 10. a) Example of stretching a quadratic curve of type 1, b) Example of stretching a quadratic curve of type 2.

In [Bayar], a model based on a cubic Bézier curve has been used easily since we hadn't to deal with the letters outlines. Now, in order to simplify the outlines computing, we used quadratic curves instead of cubic Bézier curves. There would be no loss of visual accuracy since the connections between letters have curvature vectors with small magnitudes.
The keshideh, the stretchable part in a word, is a juxtaposition of a curve in $\mathcal{B}_1$ with an other curve in $\mathcal{B}_2$. When the keshideh has to stretch of h horizontally and v vertically then each of the the components (in $\mathcal{B}_1$ and in $\mathcal{B}_2$) has to stretch of $\frac{h}{2}$ horizontally and v vertically.

Tolerance - Digital Equivalence

In order to implement a font in stretchable outlines, we use approximations. So, we need to consider some precisions or tolerances. In this subsection, we define the notions of digital equivalence and tolerance.

Definition 4 (Digital Equivalence) *Let $M_1 = (x_1, y_1)$ and $M_2 = (x_2, y_2)$ be two points in $\mathbb{Z}^2$. M_1 is digitally equivalent to M_2 if and only if $|x_1 - x_2| \leq 1$ and $|y_1 - y_2| \leq 1$. The digital equivalence is then denoted by $M_1 \equiv M_2$.*

So, for the visual aspect, a point in $\mathbb{Z}^2$ and points in his direct neighbor would then be considered as the same.
To each point in $\mathbb{R}^2$ to draw corresponds an image in $\mathbb{Z}^2$ by a function denoted $\mathcal{N}$. The tolerance of a Bézier curve can then be defined as follows :

Definition 5 (Tolerance of a Bézier curve) *Let B be a Bézier curve and $\tau = \frac{1}{n}$, $n \in \mathbb{N}^*$. The value τ is a tolerance of B if and only if for all t_1 and t_2 in $[0, 1]$ we have :*

$$|t_1 - t_2| \leq \tau \Longrightarrow \mathcal{N}(B(t_1)) \equiv \mathcal{N}(B(t_2)) \ .$$

The notions of digital equivalence and the tolerance are tools toward the formalization of the visual accuracy. We can define the notion of the optimal tolerance.

Definition 6 (Optimal Tolerance) *Let B be a Bézier curve. The optimal tolerance of B is the greater element of the set $\left\{\frac{1}{m} / m \in \mathbb{N}*, \text{ where } \frac{1}{m} \text{ is a tolerance of } B\right\}$*

We have the property :

Property 1 *Let us consider the Bézier curves B_1, B_2 and B_3. Let τ_1, τ_2 and τ_3 be their optimal tolerances respectively, then we have the properties :*

- *if B_2 is a stretched version of B_1 then $\tau_2 \leq \tau_1$,*
- *if B_2 and B_3 are curves obtained from B_1 by a Bézier refinement according to a parameter t in $[0, 1]$, then we have $\tau_2 \geq \tau_1$ and $\tau_3 \geq \tau_1$.*

Letter Outlines Computing

3.3.1 Static Letters

When a letter is static, it is directly implemented in the font with the true outline drawn by the qalam's motion (Outlines without overlapping, see figure 11). The true outline is determined with a mean external to the font applying the mathematical notions studied in the paragraph 2.

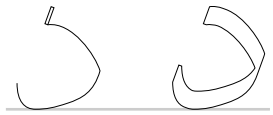


Figure 11. The Qalam's Motion and the corresponding true outlines of the standalone DEL.

3.3.2 Stretchable Letters

The outlines, in the case of variable sized letters, can be implemented in PostScript exactly as in the following example of the letter ر. For the static part of the letter ر, a non closed outline without overlapping is implemented in the font (see figure 12).The outline is then determined out of the font.

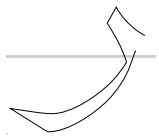


Figure 12. Non closed outline of the static part of End REH.

What about the preceding connection of the letter ر? The preceding connection of ر is a curve in $\mathcal{B}_1$ parametrized with a stretching amount h. The motion of the nib's head is shown on the figure 13-a. Let's denote B^h the trajectory of the motion's origin. For all values of h, the scalar function $\mathcal{R}_{(B^h,\overrightarrow{u}_3)}$ have an extrema on $]0,1[$ (note that $\overrightarrow{u}_2 - \overrightarrow{u}_1 = \overrightarrow{u}_3$). Let t^h be this extrema (the h here is not an exponent. It only means that t depends on h). The direct orthogonal ranges $\mathcal{R}_{(B,\overrightarrow{u}_1-\overrightarrow{u}_0)}$ and $\mathcal{R}_{(B,\overrightarrow{u}_3-\overrightarrow{u}_2)}$ are strictly monotone on $[0,1]$. So, the motions on $[0,t^h]$ and $[t^h,1]$ are normal. Consider the following notations :

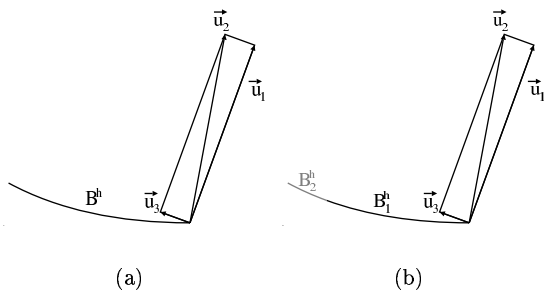


Figure 13. Motion of the stretchable part of End REH.

- t^h is the extrema of $\mathcal{R}_{(B^h,\overrightarrow{u}_3)}$ on $]0,1[$,
- B^h_1 and B^h_2 are the curves obtained by the decomposition of B^h applying the generalized Bézier algorithm of refinement [Bar85] with respect to t^h (see figure 13-b),
- $B^h_{1i} = t_{\overrightarrow{u}_i}\left(B^h_1\right)$, $i \in \{0,1,2,3\}$ ($\overrightarrow{u}_0 = \overrightarrow{0}$) and
- $B^h_{2i} = t_{\overrightarrow{u}_i}\left(B^h_2\right)$, $i \in \{0,1,2,3\}$.

Then, we have the results :

- On $[0,t^h]$ the maximal curve is B^h_{12} and the minimal one is B^h_{10} and
- On $[t^h,1]$ the maximal curve is B^h_{21} and the minimal one is B^h_{23}.

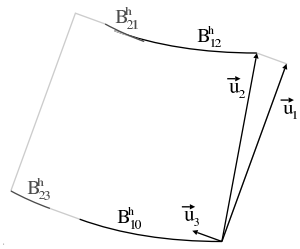


Figure 14. Overlapping of REH preceding connection.

These results are illustrated on the figure 14 with $h = 0$.
The true outline of the preceding connection can't be directly determined because there is an intersection between B^h_{12} and B^h_{21}. Note that this intersection is unique. Let r^h and s^h be the two variables on $[0,1]$ characterizing the intersection coefficients of B^h_{12} and B^h_{21} respectively. Their values are solutions of the equation 1.

$$B^h_{12}(r) - B^h_{21}(s) = (0,0) \tag{1}$$

The values of t^h, r^h and s^h change as h changes. The values are to be computed dynamically in the font. As B^h is a quadratic curve, the range $\mathcal{R}'_{(B^h,\overrightarrow{u}_3)}(t)$ is a linear function of t. Then, it is easy to determine explicitly t^h in function of h. The formula is given on the equation 2. It was in order to get an easy and explicit determination of t^h that we were to represent the stretchable parts in letters with quadratic curves instead of cubic ones. The function t^h is a monotone increasing function of h.

$$t^h = \frac{47.79081298 + h}{63.67851945 + 1.0887286138 \cdot h} \tag{2}$$

In general, the determination of the intersection between two quadratic or cubic Bézier curves is done through iterations (there is no explicit solution). Since the PostScript procedure to produce a character, in a dynamic font, is repeated whenever the letter is to draw, the determination of the

curves intersections coefficients in the procedure, applying an iterative algorithm is of a high cost. So, we have to find out a way to minimize the time necessary to compute r^h and s^h.
Let us consider once again the notations :

- τ_{12}^h the tolerance of B_{12}^h depending on h.
- τ_{21}^h the tolerance of B_{21}^h depending on h.

In the Naskh style, the stretching variable h can vary from 0 to H, where H is 6 diacritic points (12 dp for all the keshideh and 6 dp for each of its components). Consider a font in 1000 points size, $H = 580.829699$. The determination of the coefficients r^h and s^h would be based on approximations as follows.
The system Maple can be used to get the following properties :

Property 2

- *The function r^h is monotone increasing of h on $[0, H]$,* *(1)*
- *The function s^h is monotone decreasing of h on $[0, H]$,* *(2)*
- *The functions τ_{12}^h and τ_{21}^h are monotone decreasing on $[0, H]$,* *(3)*
- *and $\tau_{12}^h \leq \tau_{21}^h$ on $[0, H]$. This means that for $h \in [0, H]$, τ_{12}^h is an optimal tolerance of B_{12}^h and a tolerance of B_{21}^h.* *(4)*

Now, we have all the properties necessary to build the function giving the approximated values of r^h and s^h.

Let $r_{min} = r^0$, $r_{max} = r^H$. Consider the sequence $R = (r_i)_{i=0,\ldots,n}$ such that $r_i = r_{min} + \frac{r_{max}-r_{min}}{n} i$, where $n = \left[\frac{r_{max}-r_{min}}{\tau_{12}^H}\right] + 1$. Let $[x]$ be the integer part of x. Let's also consider the two sequences $\mathcal{H} = (h)_{i=0,\ldots,n}$ and $S = (s_i)_{i=0,\ldots,n}$ such that $\{h_i, s_i\}_{i=0,\ldots,n}$ are solutions of the equation 1 where h and s are the unknown and r is given taking r_i as value. The system Maple can help to solve all the equations. In some cases, some of these equations can require some minutes when B^h is quadratic and even some hours when B^h is cubic. That's an other reason to choose quadratic curves to encode dynamic parts. We get the following results :

- $r_{i+1} - r_i < \tau_{12}^H \ \forall i = 0, \ldots, n-1$,
- $r_{i+1} - r_i < \tau_{12}^h \ \forall i = 0, \ldots, n-1 \ \forall h \in [0, H]$ (according to the property 2-(3)). This means that $\mathcal{N}\left(B_{12}^h(r_i)\right) \equiv \mathcal{N}\left(B_{12}^h(r_{i+1})\right) \ \forall h \in [0, H]$,

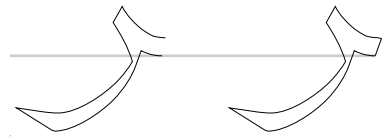


Figure 15. Closed and non closed non overlapping outline.

- $s_i - s_{i+1} < \tau_{21}^h \ \forall i = 0, \ldots, n-1 \ \forall h \in [0, H]$ (according to the property 2-(2 and 4)) . This means that $\mathcal{N}\left(B_{21}^h(s_i)\right) \equiv \mathcal{N}\left(B_{21}^h(s_{i+1})\right) \ \forall h \in [0, H]$.

We have finally an important property that would be useful for the determination of the approximation function, $\forall i \in \{0, \ldots, n-1\}$, $\forall h$, $h_i \leq h \leq h_{i+1}$, we have :

- $r_i \leq r^h \leq r_{i+1}$ and $\mathcal{N}\left(B_{12}^h(r^h)\right) \equiv \mathcal{N}\left(B_{12}^h(r_i)\right)$ and so, $\mathcal{N}\left(B_{12}^h(r^h)\right) \equiv \mathcal{N}\left(B_{12}^h(r_{i+1})\right)$.
- $s_{i+1} \leq s^h \leq s_i$ and $\mathcal{N}\left(B_{21}^h(s^h)\right) \equiv \mathcal{N}\left(B_{21}^h(s_i)\right)$ and so, $\mathcal{N}\left(B_{21}^h(s^h)\right) \equiv \mathcal{N}\left(B_{21}^h(s_{i+1})\right)$.

Then, we know very well the tolerant values of $B_{12}^h(r^h)$ and $B_{21}^h(s^h)$.
Now, consider the sequence $P = (P_i)_{i=0,\ldots,n} = (r_i, s_i)_{i=0,\ldots,n}$. Let's approximate the set of points P_i by a quadratic Bézier curve A such that $A(0) = P_0 = (r_0, s_0)$ and $A(1) = P_n = (r_n, s_n)$. In order to approximate a set of points $\{P_0, \ldots, P_n\}$ by a curve parametrized on $[0, 1]$, we may associate to every point P_i a coefficient t_i in $[0, 1]$ such that the sequence $(t_i)_{i=0,\ldots,n}$ is strictly monotone increasing and $t_0 = 0$ and $t_n = 1$.
First, let us determine $(t_i)_{i=0,\ldots,n}$.
Recall that t^h (defined in the equation 2) is a monotone increasing function of h.
Since r^h is monotone increasing, the sequence $(h_i)_{i=0,\ldots,n}$ deduced through the sequence $(r_i)_{i=0,\ldots,n}$ is monotone increasing. Consider $(t_i)_{i=0,\ldots,n}$ such that $t_i = \frac{t^{h_i} - t^0}{t^H - t^0}$.
$(t_i)_{i=0,\ldots,n}$ is a strictly monotone increasing sequence with $t_0 = 0$ and $t_n = 1$.
Consider that the quadratic Bézier curve to be determined has the control points (r_0, s_0), (x_1, y_1) and (r_n, s_n). We have to determine (x_1, y_1) in order to minimize the value D (see equation 3).

$$D = \sum_{i=0}^{n} \left(P_i - A(t_i)\right)^2 \tag{3}$$

D can be minimal if the following mandatory conditions are satisfied :

- $\frac{\partial D}{\partial x_1} = 0$ and
- $\frac{\partial D}{\partial y_1} = 0$.

$\frac{\partial D}{\partial x_1} = 0$ implies that $x_1 = \frac{\sum_{i=0}^{n} t_i(1-t_i)\left(r_i-(1-t_i)^2 r_0-t_i^2 r_n\right)}{2\sum_{i=0}^{n} t_i^2(1-t_i)^2}$ and $\frac{\partial D}{\partial y_1} = 0$ implies that $y_1 = \frac{\sum_{i=0}^{n} t_i(1-t_i)\left(s_i-(1-t_i)^2 s_0-t_i^2 s_n\right)}{2\sum_{i=0}^{n} t_i^2(1-t_i)^2}$.
Suppose that $A(t) = (r(t), s(t))$. Then A is a good approximation if the condition in the formula 4 is satisfied

$$|r(t_i) - r_i| < \tau_{12}^H \ \forall i = 0, \ldots, n \qquad (4)$$

If the condition 4 isn't satisfied the sequences $\mathcal{H}$ and P are subdivided in two pairs of sub-sequences $\left\{(h_i)_{i=0,\ldots,k}, (P_i)_{i=0,\ldots,k}\right\}$ and $\left\{(h_i)_{i=k,\ldots,n}, (P_i)_{i=k,\ldots,n}\right\}$ such that k verify the formula 5 and the process is repeated.

$$|r(t_k) - r_k| = \max_{i=0}^{n} |r(t_i) - r_i| \ \forall i = 0, \ldots, n \quad (5)$$

Concretely, for the preceding connection of the letter ر, the determination of the intersection coefficients are computed with the evaluation of a curve A_1 where h is in $[0, 179.595499[$ and another curve A_2 if h is in $[179.595499, 580.818536]$.
The approximation has been tested with arbitrary values of h and we got intersection coefficients with respect of the curves tolerances (the details are on the annexed file).
With these mechanism, we can encode the preceding connection to get ر with closed outlines and therefore to fill the letter and also a non closed outline to be used to write words in outlines (see figure 15).

4. CONCLUSIONS

This is a way to solve the problem of drawing stretchable, variable sized Arabic letters in outlines. This problem has been posed many years ago. Of course, only the Naskh style, where the nib's head is in constant inclination, has been studied. This approach will be developed in a coming paper, to styles where the nib's head can be in translation and rotation at the same time. That is of a big importance for writing Arabic alphabet based texts respecting the basic calligraphic rules. It can also help to compose calligraphic mathematical formulas.

5. REFERENCES

[Lam85] Leslie Lamport, LaTeX-A Document Preparation System, Reading MA: Addison Wesley, 1985.

[Laz01] Azzeddine Lazrek, A package for typesetting Arabic mathematical formulas, Die TEXnische Komodie, DANTE e.V., Vol.13, No. 2, pp. 54–66, 2001.

[Laz03] Azzeddine Lazrek, CurExt, Typesetting variable-sized curved symbols, EuroTEX 2003 Pre-prints : 14th. European TEX conference (Brest, France), pp. 47-71, 2003.

[Ado99] Adobe. PostScript Language Reference Manual, Library of Congress Cataloging-in-Publication Data, Third edition, Adobe Systems Incorporated, 1999.

[Wei92] D. Weise, D. Adler, TrueType and Microsoft Windows 3.1, Technical Report, Microsoft Corporation, Redmond, WA, 1992.

[Knu86] D.E. Knuth, The MetafontBook, Computers and Typesetting, Reading MA: Addison-Wesley, Vol.C, 1986.

[Ber99] Daniel M. Berry, Stretching Letter and Slanted-Baseline Formatting for Arabic, Hebrew and Persian with dittroff/fforttid and Dynamic PostScript Fonts, Software-Practice and Experience, Vol. 29, No. 15, pp.1417-1457, 1999.

[Bayar] A. Bayar and K. Sami, How a font can respect rules of Arabic calligraphy (submitted).

[Bez77] Pierre E. Bézier, Essai de Définition Numérique des Courbes et des Surfaces Expérimentales, Ph.D. dissertation, Université Pierre et Marie Curie, Paris, 1977.

[Bar85] Brian A. Barsky, Arbitrary Subdivision of Bézier Curves, Technical Report UCB.CSD 85/265, Computer Science Division, University of California, Berkeley, California 94720, November, 1985.

[Lag92] Klaus Lagally, ArabTEX - Typesetting Arabic with Vowels and Ligatures, EUROTEX'92, Prague, 1992.

 ISBN 978-86943-15-2

Shape Recognition in 3D Point-Clouds

Ruwen Schnabel Raoul Wessel Roland Wahl Reinhard Klein
Computer Graphics Group, University of Bonn
53117 Bonn, Germany
Email: {schnabel, wesselr, wahl, rk}@cs.uni-bonn.de

ABSTRACT

While the recent improvements in geometry acquisition techniques allow for the easy generation of large and detailed point cloud representations of real-world objects, tasks as basic as for example the selection of all windows in 3D laser range data of a house still require a disproportional amount of user interaction. In this paper we address this issue and present a flexible framework for the rapid detection of such features in large point clouds. Features are represented as constrained graphs that describe configurations of basic shapes, e.g. planes, cylinders, etc. Experimental results in various scenarios related to the architectural domain demonstrate the feasibility of our approach.

Keywords: shape detection, graph matching, retrieval, recognition, point-cloud

1 INTRODUCTION

The acquisition and processing of digital 3D point-clouds has received increasing attention over the last few years. While visualization of very detailed and complex point-clouds has become possible, interaction capabilities on a semantic level are still very limited. Even tasks as basic as selecting all windows in a scan of a house currently require a disproportional amount of user interaction. This is due to the fact that the acquired raw data does not provide any structural, let alone semantic, information. Therefore the extraction of semantic elements from 3D point clouds is an important topic for a wide field of applications, including architecture, cultural heritage and city model reconstruction. Further applications can be envisioned in the engineering context, e.g. supporting the automatic inventory of industrial plants or traditional reverse engineering processes.

The reasons for working on point-clouds are twofold: The first, and maybe most obvious, is that point-clouds are the native output format of laser range scanners. Moreover, point-clouds are an extremely general geometry representation as they contain no interpretation of the data. Therefore, by choosing point-clouds as the starting point of our method, we are able to process any type of 3D surface representation including polygon soups and meshes, since these representations are easily converted into point clouds by sampling.

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

Copyright UNION Agency – Science Press, Plzen, Czech Republic.

Concerning the task of shape recognition in point-clouds, two problems arise:

- The raw point-cloud contains an overwhelming amount of redundant information making it virtually impossible to operate directly on the points themselves in an efficient manner.
- For each element, we need a model that the computer can retrieve in the data. However, defining a special parameterized model for each sought feature is challenging and very inflexible in case new entity types shall be detected.

In this work we focus on an architectural application domain. Based on the observation that most man-made objects in this domain can be decomposed into parts corresponding to geometric primitives like planes, cylinders, spheres, etc. we propose novel solutions for the above mentioned problems:

Decomposition By decomposing the point data into primitive shapes, we obtain an abstraction of the point data that eliminates much of the redundancy. A topology graph captures the neighborhood relations between the primitives in a concise manner.

Constrained subgraph matching Primitive shapes are assembled into configurations characteristic for higher semantic elements, e.g. windows, roofs or columns. Such configurations can be quickly retrieved from the topology graph via constrained subgraph matching.

Our system allows even unexperienced users to quickly formulate complex configurations, since primitive shapes are easily graspable and combinable. The search results are visualized in an interactive framework, allowing for refinement of the query.

Our method was tested on a large amount of data including point-clouds from different kinds of sensors like *LIDAR (light detection and ranging)* and stereo reconstruction. We also applied our method to 3D CAD

modeled buildings that include interior structures. The approach is robust against noise, clutter, registration errors or miscalibrations which are frequently encountered in 3D laser scans.

2 RELATED WORK

Our shape matching technique is related to many works in the larger context of (partial) matching, classification and retrieval of 3D shapes. Various approaches to these challenges have been developed in the past. In this brief overview we concentrate on methods for partial matching and retrieval of 3D objects in larger 3D scenes. For a more detailed introduction to 3D matching and shape retrieval we refer to [TV04].

2.1 3D city reconstruction

Since in this work we mainly deal with data from the architectural domain, automatic reconstruction of 3D buildings and city models from aerial LIDAR data, a special case of the above formulated tasks, is especially relevant to our approach. In this setting detection is restricted to simple shapes of which most can be described by configurations of planes. Automatic building reconstruction has been studied extensively in the photogrammetry community. Most of the developed semi-automatical or manual approaches rely on user interaction or on semantic knowledge that is not contained in the 3D point-cloud. For Example, Vosselman et al. integrate LIDAR data with a 2D Geographic Information System (GIS) database and aerial photographs [Vos02]. The GIS delivers ground plan information about buildings in the point cloud. With this information at hand, the points belonging to a single edifice can be extracted and parametric building models can be fitted.

A fully automatic approach that is only relying on the information contained in the point-cloud is presented in [VKH06]. There, a region growing approach is used to detect planes in the point data. Roof-topology graphs are defined to describe configurations of planes for some simple building forms shaped like I, L and U. These configurations are sought in the set of the detected planes. In a second step, the detected simple buildings are extended to more complicated forms according to the plane configurations in the point-cloud. Compared to our approach, there are two differences: First, as only planes are detected in the LIDAR data, the approach is restricted to those shapes that can be decomposed into planar patches. Second, the method does not use any node or graph constraints during the subgraph search and is therefore susceptible to misclassifications in more general settings.

2.2 Partial matching and retrieval of arbitrary objects

The retrieval and matching of 3D objects is of particular interest to the computer graphics community. The developed methods often rely on triangle meshes or parametric representations of the objects. Among the abundance of proposed approaches, several graph-based shape retrieval methods rely on the extraction of certain geometric components and use a graph to capture the relations between these components.

Model graph-based approaches are mainly used for geometry such as is generated in CAD applications. Model graphs describe solid objects in terms of connectivity of freeform surfaces (Boundary Representation) or as a set of geometric primitives that are connected by Boolean operations (Constructive Solid Geometry). Local clique matching [EM03] [EMA03] or comparison of graph spectra [MPSR01] are used to globally determine the similarity of the graphs, i.e. no partial matching is supported. In [ZTS02], VRML objects are segmented according to different decomposition techniques. The resulting patches are assigned basic shapes like planes and spheres. An attributed decomposition graph is built containing the determined shapes as nodes. Neighboring shapes are connected by edges. The similarity between two objects is computed by matching the associated decomposition graphs using error-correcting subgraph isomorphism.

Reeb graph-based methods rely on a function that is computed on the model surface. The surface is divided into segments corresponding to intervals of this function. A skeleton graph, in which the resulting segments are represented as nodes, is built. Reeb graph-based methods are mainly used for matching of articulated objects [HSKK01][TL07]. In [PSBM07], a robust method for fast Reeb graph computation is proposed that even allows for the use of non-manifold meshes.

Skeleton graphs of 3D shapes can be computed using topological skinning of voxel representations [BNdB99], medial axis transform, ridge point tracking [CSM05] or deformable model-based reconstruction [SLSK07]. Matching of two shapes is done by comparing the associated skeleton graphs using greedy bipartite graph matching [SSGD03] or by detecting subgraph isomorphisms using decision trees [LJI+03].

3 OVERVIEW

The algorithm consists of two main steps: Firstly a shape based representation of the data is derived by detecting primitive shapes in the unstructured point data and constructing a *topology graph* that captures the neighborhood relations between the different shapes. Then, in the second step, this topology graph is searched for characteristic subgraphs corresponding to sought elements, as they have been defined by the user.

The user is able to quickly define and retrieve new entities with geometric constraints in an interactive framework. Moreover the detected subgraphs may contain optional or repetitive components, which further simplifies the definition of new entities for the user. This

way our method is very flexible and easily extensible, which renders it suitable in a broad range of applications.

4 SHAPE REPRESENTATION

The construction of a shape representation for the data constitutes the first step in our method. At this stage the fundamental building blocks are established that lay the ground for all further operations. A set of primitive shapes is detected, which is then subsumed in a topology graph.

4.1 Primitive Shapes

The aim of the shape detection is to find simple elements in the point-cloud which shall be the building blocks of more complex structures later on. We employ the algorithm presented in [SWK07] which recognizes planes, spheres, cylinders, cones and tori in the unstructured point-cloud. In this section, we will only give a very brief outline of the shape detection technique and the interested reader is referred to the original paper. The data is decomposed into disjoint sets of points, each corresponding to a detected shape proxy respectively, and a set of remaining points that consists of outliers as well as areas of more complex geometry for which primitive shapes would give an inappropriate representation. For further processing all remaining points are ignored. Points that are represented by a shape primitive are also called the *support* of a shape. Thus, given the point-cloud $P = p_1, \ldots, p_N$, the output of the shape detection is the following:

$$P = S_{\phi_1} \cup \ldots \cup S_{\phi_A} \cup R, \tag{1}$$

where each subset (the support) S_{ϕ_i} is associated with a shape primitive ϕ_i. All points in S_{ϕ_i} constitute a connected component and fulfill the condition

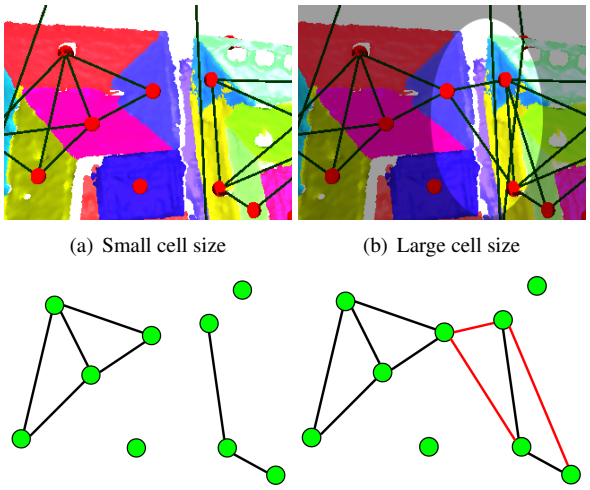
$$s \in S_{\phi_i} \Rightarrow \|s, \phi_i\| < \varepsilon \wedge \angle(n_s, n(\phi_i, s)) < \alpha, \tag{2}$$

where n_s is the normal of point s and $n(\phi_i, s)$ denotes the normal of the primitive ϕ_i at the point closest to s. The parameters ε and α are chosen by the user according to the precision of the acquisition device. The set R contains all the remaining, unassigned points.

4.2 Topology Graph

The topology graph $G(\Phi, E)$ describes the neighborhood relations between the primitive shapes detected in the point-cloud data. For each primitive ϕ_i a vertex is inserted into the graph, i.e. $\Phi = \phi_1 \ldots \phi_A$. Two shapes are connected with an edge if their supports are neighboring in space, i.e. the two vertices ϕ_i and ϕ_j are joined by an edge $e = (\phi_i, \phi_j)$ if

$$\exists p \in S_{\phi_i}, q \in S_{\phi_j} : \|p - q\| < t \tag{3}$$



(a) Small cell size (b) Large cell size

(c) Topology graph for small cell size (d) Topology graph for large cell size

Figure 1: Two houses viewed from above that are separated by a narrow alley. Primitive shapes have been detected and are depicted in random colors. a) The topology graph was built with a cell width of 50cm b) The cell width for the construction of the topology graph was set to 2m. Note that the roofs have been connected across the narrow alleyway. In c) and d) we show the resulting topology graphs. In d), the additional edges resulting from large cells are shown in red.

holds. Please note that computing the distance between the shapes directly and ignoring the support would result in many edges that have no counterpart in the data, since the shape primitives have indefinite extent.

Thus, to find the graph edges, the spatial proximity between the support of all detected shapes has to be determined. To this end we employ an axis aligned 3D grid. In a first step all points are sorted into the grid. Then for all grid cells that contain points belonging to different shapes, edges connecting the corresponding graph vertices are added to the graph, i.e. for each pair of shapes in the cell an edge is created. In order to avoid discretization dependencies due to the location of the grid cells, we use eight shifted and overlapping grids. Cells are stored in a hash table, so that memory is only allocated for occupied cells.

The width of the grid cells defines how far apart shapes are allowed to be in order to still get connected in the topology graph. Given the distance threshold t, the width of the cells is set to t and the shifted versions of the grid are created with an offset of $\frac{1}{2}t$ along the respective axes. Of course this means that shapes can get connected in some cases even though the distance between their support is in fact only less than $\sqrt{3}t$. It would be possible to eliminate these cases by checking the distance between the points in each cell, but we found this additional overhead unnecessary in practice,

as the few errors in the less restricted topology graph did not influence the performance of our algorithm. In Figure 1 resulting graphs are depicted for different cell widths.

Once the graph is complete the first step of our method is finished and a shape based representation of the point-cloud data has been derived. It can now be used to efficiently detect more complex configurations of primitive shapes that correspond to semantic entities in the data.

5 SHAPE MATCHING

In order to achieve an automatic matching between semantic entities and point-clouds, we have to find a common language for them. As we abstracted the point data to obtain a higher level description, we have to concretize the representation of feature elements in terms of primitive shape configurations.

5.1 Query graph

To this end we define a *query graph* for an element as a graph that captures its characteristic shape configuration. Basically a query graph is a topology graph with the difference that it does not stem from a point-cloud, but from knowledge about the shape of an element which is introduced by the user. The recognition of an element in the data then corresponds to a matching of the query graph to a subgraph of the topology graph. Even though subgraph matching is a NP-complete problem [Coo71], in our applications the query graphs will be small, i.e. usually less than twenty vertices, and a simple brute-force implementation of subgraph matching performs well in such a setting.

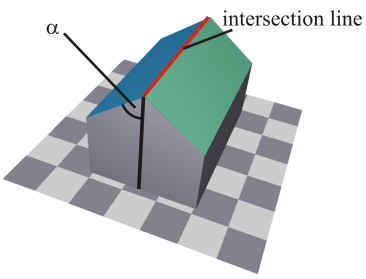


Figure 2: Illustration of the constraints that can be used to detect saddleback roofs: The angle α is constraint to be less than 90 degrees and similar for both planes. The intersection line is required to run parallel to the ground.

However, a representation solely based on topology is not sufficient to discriminate between many different feature elements. For example in the simple case of a saddleback roof (see Figure 2), the model graph consists of two vertices corresponding to planes connected by an edge. If such a graph is searched in a topology graph numerous false matches can be expected, as such a simple configuration occurs frequently. To make the detection more reliable, the user can add constraints to the query graph. For instance in the case of the saddleback roof we require the two planes to exceed a certain size, to be of similar size, and to intersect in a line that is parallel to the ground.

If we take a closer look at these geometric constraints, we find that they can be divided into classes that access different kinds of information. There are *node constraints* which only restrict the primitive shape associated with a node (e.g. type, size or orientation). There are *edge constraints* which restrict the relation between two incident shapes (e.g. angle between two planes). Any constraint not fitting into one of the first two classes belongs to the class of *graph constraints*, because it relies on the topology to be checked (e.g. sums of sizes, parallelism of disconnected planes).

Thus, when modeling a query graph the user specifies the sought shape configuration on a topological level by insertion of shape nodes and connecting edges. Geometric relations, however, are attached to these graph elements in the form of constraints which are formulated in a simple scripting language. The scripts have access to all parameters of the supported primitives as well as to the set of assigned points for each shape. Moreover predefined functions for computation of intersections, test for parallelism or orthogonality etc. exist.

5.2 Constrained subgraph matching

The outline of the recursive constrained query graph matching is illustrated in algorithm 1. To simplify the discussion of the procedure, no explicit statements are given for neither the maintenance of a data structure storing the matching, nor keeping track of visited nodes. However these actions are assumed to take place implicitly and it should be noted that they are mandatory for any correct implementation of the method. In the following the different parts of the algorithm will be described in detail:

In lines 2-8 the outer matching function is given, which searches for a suitable node in the topology graph, where the matching can be started. This outer matching function has to be started repeatedly to retrieve all possible matches.

Matching a node

In lines 10-18 the function for matching a node is sketched. First a check is made to see if all edges of the node have already been matched and if this is the case, the matching of the node has been successful (lines 11-14). Otherwise a yet unmatched edge of the node is chosen and matched to an edge of the topology graph by calling the MatchEdge function. If the edge

 ISBN 978-86943-15-2

```
1:  Input A topology graph T = (V_T, E_T) and a query
    graph Q = (V_Q, E_Q)
2:  Function MatchSubgraph(Q, T)
3:  v_Q ← StartNode(V_Q)
4:  for all v_i ∈ V_T do
5:      if CheckNodeConstraint(v_Q, v_i) then
6:          if MatchNode(v_Q, v_i) then
7:              return true
8:  return false
9:
10: Function MatchNode(v_Q, v_T)
11: if v_Q has no unmatched edge then
12:     if all nodes in V_Q are matched then
13:         return CheckGraphConstraint()
14:     return true
15: e_Q ← first unmatched edge of v_Q
16: if MatchEdge(e_Q, v_T) then
17:     return MatchNode(v_Q, v_T)
18: return false
19:
20: Function MatchEdge(e_Q, v_T)
21: for all unmatched outgoing edges e_i of v_T do
22:     if CheckNodeConstraint(dest(e_Q), dest(e_i))
        then
23:         if CheckEdgeConstraint(e_Q, e_i) then
24:             if MatchNode(dest(e_Q), dest(e_i)) then
25:                 return true
26: return false
```

Algorithm 1: Recursive Constrained Subgraph Matching

was matched successfully, MatchNode is called recursively on the same node, in order to match any remaining edges of the node (lines 15-18).

Matching an edge

In lines 20-26 the MatchEdge function is outlined. It checks if any of the unmatched edges of the given topology graph's node can be matched to the given query graph edge (line 21-25). This is the case if the end-nodes of the edges can be matched successfully - which is tested via a call to MatchNode (line 24).

Checking constraints

Constraints are always verified just before a match is established (lines 5, 13, 22, 23). In line 13 the graph constraints are checked as soon as all nodes of the query graph have been matched. If this test fails, the matching will backtrack and continue the search. As can be seen, in contrast to graph constraints, both, node and edge constraints, have the advantage that they can be checked early on during the subgraph matching procedure, as they do not rely on other parts of the graph. This is an important performance factor since this way many of the topologically correct matches can be quickly discarded, without causing extensive backtracking.

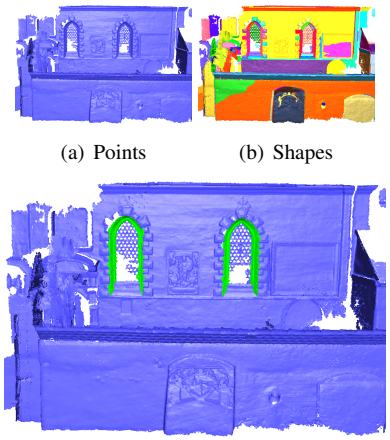
In order to avoid the need for graph constraints when using asymmetric edge constraints (e.g. sphere A has to be larger than sphere B) we also support directed edges as carriers of a constraint.

5.3 First results

At this point the methods presented so far are powerful enough to recognize large classes of semantic entities and we will first give a couple of examples illustrating the possibilities before presenting further extensions to the basic matching framework:

In Figure 3 a query graph was designed to detect Gothic windows in a scan of a medieval chapel. The windows were modeled as two spheres for the arches and two planes for the sides of the window. The spheres were constrained to have roughly equal radius and the planes to be tangential to the spheres.

To find the columns of the choir screen in Figure 4 they were modeled as a cylinder connected to two tori at both ends. The cylinder and the tori were constrained to posses the same axis of rotation (with a small tolerance of 5 degrees).



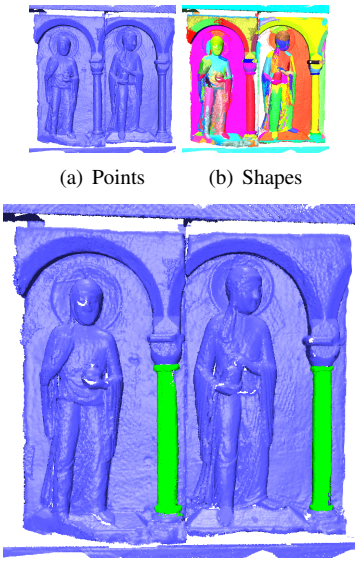
(a) Points (b) Shapes

(c) Windows

Figure 3: A scan of a medieval chapel with Gothic windows containing 4.2M points. The windows were detected by matching the query graph with subgraphs of the topology graph. In a) the original point-cloud is depicted. b) shows the support of the detected shape primitives in random colors. In c) the detected columns are highlighted in green.

5.4 Query Graph Extensions

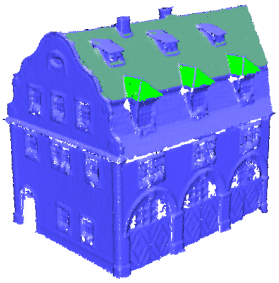
Although the given definition of a query already covers many shape configurations, there remain cases which it is still insufficient for. In the following we discuss some of these cases and demonstrate how they are overcome by extensions to the query graph model:



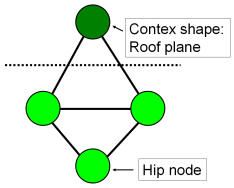
(a) Points (b) Shapes

(c) Columns

Figure 4: A scan of a choir screen consisting of 2M points. The query graph for the columns consisted of a cylinder connected to tori at both ends. In a) the original point-cloud is depicted. b) shows the support of the detected shape primitives in random colors. In c) the detected columns are highlighted in green.



(a) Matching



(b) Query graph

Figure 5: a) Detection of dormers on a roof. The roof plane shown in darker green is a context shape of the dormers. b) The query graph containing a context node.

Context nodes

Certain features benefit from a *context object* to distinguish them from other structures. For instance, a balcony needs a wall as context. Therefore we need to model the context in the query graph, but without declaring it an integral part of the balcony. This is achieved by tagging these query graph nodes as context nodes, so that after searching they can be removed from the match. In Figure 5 a) we give an example for this concept. There the roof planes have been modeled as the context shapes of the dormers, see Figure 5 b).

Optional nodes

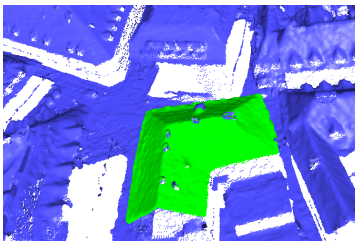
A limitation of the way we model queries so far is that we are not able to specify variants of an entity without duplicating the original query graph. For instance L-shaped roofs like the one shown in Figure 6 may occur in four variants: not hipped, hipped on either end or hipped on both ends. Thus a total of four query graphs, that only marginally differ, would have to be defined separately by the user. Since in practice this additional work may become quite burdensome, we augment the query graphs with what we call *optional nodes*. These nodes may be ignored by the matching procedure if it is unable to find any suitable counterparts in the given topology graph. To incorporate optional nodes, the matching procedure of Sec. 5.2 is extended in the following manner: First the matching is performed in the same way as described above, but ignoring any optional nodes. Then, for each matched instance of the query graph, as many optional nodes as possible are matched. To this end the graph traversal examines all possible matchings of the optional nodes but returns only those with the largest number of matches.

A problem arises if there are entire optional query graph components, instead of only single optional nodes. In such a case simply declaring all the nodes in question as optional could lead to incomplete matchings of the component. Therefore, we use a graph constraint which asserts the completeness of the matching.

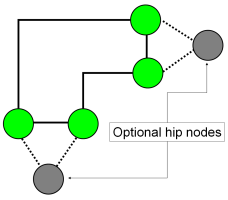
Although optional nodes increase the complexity of the search, they greatly reduce the number of required query graphs if different variants of a basic concept have to be detected. In Figure 6 the single hip of an L-shaped roof was matched by an optional node.

Multinodes

An even more complex case arises if we want to be able to model repetitive patterns like the steps of a stairway. In order to be able to model stairways with an arbitrary number of steps, a generic way of model extension is necessary. A simple approach is to define *multinodes* in the query graph that may match *several* different topology graph nodes. A multinode is defined as a query graph node that has a self-loop, i.e. an edge connecting the node with itself (this edge is implicitly considered optional by the matching algorithm). Via multinodes we are able to match arbitrarily large chains in the topology graph. This allows us to define a query graph



(a) L-shaped roof



(b) Query graph

Figure 6: An L-shaped roof may be hipped on either end. This is best modeled by optional nodes in the query graph. a) A matched L-shaped roof in a stereo reconstruction of a city containing 4M points. b) The query graph used for detection. Optional nodes are shown in grey.

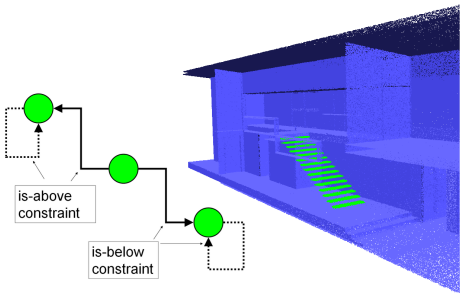


Figure 7: Detection of a stairway in a sampled CAD model of a house.

for stairways using only three nodes, as depicted in Figure 7. Note that we make use of directed edges in this example so that the multinodes do not need to match additional neighbor nodes each time the self-loop has been traversed (since the multinode does not have an outgoing edge other than the optional self-loop).

Query refinement

After the search for a query graph, the system presents the user with the results in an interactive framework that allows query refinement by changing constraints as well as query graph topology at runtime. As soon as an element of the query graph is modified, the new results are computed and displayed. This was achieved in real-time for all our tested examples.

6 CONCLUSION

Our shape detection system works on point-clouds so that we are able to work on data stemming from virtually arbitrary sources, such as terrestrial or airborne LIDAR data, polygon soups as well as ordinary meshes. As we mainly target applications in the architectural or cultural heritage domain, we safely assumed that most objects under consideration can be well represented by a set of primitive shapes. Thus we employ a fast primitive shape extraction method to effectively reduce the redundancy in the point-cloud and to derive a concise shape representation consisting of a topology graph on the shape primitives. In this graph, our system allows the detection of features that can be described as compositions of simple primitive shapes. Due to the simple structure of our representation it is not necessary for the primitive shape detection to output an optimal segmentation with a minimal number of primitives, nor to find the correct edges and transitions between different shapes. Only the detection of the relevant structures and their rough outlines has to be ensured.

Our system is unable to deal with cases for which features cannot be defined as configurations of primitive shapes, e.g. if trying to detect ornate frescos. However, we have demonstrated that for a wide range of frequently encountered structures our approach is very well suited and is able to deliver results as expected by the user. The user is able to specify the sought structures in a general way, even permitting fuzzy search within the limits of the graph constraints and the inclusion of optional components.

A potential drawback of our method could arise if large topology graphs with a lot of nodes and edges are used and at the same time the node and edge constraints of the query graph are chosen in a way that a wrong match will not be encountered early on during the matching procedure. Since the search for subgraph isomorphisms is a NP-hard problem, the retrieval performance might degenerate. However, such pathological cases are unlikely and in practice we observe very fast response times of the system, i.e. in the order of a few milliseconds (see timings in Table 1).

6.1 Future work

Future work concerning our method should address the improvement of usability. Up to now, the query graphs and the attached constraints are defined by hand. To make this process more comfortable for less experienced users we propose the development of a graphical user interface in which query graphs and constraints can easily be defined. A further step of research would be the automatic extraction of query graphs from modeled or scanned objects by means of statistical learning. Techniques like relevance feedback could be used to further enhance the retrieval performance.

dataset	#nodes	#edges	top. graph	matching
chapel (Figure 3)	232	406	1.8s	< 10ms
choir screen (Figure 4)	537	2731	1.8s	< 10ms
dormer roof (Figure 5)	106	138	0.27s	< 10ms
city model (Figure 6)	431	351	2.5s	< 10ms
CAD-house (Figure 7)	160	513	3.9s	< 10ms

Table 1: Some statistics on test models. *#nodes* describes the number of primitive shapes that were found in the point cloud and by that the number of nodes in the resulting topology graph. *#edges* states how many edges describing neighborhood relations between the primitive shapes were found. *top.graph* shows how long it took to build the topology graph. The last column describes how much time was needed for matching the query graph.

Moreover, we plan to apply our system to basic point-cloud editing operations such as copy and paste of semantic units. Another interesting avenue of future research is exploiting the ability to detect self-similarities in the data for compression. Replacing instances of a query graph by generic representations might lead to very high compression ratios.

ACKNOWLEDGEMENTS

This work was partially funded by the German Science Foundation (DFG) under grant GZ 554975(1) Oldenburg BIB 48 OLof 01-02 *Probado* and as part of the bundle project "Abstraction of Geographic Information within the Multi-Scale Acquisition, Administration, Analysis and Visualization". The city model of Graz (Figure 6) is courtesy of German Aerospace Center (DLR) – Institute of Robotics and Mechatronics and was derived by semi-global matching from Vexcel Imaging Graz™ imagery. The model of the rochus chapel (Figure 3) is courtesy of Zoller+Fröhlich GmbH.

REFERENCES

[BNdB99] Gunilla Borgefors, Ingela Nyström, and Gabriella Sanniti di Baja. Computing skeletons in three dimensions. *Pattern Recognition*, 32(7):1225–1236, 1999.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM Press.

[CSM05] Nicu D. Cornea, Deborah Silver, and Patrick Min. Curve-skeleton applications. In *IEEE Visualization*, page 13. IEEE Computer Society, 2005.

[EM03] M. El-Mehalawi. A database system of mechanical components based on geometric and topological similarity. part ii: indexing, retrieval, matching, and similarity assessment. *Computer-Aided Design*, 35(1):95–105, January 2003.

[EMA03] Mohamed El-Mehalawi and Allen. A database system of mechanical components based on geometric and topological similarity. part i: representation. *Computer-Aided Design*, 35(1):83–94, January 2003.

[HSKK01] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proceedings of ACM SIGGRAPH*, 2001.

[LJI+03] K. Lou, S. Janyanti, N. Iyer, Y. Kalyanaraman, S. Prabhakar, and K. Ramani. A reconfigurable 3d engineering shape search system part ii: database indexing, retrieval and clusturing. In *DETC*, 2003.

[MPSR01] David McWherter, Mitchell Peabody, Ali C. Shokoufandeh, and William Regli. Database techniques for archival of solid models. In *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 78–87, New York, NY, USA, 2001. ACM Press.

[PSBM07] Valerio Pascucci, Giorgio Scorzelli, Peer-Timo Bremer, and Ajith Mascarenhas. Robust on-line computation of reeb graphs: simplicity and speed. *ACM Trans. Graph.*, 26(3):58, 2007.

[SLSK07] Andrei Sharf, Thomas Lewiner, Ariel Shamir, and Leif Kobbelt. On–the–fly curve-skeleton computation for 3d shapes. In *Eurographics*, pages 323–328, Prague, september 2007.

[SSGD03] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval, 2003.

[SWK07] R. Schnabel, R. Wahl, and R. Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, June 2007.

[TL07] Gary K. L. Tam and Rynson W. H. Lau. Deformable model retrieval based on topological and geometric signatures. *IEEE TVCG*, 13(3):470–482, 2007.

[TV04] J. W. Tangelder and R. C. Veltkamp. A survey of content based 3d shape retrieval methods. In *SMI '04: Proceedings of the Shape Modeling International 2004 (SMI'04)*, pages 145–156, Washington, DC, USA, 2004. IEEE Computer Society.

[VKH06] Vivek Verma, Rakesh Kumar, and Stephen Hsu. 3d building detection and modeling from aerial lidar data. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2213–2220, Washington, DC, USA, 2006. IEEE Computer Society.

[Vos02] George Vosselman. Fusion of laser scanning data, maps, and aerial photographs for building reconstruction. In *Geoscience and Remote Sensing Symposium*. IEEE International, 2002.

[ZTS02] Emanuel Zuckerberger, Ayellet Tal, and Shymon Shlafman. Polyhedral surface decomposition with applications. *Computers and Graphics*, 26(5):733–743, October 2002.

The Room Connectivity Graph: Shape Retrieval in the Architectural Domain

Raoul Wessel
Computer Graphics Group
University of Bonn
53117 Bonn, Germany
wesselr@cs.uni-bonn.de

Ina Blümel
German National Library
of Science and Technology
30167 Hannover, Germany
ina.bluemel@tib.uni-hannover.de

Reinhard Klein
Computer Graphics Group
University of Bonn
53117 Bonn, Germany
rk@cs.uni-bonn.de

ABSTRACT

While advances in CAD modeling techniques led to an ever increasing number of available architectural 3D models, reusability of these models as templates or as inspiration sources is still very limited. One reason for this is that there exist basically no shape retrieval methods specialized in the architectural domain. In this work, we therefore present a method to efficiently characterize 3D architectural models according to the underlying arrangement of their rooms by a *room connectivity graph*. In this graph, rooms are represented by attributed nodes. Connections between rooms, i.e. doors or windows, are represented by attributed edges. We show that these attributed graphs can be used for an efficient retrieval of 3D architectural models using fast graph matching techniques.

Keywords: 3D Shape Retrieval, Graph Matching

1 INTRODUCTION

3D modeling plays a more and more important role in the architectural domain. Along with the increasing amount of available 3D models of buildings comes the requirement to search building databases for those models that are similar to a query object, either for inspirational purposes or for reuse.

Common state-of-the-art 3D shape retrieval techniques mostly rely on the extraction of global or local geometric features and shape descriptors. Two 3D shapes are supposed to be similar if the distance between their shape descriptors is small. These methods are not well suited for the domain of architectural 3D models, as they are strongly characterized by topological properties of their underlying 2D floor plans rather than by local geometric 3D features.

While drafting, architects are particularly using floor plans, because those can easily demonstrate geometry and structure of a building. Taking all desirable drafting opportunities into consideration, floor plans are most unambiguous showing the spatial organization (development structure, topology and disposition of rooms, [Sch04]). Therefore they are a major ingredient for architectural drafting.

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

Copyright UNION Agency – Science Press, Plzen, Czech Republic.

For bottom-up-design of buildings the topology which means the arrangement of space is important, whereas for top-down-design the outer geometry always act as starting point. There exist basic characteristic floor plan geometries (for instance comb type, linear type, central type) that are also used to categorize a given collection of buildings [Neu05]. A floor plan often implies the scale of a building, the style (e.g. gothic, modernism, deconstructivism) and its content (as use, function, appropriateness), see [Pev79].

In this paper, we present a shape retrieval technique for 3D architectural models that is tailored to the specific requirements of architectural data and overcomes the limitations of common 3D shape retrieval algorithms. We propose the use of *room connectivity graphs* to characterize building models according to the topology of their underlying floor plans. In these graphs, rooms are represented by attributed nodes. Attributes might for example be the area and the extent of the room. Connections between rooms are represented by edges. These edges are also assigned attributes characterizing the type of room connection, i.e. door or window. For each floor of the underlying building, one such room connectivity graph is computed.

The extraction of the room connectivity graphs is not trivial as two major requirements need to be fulfilled to make the approach feasible for a large number of arbitrary 3D building models:

- **Format independence** Especially in the domain of 3D CAD modeling there exists an abundance of different file formats. Room connectivity graph extraction should therefore rely on a simple model

representation that can easily be created from the underlying file format.

- **Robustness towards modeling errors** 3D meshes often contain T-vertices, intersecting triangles, wrong connectivity, etc. Additionally, we also encountered a lot of 3D building models in which separating walls between two rooms have gaps of several centimeters (see Figure 3a). These gaps were obviously not intended by the modeler and should therefore be eliminated.

To deal with these requirements, our method for the room connectivity graph extraction relies only on simple polygon soups that can easily be derived from almost any 3D representation. For each building story we compute several cuts between the polygon soup and a plane parallel to the ground. A 2D halfedge structure is built from the resulting line segments in which faces that correspond to rooms in the 3D building model can be determined. Unintended gaps between walls are closed using a technique that is inspired by topological simplification [SZL92]. Given one graph representation for each building story, determining those models in a database that are similar to a query object results in determining subgraph isomorphisms of the associated graphs with respect to node and edge attributes.

The contribution of our work is threefold. First, we introduce the room connectivity graph as a basic structure for 3D object retrieval in the architectural domain. Second, we describe a robust method to extract room connectivity graphs from polygon soup. This way we support most of the available 3D formats. Third, we show that the room connectivity graph concept is feasible for efficient and fast retrieval of architectural 3D models even in large databases.

2 RELATED WORK

In contrast to our approach, most methods for retrieval, classification and matching of shapes concentrate on arbitrary 3D objects instead of a special domain like architecture. In this section we focus on graph-based methods as they are most related to our approach. A more detailed introduction to 3D matching and shape retrieval can be found in [TV04].

Graph-based methods Graph-based shape retrieval methods rely on the extraction of certain geometric components and use a graph to show how these components are linked together. In contrast to our approach, these methods rely on low-level geometric 3D components rather than high-level semantic features like rooms, doors and windows.

Model graph-based approaches are mostly used for solid models like those in CAD applications. They describe solid objects in terms of connectivity of freeform surfaces (*Boundary Representation*) or as a set of geometric primitives that are connected by Boolean operations *(Constructive Solid Geometry)*. Local clique matching [EM03], [EMM03] or comparison of graph spectra [MPSR01] are used to determine the similarity of the graphs. In [ZTS02], VRML objects are segmented according to different decomposition techniques. The resulting patches are assigned basic shapes like planes and spheres. An attributed decomposition graph is built containing the determined shapes as nodes. Neighboring shapes are connected by edges. The similarity between two objects is computed by matching the associated decomposition graphs using error-correcting subgraph isomorphism.

Reeb graph-based methods rely on a function that is computed on the model surface which is supposed to be a compact manifold. The surface is divided into segments corresponding to intervals of this function. A skeleton graph in which the resulting segments are represented as nodes is built. Reeb graph-based methods are mainly used for matching of articulated objects [HSKK01], [TL07]. In [PSBM07], a robust method for fast Reeb graph computation is proposed that even allows for the use of non-manifold meshes.

Skeleton graphs of 3D shapes can be computed using topological skinning of voxel representations [BNdB99], medial axis transform, ridge point tracking [CSM05] or deformable model-based reconstruction [SLSK07]. Matching of two shapes is done by comparing the associated skeleton graphs using greedy bipartite graph matching [SSGD03] or by detecting subgraph isomorphisms using decision trees [LJI+03].

View-based methods The idea behind view-based methods is similar to our approach in a way that the 3D shape retrieval problem is transformed to several problems involving data of lower dimension. First, a set of view-dependent shape descriptors is computed. The similarity between two 3D objects is determined by the comparison of the associated descriptor sets. Descriptors based on pure 2D information like silhouettes [DYCO03], [MD06] are used as well as descriptors based on 2.5D depth-buffer information [Vra04]. Unfortunately view-based methods in general concentrate on shape instead of topology and are therefore not suited for retrieval in the architectural domain.

3 ROOM CONNECTIVITY GRAPH EXTRACTION

The extraction of room connectivity graphs consists of three steps, the detection of the building stories, the determination of story rooms and finally the determination of doors and windows connecting these rooms.

The prerequisite for our method is a polygon soup representation of the building. The units of measurement in the model as well as the plane corresponding to

the ground must be known. Note that this does not imply a severe restriction, as architectural models in general provide measurement information. The X-Y plane is the one corresponding to the ground in almost all cases.

3.1 Automatic Story Detection

The idea for the automatic detection of stories is given by the fact that in general all building stories are bordered by a flooring and a ceiling both of which result in patches that are parallel to the ground. The detection of floorings and ceilings according to these patches faces several problems. First, architectural models contain furniture and staircases that include patches which are also parallel to the ground. Second, the *floor construction*, i.e. the volume between a ceiling and the flooring of the room above, often contains additional large patches that are parallel to the ground, see e.g. the green planes in Figure 1. In the following we will show how to overcome these difficulties and identify those patches that correspond to floorings and ceilings.

First, we determine the set of polygons that are parallel to the ground. For each of these polygons we compute the distance to the ground. Those polygons that have approximately the same distance to the ground (we use a maximum deviation of 1mm) are put into a common bin. For each resulting bin we compute the sum of the included polygon areas. Patches caused by furniture or staircases are much smaller than those caused by floorings, ceilings and floor constructions. We therefore discard all bins with an area sum below a certain threshold t.

To get rid of those patches caused by the flooring construction, we use the fact that the thickness of flooring constructions is limited. It is generally determined by the span, the load and the construction material, see e.g. in [Hol07]. In buildings up to 4 floors the distance between a ceiling and the flooring of the room above usually is less than 0.4 meters, whereas multistory buildings, industrial buildings or hall structures may have floor profiles up to 2 meters and more. As we are mainly dealing with few-floors residential buildings, we apply another binning scheme and collect all patches within the distance of 0.4 meters in one bin. Within one bin, that patch with the maximum distance to the ground represents a flooring (see e.g. the blue planes in Figure 1) and that with the minimum distance represents a ceiling (see e.g. the red planes in Figure 1). All patches in-between belong to the flooring construction and are therefore discarded. In case there is only one patch contained, it represents both, the flooring of the upper story and the ceiling of the lower story.

We encountered some building models in which the flooring of the lowest story is not modeled. We therefore check whether the lowest detected ceiling patch is further apart from the ground than the minimum common story height (i.e. about 2.40m). In case the building has no flat roof, we test if the distance of the highest flooring to the ground is smaller than the bounding box distance to the ground, which means that an attic is located above the highest flooring.

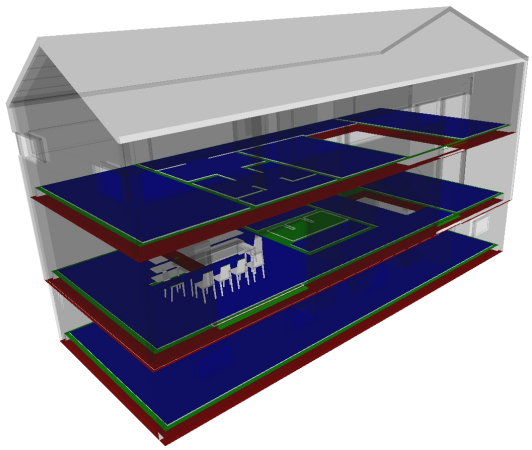


Figure 1: Building containing three stories. Red planes represent ceilings, blue planes represent floorings. The floor constructions to which the green planes belong are located between these planes.

3.2 Floor plan generation

As in architecture 2D floor plans are used to unambiguously show the spatial organization of a story, we restrict ourselves to the use of horizontal cuts instead of the 3D representation to efficiently extract rooms, windows and doors from building models. A horizontal cut of a 3D polygon soup results in a set of two-dimensional line segments that do not provide any connectivity. We transfer this set into a two-dimensional halfedge representation in which the line segments are connected with respect to their spatial relationship. Computing three cuts for each story at different height levels, we use the resulting halfedge structures to derive the room connectivity graph.

We represent each line segment that was created by the horizontal cut by two vertices v_i and v_j, a halfedge e_{ij} going from v_i to v_j and another halfedge e_{ji} going from v_j to v_i. We denote the set of vertices by V and the set of halfedges by E.

In the first preprocessing step, we determine the connectivity between the vertices. Two vertices v_i and v_j with $i \neq j$ are merged if the distance $d(v_i, v_j)$ between them is below a certain threshold ε (see Figure 2a). In our experiments we often encountered buildings containing double-sided modeled polygons, resulting in two line segments that are very close to each other. After the vertex merging it therefore might happen that the connection from vertex v_i to vertex v_j is represented by more than one halfedge. In this case we eliminate all halfedges from v_i to v_j but one.

In the second step, we determine the connectivity between vertices and halfedges. For each halfedge e_{ij} we determine the distance $d(e_{ij}, v)$ to every vertex $v \in V \setminus \{v_i, v_j\}$. If $d(e_{ij}, v)$ is below the above introduced threshold ε, e_{ij} and its opposite halfedge e_{ji} are split into two halfedges. The resulting halfedges are connected to the vertex (see Figure 2b). In all our experiments we used $\varepsilon = 1mm$.

In the next preprocessing step we eliminate intersecting halfedges. For all halfedges we check if it intersects with another halfedge. Intersecting halfedges e_{ij} and their opposite halfedges e_{ji} are split and a new vertex is inserted at the point of intersection (see Figure 2c).

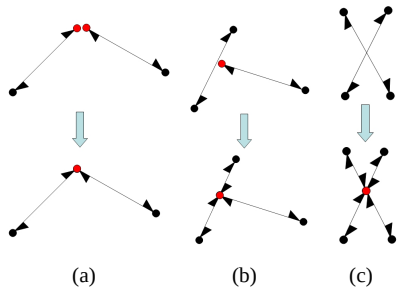


Figure 2: Line segment processing steps. a) Merging of close vertices. b) Halfedge split due to close vertex. c) Halfedge split and vertex insertion due to halfedge intersection.

Note that the resulting graph still contains unintended gaps due to modeling errors like that in Figure 3a. In the beginning of the room detection described in the next paragraph we do not take care of these inconsistencies but resolve them only after a first version of the room connectivity graph was constructed.

3.3 Room Detection

All rooms of one story are contained as faces in a halfedge graph constructed from a horizontal cut of the polygon soup that is located marginally below the story ceiling. At this height, no line segments corresponding to doors and windows will be contained in the cut. Given the halfedge graph representation of this cut, we determine the graph faces by walking along the halfedges, see Algorithm 1.

Considering the resulting faces, two cases must be distinguished. A face was either traversed inside (*inside face*) or outside (*outside face*). Inside faces correspond to either rooms or walls of the story building. Outside faces either represent rooms or walls lying completely inside another face without being connected to it or they correspond to the facade. To distinguish between inside and outside faces, we compute the angle between the incoming halfedge and the exiting halfedge with respect to clockwise direction for each of the n vertices during the face traversal. If all angles sum up to $(n-2)\pi$ (i.e.

Input Halfedge graph $G = (V, E, F)$

Function DetermineFaces(Graph G)
for all $e_{ij} \in E$ **do**
  if e_{ij} was not visited yet **then**
    Create new face f
    Assign f a pointer to e_{ij}
    TraverseHalfedge(e_{ij}, f, e_{ij})
    Add f to F

Function TraverseHalfedge(halfedge e, face f, starting halfedge e_{start})
Mark e as visited
Assign e a pointer to f
$v_{target} \leftarrow$ target vertex of e
$e_{next} \leftarrow$ next outgoing halfedge of v_{target} in clockwise direction with respect to e
if $e_{next} \neq e_{start}$ **then**
  TraverseHalfedge(e_{next}, f, e_{start})

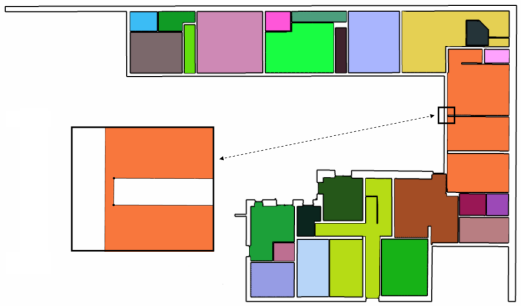
Algorithm 1: Face Determination

the sum of interior angles in a polygon), an inside face was determined, if the angles sum up to $(n+2)\pi$ an outside face was determined. After all faces have been extracted, one arbitrary vertex of each outside face is selected and it is tested if it lies inside one of the inside faces. By that we can determine faces corresponding to facades. Note that in case the building model consists of several tracts (see e.g. Figure 5), there also exist several facade faces for each of which we store a node in the room connectivity graph and assign it the attribute `facade`. All other outside faces are discarded.

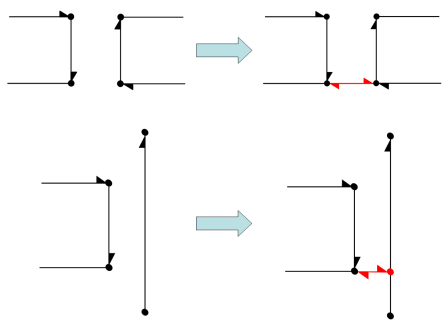
The inside faces represent either rooms or walls of the story building. In the next step we identify those faces that belong to rooms. First, all faces with an area of less than $1m^2$ are discarded, as according to DIN guidelines (see [Neu05]) rooms are larger than this size in general. Although this eliminates much faces, there still remain faces representing walls. To deal with those faces, we additionally consider the room extent. In general, walls have an elongated shape while rooms correspond to rather quadratic-shaped structures. Considering a wall of width w with an area A and a quadratic room of the same area A, then the extents of the wall E_w and of the room E_r are given by

$$\begin{aligned} E_w &= 2\left(\frac{A}{w} + w\right), \qquad (1) \\ E_r &= 4\sqrt{A}. \end{aligned}$$

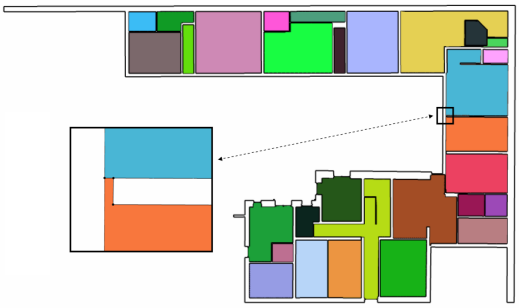
Common wall widths range from 6cm to 40cm. Using $w = 40$cm, the wall extent evaluates to $E_w = 5A + 0.8m$ which is much more than E_r for the remaining face areas with $A \geq 1m^2$. This implies that in general the ratio α between extent and area is larger for walls than for



(a) Unintended gaps between walls result in rooms that are not separated



(b) Gap-closing operations. Insertion of two halfedges between existing nodes and insertion of a new node with subsequent insertion of two halfedges.



(c) The same as a) after the gap closing.

Figure 3: Gap-closing

rooms. To exactly determine up to which α value faces should considered to be rooms, we divided all faces of 20 halfedge graphs created from different stories by hand into rooms and walls and computed their α values. Using Bayesian decision theory [DHS01] we compute the a-posteriori probabilities for rooms and walls, given the ratio observation α:

$$\begin{aligned} p(room|\alpha) &= \frac{p(\alpha|room)p(room)}{p(\alpha)}, \qquad (2) \\ p(wall|\alpha) &= 1 - p(room|\alpha), \end{aligned}$$

where $p(room)$ is given by the ratio between the total number of rooms and the total number of faces. To estimate the prior $p(\alpha)$ and the conditional probability $p(\alpha|room)$ we use the non-parametric Parzen window method [DHS01].

Given unclassified faces, we use the estimated a-posteriori probabilities to eliminate faces corresponding to walls. For each detected room a node is stored in the room connectivity graph and is assigned `area` and `extent` as additional attributes.

Figure 3a shows the problem of gaps between walls creating connections between rooms that actually were intended to be separated. In the data we found gaps ranging to a size of about 10cm. A naive approach to deal with this problem would be to increase the threshold ε that is used for the construction of the halfedge graph. By that, gaps up to a size of ε would be closed. The drawback of this method is that the larger ε is chosen, the more vertices are moved and the floor plan shape changes. Walls will disappear due to this simplification and structures like windows and doors will become more and more unrecognizable.

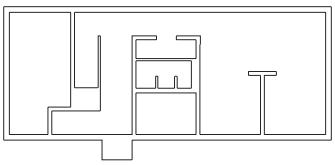
We follow a different approach that is inspired by topological simplification techniques which were invented for mesh simplification, see [SZL92]. While this method concentrates on determining simplification operations that preserve the topology, our algorithm identifies operations causing significant topology changes, i.e. closing gaps that create unintended room connections. A gap-closing operation can either be the insertion of two halfedges between two vertices or the insertion of a new vertex on a halfedge including the split of this halfedge and a subsequent insertion of two halfedges, see Figure 3b. We denote a gap-closing operation to be *valid* if it splits the face into two faces that still satisfy the room conditions (i.e. minimum area of $1m^2$ and $p(room|\alpha) > 0.5$). For each face we perform a sequence of virtual gap-closing operations, until a valid one is found and conducted. The resulting faces are tested for valid gap-closing operations recursively. We consider gaps up to a size of 375mm (minimum window width according to DIN, see [Neu05]), which is large enough to close all unintended gaps we found in our data and is still small enough to not close window openings unintentionally. By that, our method preserves the floor plan shape while handling all unintended gaps, see Figure 3c.

3.4 Door and Window Detection

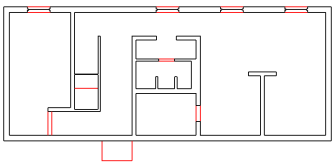
For the detection of building elements that connect rooms, we compute two more horizontal cuts with the polygon soup. The first cut is located marginally above the story flooring, such that it will cut through the doors but not through the windows. The second cut is located at breast height (i.e. 1.40m above the flooring), such that it will cut through both, the windows and the doors. For both cuts, rooms are detected in the above described way. Windows and doors produce geometric inconsistencies in a wall. By determining the inconsistencies

between the three cuts, these elements can be identified. Inconsistencies between the cut at breast height and the cut below the ceiling indicate the presence of either windows or doors. If a corresponding inconsistency is also found in the cut above the flooring, it indicates the presence of a door. In Figure 4 we show three cuts of one story at the different height levels and the inconsistent line segments that are contained in the high cut but not in the lower cuts.

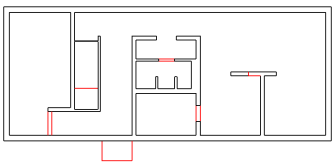
Once the inconsistent line segments have been identified, we must determine whether they really correspond to doors or windows as there are also inconsistencies caused by other structural differences. Figure 4 shows that each door and window causes one pair of inconsistent segments in the two lower cuts. According to DIN standards (see [Neu05]), there exist minimum widths for windows (375mm) and doors (55cm), such that inconsistent segments caused by these elements must provide according minimum lengths. All pairs of inconsistent line segments are tested whether they provide these characteristics. Once a pair of segments is identified to be a door or a window, we determine to which faces the according halfedges in the cut below the ceiling belong to (note that this might also be the face representing the facade). We finally add an edge in the room connectivity graph connecting the room nodes representing the determined faces. As an attribute, the edge is either assigned `door` or `window`.



(a) Cut below the ceiling



(b) Cut at breast height



(c) Cut above the flooring

Figure 4: Cuts through the story of a building. In b) and c) those line segments that are only contained a) are shown in red. Inconsistent segments caused by doors are contained in both lower cuts whereas inconsistent segments caused by windows can only be found b).

4 RETRIEVAL WITH ROOM CONNECTIVITY GRAPHS

The room connectivity graph represents a basic structure for retrieval of building models in databases. With this representation at hand, similarity between an attributed query graph and a building model in a database can be determined by checking whether the query graph is isomorphic to a subgraph of the room connectivity graph of the building model, with respect to the attributes. Subgraph-isomorphism in general is known to be NP-complete [Coo71]. However, there are two aspects that make this approach feasible to this particular retrieval problem. First, the graphs we consider are in general not very large. Second, the attributes that are assigned to the nodes and the edges efficiently accelerate the graph matching process.

While determining a subgraph isomorphism between an attributed query graph and a room connectivity graph, we only match nodes and edges of the two graphs if their attributes are similar. We therefore use global constraints describing the amount of similarity the nodes or edges must provide such that they can be matched. Considering the edge attributes we introduced, they result in a constraint involving the edge type: Two edges can only match if they represent the same structure, that is either window or door. Considering the node attributes, they result in three constraints, that is: First, two nodes can only match if they represent the same structure, that is either a room or the facade, second, two nodes can only match if the area of their associated rooms differs only by a certain amount d_A of square meters and third, two nodes can only match if the extent of their associated rooms differs only by a certain amount d_E of meters. Although we restricted our experiments to these four constraints, the attributes introduced so far allow for the construction of further constraints involving for example the ratio between area and extent. Note that all constraints are global, i.e. they count for all the nodes and edges, respectively.

In case the graphs to be matched both provide a facade node, these nodes should be matched first to further accelerate the retrieval. If no such node exists in the query graph, an arbitrary node can be chosen.

5 RESULTS

For our experiments we used a database consisting of 100 residential building models that were created by students of architecture. For the room connectivity graph extraction all thresholds were chosen as described above. The threshold t for the automatic story detection was set to $10m^2$. The total processing of the whole database took about 7 minutes (see Table 1) and

 ISBN 978-86943-15-2

created 289 room connectivity graphs, one for each story. In Figures 5 to 9 examples of these graphs are shown. Each colored face represents a room node in the graph. Edges between room nodes are shown including the type connection (either `door` or `window`). Figure 9 shows a problem we discovered in some building models. The arrow points to a structure representing a window. In contrast to the other windows in this story, the window pane and the facade are exactly in one line. Therefore, the wall structure provides no inconsistency and the window was not detected. Timings for the extraction of the depicted room connectivity graphs can be found in Table 1.

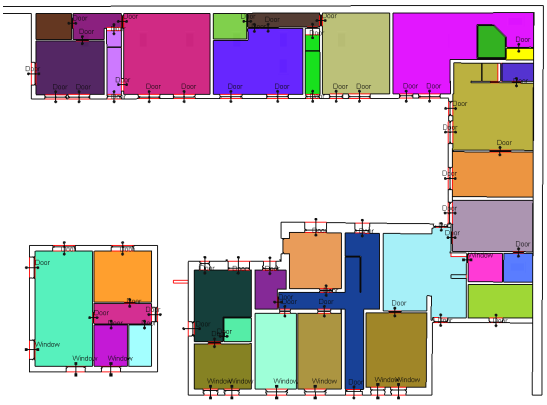


Figure 5: Ground floor of a complex building containing two tracts. The resulting room connectivity graphs consists of two connected components.

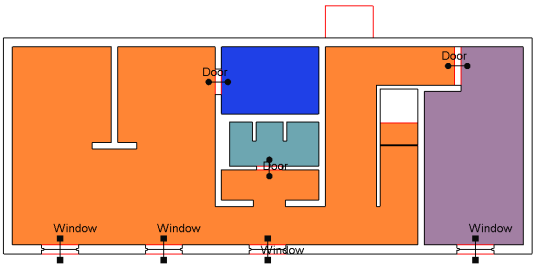


Figure 6: Cellar of a residential building.

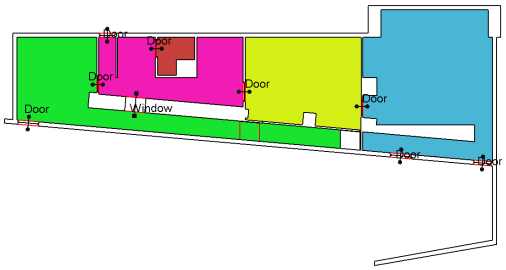


Figure 7: Ground floor of a residential building.

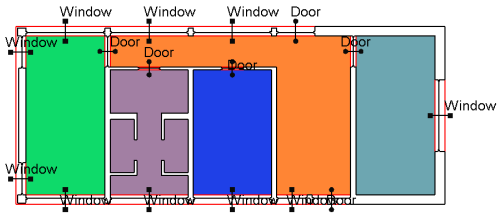


Figure 8: Ground floor of a residential building.

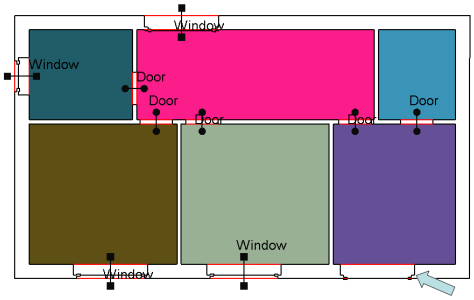
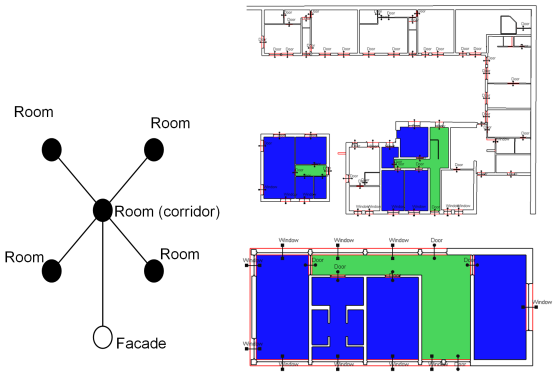


Figure 9: First floor of a residential building. The arrow points to a window that could not be detected by our method as the window pane is positioned exactly in one line with the facade.

Object	#Graphs	Extraction	Retrieval
Figure 5	2	10.451s	< 10ms
Figure 6	3	0.967s	< 10ms
Figure 7	4	4.828s	< 10ms
Figure 8	3	1.077s	< 10ms
Figure 9	2	0.421s	< 10ms
Database	289	436.512s	0.173s

Table 1: Statistics on room connectivity graph extraction. *#Graphs* describes the number of room connectivity graphs that were extracted from the underlying 3D model. The time that was spent for the extraction is given by *Extraction*. *Retrieval* states how long it took to search the extracted room connectivity graph for the query graph shown in 10a. All experiments were run on an AMD Athlon^{TM} 64 with 2 GHz and 2 GB RAM.

Figure 10 shows results of our retrieval experiments. As query we used a graph representing the structure of a typical apartment including a corridor that leads to four rooms (Figure 10a). The results shown in Figure 10b were generated using the constrained subgraph matching as described in Section 4 without considering the area and extent attributes. In other experiments we also included these attributes and could thereby further refine the retrieval result. The time consumption for searching all 289 room connectivity graphs for the query graph in Figure 10a was 0.173s (see Table 1 for more details).



(a) (b)

Figure 10: a) Graph representing a typical apartment. All edges represent doors. b) Amongst the floor plans in Figure 5 to 9, the query graph was found twice in the floor plans of Figure 5 and once in that of Figure 8. The corridor room is shown in green, the other rooms are shown in blue.

6 CONCLUSION

In our work we presented a new method to efficiently characterize architectural data by room connectivity graphs. The method only relies on polygon soup representations and is therefore feasible for a large number of 3D models. Our approach is robust towards modeling errors like unintended gaps between rooms. In our experiments we showed that the extracted room connectivity graphs can be used for fast and efficient shape retrieval in architectural databases.

Additionally to the room area and extent it would also be interesting to include attributes that describe the shape of the room more precisely. 2D descriptors like Zernike moments or centroid distance-based ones could therefore be used to further enhance retrieval results. Future work should also consider building units that connect different stories. By that, the room connectivity graphs of each story will be interlinked by edges representing staircases and elevator shafts.

ACKNOWLEDGEMENTS

This work was partially funded by the German Research Foundation (DFG) under grant GZ 554975(1) Oldenburg BIB 48 OLof 01-02 *Probado*.

REFERENCES

[BNdB99] Gunilla Borgefors, Ingela Nyström, and Gabriella Sanniti di Baja. Computing skeletons in three dimensions. *Pattern Recognition*, 32(7):1225–1236, 1999.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM Press.

[CSM05] Nicu D. Cornea, Deborah Silver, and Patrick Min. Curve-skeleton applications. In *IEEE Visualization*, page 13. IEEE Computer Society, 2005.

[DHS01] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley, 2001.

[DYCO03] M. Ouhyoung X.-P. Tian Y.-T. Shen D.-Y. Chen and M. Ouhyoung. On visual similarity based 3d model retrieval. In *Eurographics*, pages 223–232, Granada, Spain, 2003.

[EM03] M. El-Mehalawi. A database system of mechanical components based on geometric and topological similarity. part 2: indexing, retrieval, matching, and similarity assessment. *Computer-Aided Design*, 35(1):95–105, January 2003.

[EMM03] Mohamed El-Mehalawi and Allen Miller. A database system of mechanical components based on geometric and topological similarity. part 1: representation. *Computer-Aided Design*, 35(1):83–94, January 2003.

[Hol07] Klaus Holschemacher. *Entwurfs- und Berechnungstafeln*. Bauwerk Verlag GmbH, Berlin, 2007.

[HSKK01] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proceedings of ACM SIGGRAPH*, 2001.

[LJI+03] K. Lou, S. Janyanti, N. Iyer, Y. Kalyanaraman, S. Prabhakar, and K. Ramani. A reconfigurable 3d engineering shape search system part ii: database indexing, retrieval and clusturing. In *DETC*, 2003.

[MD06] Ameesh Makadia and Kostas Daniilidis. Light field similarity for model retrieval. In *SHREC2006, 3d Shape Retrieval Contest*, June 2006.

[MPSR01] David McWherter, Mitchell Peabody, Ali C. Shokoufandeh, and William Regli. Database techniques for archival of solid models. In *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 78–87, New York, NY, USA, 2001. ACM Press.

[Neu05] Ernst Neufert. *Bauentwurfslehre*. Vieweg, 38th edition, September 2005.

[Pev79] Nikolaus Pevsners. *A History of Building Types*. Princeton University Press, December 1979.

[PSBM07] Valerio Pascucci, Giorgio Scorzelli, Peer-Timo Bremer, and Ajith Mascarenhas. Robust on-line computation of reeb graphs: simplicity and speed. *ACM Trans. Graph.*, 26(3):58, 2007.

[Sch04] Friederike Schneider. *Floor Plan Manual Housing*. Birkhäuser, third edition, April 2004.

[SLSK07] Andrei Sharf, Thomas Lewiner, Ariel Shamir, and Leif Kobbelt. On–the–fly curve-skeleton computation for 3d shapes. In *Eurographics*, pages 323–328, Prague, september 2007.

[SSGD03] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval, 2003.

[SZL92] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. *Computer Graphics*, 26(2):65–70, 1992.

[TL07] Gary K. L. Tam and Rynson W. H. Lau. Deformable model retrieval based on topological and geometric signatures. *IEEE TVCG*, 13(3):470–482, 2007.

[TV04] J. W. Tangelder and R. C. Veltkamp. A survey of content based 3d shape retrieval methods. In *SMI '04: Proceedings of the Shape Modeling International 2004 (SMI'04)*, pages 145–156, Washington, DC, USA, 2004. IEEE Computer Society.

[Vra04] Dejan V. Vranić. *3D Model Retrieval*. PhD thesis, University of Leipzig, Germany, 2004.

[ZTS02] Emanuel Zuckerberger, Ayellet Tal, and Shymon Shlafman:. Polyhedral surface decomposition with applications. *Computers and Graphics*, 26(5):733–743, October 2002.

Automatic Zoom and Pseudo Haptics to Support Semiautomatic Segmentation Tasks

Wolfram Schoor
Fraunhofer Institute for Factory Operation and Automation (IFF)
Sandtorstrasse 22,
39106 Magdeburg, Germany
wolfram.schoor@iff.fraunhofer.de

Felix Bollenbeck
Leibniz Institute of Plant Genetics and Crop Plant Research (IPK)
Corrensstrasse 3,
06466 Gatersleben, Germany
bollen@ipk-gatersleben.de

Marc Hofmann
Fraunhofer Institute for Factory Operation and Automation (IFF)
Sandtorstrasse 22,
39106 Magdeburg, Germany
marc.hofmann@iff.fraunhofer.de

Rüdiger Mecke
Fraunhofer Institute for Factory Operation and Automation (IFF)
Sandtorstrasse 22,
39106 Magdeburg, Germany
ruediger.mecke@iff.fraunhofer.de

Udo Seiffert
Leibniz Institute of Plant Genetics and Crop Plant Research (IPK)
Corrensstrasse 3,
06466 Gatersleben, Germany
seiffert@ipk-gatersleben.de

Bernhard Preim
Otto von Guericke University,
Universitätsplatz 2,
39106 Magdeburg, Germany
preim@isg.cs.uni-magdeburg.de

ABSTRACT

This paper presents a new technique for navigating large amounts of biological image data during segmentation. Digitized microtome tissue slices of barley grains constitute the image data. Automatic zoom is dependent on mouse speed, thus enabling users to navigate an image more efficiently and with a tighter focus during segmentation. The user remains oriented and smoothly moves through the data. Furthermore, pseudo haptic feedback based on image cost map data compensates for user inaccuracies generated, for example, by shaky hands. A prototype was implemented and tested. An informal usability study revealed that users are inclined to prefer automatic zooming and pseudo haptics for semiautomatic segmentation tasks.

Keywords: Interaction techniques, zoom techniques, pseudo haptic, segmentation.

1 INTRODUCTION

Navigation techniques support the exploration of huge information spaces by mapping a subset of information onto the limited screen space with the aim of plausibly mapping transitions between successive subsets to meet user requirements. This can be achieved by panning (scrolling) and zooming in the two-dimensional continuous space. Zooming allows a user to view specific targets on different scales. Panning can be used to visit different locations on the same scale. Panning and zooming are the de facto standard for the navigation of large information spaces [vWN03].

Typical interfaces for manual and semiautomatic image data segmentation lack effective navigation. Users have to zoom in for detailed segmentation in difficult regions and then zoom out and pan for rapid segmentation in regions with clearly defined edges. Using GUI buttons to adjust the zoom to the required scale increases the time expended and can waste additional resources. A rate-based scrolling interface such as [ZSS97] can map input device speed to zooming speed. This is done with a mouse wheel for instance. While these interfaces require less time, they can also cause disorientation in the information space [JF98]. Another important aspect of navigation is the efficient mapping of a user's interaction onto the screen. Pseudo haptic feedback [LBE04], [DLB$^+$05] can enhance user interaction by supporting a user on the basis of the underlying image data. Undesired movements such as hand tremors are diminished. Speed and accuracy increase. Figure 1 is an image for such a segmentation task.

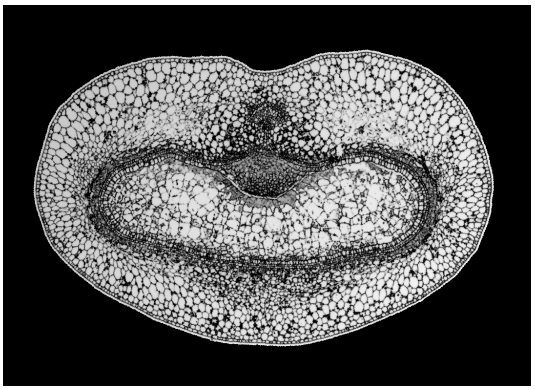


Figure 1: Slice of *Hordeum vulgare*

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 2008 conference proceedings
WSCG'2008, February 4 – February 7, 2008
Plzen, Czech Republic.

Copyright UNION Agency – Science Press

2 RELATED WORK

2.1 Complex Image Data Segmentation

Different segmentation methods are employed in different fields of application. The live wire technique [SPP00], [GDGC02], also known as intelligent scissors [MB99], [MB98], was selected for this research.

All image segmentation algorithms are intended to deconstruct an image into segments with specific *meaning* for a given application [HS85] or, expressed more technically, to identify regions that *uniformly* relate to a specific criterion such as texture or image intensity [HS85]. The literature, specifically [LM01], [Hah05] or [Kan05], presents a variety of classification approaches. With regard to interaction, this paper classifies image segmentation according to the following techniques.

Manual segmentation is the easiest image segmentation technique and is commonly used for high level segmentation tasks, especially when a priori knowledge of the image data is required. The user specifies a set of control points that will be connected afterward. The contour can additionally be smoothed by familiar interpolation techniques [Far88]. The main disadvantage of this technique is the time and concentration that precise segmentation requires since numerous control points have to be specified manually.

Automatic segmentation algorithms can be far more efficient than manual segmentation but can also fail to correctly segment the objects of interest, resulting in an inacceptable loss of accuracy. The advantage of these algorithms is that they do not require user input for segmentation. Calculation time depends on the complexity of the algorithm and ranges from real time to several hours [BBS+06]. Lucheese et al. [LM01] and Hahn [Hah05] survey automatic methods, including histogram threshold, boundary-based, clustering and neural network algorithms.

Semiautomatic segmentation techniques represent a compromise between powerful low level segmentation algorithms and the knowledge required of the user. The user has to set a limited number of parameters specific to the dataset or support the segmentation process with preexisting knowledge. Semiautomatic methods function efficiently and accurately where automatic methods fail or cannot be applied. While semiautomatic algorithms are more time consuming than most automatic algorithms, the results of the segmentation process can nevertheless be used as training data for other automatic segmentation algorithms. Semiautomatic algorithms include region growing [AB94], [KWT88] and live wire [MB98], [SPP00]. Given the nature of the datasets considered here, difficult to predict shapes like those created by snakes are not adequate. Apparent gaps in the shapes also make region growing unfeasible. Since it is more easily controlled and more predictable, interactive segmentation that iteratively generates the correct curve segments promises better results. For these reasons, live wire was chosen for further investigations of semiautomatic segmentation algorithms.

2.2 Navigation and Interaction Techniques

The datasets consist of large and highly detailed images. The goal of a semiautomatic segmentation application is to enable users to access data with different levels of detail easily and efficiently as well as to maintain an overview of the data. The limited screen space necessitates different approaches, all of which should fulfill Shneiderman's maxim of visual seeking: "Overview first, zoom and filter, then details on-demand." [Shn96].

One such method is the space deformation approach. All information is visible simultaneously and important areas become magnified similar to the fisheye technique [Fur86]. The advantage of this approach is its simultaneous provision of focus and overall context. However, the magnification remains bound to the position of the cursor. Large scales produce substantial distortions around a magnified area. Nonetheless, this information may be important for the next path segments. The zoom's amplitude of magnification must be adjusted manually.

Another method is the information deformation technique [AS07], [WGCO07], which shrinks an image space content-aware to fit the screen space. One problem with these algorithms is that only limited information loss can be compensated without losing global context or experiencing undesired effects. Thus, global information can be mapped to the limited screen space only up to a low zoom level.

An overview window can generate focus and context within the representation of an image on a specific zoom level. Again, other zoom levels must be adjusted manually.

These methods can be supported by speed-dependent automatic zooming (SDAZ) as first set forth by Igarashi and Hinckley [IH00] in 2000, although already in use in such games as GTA (DMA Design 1997). Other examples of SDAZ are [vWN03], [vWN04] or [Kre05]. The main challenge is automatically adjusting the zoom level on the basis of a user's speed so that the user can navigate within the image data with a tighter focus and fully concentrate on the task at hand without being too distracted by navigation. SDAZ has improved navigation speed in numerous applications. Moreover, its fewer different interaction steps make it user friendly. Smooth transitions between successive subsets of information space are essential to render the relation to movement and the global information space comprehensible to the user.

Another well known interaction approach is pseudo haptic feedback as applied in [LBE04] or [DLB+05].

The change in the control to display ratio (C/D ratio) conveys an impression of a surface's structure or ruggedness to users, similar to sensing bumps or hills. The well established principle of gravity active points [BS86] is closely related to pseudo haptics and was used to position objects precisely.

3 DATA AQUISITION

Each of the datasets segmented consisted of approximately 2000 slices of a barley corn (*Hordeum vulgare*) $3\,\mu m$ high. Captured with a color CCD camera, the images originally measured 1600 x 1200 pixels with a pixel size of 1.83 x 1.83 μm. Given the very high correlation of the color values (see the line approximation in the RGB space in Figure 2), the images were converted to a gray scale with a depth of 8 bits for better performance.

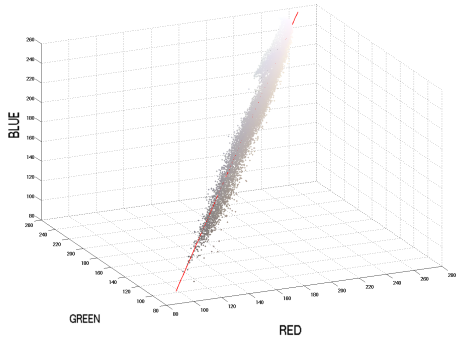


Figure 2: Distribution of the color values

In addition, automatic background elimination and alignment were applied to the datasets [GDB+07]. This reduced the size of each dataset to 5 GB.

An initial analysis of the data and related datasets revealed a high degree of complexity in the images. The image data contains similar, no or only partially continuous edges and the intensity changes in many images. Only vague information exists about the resulting structures being segmented (see Figure 1 and especially the areas encircled in red in Figure 3).

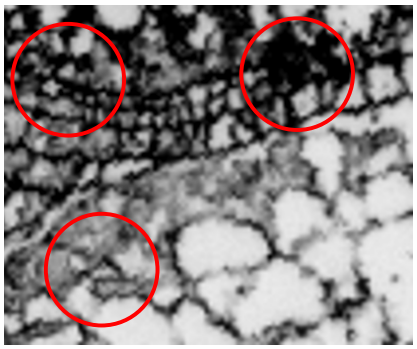


Figure 3: Detail of a highly complex slice of a microtome tissue

Figure 4 illustrates the complete segmentation of an image slice, manually segmented with the commercial visualization software package Amira [Kon07]. The total segmentation time for all tissues in Figure 4 was approximately 23 minutes.

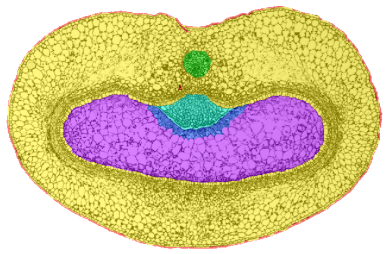


Figure 4: Complete segmentation of all basic tissues

4 AUTOMATIC ZOOMING AND PSEUDO HAPTIC FEEDBACK

Concepts were formulated and implemented to adapt and enhance the interaction and navigation techniques relevant for image segmentation.

4.1 Concept

Automatic zooming involves identifying constraints that describe the relation between zooming and cursors speed. To calculate the zoom level of the next time step (see Equation 1), certain terms must be taken into account. The preceding zoom level $Zoom_{OLD}$ is scaled with a $Speed_{AVG}$ term averaging cursor speed over a distinct time interval, a $Trend$ term calculated from recent time intervals and a $Prop$ term derived from the costs of the propagated path in the cost map.

$$Zoom_{NEW} = Min(0, (Max(1, (Zoom_{OLD} \cdot Speed_{AVG} \cdot Trend \cdot Prop))) \quad (1)$$

Another key concept is the integration of pseudo haptic feedback in segmentation for which the control to display ratio is an important term, which describes the relationship between the movement of the control device (e.g. mouse) and the actual movement of the cursor on the screen. The control to display ratio's nonlinear anisotropic behavior supports users when they are performing difficult segmentation tasks. As opposed to the calculation of segmentation in the direction of the gradient (orthogonal to the direction of the edge) with a larger C/D ratio, segmentation along an edge is calculated with a fixed C/D ratio. Consequently, users must amplify their movements to obtain the same results from cursor movement in the screen space. This eliminates small, undesired user hand motions such as shaking in the direction of segmentation. Since user

speed is related to cursor movement, this technique indirectly influences the automatic zoom. Since the C/D ratio is initially a fixed value in a system, only changes of the C/D ratio must be calculated. This can be done based on cursor position (Equation 2).

$$CDR_{NEW} = \alpha \cdot \sqrt{\delta v^2 + Cost_{NORM}^2} \tag{2}$$

where δv is the magnitude of the direction change of the cursor in a time step, α a constant indicating the scale factor and $Cost_{NORM}$ the normalized costs from the cost map (see Equation 3). Figure 5 visualizes the change of the C/D ratio, the red arrows illustrating the movement of the cursor in the 2-D information space and the blue arrows indicating the respective movement of the input device necessary. The influence of the 3-D cost distribution on the change of the C/D ratio during segmentation counter the direction of the edge (i.e. uphill) is easily recognizable.

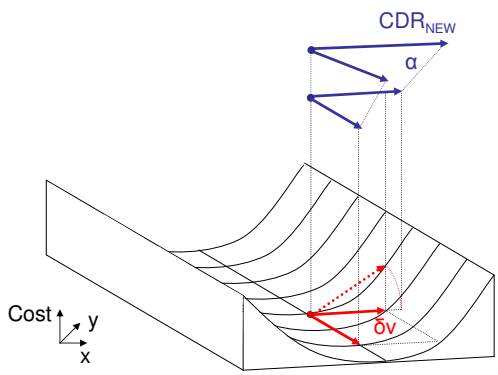


Figure 5: C/D ratio increases in relation to the image costs

4.2 Implementation

The algorithms were implemented in C++ and the graphic user interface was generated with WxWidgets 2.8[1].

The live wire algorithm was implemented largely based on [GDGC02]. Good results were achieved by using the parameters Laplace with a weight wL of 20 ($5x5$ and $9x9$ filter), gradient magnitude with a weight wG of 30 and the point-line distance with a weight wD of 1 to generate the cost map as in Equation 3.

$$\begin{aligned} Cost = {} & wL \cdot LaPlaCost \\ & + wG \cdot GradientCost \\ & + wD \cdot (0.25 \cdot PointLineDist^2) \end{aligned} \tag{3}$$

Furthermore, an image pyramid was used to speed up algorithm performance to determine initial cost. Moreover, an exact cost calculation and cost update is preferably performed in the neighborhood of the cursor by overlapping tiles. Since OpenGL[2] can only handle images of a limited texture size, each image had to be subdivided into manageable texture sizes, whereas the image is loaded completely into the RAM for calculation and a tile class executes the calculation loops in the respective RAM limited by tile size.

Control points were introduced to attach correct presegmented parts of the shape. When the current path segment reaches a predefined length, control points with a specific path distance as well as the preceding path segment are set. These minor changes make real time interaction with the image data possible. This is a fundamental prerequisite to conducting usability studies based on the algorithms presented here.

The following is a basic technical description of the automatic zoom's kernel.

```
Pseudocode of the zoom function:

//initialize sum of the speeds with 0
sumSpeed = 0;
//initialize number of sampled speeds with 0
sumCount = 0;
//initialize change of zoom with 0
deltaZoom = 0;

//test if the mouse was moved
if dist(p, q) > 0 do
   //increase sumSpeed by speed
   //of last moved distance
   sumSpeed += dist(p,q) / passedTime;
   sumCount++;
end

//evaluate average speed every 500 msec
if 05-Sec-Timer do
   //calculate avg speed over last 500 msec
   avrSpeed = sumSpeed / sumCount;
   //reset sum of speed
   sumSpeed = 0;
   //reset count of sampled speeds
   sumCount = 0;
   //check if avg speeds exceeds maximum value
   if avrgSpeed > maxSpeed do
      //decrease deltaZoom
      deltaZoom -= 5;
   end

   //check if avg speeds is smaller
   //than minimum value
   if avrgSpeed < maxSpeed do
      //increase deltaZoom
      deltaZoom += 5;
   end
   //restart the 3 seconds timer
   03-Sec-Timer.restart();
end

if 01-Sec-Timer do
   //check if deltaZoom exceeds
   if  |deltaZoom| > maxDelta do
      //if deltaZoom is smaller than 0 zoom out
      if deltaZoom < 0 do
         decreaseZoom by 1;
      //if deltaZoom is bigger than 0 zoom in
```

[1] www.wxwidgets.com

[2] Hardware dependent.

 ISBN 978-86943-15-2

```
        else
            increaseZoom by 1;
        end
    end
    //restart the 1 seconds timer
    01-Sec-Timer.restart();
end
```

In the brief description of pseudo haptic feedback below, the new C/D ratio ensues from the related cost map and the current mouse movement.

```
Pseudocode for pseudo haptic feedback:


// 2-D vector from last mouse position
// to current mouse position
deltaMouse = currMousePos - lastMousePos;

// calculate normalized cost from last
// mouse position to current mouse position
currCost = CalcCost(lastMousePos, currMousePos);

// calculate length of mouse change
deltaV = Length(deltaMouse);

// calculate new C/D ratio
ratio = CDRmax * sqrt(deltaV*deltaV +
        currCost*currCost);

// modify new mouse position via ratio
currMousePos = lastMousePos + deltaMouse * ratio;
```

5 USABILITY STUDY

A preliminary usability study investigated the strengths and limitations of automatic zooming and pseudo haptic feedback for semiautomatic segmentation.

5.1 Setup and Object of Investigation

The participants of the study were divided into four groups based on their abilities and ages[3] (see Table 1). Three different segmentation methods (manual segmentation, conventional live wire and improved live wire with automatic zoom and pseudo haptics) were tested. The order of segmentation methods was altered for each participant so that the learning rate influenced each segmentation time.

Number	Age	Skills
1	Young	Good
2	Young	Moderate
3	Medium	Good
4	Medium	Moderate

Table 1: Study group

The first aspect demonstrated was the completion time of a predefined segmentation task. The pool of participants consisted of sixteen untrained individuals. Hence, the training scenario consisted of a specially generated dataset. The original data was enriched with a superimposed ideal segmentation shape of a pericarp, the tissue surrounding a seed that develops from the ovary wall of a flower. An initial manual segmentation performed by four experts defined the gold standard segmentation.

Augmentation without zooming is represented as a thin dashed line. The thickness of the line is proportional to the magnitude of the edge. When the zoom was used in critical segmentation regions, several lines were inserted to compel the user to apply the global context. Figure 6 shows two different zoom levels of the dataset.

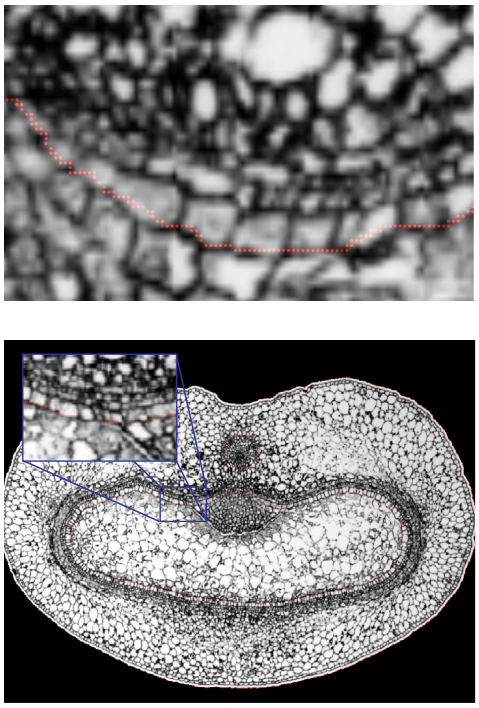


Figure 6: Top: Detail of dataset with zoom
Bottom: Detail without zoom

The second aspect is the accuracy of the segmentation results.

Errors were quantified as the ratio of misclassified pixels to the total number of pixels (error rate). Standard deviation or the mean from the gold standard are a more informative comparative characteristic of accuracy. The point with the closest Hausdorf distance was selected to calculate this.

5.2 Observations and Comments

In their responses to an informal questionnaire, most participants indicated that the conventional pan and zoom mechanism buttons/scrollbars/mouse wheel are easy to use but require too many discrete manipulation steps to navigate between two separate points with different zoom levels. Thus, users sometimes lose their focus on the task at hand. In other words, navigation

[3] Under twenty-nine years of age = young.

requires too much attention from users. Furthermore, users complained, that the discrete steps in the visualization of different zoom levels were laborious. Hence, many users avoided manual zoom changes. All participants experienced the automatic zoom as unfamiliar in the beginning but were able to adapt quite quickly. Users were able to navigate with a tighter focus and concentrate on the segmentation task. Pure manual segmentation of the dataset was a Sisyphean task in comparison with the two semiautomatic segmentation methods with speed-dependent automatic zooming and pseudo haptic feedback or conventional interface control. Virtually all the participants rated the pseudo haptic feedback as an excellent solution enabling direct manipulation with small control amplitudes and the elimination of inaccuracies.

5.3 Results

The formal user study delivered virtually all the preliminary results. Both methods using semiautomatic segmentation facilitate significantly faster completion times than manual selection. Speed-dependent automatic zoom combined with pseudo haptic feedback is generally not faster than the conventional method. This slightly contradicts the subjective impressions of the majority of the participants (see Section 5.2) and may be attributable to the dynamics of automatic zooming and panning. Figure 7 presents the results of the completion time test for each segmentation method. The standard deviation of the completion time is visualized with continuous T-lines for each group.

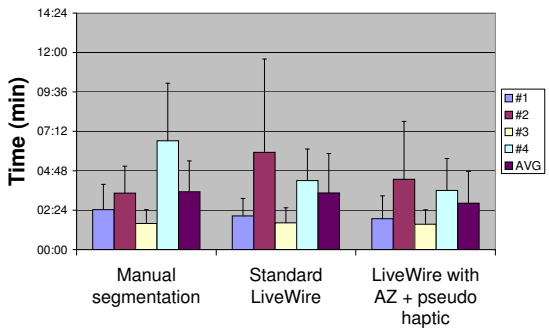


Figure 7: Completion times of manual segmentation, standard live wire and live wire with automatic zoom (AZ) and pseudo haptics

Groups 2 and 4 experienced consistently longer segmentation times with strong deviations (nearly equaling the average segmentation times). Groups 1 and 3 had far better segmentation times than groups 2 and 4, which consisted of test persons with average computer skills. A significant and unexpected finding was that experienced users' segmentation times hardly varied (only 6 %) regardless of whether they were using the improved live wire with automatic zoom (AZ) and pseudo haptics or the conventional live wire. On the other hand, members of groups 2 and 4 were on average 28 % and 14 % faster with the automatic zoom and pseudo haptics than with the standard live wire method. This may have been because the automatic navigation provides inexperienced users better support, whereas experienced users are able to navigate and simultaneously concentrate on their current tasks (just as in game playing for example). Disregarding extreme individual deviations, the ages of users with the same level of knowledge appear to play no role in the average segmentation time. The average segmentation times of all users are presented in Figure 7 (violet bars).

Figure 8 (bottom) lists the error rates of all participants, i.e. the ratio between misclassified pixels to the total number of pixels. Figure 8 (top) presents the average pixel deviation (mean) and the standard deviation from the mean in relation to the gold standard segmentation. The value of the standard deviation from the mean indicates local misclassifications. Since accuracy did not deviate strongly among the user groups, only the averaged results per segmentation method are presented here.

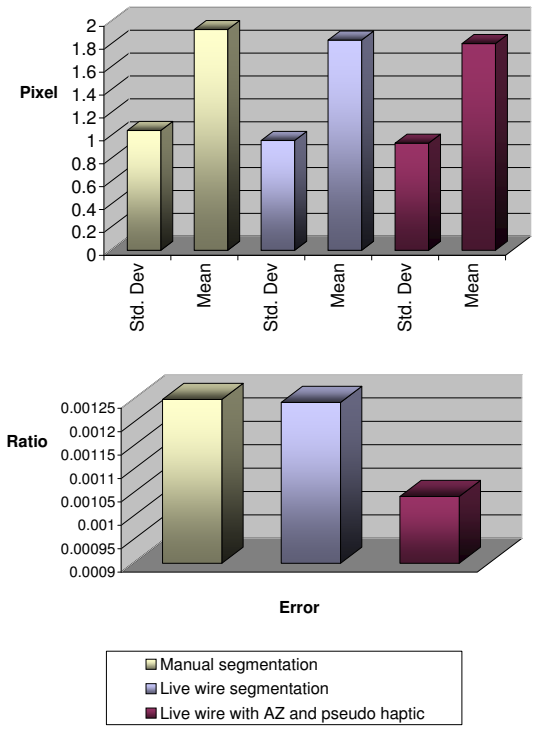


Figure 8: Test subject' accuracy with the segmentation methods tested (top) mean and standard deviation (bottom) error rate

Figure 9 visualizes a typical segmentation process. The transitions in the automatic zoom curve are smoother than in the manual zoom curve. Furthermore, the completion time is significantly shorter.

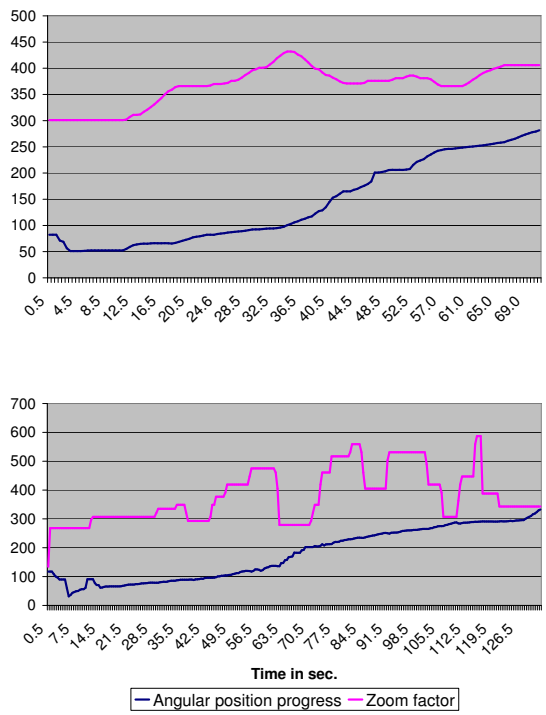


Figure 9: Comparison of one representative participant's unwound segmented shape with automatic zoom (top) and with manual zoom adjustment (bottom)

6 FUTURE DIRECTIONS

One challenge will be to design plausible transitions between different zoom levels that allow users to change speeds quickly without causing any confusion. At fast speeds, different users sometimes expect different system feedback from the same user action. Further research is therefore required, especially on adjusting the zooming parameters to individual users. Graphically manipulating the zoom function is planned, thus making it possible to save results in a specific user file.

Development of a board-like input device for two hands is also planned: Such a device would increase the speed and accuracy of segmentation as Kron and Schmidt [KS05] have demonstrated. Another approach deserving further investigation is inverse panning: The user remains stationary and the image data moves through the screen space, similar to cutting wood with a jigsaw.

Flicking movements represent another technique [MC03], [DAZ99] and can be employed to quickly and precisely access a single control from a small group of plausible candidates. Whether the integration of additional navigation controls such as a default zoom level or a last contour segment reset can be suitably mapped to distinct mouse movements deserves more investigation.

While the informal user study demonstrated the capabilities expected for a biological test dataset, the results have to be verified for different segmentation scenarios especially in the context of practical use. Furthermore, the refined prototype algorithms must be studied additionally to determine whether the haptic or the automatic zoom or a combination of the two improve the standard live wire technique most.

7 CONCLUSION

This paper introduces a new technique that improves semiautomatic segmentation. The automatic zoom is based on a user's interaction speed and the underlying image data. Slow user input produces an in-depth view of an image, thus making it easier to segment unclear or overly detailed regions. Fast user input indicates strong edges that can typically be segmented easily with live wire. This can be done on a lower zoom level on which the user can still recognize the edges and maintain an overview. Pseudo haptic feedback additionally supports accurate segmentation in complex areas. Movement toward the edge direction is penalized with a larger C/D ratio, thus increasing control device amplitude.

In an informal usability study, most users preferred this technique over segmentation with manual zoom. Moreover, they described their impression of pseudo haptics as a "magical force" that binds the cursor to the correct position in an image or leads it there.

In terms of objective results, the approach presented here increased accuracy. Misclassifications (error rate) were reduced by 16 % and the mean and standard deviation of the Hausdorf distances from the gold standard of manual segmentation was 10 % and 10 % lower. The new approach additionally cut all users' segmentation times by nearly 20 %. It increased accuracy with 16 % fewer misclassifications than the standard live wire method. However, the mean and standard deviation from the target contour were only reduced 1 % and 2 %, respectively. Average segmentation speed was reduced approximately 18 %.

8 ACKNOWLEDGEMENT

This research work is being supported by the BMBF grants 0313821B and 0313821A.

9 REFERENCES

[AB94] R. Adams and L. Bischof. Seeded region growing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(6):641–647, 1994.

[AS07] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 10, New York, NY, USA, 2007. ACM Press.

[BBS+06] C. Brüß, F. Bollenbeck, F.-M. Schleif, W. Weschke, T. Villmann, and U. Seiffert. Fuzzy image segmentation with fuzzy labelled neural gas. In *Proc. of the 14th European Symposium on Artificial Neural Networks ESANN*, 2006.

[BS86] E. A. Bier and M. C. Stone. Snap-dragging. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 233–240, New York, NY, USA, 1986. ACM Press.

[DAZ99] M. S. Dulberg, R. S. Amant, and L. Zettlemoyer. An imprecise mouse gesture for the fast activation of controls. In *In Proceedings of INTERACT' 99*, pages 375–382, 1999.

[DLB+05] L. Dominjon, A. Lecuyer, J. Burkhard, P. Richard, and S. Richir. Influence of control/display ratio on the perception of mass of manipulated objects in virtual environments. In *Proceedings of IEEE Virtual Reality*, pages 19–25. IEEE Computer Society Press, 2005.

[Far88] G. Farin. *Curves and surfaces for computer aided geometric design: a practical guide*. Academic Press Professional, Inc., San Diego, CA, USA, 1988.

[Fur86] G. W. Furnas. Generalized fisheye views. In *CHI '86: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 16–23, New York, NY, USA, 1986. ACM Press.

[GDB+07] S. Gubatz, V.J. Dercksen, C. Brüšs, W. Weschke, and U. Wobus. Analysis of barley (Hordeum vulgare) grain development using three-dimensional digital models. *The Plant Journal*, 6:preprint, 2007.

[GDGC02] O. Gérard, T. Deschamps, M. Greff, and L. D. Cohen. Real-time interactive path extraction with on-the-fly adaptation of the external forces. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III*, pages 807–821, London, UK, 2002. Springer-Verlag.

[Hah05] H. K. Hahn. *Morphological Volumetry*. PhD thesis, Universität Bremen, 2005.

[HS85] R. M. Haralick and L. G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29(1):100–132, 1985.

[IH00] T. Igarashi and K. Hinckley. Speed-dependent automatic zooming for browsing large documents. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 139–148, New York, NY, USA, 2000. ACM Press.

[JF98] S. Jul and G. W. Furnas. Critical zones in desert fog: aids to multiscale navigation. In *UIST '98: Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 97–106, New York, NY, USA, 1998. ACM Press.

[Kan05] H. W. Kang. G-wire: A livewire segmentation algorithm based on a generalized graph formulation. *Pattern Recognition Letters*, 26(13):2042–2051, 2005.

[Kon07] Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB). Amira - Advanced 3D Visualization and Volume Modeling, 2007. www.amira.zib.de last visit: Sep. 6th 2007.

[Kre05] I. P. Krepps. Design and evaluation of a multi-floor speed dependent automatic zoom interface. Master's thesis, Royal Military College of Canada, April 2005.

[KS05] A. Kron and G. Schmidt. Haptic telepresent control technology applied to disposal of explosive ordnances: Principles and experimental results. In *Proc. of ISIE'05, IEEE International Symposium on Industrial Electronics*, volume 4, pages 1505– 1510, 2005.

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, V1(4):321–331, January 1988.

[LBE04] A. Lécuyer, J.-M. Burkhardt, and L. Etienne. Feeling bumps and holes without a haptic interface: the perception of pseudo-haptic textures. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 239–246, New York, NY, USA, 2004. ACM Press.

[LM01] L. Lucchese and S. K. Mitra. Color image segmentation: a state-of-the-art survey. In *Proceedings of the Indian National Science Academy*, pages 207–221, 2001.

[MB98] E. N. Mortensen and W. A. Barrett. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60(5):349–384, September 1998.

[MB99] E. N. Mortensen and W. A. Barrett. Toboggan-based intelligent scissors with a four-parameter edge model. *cvpr*, 02:2452, 1999.

[MC03] M. Moyle and A. Cockburn. The design and evaluation of a flick gesture for 'back' and 'forward' in web browsers. In *AUIC '03: Proceedings of the Fourth Australasian user interface conference on User interfaces 2003*, pages 39–46, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[Shn96] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *VL '96: Proceedings of the 1996 IEEE Symposium on Visual Languages*, page 336, Washington, DC, USA, 1996. IEEE Computer Society.

[SPP00] A. Schenk, G. P. M. Prause, and H.-O. Peitgen. Efficient semiautomatic segmentation of 3d objects in medical images. In *MICCAI '00: Proceedings of the Third International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 186–195, London, UK, 2000. Springer-Verlag.

[vWN03] J. J. van Wijk and W. A. A. Nuij. Smooth and efficient zooming and panning. *INFOVIS*, 2003.

[vWN04] J. J. van Wijk and W. A. A. Nuij. A model for smooth viewing and navigation of large 2d information spaces. *IEEE Trans. Vis. Comput. Graph.*, 10(4):447–458, 2004.

[WGCO07] L. Wolf, M. Guttmann, and D. Cohen-Or. Non-homogeneous content-driven video-retargeting. In *Proceedings of the Eleventh IEEE International Conference on Computer Vision (ICCV-07)*, 2007.

[ZSS97] S. Zhai, B. A. Smith, and T. Selker. Improving browsing performance: A study of four input devices for scrolling and pointing tasks. In *INTERACT '97: Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*, pages 286–293, London, UK, UK, 1997. Chapman & Hall, Ltd.

Automatic Axis Extraction from "Tubular" Structures In CT Images- An Application in Virtual Colonoscopy

José Tiberio HERNANDEZ
University of los Andes
IMAGINE: R&D in Visual Computing
Cra.1E No.18A-10
Colombia (57001), Bogota, D.C.
jhernand@uniandes.edu.co

Juan Manuel CALDERON
University of los Andes
IMAGINE: R&D in Visual Computing
Cra.1E No.18A-10
Colombia (57001), Bogota, D.C.
juan_mch@yahoo.com

ABSTRACT

This paper presents a new method for axis extraction from "tubular" structures in CT images. The axis is important in virtual colonoscopy systems because it offers more efficient measurement features and better interactive visualization possibilities.

The proposed method is based on the "pseudo-tubular" structure of the Colon. To find a good estimation of the Colon axis, the method is organized in two steps: In the first step, the voxels are organized in sequential clusters along the structure using a region-growing strategy and a cylindricity criterion. Each cluster has a distinct point (which is a good candidate to be part of the axis, identified by the cylindricity criterion), and an internal partial ordering of the cluster's voxels. This information structure allows us, in the second step, to propose three instances (one per principal coordinate axis) of an estimation problem to be solved with the Takagi-Sugeno-Kang Method (TSK). Based in the resulting solutions it is possible to build a piece-wise linear estimation of the colon axis.

This method has produced good results. It is robust in face of the "pseudo-tubular" shape of the colon. It has been tested on a set of Colon CT images from previously prepared patients. The results are illustrated with animations of virtual colonoscopy and a set of measurements and visualizations. Medical practitioners had a good perception of the method's potential in the field of virtual colonoscopy applications.

Keywords

Virtual colonoscopy, axis extraction, 3D Image processing, Medical Imaging

1 INTRODUCTION

Virtual colonoscopy is an alternative to diminish invasive practices in colonoscopy procedures used in colorectal cancer detection. A 3D model of the patient's colon is created from CT images. Using this geometrical model, the expert can perform visual examination and polyp detection. [1].

The main steps in virtual colonoscopy, from a computational point of view are:

- Image acquisition from the patient's colon region
- Segmentation of the colon in the image
- Axis (or central line) computation
- Interactive visualization (guided by the axis)
- Measurement, polyp detection and localization tools

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

This technique, although less invasive than the optical colonoscopy, requires quite a demanding preparation on the patient´s side. Nevertheless, this practice has had an important impact in the reduction of mortality due to colon cancer.

Research initiatives meant to improve virtual colonoscopy systems [1],[2] focus on the reduction of the invasive aspects of patient preparation with new methods of colon segmentation [3], and the improvement of visualization and measurement tools to be able to identify and locate polyps and other pathologies [4].

This paper presents a new axis extraction approach starting from a segmented 3D Image of the colon. The results obtained allow us to expect a "good behavior" in tubular structures that, like the colon, have complex shapes and characteristics such as local diameter variations and big curvatures. Other interesting tools for virtual colonoscopy based on axis computation are also presented: interactive navigation, measurement, and section height visualization.

2 AXIS EXTRACTION – GENERAL CONCEPTS

The extraction of the axis of a "tubular" structure such as the colon, involves finding a line whose points fulfill a series of characteristics [4]:

- **Centricity:** The points should be situated as far as possible from the surface or walls of the structure. Each point of the axis defines a section in the perpendicular plane to the axis at that point. In that section, the point of the axis should maximize the minimum distance to the border.
- **Connexity**: The points of the axis must, of course, be connected, and the line must have a "smooth" behavior (no discontinuities in its second derivative)
- **Singularity:** The axis must be one voxel in width.
- **Efficiency:** Computation time should be short.
- **Automation**: The method should not have to require human intervention.

The task may be approached in different ways. We may classify them in the following categories:

- **Manual extraction**: Based on human expert interactive work on the 2D images of the CT. Manually, the expert points out a pixel as part of the central axis of the segmented volume [4].
  Advantages: We will always have exactly one point per slice of the CT image.
  Disadvantages: The extracted line may or may not be precisely located on the center line. Human expert work is based on the CT slices, not on perpendicular sections of the segmented 3D images. Moreover, connectivity criteria are not guaranteed and, obviously, the procedure is not automatic.
- **Thinning**: Based on the successive elimination of border voxels from the segmented volume, in order to keep the "resulting" center. [5], [6].
  Advantages: The axis points determined are (by construction) connected and far from the borders of the segmented volume.
  Disadvantages: High sensitivity to "local pathologies" of the segmented structure (curvature, diameter) and to limitations of the initial segmentation.
- **Distances Map:** Based on the computation of the distance from each voxel to the nearest border in the segmented volume. Each voxel is labeled with the negative of this distance. The proposed axis is the traversal minimal cost path [7], [8], [9], [10], [11].
  Advantages: Based on the end points, this is an automatic process. The proposed axis is continuous and is located far from the borders of the segmented volume. In vascular structures, these methods present a very good behavior.

3 PROPOSED METHOD

The proposed method improves the geometric quality and robustness of the automatic solution. The main characteristic is the use of a "fuzzy" method (robustness) for piecewise curve adjustment (geometric qualities). Based on the 3D tubular characteristics of data, we propose a preprocessing stage that improves the conditions in which the Takagi-Sugeno-Kang (TSK) technique [12], [13] may be used.

The method is presented in two phases:

First Phase, **data structuring**, seeks the creation of a first rough estimation of a centerline of the tubular structure. This line is called ***primitive centerline.*** The primitive centerline is a polyline whose first point (in the particular case of the Colon`s images) can be computed automatically. Beginning at this point (the chosen point), and based on a growing strategy using the "Manhattan distance", and on a cylindricity criterion, we propose the next "chosen point" (the end of the axis of the biggest "cylindrical" section compliant with the cylindricity criterion). The sequence of chosen points is the primitive centerline of the tubular structure. The voxels are labeled with the Manhattan distance from the initial point (fig.1). Each vertex of the primitive center line has a "cylindrical section" of the structure associated to it (fig.3).

Once the primitive centerline has been established, voxels conforming the colon are also labeled with membership values based on their distance to each vertex of the primitive centerline. The voxels are organized in "clusters" associated to each vertex. The primitive centerline´s vertices generate a total order among clusters. The voxels within each 'cluster' are ordered considering their proximity to the neighboring 'clusters' (the previous one and the next one).

The second phase of the method, **Axis computation**, consists of a non-linear piecewise estimation of the colon's axis, by using the Takagi-Sugeno-Kang method (TSK) [14],[15],[16]. To implement this method, the ordering of voxels from the previous phase is used to generate three TSK instances in one dimension (one for each Cartesian coordinate). With the same order, the three solutions are integrated in one 3D line which is our proposal for the centerline of the tubular structure being analyzed (the colon, in our case).

Based on this centerline, an interactive navigation system can be guided inside the structure, different normalized measurements can be made on the sections (perpendicular to the axis), and then the morphological characteristics of the internal surface can be automatically estimated.

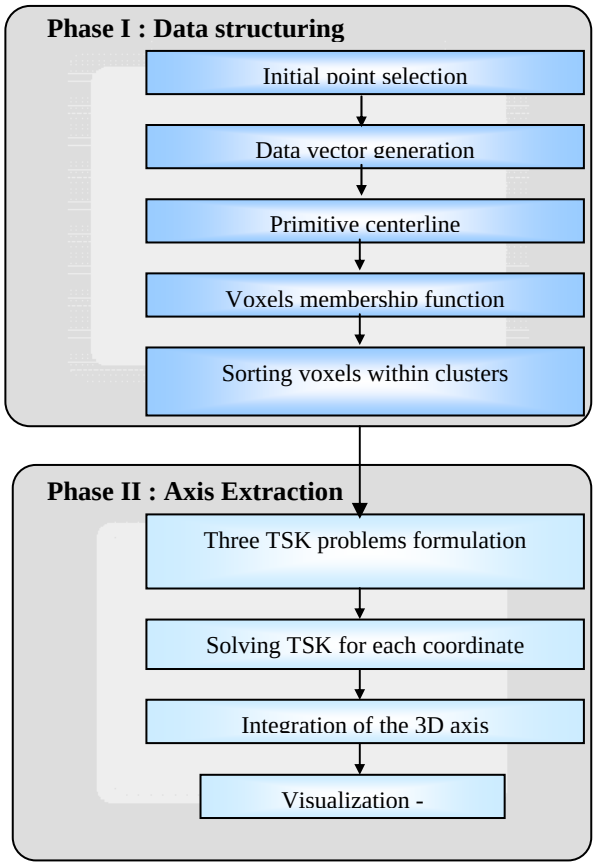


Diagram of the proposed method

In this section, each one of the phases of the proposed method is introduced in detail.

Information structuring

The method assumes that it has a 3D segmented image of the tubular structure i.e. the colon. The output of this phase is: a sequence of points which are the first point of the axis of "clusters" with pseudo-cylindrical shape (the primitive centerline) and the clusters with an order defined among its voxels. This phase has the following steps:

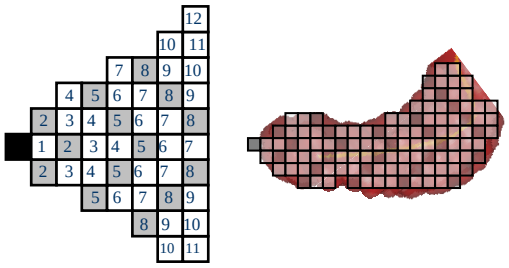
1) Selection of an initial point $\mathbf{P_0}$ in one of the ends of the tubular structure:

In the case of the colon, the initial point chosen is the voxel in the lowest level of the z axis. (We can assume that it is the beginning of the rectum).

2) Data vector generation of each voxel

a) Manhattan distance to P_0

The Voxel $\mathbf{P_0}$ is labeled with zero value (the initial voxel). Remaining voxels are labeled with the value of their Manhattan distance to $\mathbf{P_0}$. In this context, the set of voxels with the same label value (the Manhattan distance to $\mathbf{P_0}$) will be called "layer" (fig.1.a).



(a) (b)

Fig. 1. (a) Layer generation from the manhattan distance, (b) $\mathbf{P_0}$ Layers and manhattan distance inside the colon's human model from the $\mathbf{P_0}$ point

b) Distance to the nearest border

Let us define a "border voxel" as a voxel in the structure for which at least one of its 6-neighbors doesn't belong to the structure. For each voxel of the structure we compute the distance to the closest "border voxel" in one of the 6 main directions.

3) The sequence of points for the primitive centerline

The primitive centerline is a polyline through the tubular structure. Its segments define sections which also have a tubular shape according to a cylindrical criterion.

a) Criteria for the tubular sections

The colon is supposed to be divided in sections of uniform tubes.

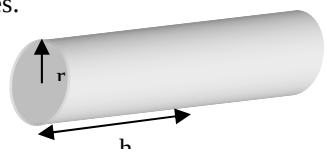


Fig. 2. R and H Parameters of a tubular section

Each section of the tube, as an approximation to the original structure, must have a height **h** greater or equal to two times its **r radius.** This way its main axis is in a direction close to the true axis of the structure.

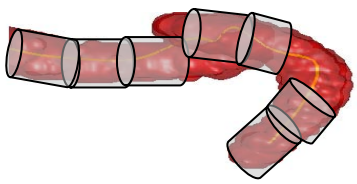


Fig. 3. Tubular sections throughout the colon

In order to build the cylindrical segments we can use two parameters: volume V and height h.

Volume (V) is estimated by the number of voxels present in the section.

 ISBN 978-86943-15-2

Height (h) is estimated by the number of layers present in the section. (The layers were generated using the Manhattan distance)

Based in the volume formula for a cylinder ($V = \pi r^2 h$) and the proposed relation ($h > 2r$), we have a criteria for an acceptable cylindrical shape:

$$h^3 > \frac{4V}{\pi} \qquad (1)$$

Where h, the length of the section, is estimated by the number of layers, and V, the volume of the section, is estimated by the number of voxels in the section.

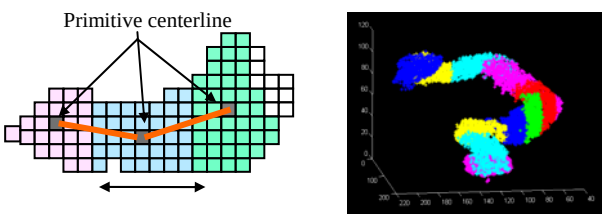
b) The primitive centerline algorithm

i. $\mathbf{P_0}$ is defined as the original point (layer 0). $\mathbf{P}_0$ is taken as new "chosen point" **Pi**. A new section (future tubular section) is initialized with **Pi** as member.

ii. With $\mathbf{P_i}$ as the starting point, we include in the current section the voxels of the next "layer", in increasing order of layers, until condition (1) is true or there are no more layers to include.

iii. From the voxels in the last layer included in the current section, we choose the voxel with the greatest border distance. This is the new "chosen point" $\mathbf{P_{i+1}}$. A new section (future tubular section) is initialized with $\mathbf{P_{i+1}}$ as member. Increment **i** and return to (ii)

The sequence <P_o,.... P_i,... P_n> is a polyline which runs through the tubular structure and whose points are (at least in the main directions) as far as possible from the border. This line will be called the **primitive centerline of the structure.** It will be used to define a sequence of "clusters" of voxels throughout the structure.



(a) (b)

Figura 4. (a) Clusters created in a section of the colon. (b) Clusters throughout the colon's structure.

c) Clusters

A cluster is a set of voxels associated to a particular vertex of the primitive centerline. We will define a fuzzy characteristic associated to each voxel called membership (how much does a voxel belong to each one of the cluster generated by the vertex of the primitive centerline).

4) Calculating the membership of each voxel to clusters

Each voxel is associated with a level of membership regarding each one of the clusters generated by each one of the vertex of the primitive centerline.

Calculation of the level of membership is determined by:

$$\mu_{ij} = (\frac{1}{D_{ij}})Dm_j$$

Where:

μ_{ij} = Membership of the i voxel to the cluster associated with the j point in the primitive centerline

D_{ij} Euclidian distance from i voxel to j point in the primitive centerline

Dm_j Minimum distance between all the voxels (not equal to j) in the structure to the j point in the primitive centerline

For each voxel in the structure, a membership value associated to each one of the vertex of the primitive centerline is computed. The voxel will belong to the cluster where the membership value is the greatest.

For voxels with two vertexes with the highest membership, the cluster assigned will be the one of the first vertex in the primitive centerline.

Each voxel will have three membership values: the value associated to its cluster, to its previous cluster (except the first one) and to the next cluster (except the last one).

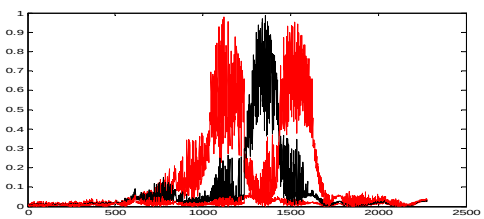


Fig. 5. Membership values of the voxels of the image. Colors illustrate the cluster to which they are assigned.

5) Ordering within each cluster

To build a total order for the voxels in a cluster (which will be used for the linear approximation later), we define a new parameter. The "relative membership" of a voxel in a cluster is the difference between its memberships to the next cluster and previous cluster.

Once the "relative membership" value is calculated, voxels are sorted using this parameter. We will find a total order inside the clusters, for the sequence of clusters, defined by the vertex in the primitive centerline.

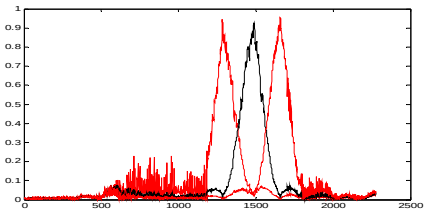


Fig. 6. Membership values of the ordered voxels.

The axis computation

Based on the structured information produced in phase one, the axis estimation for the tubular structure being analyzed is presented below:

1) *Generation of three Takagi-Sugeno-Kang (TSK) approximation instances*

The proposed method computes the axis estimation as a piecewise approximation for the set of voxels grouped in clusters. The problem statement is the estimation of three variables (the three coordinates) for each axis point based on a sequence of clusters and inside each cluster a sequence of voxels (based on the relative membership parameter). In order to reduce the complexity of the problem we propose three 1-variable problems, following a common total order of the data.

This approach allows us to compute a non-linear approximation for each coordinate and, based on these results, the final axis estimation.

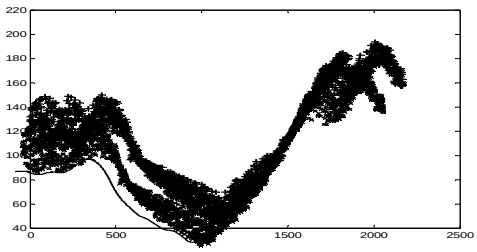


Fig. 7. Values of the x variable regarding the order parameter

2) *TSK Estimating for each component*

The next step is to find a non-linear model that can be adjusted to the information found when the coordinates of the voxels are separated according to the axes. We use the TSK (Takagi Sugeno Kang) method, well known in Fuzzy Control area (non-linear systems modeling).

We will follow the steps to generate a non-linear model in TSK method:

i.Create clusters from the set of information

ii.Obtain fuzzy membership functions of antecedents

iii.Obtain the consequents membership functions

iv.Calculate the system's final model.

i. Create clusters from the information set

In a regular TSK process, initial information must be exposed to a clustering process where the main objective is to find sub-groups within the data universe that have similarity or proximity. In our case, this step of the process has already been done. The structure was divided in clusters by calculating memberships (which comply with the TSK conditions).

ii. Obtain the antecedents` membership function.

Once the partition of the inputs in fuzzy sets is conducted, it is necessary to obtain the membership functions, so that it is possible to calculate the level of membership for any data belonging to a cluster, in order to obtain the appropriate model. This process is conducted through the association of a Gaussian membership function to each one of the clusters. Such function will be responsible of conducting smooth transitions between different linear pieces to be calculated later on.

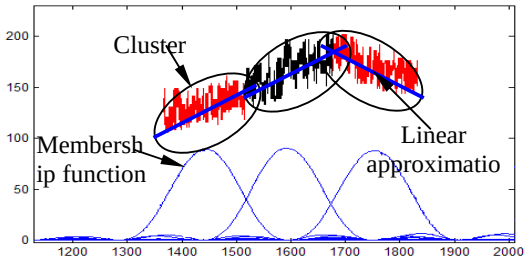


Fig. 8. Membership function and linear approximation

iii. Obtain the consequent membership function

As mentioned previously, the TSK method is based on the use of linear piece-wise approximations to generate the final non-linear approximation.

Linear approximations (straight lines) are generated for each one of the groups of information of each cluster. This straight line is named "consequent".

iv. Calculate the final model of the system.

Based on the previous steps of the process, the required data is available to compute the final model of a variable. We should multiply the membership functions with the consequents in order to obtain the final model:

 ISBN 978-86943-15-2

$$y(x) = \sum_{i=1}^{r} \left(\left(\frac{\mu_i(x)}{\sum_{j=1}^{r} \left(\mu_j(x)\right)} \right) \bullet \left(a_i x + b_i\right) \right)$$

where :

r = Number of clusters

$\mu =$ Membership level

$a_i x + b_i =$ Straight line approximation to the information

The result obtained from the application of the previous equation, generates the following approximation of the evolution of the coordinate along the axis.

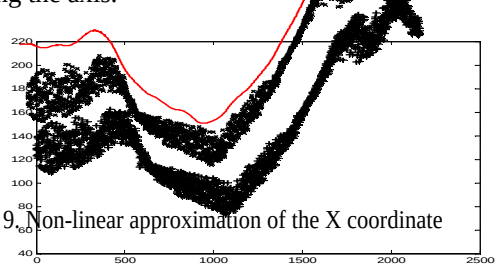


Fig. 9. Non-linear approximation of the X coordinate

The resulting model is continuous, smooth and seeks to go through the intermediate points of the data from the original system.

This process is applied to each one of the three systems that were built (one for each coordinate).

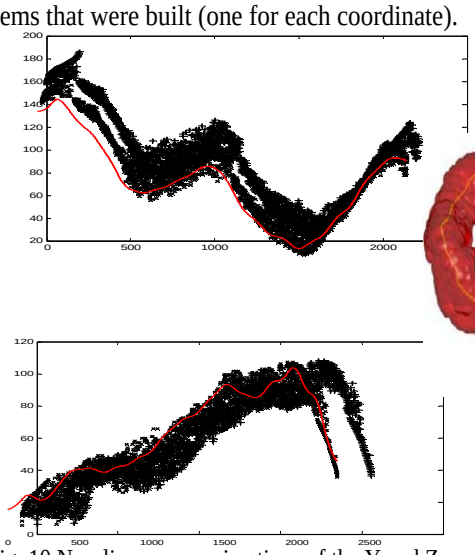


Fig. 10 Non-linear approximations of the Y and Z coordinates

3) Integration of the 3D model

The previous process produces an approximation for each one of the coordinates. We can integrate these results using the total order established in a previous step. In this way we find a 3D model of the structure's axis. For our specific example (the colon) the 3D line is obtained in fig. 11:

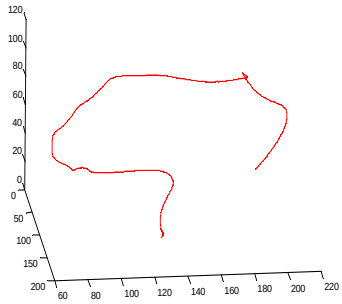


Fig. 11: Integrated solution for the axis of the tubular structure.

4 RESULTS

An implementation of the method presented was built using the Matlab 6.5 software. The input images were obtained from a basic segmentation system for Colon axial tomography, and from a tubular geometrical model generator. The Axis extraction method was applied to these input data and the results are presented bellow.

Two Colon tomographies were used to apply the segmentation process and axis extraction.

	Dimension (Voxels)	(m.m /voxel)	Axis Length (mts)	computation time
Colon 1	256X256X130	1.4x1.4x1.4	1.54	34 Sec
Colon 2	256X256X499	1.4x1.4x1.0	2.48	52 Sec

Table 1. Dimensions of the models and axis

Table 1 shows the main images characteristics, the length of the obtained axis and the computation time of the extraction method running on Matlab in a current laptop computer.

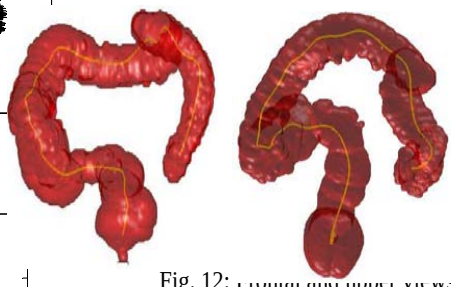


Fig. 12: Frontal and upper views of the colon, with the extracted axis in yellow.

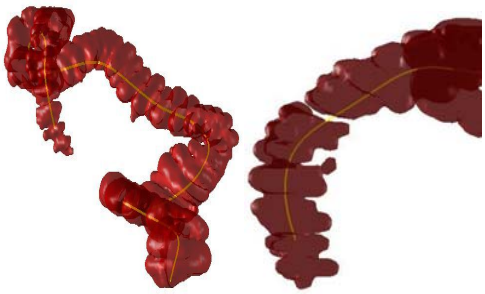


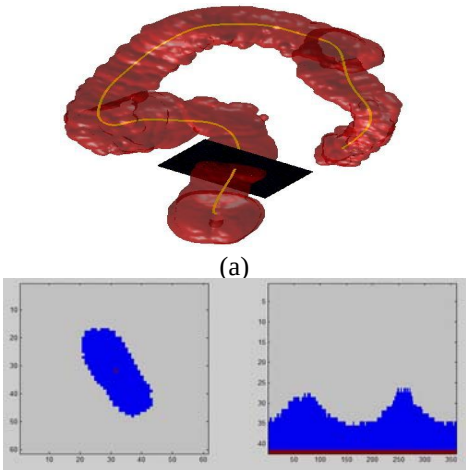
Fig. 13: Side/lateral views of the colon 2 and detail of the ascendant colon

 ISBN 978-86943-15-2

A discontinuity of the ascendant colon can be observed in fig.13; we can observe that the algorithm outlines the axis in spite of this lack of data.

Based on the axis, we can obtain perpendicular sections along the axis and complementary measurements can be done (diameter and perimeter evolution, for example). Fig. 14(a) shows the axis and one perpendicular section with the profile generated from it -14(b).

In a similar way, we show in Fig. 15, a "terrain" view of the inside surface of the colon. It was obtained using the sections calculated along the extracted axis. This particular view can be very useful for polypus detection studies [19].



(a)

(b)

Fig. 14. (a) Perpendicular cut to the colon's human model (b) Cut's profile/section and relation of distances from the axis to the surface of the cut.

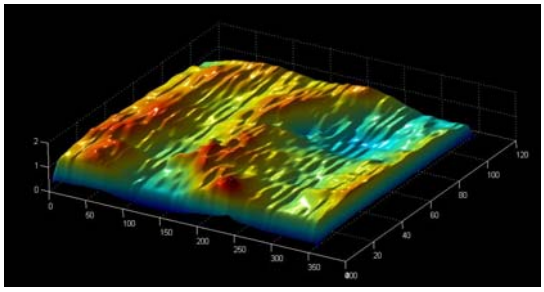


Fig. 15 'terrain' view of the internal wall of the colon throughout the axis

And, finally, when the axis is obtained, it is also possible to guide an interactive virtual colonoscopy, where a virtual camera navigates inside the colon with the support of the measurement facilities presented above.

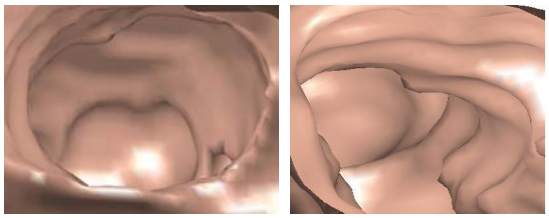
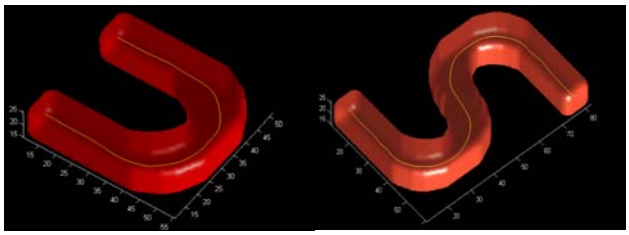
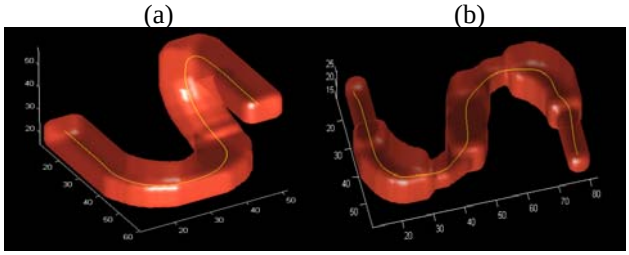


Fig.16. Virtual navigation images

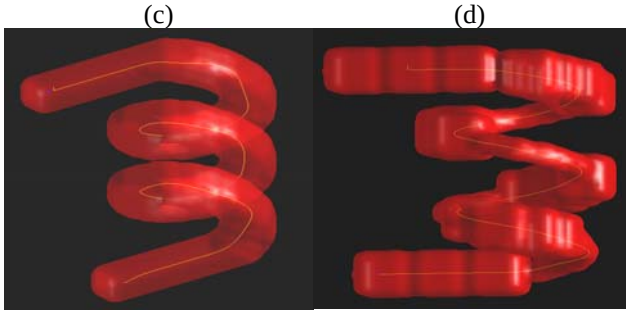
In addition to the tests shown previously, the method was applied to tubular structures generated virtually for which the axis was exactly known. Four models were created for this, which are shown in fig. 17.



(a) (b)



(c) (d)



(e) (f)

Fig. 17. (a) U shaped structure, (b) S shaped structure in the XY plane (c) S shaped structure in the XYZ plane (d) S shaped structure with changes in the radius, (e) (f) helical shape structure

Table 2 shows the error between the real axis and the automatic extracted axis using the proposed method.

Structure	Error
Model a	0.013%
Model b	0.2%
Model c	1.3%
Model d	2.1%
Modelo e	3.2%
Modelo f	4.1%

Table 2. Error in extracted axis for the generated models

 ISBN 978-86943-15-2

5 CONCLUSIONS

An automatic method for the axis extraction of pseudo tubular structures has been presented. We have illustrated the results using an implementation oriented towards virtual colonoscopy.

The use of criteria based on the tubular structure of a 3D image (cylindrical, layers based on Manhattan distance, axial distance to the surface), allows us to define and compute "tubular" clusters. Based on these clusters, an automatic formulation of three TSK estimation problems was done, and its results were then integrated in a robust estimation of the 3D axis of the input tubular structure.

The behavior of the presented method was illustrated with real colon images and with synthetic models. The results are interesting because of the geometric quality and the robustness of the different pseudo tubular forms and little discontinuities in the input data.

The frequency of pseudo tubular structures in biomedical problems gives this method a good potential for applications. The colon's characteristics as a pseudo tubular structure allow us to show results with a good potential impact in virtual colonoscopy applications.

6 REFERENCES

[1] Liang, Z "*Virtual Colonoscopy: An Alternative Approach to Examination of the Entire Colon*". In; www.viatronix.com/pdfs/VC_whitepaper.pdf. Innervision 2001

[2] Chen, Liang, Wax, Li y Kaufman, " *A novel Approach to Extract Colon Lumen from CT Images for Virtual Colonoscopy*". IEEE Transactions on Medical Imaging, Vol. 19, No. 12, Dec. 2000, pp. 1220-1226.

[3] Martinez D., Hernandez JT, Florez L., *"Contributions to colon segmentation without previous preparation in computer tomography image"*. WSCG 2007, 15-th Intnal. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision, Plzen, February 2007 pp:211-219.

[4] Ming Wan, Zhengrong Liang, Lichan Hong, Arie Kaufman, *"Automatic Centerline Extraction for Virtual Colonoscopy"*, IEEE Transactions on medical imaging, Vol 21, Nº12, December 2002

[5] Robert J.T. Sadleir, Paul F. Whelan,. *"Colon Centerline Calculation for CT Colonography using Optimised 3D Topological Thinning*". 1st International Symposium on 3D Data Processing Visualization and Transmission (3DPVT 2002), Padova, Italy, Jun 19-21, 2002

[6] Lordanescu, G; Summers, RM *"Automated Centerline for computed tomography colonography*".Academic Radiology, Vol.10, No.11, 2003, pp:1291-1301

[7] Ming Wan, Frank Dachille, Arie Kaufman, *"Distance-Field Based Skeletons for Virtual Navigation"* IEEE Proc. on visualization. 2001.

[8] Ingmar Bitter, Kaufman, Fellow, Mie Sato, *"Penalized-Distance Volumetric Skeleton Algorithm"*, IEEE Transaction on visualization and computer graphics, Vol 7, Num 3. 2001.

[9] D Chen, B Li, Z Liang, M Wan, A Kaufman, M Wax *"A tree-branch searching, multiresolution approach to skeletonization for virtual endoscopy"*. Proc. SPIE Med. Imag, 2000 In: www.cs.sunysb.edu/~vislab/papers/tree-ske.pdf

[10] JH Chuang, CH Tsai, MC Ko *"Skeletonisation of three-dimensional object using generalized potential field***",** IEEE Transaction on Pattern Analysis and Machine Intelligence. 2000 Volume 22 Issue 11.

[11] Carrillo,JF, Orkisz,M ,Hernández,M :*Extraction of 3D vascular tree skeletons based on the analysis of connected components evolution* In: Computer Analysis of Images and Patterns, 2005, Versailles. LNCS- Springer. Berlin 2005. v.3691. p.604 - 611

[12] Li-Xin Wang. *"A course infuzzy systems and control*". Prentice Hall PTR, Upper Saddle River, 1997

[13] Tomohiro Takagi, Kasushi Kawase. *"A trial for data retrieval using conceptual fuzzy sets"*. IEEE Trans. on fuzzy systems, Vol 9, Nº4. 2001.

[14] Hao Ying. *"General SISO Takagi-Sugeno fuzzy systems with linear rule consequent are universal approximators"*. IEEE Transactions on fuzzy systems, Vol 6. Nº 4, Nov. 1998.

[15] Ke Zeng, Yao Zhang, Wen-Li Xu. *"A comparative study on sufficient conditions for Takagi-Sugeno Fuzzy systems as universal approximators"*. IEEE Transactions on fuzzy systems, Vol 8, Nº 6. Dcember 2000.

[16] Driankov Dimiter, Hellendoorn Hans. *"Fuzzy model identification"*. Springer. 1997

[17] M Macari, EJ Bini *"CT Colonography: where Have We Been and Where Are We Going". in: Radiology* 2005 Special Reviews. *www.* radiology.rsnajnls.org/cgi/reprint/237/3/819.pdf

[18] YP Li, T Zielinska, MH Ang, W Lin "Vehicle Dynamics of Redundant Mobile Robots with powered Caster Eheels". Proc. of the sixteenth CISM_IFToMM Symposium. 2006 pp:221-228

[19] Summers RM, Beaulieu CF, Pusanik LM, Malley JD, Jeffrey RB Jr, Glazer DI, Napel S. *"Automated polyp detector for CT colonography feasibility study"*. In: Radiology 2000 Jul;216(1):284-90

Dynamic Mesh Refinement on GPU using Geometry Shaders

Haik Lorenz
Hasso-Plattner-Institute
University of Potsdam
August-Bebel-Str. 88
14482 Potsdam, Germany
haik.lorenz@hpi.uni-potsdam.de

Jürgen Döllner
Hasso-Plattner-Institute
University of Potsdam
August-Bebel-Str. 88
14482 Potsdam, Germany
doellner@hpi.uni-potsdam.de

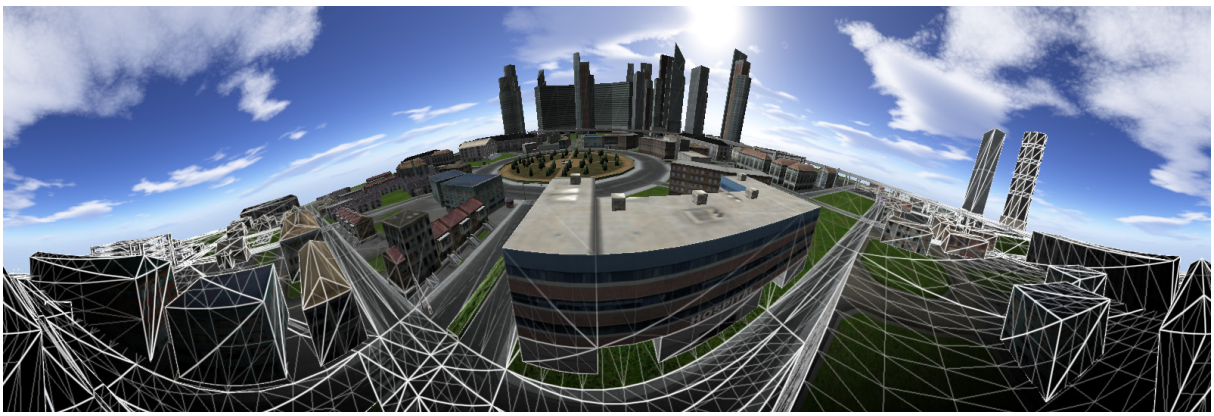


Figure 1. A 360° panorama view generated with dynamic mesh refinement. The frame's input mesh is hinted with thick lines. Actually rendered primitives have thin lines.

ABSTRACT

This paper presents a real-time rendering technique for dynamic, incremental 3D mesh refinement. The technique can be applied to any triangulated 3D mesh with arbitrary topology and connectivity. The functionality relies on geometry shaders that are used to amplify or remove geometry based on precalculated refinement patterns. For each triangle, the instantiated refinement pattern is selected dynamically. Due to limitations of current hardware, on-the-fly pattern instantiation cannot be implemented on the GPU. Instead, the complete refined mesh must be generated through pattern copying. We propose an incremental approach where the refined mesh is generated by using the previous refined mesh as primitive source. This algorithm runs exclusively on the GPU and requires no continuous data exchange between CPU and GPU. Due to the necessary mesh generation, the approach is particularly suitable for applications with small refinement levels. It complements traditional pattern-based refinement approaches that deliver high throughput for large refinement levels, but incur a substantial CPU-GPU communication overhead otherwise. Interesting applications include view-dependent mesh smoothing and interactive non-planar projections. In these areas, our algorithm enables efficient vertex-based implementations due to adaptive refinement.

Keywords

Geometry shaders, GPU, mesh refinement, refinement patterns, barycentric coordinates

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

1. INTRODUCTION

Geometry specification and its efficient rendering is a crucial aspect of computer graphics. For high-performance rendering, most parts of the graphics pipeline have become implemented in graphics hardware and are fed with primitives through standardized APIs. Real-time applications are forced to limit the number of primitives passing through the graphics pipeline. For fine meshes, this translates to *mesh simplification*, where a detailed mesh is

degraded into a coarse mesh while limiting visual artifacts. The simplification is either offline (statically) or online (dynamically). In general, level-of-detail algorithms (LoD) follow this paradigm.

If geometry is available in form of some compact functional description, e.g., subdivision surfaces, *geometry synthesis* is required for generating a mesh representation for processing by graphics hardware. Online synthesis allows for adapting the mesh to view parameters and thus limiting its size. In real-time applications, geometry synthesis is often posed as *mesh refinement* problem where the intended geometry is defined as function over a coarse base mesh [Bou07]. The base mesh is then refined so that the geometry function can be evaluated per vertex with sufficient frequency in screen space.

The same idea of using mesh refinement to ensure sufficient vertex frequency can help to map computer graphics algorithms to graphics hardware. In a *vertex-based approximation* scheme, complex, non-linear computations are evaluated per vertex and linearly interpolated in between. Examples include global illumination or non-planar projections.

Our motivation is the use of mesh refinement for view-dependent geometry synthesis and vertex-based approximation techniques. A number of hardware-accelerated techniques have been proposed for generic [Bou07] or special-purpose [Bun05, Shi05] mesh refinement. However, all of these techniques require continuous data transfer between CPU and GPU proportional to the input mesh size. While exhibiting great performance for high refinement levels, this makes them inefficient for view-dependent techniques, whose refinement level is dynamic and rather low on average.

We propose a dynamic approach fully operating on the GPU to make mesh refinement efficient for view-dependent problems. Our technique uses the idea of *barycentric refinement patterns* as proposed by Boubekeur et al. [Bou05b, Bou07] and follows their coarse algorithm. However, we relieve the CPU from selecting and instancing the refinement patterns by exploiting recently introduced *geometry shaders* [Bly06]. This enables us to dynamically amplify or remove geometry on the GPU, but at the same time shader limitations require us to create and store a representation of the complete refined mesh. Even though being able to provide arbitrary refinement levels, the mesh creation makes our approach most efficient for low average refinement levels and thus complements Boubekeur's method, as suggested in [Bou07].

Our algorithm exhibits four major properties that make it particularly suitable to add view-dependency to both, geometry synthesis and vertex-based approximation techniques. It is

- *generic*, i.e., it operates on a triangular mesh of arbitrary topology and connectivity. It does not impose any restrictions on the refinement strategy or rendering technique and thus is adaptable to a wide range of applications.
- *pattern-based on triangle level*, i.e., it uses precalculated refinement configurations to replace each original input triangle. The patterns can have any suitable structure. Patterns are not limited in their vertex count.
- *dynamic*, i.e., it selects each triangle's refinement pattern for each frame. Then, a refined mesh is build and rendered accordingly.
- *constant and minimal in communication overhead*, i.e., it fully saturates the GPU regardless of the input or rendered mesh size.

Our first contribution is the removal of communication overhead. While the first three properties are inherited from Boubekeur's approach, we remove the need for per-triangle draw calls despite a non-uniform refinement. Our second contribution is an incremental refinement scheme that enables fast arbitrary geometry amplification with geometry shaders despite their output limit.

The paper is organized as follows: Section 2 gives an overview over related work, Section 3 briefly introduces geometry shaders, Section 4 gives an in-depth description of our generic refinement algorithm along with its limitations, Section 5 demonstrates the refinement for two examples, Section 6 provides performance results and comparisons, and Section 7 concludes.

2. RELATED WORK

This section concentrates on hardware-accelerated mesh refinement algorithms. We distinguish two groups based on application dependency.

In application-independent refinement, Boubekeur's work [Bou05b, Bou07] is of most importance to our method, as described earlier. His methods are applicable to any triangulated mesh. The refinement presented by Bokeloh and Wand [Bok06] needs to restrict the input to rectangular patches that become hierarchically subdivided. They cast the refinement problem as image upscaling and use fragment shaders for implementation on the GPU. The new image data is then interpreted as vertex data for rendering.

Application-dependent refinement is used for geometry synthesis. For example, various approaches to subdivision surface rendering exist. Hardware-accelerated methods use patch-based approaches as introduced by Pulli and Segal [Pul96]. Bunnell [Bun05] describes a GPU implementation for adaptive tessellation including displacement mapping. While this approach uses a 2D texture to

represent surface patches, Shiue et al. [Shi05] uses a spiraling scheme to unfold each patch into a 1D texture. Both methods then use image operations to upscale the texture for actual geometry synthesis and recast the final texture as vertex buffer for rendering.

Guthe et al. [Gut05] present a synthesis approach to trimmed NURBS. They use predefined sampling grids similar to our barycentric refinement patterns to render a hierarchy of bi-cubic approximation surfaces. They focus on artifact-free and efficient trimming of the resulting rendering.

Hardware-accelerated mesh smoothing has been described by Losasso et al. [Los03]. They use geometry images [Gu02] to represent the control points for a B-Spline-based surface approximation. The control points are then transformed to a refined surface by performing image convolution operations on GPU. Finally, the image is interpreted as vertex data and rendered.

A new generic approach to rendering has been proposed by Whitted and Kajiya [Whi05]. They outline changes to the current graphics hardware to enable processing of fully procedural geometry. As they propose point sampling in screen space, such a rendering processor eliminates the need for mesh refinement.

Geometry shaders are a recent extension of the hardware graphics pipeline [Bly06]. So far, their use has rarely been reported in literature. Tariq [Tar06] presents some examples provided by nVidia, including fur rendering, cloth animation, and isosurface extraction. DeCoro and Tatarchuk [DeC07] exploit geometry shaders for mesh simplification through vertex clustering. An interesting future development is the proposal of a dedicated hardware tessellation unit [Tat07].

3. GEOMETRY SHADERS

Geometry shaders [Bly06] introduce an additional processing unit in the graphics pipeline. This unit is located after vertex processing and before clipping and rasterization. It operates on whole primitives (points, lines, or triangles with or without newly introduced adjacency information) having random access to all a primitive's transformed vertices and their attributes. The geometry shader program is invoked once for each assembled input primitive. The program then emits zero or more primitives, whose type is fixed but can differ from the input primitive type. The output completely replaces the input, allowing for geometry amplification and deletion.

The amount of output data cannot exceed a certain hardware limit, typically 4096 bytes. The programmer can further restrict the maximum output size, since the shader's performance and efficiency improves significantly with smaller maximum output sizes, regardless of the actual data output per invocation.

An additional feature introduced together with geometry shaders is *transformation feedback,* a.k.a. *stream out*. This allows for capturing transformed vertices into a buffer in sequential order before clipping but after the geometry shader stage. The exact number of vertices captured in a buffer can be queried after command completion. As the capture buffer resides in GPU local memory, it can be used as input to subsequent rendering commands without copying. Optionally, the vertices can be discarded after capturing without rasterization taking place.

4. DYNAMIC MESH REFINEMENT

4.1 Overview

Our refinement approach is pattern-based, i.e., each triangle of the original mesh is replaced by a generic precomputed refinement pattern consisting of *subtriangles* before rendering. Each selected pattern is then adapted to and rendered instead of its originating triangle.

Except for pattern selection, our approach works on single triangles only. Thus, we impose no restrictions on the original triangular mesh's topology or connectivity. In conjunction with a local pattern selection scheme, e.g., based on the screen space edge length, the dynamic mesh refinement is instantly applicable to any (indexed or non-indexed) triangle soup accepted by standard graphics APIs. No additional attributes or data structures (e.g. half-edge structures) are required.

The pattern selection is computed on GPU for each frame and can follow any scheme suitable for the target application. Moreover, the selection can easily take backface and view volume culling into account to avoid unnecessary refinement.

The core of our method is an incrementally updated intermediate mesh of all subtriangles. It is motivated by three observations. First, current GPUs are not capable of dynamically instantiating geometry without the control of the CPU. Second, efficient use of geometry shaders requires to minimizing each invocation's output by all means (cf. Figure 8). Third, in an incremental approach the geometry shader's output limit only limits the growth of a triangle's refinement instead of its size.

The intermediate mesh is a concatenation of one plain pattern copy per input triangle. During update, each triangle's existing copy is replaced by the newly selected one. This update is computed on subtriangle level using geometry shaders on the GPU without the need for CPU control and thus without continuous communication between CPU and GPU. This removes the bottleneck of Boubekeur's

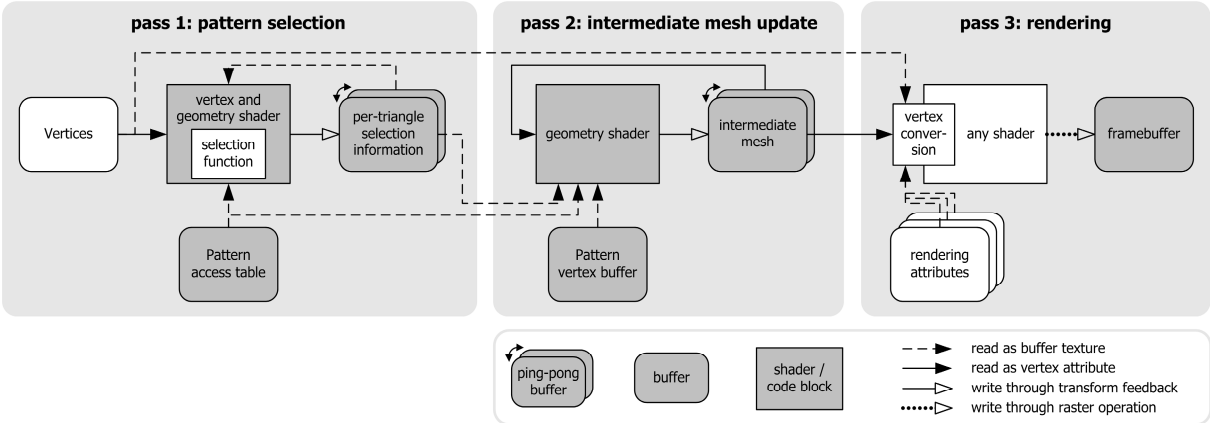


Figure 2. Algorithm outline. White boxes mark application dependent parts.

method. However, the need for creating and storing the complete intermediate mesh favors applications that need only low average refinement levels.

Finally, the whole intermediate mesh is rendered. Only then, the generic pattern subtriangles are adapted to their originating triangles. Vertex positions and other rendering attributes are computed and immediately used for further processing. This includes per-vertex approximations or the evaluation of surface descriptions in geometry synthesis. To a subsequent rendering technique, refined vertices become indistinguishable from original vertices after conversion from their barycentric form. Thus, our method can be used for any rendering effect.

The result is a three-pass-algorithm as outlined in Figure 2. Application independent parts are shown in gray. Each pass requires exactly one vertex array draw call with all information residing in GPU local memory. Hence, the CPU utilization and communication costs between CPU and GPU are negligible and constant regardless of the input or intermediate mesh's size. In the following, we introduce our refinement patterns and explain each pass in detail.

4.2 Refinement Patterns

Similar to [Bou07], each refinement pattern represents a triangle tessellation encoded in barycentric coordinates. A barycentric vertex $(u,v,w)^T$ represents a point inside a triangle as weighted sum of the triangle's vertices with $u+v+w=1$ and $u,v,w \geq 0$. For efficient storage, the third coordinate can be omitted as $w=1-u-v$ holds.

The exact pattern structure is application-dependent. As all patterns are precomputed, any suitable structure can be used. After generating all patterns, the resulting barycentric subtriangles are stored in a combined vertex buffer. A separate table buffer stores for each possible refinement pattern its triangle count and starting location in the combined vertex buffer.

In contrast to [Bou07], our method does not allow for optimizing refinement patterns through the use of indexing. If we used indexing, the same pattern vertex index in different pattern copies referred to different original triangles. As we render the whole refined mesh with a single draw call, the graphics hardware then might wrongly reuse cached vertex shader results belonging to another original triangle. Non-indexed stripping can be used provided that only a single strip for the whole refined mesh is generated [Eva96, Reu05, Dia06], and that artifact-free rendering is achieved.

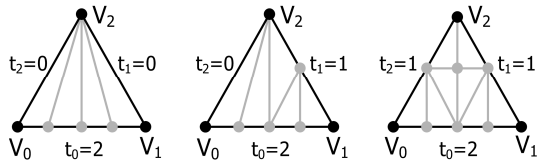


Figure 3. Some sample refinement patterns.

Our examples in Section 5 use patterns controlled by their edges' tessellation degrees t_0, t_1, and t_2 with a regular and homogeneous interior triangulation. The borders are subdivided in a recursive fashion into 2^{t_i} sections with $t_i \in [0;t_{\max}]$ denoting the edge's refinement depth. This results in $t_{\max}^3$ possible refinement patterns. Some examples are shown in Figure 3.

4.3 Pattern Selection

Pattern selection is the first pass of our algorithm. For each input triangle, it selects one pattern from the available precomputed refinement patterns. The selection function is fully application-dependent and maps each triangle to a pattern index. However, usually two goals are desired: a crack-free and minimal refinement.

Since our approach uses a local refinement, cracks can only be prevented by ensuring identical tessellation of the shared edge of two adjacent

triangles. This can be achieved by selecting a pattern based on each edge's tessellation degree t_i as implemented in the examples.

Ensuring a minimal refinement obviously improves total performance. Most importantly, hidden triangles can be left unrefined to save computation time. It is not possible to omit such triangles altogether, which is explained in Section 4.4. GPU-friendly hidden triangle detection methods include backface and view volume culling. Additionally, any screen space based selection function inherently limits the applied refinement to the current view's demands. Examples include screen space error estimators for subdivision surfaces [Bun05], height fields [Lin96], or texture deviation [Coh98].

Pattern selection is implemented in a geometry shader operating on triangles, whose input depends on the requirements of the selection function. Usually, this includes only a subset of the available rendering attributes, such as vertex position and normal. Additionally, the previous frame's selection is fed into the shader. Its output only contains 3 values per triangle: the selected pattern's index p, the number of required subtriangles s_r, and the number of available subtriangles s_p from the previous frame. For proper operation, the pattern growth, i.e., the ratio s_r to s_p, must not exceed the output limit of the update pass, even though this might introduce temporary cracks. Instead of being rendered, this output in form of point primitives is captured in a buffer (using stream out or transformation feedback) and then discarded before rasterization.

The previous frame's selection is accessible through the use of *buffer textures*. This enables a shader to randomly access an arbitrary buffer residing on the GPU via the 1D texture interface. With the primitive ID as index, a triangle's number of subtriangles can be read from the previous frame's selection buffer and copied to the number of available subtriangles for the current frame.

4.4 Intermediate Mesh Update

The update pass uses only the previous frame's intermediate mesh consisting of subtriangles as primitive input. Each subtriangle is identified by the original mesh's triangle id i, and the subtriangle id j within the pattern. Together with the barycentric coordinates (u,v), this information is stored in every vertex. Additionally, the geometry shader has access to the selection pass's output buffer, the precomputed pattern vertex buffer, and the accompanying table buffer via buffer textures.

For each original triangle, the update pass completely replaces an existing refinement by a copy of the newly selected refinement pattern. The copying task is evenly distributed across all respective subtriangles, such that the output of a single geometry shader invocation is minimized.

For each subtriangle (i, j), the geometry shader fetches the original triangle's pattern selection (p, s_r, s_p). It can then determine the corresponding range of subtriangles from pattern p. It emits each new subtriangle by reading its barycentric vertices from the pattern vertex buffer and emitting them augmented by the triangle id i and a new subtriangle id j'. Similar to pass 1, the shader output is captured in a buffer and discarded before rasterization.

For proper refinement, this algorithm requires at least one subtriangle per original triangle. Once an original triangle is lost from the intermediate mesh it can never reappear. Consequently, hidden triangles must not be omitted but kept in the intermediate mesh as a single subtriangle. Additionally, the intermediate mesh must be initialized once to one subtriangle per original triangle.

The buffers for storing the intermediate mesh need to be sufficiently sized in advance. Transformation feedback allows for buffer overrun detection, but for best performance, buffer reallocation should be avoided.

4.5 Rendering

The final pass comprises the conversion of the intermediate mesh's barycentric vertices to the intended geometry and the actual image generation. It is fully application dependent.

The conversion part is of most interest to geometry synthesis, as it implements the surface description evaluation. E.g., it can use the barycentric coordinates for procedural generation, as texture coordinates for displacement mapping, or as parameters for spline surface evaluation. For vertex-based approximations, usually a simple weighted sum for interpolation in object space is sufficient. If an interpolation in screen space is desired, the barycentric coordinates require a preceding transformation, e.g., by hyperbolic interpolation [Bli92].

Image generation is not affected by our mesh refinement. Since any rendering attribute can be computed during vertex conversion, any rendering technique is applicable. This includes multi-pass techniques, even though each additional pass might require its own vertex conversion. Alternatively, conversion results can be stored using transformation feedback and reused in each subsequent pass.

5. APPLICATIONS

For usage, pass 1 and 3 from our algorithm need to be adapted (white boxes in Figure 2). In pass 1, a suitable pattern selection function needs to be implemented in a shader. Pass 3 implements the application-specific rendering. This pass can use the original shaders as if no mesh refinement was in place. Only the vertex shader portion needs to be extended by a vertex conversion function converting the generic barycentric pattern coordinates to the actual vertex attributes as needed for further processing.

In the following, we demonstrate our refinement for one geometry synthesis and one vertex-based approximation technique.

5.1 Curved PN-Triangles

Curved PN-Triangles have been proposed by Vlachos et al. [Vla01] as a simple heuristic to smoothing a triangular mesh. It has been designed with a dedicated hardware implementation and readily available input in mind [ATI01]. It only uses vertex positions and normals to construct a triangular Bézier surface over the triangle. This surface is then used for geometry synthesis. For many coarse meshes, this method instantly generates visually pleasing results without any change to the original mesh. A respective tessellation unit can be found on some graphics hardware. The original approach has been further investigated, e.g., by Boubekeur et al. [Bou05a] or Choi et al. [Cho04].

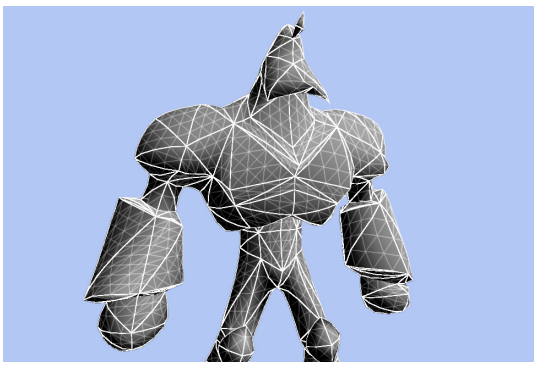


Figure 4. PN-triangles. Thick lines show the original mesh, thin lines the refined mesh.

We demonstrate an implementation of the original approach based on our dynamic mesh refinement. Since the curved PN-triangle surface closely follows the original mesh, the original triangle's screen space edge length is a reasonable basis for a view-dependent metric. At the silhouette, this is not sufficient as some minimal refinement should be retained to prevent popping artifacts. Consequently, the normal orientation is incorporated into the pattern selection. In pass 3, we use the formulas given in [Bou05b] to calculate the refined vertex's position and normal from the barycentric coordinates. The result is shown in Figure 4.

5.2 Cylindrical Projection

Current graphics hardware relies on linear interpolation (in homogeneous space) during rasterization. This prevents a straight-forward vertex-based implementation of non-planar projections or view deformations. The actual result can only be approximated due to the missing correct non-linear interpolation. Traditionally, in real-time applications an image space solution involving multiple render-to-texture passes and subsequent image warping [Yan05] has been used. This solution trades approximation errors for warping related interpolation errors.

As alternative, Spindler et al. [Spi06] proposed "camera textures", a generic object space approach to per-vertex view deformation. If the input mesh is sufficiently dense, approximation errors become negligible. For coarse meshes, they suggest the use of a dynamic mesh refinement.

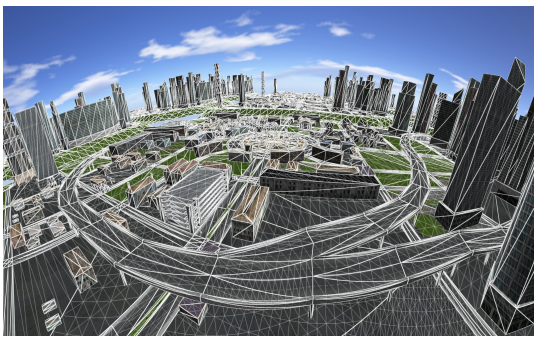


Figure 5. A cylindrical projection with 160° field of view. Our refinement is usable with any rendering technique. Geometry is hinted as in Figure 4.

We implement a per-vertex cylindrical projection to demonstrate a vertex-based approximation relying on mesh refinement. The pattern selection function is based on an edge's horizontal screen length. Since the vertical direction does not introduce approximation errors (it uses a perspective projection), this reduces the overall refinement level. Vertex conversion in pass 3 is a simple weighted sum. Figure 5 shows the result and hints both, the unrefined and refined mesh. This example also exhibits the use of additional rendering attributes (texture coordinates etc.).

6. RESULTS

For performance evaluation, we compare our method (DMR) to [Bou07] (ARP) for the cylindrical projection described in Section 5.2. Both implementations use identical refinement patterns, represented as non-indexed triangles for DMR and as indexed degenerated tristrips for ARP. Our pattern selection pass (pass 1) with an

 ISBN 978-86943-15-2

additional buffer readback has been added to ARP to enable identical dynamic refinement. Both methods are implemented with one frame latency per pass to hide dependencies between passes.

The measurements have been taken on a PC with an AMD Athlon 64 X2 4400+ with 2GB RAM and an nVidia GeForce 8800GTS with 640MB RAM. The viewport size was 1600x1200. For optimal performance, the programs have been forced to run on a single core during measurements. The measurements use the same flight path through a textured small (13639 triangles) and large (122751 triangles) city data set.

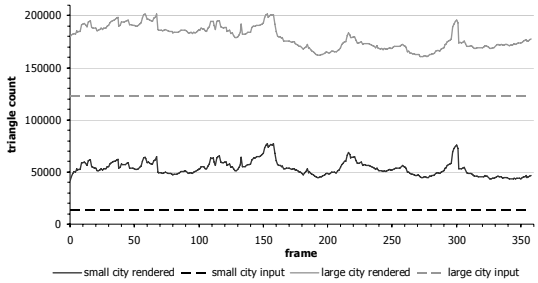


Figure 6. Triangle count for the test animation.

Figure 6 shows the input and rendered triangle count for each frame of the flight path at a maximum horizontal edge length of 10 pixels. It shows the dynamic view-dependent refinement. As expected, the refinement ratio is rather low and almost constant. Since the small data set is only a portion of the large one, the curves look very similar.

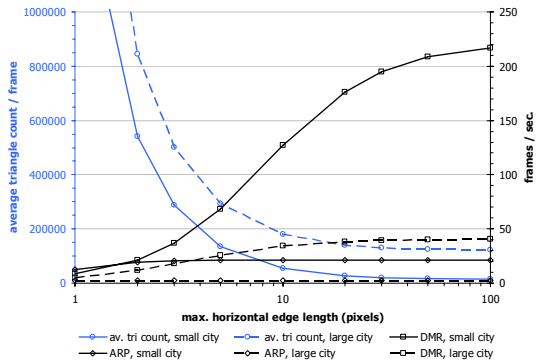


Figure 7. Cylindrical projection performance for varying refinement settings. The x-axis uses a logarithmic scale.

Figure 7 shows the resulting frame rates and average triangle counts for varying maximum edge length. For the tested use cases, Boubekeur's method is almost independent of the rendered triangle count and only depends on the input triangle count, i.e. CPU-GPU communication is the method's bottleneck as described in [Bou07]. Our method scales with both, input and rendered triangle count. By removing the communication overhead, our method can always fully saturate the GPU while ARP requires a sufficient refinement ratio for that. Consequently, our method outperforms ARP for reasonable error bounds despite the more complex algorithm. Note, that we were not able to reproduce the frame rates for ARP reported by [Bou07]. This has no impact on its general performance behavior.

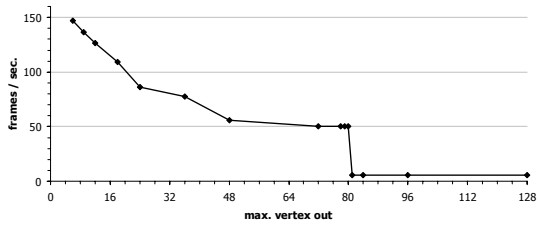


Figure 8. Performance results for varying geometry shader output limit.

Figure 8 shows the geometry shader's performance for different output limits, measured for the small city data set at a maximum horizontal edge length of 10 pixels. The actual refinements and thus geometry shader outputs for pass 2 are identical for all test runs (cf. Figure 6), only the allowed maximum vertex output varies. Up to a maximum output of 80 vertices, the curve drops as expected. Starting from 81 up to the maximum of 1024 vertices (not shown), the frame rate suddenly drops to a constant 5.75 fps. This curve shows the importance of choosing a sensible maximum vertex output. We use 12 vertices, i.e., 4 triangles, as trade-off to allow for fast pattern growth (e.g., for an invisible triangle entering the viewport) and high frame rates.

7. CONCLUSIONS

We presented a novel approach to generic mesh refinement that first leverages the power of geometry shaders to run exclusively on the GPU in an incremental multi-pass scheme. Without continuous communication between CPU and GPU, our method is most efficient for problems with large input triangle meshes and/or low refinement ratios. In particular, view-dependent approaches to geometry synthesis or vertex-based approximations profit from our approach. Nevertheless, it does not replace previous methods, but rather complements them as its efficiency drops with higher refinement ratios.

Our current implementation largely depends on the design of DirectX 10 class GPUs. It could be improved by using indexed tristrips for the refinement patterns. For that, a new hardware feature is required. Similar to the existing "primitive restart" feature, a prospective "next instance" feature could enable a geometry shader to emit blocks of indices separated by a special index

to a single buffer. During vertex fetch, this special index increments the instance ID instead of provoking a vertex. In a subsequent pass, the blocks could then be identified as separate instances and processed with varying shader parameters accordingly.

Future developments, such as a separate programmable hardware tessellation unit [Tat07], will hopefully ease generic mesh refinement and improve its performance. Our future work concentrates on applications for mesh refinement. In particular, we explore the use of vertex-based approximations as alternative approach to existing solutions. The cylindrical projection described in Section 5.2 represents an example and initial result that competes with traditional image warping techniques.

Finally, we explore a unified approach to mesh refinement and mesh simplification. We hope to find a continuous level-of-detail algorithm that not only simplifies distant meshes to gain performance, but also refines close meshes to gain visual detail. For example, outdoor scenes could benefit from such an algorithm as only a single medium sized mesh per object can be used for all rendering.

8. REFERENCES

[ATI01] ATI Inc. TRUFORM white paper. 2001.

[Bli92] Blinn, J. Hyperbolic Interpolation. IEEE Computer Graphics and Applications 12, No. 4, pp. 89-94, 1992.

[Bly06] Blythe, D. The Direct3D 10 system. In Proc. of ACM SIGGRAPH, ACM Press, pp. 724-734, 2006.

[Bok06] Bokeloh, M. and Wand, M. Hardware accelerated multi-resolution geometry synthesis. In Proc. of the 2006 symp. on Interactive 3D graphics and games, ACM Press, pp. 191-198, 2006.

[Bou05a] Boubekeur, T., Reuter, P. and Schlick, C. Scalar Tagged PN Triangles. In Eurographics (Short Paper), pp. 17-20, 2005.

[Bou05b] Boubekeur, T. and Schlick, C. Generic mesh refinement on GPU. In Proc. of the ACM SIGGRAPH/EUROGRAPHICS conf. on Graphics hardware, ACM Press, pp. 99-104, 2005.

[Bou07] Boubekeur, T. and Schlick, C. A Flexible Kernel for Adaptive Mesh Refinement on GPU. Computer Graphics Forum OnlineEarly Articles, 2007.

[Bun05] Bunnell, M. Adaptive Tesselation of Subdivision Surfaces with Displacement Mapping. In M. Pharr (eds.). GPU Gems 2, Addison-Wesley, pp. 109-122, 2005.

[Cho04] Choi, Y.-S., Chung, K.-S. and Kim, L.-S. Adaptive Tessellation of PN Triangles Using Minimum-Artifact Edge Linking. IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences E87-A, No. 10, pp. 2821-2828, 2004.

[Coh98] Cohen, J., Olano, M. and Manocha, D. Appearance-preserving simplification. In Proc. of the 25th annual conf. on Computer graphics and interactive techniques, ACM Press, pp. 115-122, 1998.

[DeC07] DeCoro, C. and Tatarchuk, N. Real-time mesh simplification using the GPU. In Proc. of the 2007 symp. on Interactive 3D graphics and games, ACM Press, pp. 161-166, 2007.

[Dia06] Diaz-Gutierrez, P., Bhushan, A., Gopi, M. and Pajarola, R. Single-strips for fast interactive rendering. Vis. Comput. 22, No. 6, pp. 372-386, 2006.

[Eva96] Evans, F., Skiena, S. and Varshney, A. Optimizing triangle strips for fast rendering. In Proc. of the 7th conf. on Visualization '96, IEEE Computer Society Press, pp. 319-326, 1996.

[Gu02] Gu, X., Gortler, S. J. and Hoppe, H. Geometry images. In Proc. of the 29th annual conf. on Computer graphics and interactive techniques, ACM Press, pp. 355-361, 2002.

[Gut05] Guthe, M., Balázs, Á. and Klein, R. GPU-based trimming and tessellation of NURBS and T-Spline surfaces. ACM Trans. Graph. 24, No. 3, pp. 1016-1023, 2005.

[Lin96] Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L. F., Faust, N. and Turner, G. A. Real-time, continuous level of detail rendering of height fields. In Proc. of the conf. on Comp. graphics and interactive techniques, ACM Press, pp. 109-118, 1996.

[Los03] Losasso, F., Hoppe, H., Schaefer, S. and Warren, J. Smooth geometry images. In Proc. of the 2003 Eurographics/ACM SIGGRAPH symp. on Geometry processing, Eurographics Association, pp. 138-145, 2003.

[Pul96] Pulli, K. and Segal, M. Fast rendering of subdivision surfaces. In Proc. of the eurographics workshop on Rendering techniques '96, Springer-Verlag, pp. 61-70, 1996.

[Reu05] Reuter, P., Behr, J. and Alexa, M. An Improved Adjacency Data Structure for Fast Triangle Stripping. Journal of Graphics Tools 10, No. 2, pp. 41-50, 2005.

[Shi05] Shiue, L.-J., Jones, I. and Peters, J. A realtime GPU subdivision kernel. ACM Transactions on Graphics (TOG) 24, No. 3, pp. 1010-1015, 2005.

[Spi06] Spindler, M., Bubke, M., Germer, T. and Strothotte, T. Camera textures. In Proc. of the 4th international conf. on Computer graphics and interactive techniques in Australasia and Southeast Asia, ACM Press, pp. 295-302, 2006.

[Tar06] Tariq, S. Next Generation Effects in DirectX10. In NVidia tutorials session at SIGGRAPH, 2006.

[Tat07] Tatarchuk, N. Real-Time Tessellation on GPU. In Course 28: Advanced Real-Time Rendering in 3D Graphics and Games. ACM SIGGRAPH 2007, 2007.

[Vla01] Vlachos, A., Peters, J., Boyd, C. and Mitchell, J. L. Curved PN triangles. In Proc. of the symp. on Interact. 3D graphics, ACM Press, pp. 159-166, 2001.

[Whi05] Whitted, T. and Kajiya, J. Fully procedural graphics. In Proc. of the ACM SIGGRAPH/EUROGRAPHICS conf. on Graphics hardware, ACM Press, pp. 81-90, 2005.

[Yan05] Yang, Y., Chen, J. X. and Beheshti, M. Nonlinear Perspective Projections and Magic Lenses: 3D View Deformation. IEEE Comput. Graph. Appl. 25, No. 1, pp. 76-84, 2005.

A Hybrid Technique for Creating Meaningful Abstractions of Dynamic 3D Scenes in Real-Time

Niall Redmond
Graphics, Vision and Visualization
Group, Trinity College Dublin, Ireland
redmondn@cs.tcd.ie

John Dingliana
Graphics, Vision and Visualization
Group, Trinity College Dublin, Ireland
john.dingliana@cs.tcd.ie

ABSTRACT

We present an approach for creating non-photorealistic renderings of 3D scenes in real-time. We employ a hybrid system which uses both image-space and object-space techniques for creating fast and effective results. A reliable method of edge detection is presented to find all important edges within a scene. This edge detection technique is combined with a painterly renderer to render the scene using different levels of abstraction. This is used to increase the saliency of important objects and remove extraneous detail. 3D object information is used to apply an object-based segmentation technique which allows each scene object to be rendered using a single abstraction level depending on its scene importance. The abstraction techniques are implemented on the GPU which helps the system achieve interactive rates.

Keywords: NPR, Adaptive, Real-time, GPU.

1 INTRODUCTION

Non-photorealistic rendering (NPR) has become a popular technique for both stylization of images and also for emphasizing important communication content within images or scenes. NPR can create simple renderings of complicated scenes to make them faster and simpler to comprehend. This can be done by borrowing techniques which have long been used by artists to control a viewers focus on an image. NPR is therefore heavily concerned with human perception of scenes and images.

Our main goal is to use non-photorealistic abstraction to produce meaningful renderings which emphasize vital features of an image and reduce the impact of unimportant information. We present a system which uses 3D object information in the scene to create meaningful and natural non-photorealistic renderings. By doing this we aim to allow for user judgment to be effectively influenced and guided in interactive scenes. Preliminary results for the system were presented in [19].

The majority of existing automatic NPR techniques for 3D scenes are not applicable to generic scenes as they are too expensive to run in real-time, are tailored for limited scene sizes or will only work for a single type of 3D model, such as polygon models [16] or volumetric models [23]. These types of object based methods require complete knowledge of the geometry detail in the scene and they are often unsuitable where important detail is only available in texture. Image-based abstraction techniques can solve these problems as access to the model and scene data is not necessary and texture detail can be taken into account. However most existing image based abstraction techniques are suited only for single images or video and do not exploit available 3D object information. We present a system which uses a hybrid approach to create non-photorealistic renderings. Image-space techniques are used to ensure interactive rates while object-space data is also used to create renderings which can adapt to the scene and create natural looking results.

Figure 1: Example of our object based abstraction

To demonstrate our system we use Virtual Dublin [7], a virtual urban environment, developed in Trinity College, which runs in real-time. This is a typical interactive application where our system could be used to emphasize or omit detail in order to influence a users

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 2008 conference proceedings, ISBN
WSCG'2008, February 4 – February 7, 2008
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

choices in the scene. It is also an example of a large complex scene, containing over 80,000 polygons. The building objects themselves are relatively coarse in their geometry but the textures contain much detail. Our abstraction techniques could be used to highlight particular historical landmarks or popular sights in Virtual Dublin for tourism purposes.

For our system to be able to run effectively in interactive applications it is necessary for it to run in real-time. We achieve this using image space NPR methods which are implemented on GPU pixel shaders. Past research shows that image space NPR techniques can be used effectively to produce real-time applications [25] [11].

The paper is organised as follows. Section 2 reviews past research in various types of non-photorealistic rendering, studies of NPR with different levels of abstraction and also research in the relationship between psychology and NPR. The non-photorealistic rendering methods used are described in Section 3. Section 4 contains information about the different levels of abstraction created, the method of segmenting the scene and the system used to determine each objects' abstraction levels. Section 5 contains results obtained from our system while Section 6 discusses the work and puts forward possible future work.

2 RELATED WORK

There has been much research into various types of automatic non-photorealistic rendering, both using image-space and object-space techniques [8] [16] [25]. The most popular styles include painterly rendering, pencil shading, silhouette drawing and cel-shading. Painterly rendering is the most heavily researched style of NPR. Many different techniques exist for creating painterly renderings although the majority of these are only suitable for single images and not for real-time applications due to processing time and frame to frame coherence.

Among the earliest work on painterly rendering was the research by Haeberli in 1990 [4]. Haeberli introduced a semi-automated system which gave the user control over the painterly image that was being created. While it was a simplistic method which required a lot of user interaction it proved that convincing painterly renderings were possible. Following on from that automatic methods were presented for creating painterly renderings for both single images and 3D scenes. For image based painterly renderings [13] [8] the automatic simulation of various size brushstrokes meant a large amount of processing. Similarly the 3D particle system presented in [16] for creating painterly renderings for 3D scenes was an expensive one and impossible to run for large scenes in real-time.

Real-time methods for obtaining NPR results are presented in [25] where image based techniques are used on video to produce non-photorealistic results. Each frame of the video is put through a series of image processing filters. First an anisotropic diffusion filter is used to produce a painterly like effect, then an edge detector and finally a colour quantization filter. A simliar filter to the anisotropic diffusion filter used by Winnemoller et al. was earlier put forward in [14]. Here an edge preserving smooth filter is presented which gives a painterly effect. This filter, known as the Kuwahara filter, works on a per pixel basis and is both fast and effective.

Santella and DeCarlo have done much research on the creation and evaluation of non-photorealistic images with various levels of detail. In [21] eye tracking data is used to guide automatic painterly renderings of images. The eye-tracker is used to first retrieve the visually important parts of an image then abstraction is performed. The abstraction uses brushstrokes of various sizes depending on the visual importance of each point. Following on from that paper Santella and DeCarlo evaluated their work in [22]. Eye-tracking data was again used, this time to investigate whether images created using the original system successfully focussed a users interest on particular parts of an image. The results validated the design of their original system and showed that a users focus can be guided using NPR. The renderings produced by Santella and DeCarlo are quite effective although the technique is only applicable to single images. The images also suffer from unwanted background detail near focus points in the image due to the constrains of the purely image-space technique.

In [6] the relationship between NPR and psychology is explored. Here it is found that NPR can influence user perception and judgement. Users tended to select objects with bold edges over other objects within a scene. It was also shown that NPR can be used to guide user behaviour in navigation and exploration, users viewed paths with higher detail as more interesting for exploration relative to paths with low detail.

Edges can be an important part of non-photorealistic rendering. The addition of edges to a scene can help create clearly distinct regions and bold edges can help to direct user focus to particular objects. Through perceptual tests in [5] it is shown that humans are good at inferring the shape of an object from its line drawing alone. Edges can therefore be useful when rendered on their own and much research has gone into the generation and stylization of edges [17] [12] [3].

Object space algorithms can find edges very effectively but are very expensive and dependant on the com-

plexity of the scene. Image space edge detectors, despite not being as accurate as object space edge detectors, can achieve good results and run at interactive rates. Some hybrid techniques exist which use image-space edge detectors to find discontinuities in the depth buffer and therefore find 3D edges [20]. These methods can find object silhouettes quite well although they are dependant on the size of the depth buffer and miss edges between objects at the same depth level.

The Canny edge detector [2] is known as the optimal image space edge detector due to its accurate and noise-free results but the complexity associated with it makes it impossible to run in real-time. The Marr-Hildreth algorithm [15] for detecting edges uses the Laplacian of Gaussian function on an image and then looks for the zero crossings in the filtered result. The Difference of Gaussian method is a fast approximation of the Marr-Hildreth algorithm.

NPR has also been used successfully in visualization. [1] presents a technique for using NPR for medical visualizations with emphasis and focussing methods. Silhouettes and hatching are used among other methods. The results are effective although impossible to run in interactive applications. A method for pen-and-ink rendering for volumetric visualization is described in [23] where NPR is shown to be an effective mechanism for improving interpretation of images. NPR could be used for other types of visualization such as vehicle or housing projects.

Virtual Dublin [7], a large-scale immersive urban simulation developed in Trinity College, is used to demonstrate our system. It simulates Dublin city centre and the surrounding areas and contains over 80,000 polygons. Virtual Dublin is an ongoing project with many research areas, such as crowd simulation, traffic simulation, and multisensory perception. Our abstraction system could be used effectively with an application such as Virtual Dublin to help guide a user around the city.

3 RENDERING TECHNIQUES

Our system uses image space techniques implemented on the GPU, which work on a frame to frame basis. To effectively highlight important content within a scene a painterly renderer and an edge detector were used. First the Kuwahara filter is used to abstract the scene in a painterly style and then a Difference of Gaussian edge detector is used to highlight the significant edges in the image.

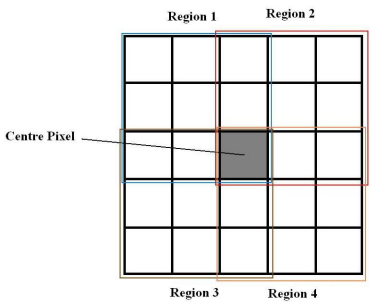


Figure 2: Example of the layout of a 5x5 Kuwahara filter containing 3x3 sampled regions. Each of the four regions is indicated using a seperate colour

3.1 Painterly Abstraction

To produce effective non-photorealistic renderings we implemented a painterly abstraction technique to obtain various levels of detail within an image. The Kuwahara filter, an edge-preserving smooth filter, was chosen as it is suitable for real-time applications, it gives a convincing painterly effect and it has good frame-to-frame coherence with no flicker.

The Kuwahara filter works on a per-pixel basis throughout the image. The filter calculates the mean colour and variance in each of four adjacent regions for each pixel. These regions are generated by dividing the surrounding area into four overlapping windows, each containing the centre pixel itself (see Figure 2). The output colour for each pixel is the mean colour for the adjacent window with the smallest variance. This has the effect of smoothing internal regions while keeping the edges between colours sharp.

Using varying sampled region size can give different levels of abstraction. Large filters, such as 9x9, produce more smoothing but don't preserve edges as accurately. Large filters are also significantly slower. If a larger level of abstraction is required and edges still need to be preserved it is preferable to apply a smaller filter iteratively (e.g. 5x5). This has the effect of preserving edges and increasing the abstraction. Using various filter sizes in combination with applying filters iteratively can produce many different levels of abstraction (see Figure 4).

3.2 Edge Detection

As discussed in Section 2, edges are an excellent way of directing user focus. Image-space edge detectors are not as accurate as object-space edge detectors but are needed for speed and also for finding edges within textures which may appear in a scene. Texture edges are important within a large environment such as Virtual Dublin because large amounts of complex textures are

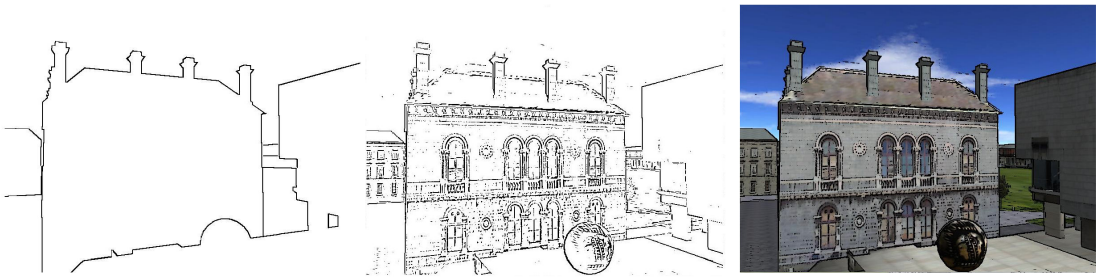


Figure 3: Edge detection example: 3D Silhouette edge image(Left), Image space edge image(Middle) and the two overlaid on the original image to display all scene edges(Right)

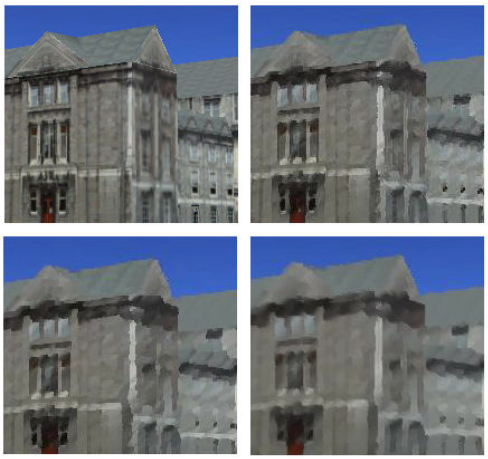


Figure 4: A section of an image when there is no abstraction(top left), a 3x3 Kuwahara filter(top right), a 3x3 Kuwahara filter applied twice(bottom left) and a 5x5 Kuwahara filter(bottom right)

typical of this sort of application. These textures may contain important information which may be emphasized to the user.

Image-space edge detectors can achieve good results and work at interactive rates although they can be unreliable due to the fact that some 3D edges can be missed due to colour similarities. We present a hybrid edge detection system which uses both object-space and image-space techniques to reliably find all edges in the scene, including 3D object silhouettes and edges which exist within textures.

Difference of Gaussian (DoG) edge detection was chosen to be the image-space edge detector used in our system. More accurate but expensive detectors exist such as Canny, while the Sobel filter is faster but too unreliable. DoG was seen as a good compromise between speed and accuracy. Difference of Gaussian edge detection works by simply applying a gaussian blur to an image using two different blur levels. The resulting edges can then found by calculating the difference between the two blurred images and finding all pixels with a difference under a threshold.

The Gaussian blur function can be seen in Equation 1 below, the function is used for creating a Gaussian mask from a sigma value. The mask is then passed through the image to blur each pixel. Each of the two blur images is created using a different sigma value. The Marr-Hildreth [15] ratio of 1.6 between the blurred images was used to give the most effective result.

$$G(u,v) = (\frac{1}{2\pi\sigma})e^{\frac{-(u^2+v^2)}{2\pi^2}} \quad (1)$$

σ = standard deviation of the Gaussian distribution
(u,v) = distance from centre of mask

The Difference of Gaussian edge detection is run for the framebuffer image to find the majority of the scene edges, including any edges which might occur within object textures. The same DoG edge detection process is then run on the false colour image generated from the first-pass rendering (see Section 4.2). By doing this all 3D silhouettes edges are found for scene objects. As can be seen from Figure 3 the 3D silhouette image finds edges which the image-space detector might miss, in the example edges are missed particularly around the roof. Implementing these two edge detectors gives a reliable method of finding all significant edges within a scene.

While DoG edge detection is run for both the framebuffer image and the false colour image, a different edge threshold is used for each image. This is because all edges from the simple false colour image are desirable whereas only the strongest edges from the framebuffer image are needed. DoG edge detection can be implemented on a pixel-by-pixel basis making it ideal for GPU pixel shaders. The edges found in the edge detection process can then be overlaid on the original image to give convincing results containing all scene edges (see Figure 3).

4 SCENE ABSTRACTION

4.1 Abstraction Levels

Using the two NPR techniques described, it is possible to produce different levels of abstraction. These levels

Figure 5: Examples of our abstraction levels. Left (level 1, most salient), Middle (level3) and Right (level 5)

can be used to increase the contrasts between scene objects and therefore guide a users focus within a scene more successfully. In addition to these methods the implementation of a fading algorithm is also introduced to increase object saliency.

The fading algorithm is done in two parts, first the colour of an object can be altered by utilizing simple operations on the RGB values of the object. This fades the objects colour and therefore lowers its visual saliency in the scene. Edges can also be faded out by altering the alpha value of each edge pixel to increase its transparency and blend it with the colour from the underlying normal rendering. This serves to make some edges bolder than others and give them more prominence in an image. Combining these techniques enable objects of high importance to be rendered with increased saliency by using clearer abstraction levels and bolder edges. Similarly objects of lesser relevance can be rendered with large abstraction levels and weak edges.

Our system can use five levels of abstraction within scenes depending on object significance. Application of the Kuwahara filter with a combination of varying filter size and multiple applications of the filter allows the generation of the different abstraction levels. These multiple degrees of painterly rendering allow for smooth abstraction transitions within a scene.

Five levels of abstraction for both colour and edge fading are again used to match with the five painterly abstraction levels. Five levels were chosen as we found it to be a sufficient number to successfully abstract a scene smoothly. It is possible to introduce more levels of abstraction if necessary. An example of three of the finished abstraction levels can be seen in Figure 5 and a detailed description of the five degrees of abstraction we use can be seen in Table 1.

Level	*Kuwahara*	*RGBFade*	*EdgeFade*
1	None	No change	No transparency
2	5x5 filter	0.12 addition to each colour	0.08 transparency with alpha value
3	5x5 filter applied twice	0.18 addition to each colour	0.16 transparency with alpha value
4	9x9 filter	0.24 addition to each colour	0.24 transparency with alpha value
5	9x9 filter applied twice	0.30 addition to each colour	0.30 transparency with alpha value

Table 1: Detailed description of the five levels of abstraction used

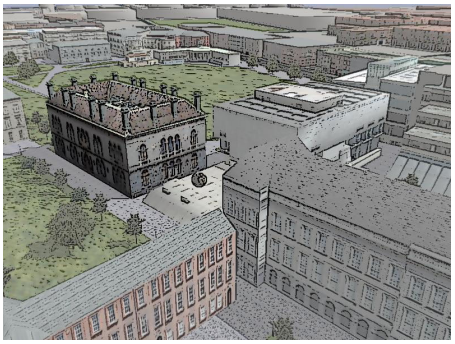


Figure 6: An example of a scene with object based-segmentation. One object is rendered clearly while the rest of the scene is abstracted

4.2 Scene Segmentation

To create meaningful non-photorealistic renderings which fully highlight important objects and find all edges it is necessary to use the available 3D object information. We implemented a method which uses false colour rendering of a scene to find edges and guide abstraction. False colour rendering is a popular technique used in synthetic vision for object identification [18] and similar techniques in NPR use an ID-buffer to segment a scene e.g. in [24] to give each object a different halftoned texture.

When rendering each frame of the scene we use a two pass system. We first render the scene with each object rendered in a distinct colour. The amount of unique colours available depends on the depth of each component in the colour buffer. When rendering during the false colour pass it must be ensured that all objects are rendered in flat colour, therefore lighting, textures etc. must be disabled. The simplicity of the rendering pass makes it fast enough for interactive applications. The

false colour rendering pass is a useful tool for object identification on each frame.

4.3 Object-based Abstraction

It is desirable to render each object in an abstraction style which suits that objects importance in the scene. This allows the effective rendering of scenes which we propose will draw an users attention to certain objects and paths. We put forward an object classification algorithm for doing this based around the most visually important objects within the scene.

The information gained from the false colour rendering can be used to calculate which abstraction level each object should be rendered in. The first step to doing this is calculating the centre point of each visible salient object. These visually important objects can be pre-defined, set by the current mouse position or image centre, or be dependant on a users current goal or interest. For each pixel in the image the distance to the nearest important objects centre point is found.

After the image has been fully traversed the abstraction levels for each scene object must be calculated. This is necessary for creating meaningful NPR renderings which increase saliency of certain objects but also fit in naturally with the scene. As can be seen from Figure 9 there is an obvious difference between the rendering done using only distance as a guide for abstraction compared with the rendering which does the abstraction on a per-object basis.

The per-object abstraction uses the false colour image and distance information for each pixel to calculate the average distance for each scene object. This is done by first inspecting the false colour image to discover which pixels belong to a certain object and then using the distance information for those pixels to find an average. Depending on the result each object can be defined as belonging to one of the five abstraction levels. This information can then be passed to the pixel shader to do the necessary abstraction.

The pixel shader, written in CG, is responsible for implementing the abstraction itself. Only one pixel shader is used to do all abstraction. The work performed by the shader depends on which abstraction level has been defined for each pixel. The shader first performs the Kuwahara filtering followed by the edge detection process. It is done in this order as some edges may be altered in the painterly process, especially for high levels of abstraction. Finally the edge and colour fade are performed on the pixels values, again dependant on the abstraction level for the pixel. The final output pixel

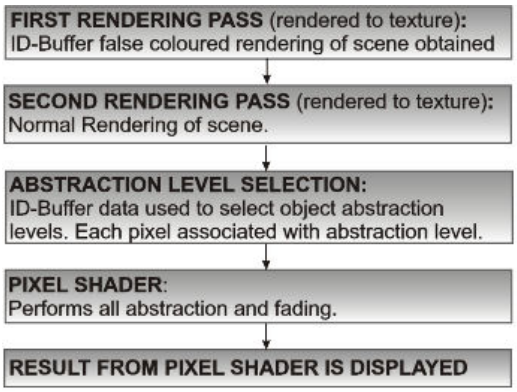


Figure 7: Overall algorithm for the system

is then written to a texture which can be displayed on-screen when the shader has completed. The overall algorithm for our system can be seen in Figure 7.

5 RESULTS

We implemented our system using C++ with OpenGl and the GPU pixel shader was written in CG. The system was tested on a Pentium D, 3.72GHz computer with 2GB of RAM using an NVidia GeForce 7950 GX2 GPU.

The implementation of the abstraction techniques on the GPU coupled with the fast false colour rendering techniques mean that interactive frame times are achieved, which can be seen from table 2, below. The major advantage of using image space methods such as the ones implemented here is that they are dependant on screen resolution rather than scene complexity which means frame rates can be guaranteed. The object segmentation algorithm does depend on scene complexity but the speed of the false colour rendering makes the slow-down minimal, even for large complex scenes such as Virtual Dublin, on which this system was tested.

RenderingStyle	640*x*480	800*x*600	1200*x*800
Regular rendering (no abstraction)	102	100	98
2 pass rendering (false colour pass)	54	53	52
Edge detection (with 2 pass)	50	47	45
Painterly rendering (with 2 pass)	51	48	47
Full abstraction system	46	43	42

Table 2: Frames per second achieved by the system using various abstraction options on Virtual Dublin at a different resolutions.

As can be seen from the frame times, the biggest slowdown of the system is the false colour rendering

Figure 8: Example of two scene images with and without abstraction. The yellow markers indicate the most salient points of each image

although Virtual Dublin still runs at interactive rates. The two pass rendering method is also vital for our system and is the fastest way of achieving object segmented renderings. The table also shows the fact that the image based abstraction methods do depend on image resolution although the slowdown is minimal for large changes in resolution. This is largely thanks to the speed of the algorithm when implemented on the GPU.

The results from our system were evaluated using a saliency mapping algorithm [9]. This technique uses a multi-scale approach for finding the location of salient regions based on local image structure. Regions of high contrast are found for colour, intensity and orientation of an image on a number of scales. The results are then normalised and summed to create the final saliency map for an image.

We used the saliency mapping system to confirm that our techniques can successfully increase the saliency of certain objects within a scene. Figure 8 shows two scene images both with and without our abstraction methods. This figure proves that adding abstraction to a scene image can change the most salient point to the object we want to emphasize in the scene. While user studies would provide more worthwhile information about the effectiveness of our system these preliminary tests show that our methods are successful and could be used to guide a users focus in a scene.

The objects of interest in each scene could be chosen in a number of ways depending on the application. They could be pre-defined to encourage a user to take certain routes or to look at certain objects within the scene. In an application such as Virtual Dublin this could be highlighting historical monuments or popular museums. The visually important object at any one time could also be dependant on a users goal in an interactive application at any particular time to make the current objective clear. Along with tourism the system would be useful in games or route-finding applications.

6 CONCLUSION AND FUTURE WORK

We have presented a novel technique for generating meaningful non-photorealistic renderings of large 3D scenes in real-time. Our main contribution is a hybrid method for rendering each scene object in a particular abstraction level which is suitable for its importance in the scene. We have shown that our methods could be used to create more salient objects and guide a users focus, thereby directing and influencing user judgement. We also present a novel hybrid method for effectively combining object space edges and edges within textures to find all scene edges.

Our system successfuly provides interesting variable abstractions of a scene in real-time which was our main goal. We also validate our abstraction methods by showing that scene saliency can be altered successfully. To fully investigate the effectiveness of the system at directing user attention, we propose future work in the field of perception.

An eye-tracker may be used such as in previous work like [22] where it is proved that eye-tracking data can be a useful tool for the evaluation of NPR systems or in [10] where saliency-guided enhancement was evaluated with eye-tracking data. In addition, eye-tracking experiments may be useful in tuning the abstraction parameters to create the most effective abstractions of the scene. Further research could include task based experiments to evaluate the success of the renderings. Comparison and evaluation of tasks performed both with and without abstraction would give an accurate measure of the effectiveness of the object based abstraction system.

REFERENCES

[1] Christian Tietjen Bernhard Preim and Christian Dorge. Npr, focussing and emphasis in medical visualizations. *Simulation and Visualisierung*, pages 139–152, 2005.

[2] J.F. Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8:769–798, 1986.

[3] Cassidy J. Curtis. Loose and sketchy animation. In *SIGGRAPH '98: ACM SIGGRAPH 98 Electronic art and animation catalog*, page 145. ACM Press, 1998.

[4] Paul Haeberli. Paint by numbers: abstract image representations. In *SIGGRAPH '90*, pages 207–214. ACM Press, 1990.

 ISBN 978-86943-15-2

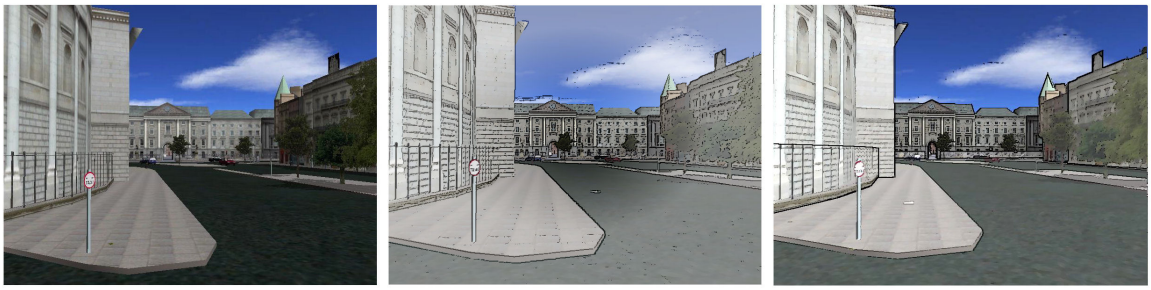


Figure 9: A comparison between the original rendering, a purely distanced based abstractive rendering and our finished hybrid abstraction method, which focuses on the far building at the centre of the image

[5] Nick Halper, Mara Mellin, Christoph S. Herrmann, Volker Linneweber, and Thomas Strothotte. Surface gradients, contours and the perception of surface attitude in images of complex secenes. *Perception*, 25:701, 1996.

[6] Nick Halper, Mara Mellin, Christoph S. Herrmann, Volker Linneweber, and Thomas Strothotte. Psychology and non-photorealistic rendering: The beginning of a beautiful relationship. pages 277–286, 2003.

[7] John Hamill and Carol O'Sullivan. Virtual Dublin - A Framework for Real-Time Urban Simulation. *Journal of WSCG*, 11:221–225, February 2003.

[8] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. *Computer Graphics*, 32(Annual Conference Series):453–460, 1998.

[9] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[10] Youngmin Kim and Amitabh Varshney. Saliency-guided enhancement for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):925–932, 2006.

[11] Allison W. Klein, Wilmot Li, Michael M. Kazhdan, Wagner T. Corrêa, Adam Finkelstein, and Thomas A. Funkhouser. Non-photorealistic virtual environments. In *SIGGRAPH '00*, pages 527–534. ACM Press/Addison-Wesley Publishing Co., 2000.

[12] Adam Lake, Carl Marshall, Mark Harris, and Marc Blackstein. Stylized rendering techniques for scalable real-time 3d animation. In *NPAR '00:*, pages 13–20. ACM Press, 2000.

[13] Peter Litwinowicz. Processing images and video for an impressionist effect. In *SIGGRAPH '97*, pages 407–414. ACM Press/Addison-Wesley Publishing Co., 1997.

[14] S. Eiho M. Kuwahara, K. Hachimura and M. Kinoshita. *Digital processing of biomedical images*. Plenum Press, 1976.

[15] D. Marr and E. Hildreth. Theory of Edge Detection. *Royal Society of London Proceedings Series B*, 207:187–217, February 1980.

[16] Barbara J. Meier. Painterly rendering for animation. *Computer Graphics*, 30(Annual Conference Series):477–484, 1996.

[17] J. D. Northrup and Lee Markosian. Artistic silhouettes: a hybrid approach. In *NPAR '00:*, pages 31–37. ACM Press, 2000.

[18] Hansrudi Noser, Olivier Renault, Daniel Thalmann, and Nadia Magnenat Thalmann. Navigation for digital actors based on synthetic vision, memory, and learning. *Computers and Graphics*, 19(1):7–19, 1995.

[19] Niall Redmond and John Dingliana. Adaptive abstraction of 3d scenes in real-time. In *EUROGRAPHICS '07: Proceedings of the Short Paper Programme*, pages 77–80, September 2007.

[20] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-d shapes. *SIGGRAPH Comput. Graph.*, 24(4):197–206, 1990.

[21] Anthony Santella and Doug DeCarlo. Abstracted painterly renderings using eye-tracking data. In *NPAR '02:*, pages 75–ff. ACM Press, 2002.

[22] Anthony Santella and Doug DeCarlo. Visual interest and npr: an evaluation and manifesto. In *NPAR '04:*, pages 71–150. ACM Press, 2004.

[23] S. M. F. Treavett and M. Chen. Pen-and-ink rendering in volume visualisation. In *VIS '00*, pages 203–210. IEEE Computer Society Press, 2000.

[24] O. Veryovka and J. Buchanan. Comprehensive halftoning of 3D scenes. In *Computer Graphics Forum (Eurographics '99)*, volume 18(3), pages 13–22. The Eurographics Association and Blackwell Publishers, 1999.

[25] Holger Winnemoller, Sven C. Olsen, and Bruce Gooch. Real-time video abstraction. In *SIGGRAPH '06*, pages 1221–1226. ACM Press, 2006.

Discovering the Covered: Ghost-Views in Information Visualization

Martin Luboschik, Heidrun Schumann
Institute of Computer Science, University of Rostock, Albert-Einstein-Str. 21, 18051 Rostock, Germany
{luboschik,schumann}@informatik.uni-rostock.de

ABSTRACT

A not negligible number of information visualization techniques uses 3D-geometry to visualize data and structures. Thereby, constantly growing data volumes influence the final visual representation and often result in the occlusion of certain items. Therefore, different approaches have been developed that mainly manipulate item positions to uncover specific items of interest or otherwise use filtering and information hiding to reduce the amount of visible items. This paper presents a novel method to adapt 3D-views from information visualization by the use of the well-known illustrative technique *ghost-view* to successfully address this occlusion problem. Applying ghost-views to 3D information visualization techniques ensures the visibility of selected items by view-dependently manipulating the transparency of unselected data: without any manipulation of positions or continuous context suppression. Our approach is applicable to most 3D visualization techniques. It is interactive and easy to adapt to existing visualization environments.

Keywords: Information visualization, illustrative rendering, ghost-views.

1 INTRODUCTION

In various fields of information visualization, 3D-visualization techniques have been developed to visualize information structures (e.g. in [RG93] [vHvW02] [BCS04]...) and information objects (e.g. [MRC91] [TK98] [TSWS05] ...).

Since the visualization of constantly growing large data volumes is a general aim of information visualization, this becomes a problem for 3D visualization techniques due to the growing probability of occlusions. Card et. al. points out that occlusions are a major problem in 3D information visualization (see [CMS99]).

In the field of information visualization, existing solutions to solve the occlusion problem mainly propagate two strategies: First a view dependent arrangement / distortion of 3D graphic primitives (e.g. in [SCCF96]) is applied that reduces the complexity in presentation space and shows objects that are occluded otherwise. The second approach, called information hiding, reduces complexity in information space resulting in a continuous reduction of visible representatives (see [Fur81]). However, the first approach is not applicable in scenarios where the graphic primitives position is necessary for interpretation of the information. The second approach will fail if the context has to be known to understand the visualized data.

However, in the field of technical and medical illustrations ghost-views are used to facilitate a view on objects of interest that are encapsulated by 3D geometry. These views have been successfully adapted to the field of volume visualization and illustrative rendering of 3D models but are rarely considered in 3D information visualization (Section 2).

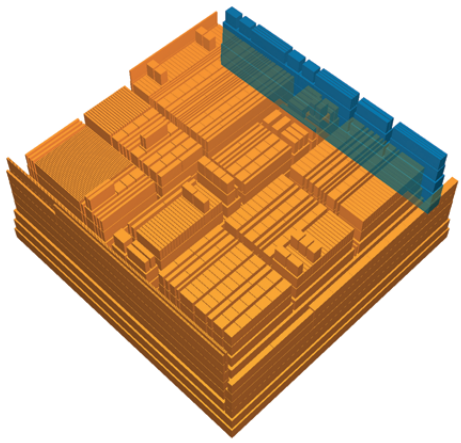


Figure 1: This figure shows the application of the proposed ghost-view technique to a 3D hiearchy visualization called steptree (see [BCS04]).

We propose the application of ghost-views to 3D information visualization techniques to guarantee the visibility of selected items without arrangement manipulation or continuous context suppression (Section 3). Our general approach that takes advance of modern graphics hardware (Section 5), is applicable to several 3D information visualization techniques – demonstrated by two examples (Section 4). The results of our approach are discussed in Section 6.

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

Figure 2: The figure shows a small setup of four boxes, whereby the light blue are unimportant objects and the orange / red boxes are objects of interest (left, shown from top). In the center, the conflict between the objects of interest is solved using standard depth sorting. At the right, gradual differences between the objects of interest ensure the visibility of the red box (importance maps are shown below).

2 BACKGROUND

A ghost-view is a specific illustration technique that is used to make objects visible that are enclosed by 3D geometry [Die05]. The transparency of the occluding geometry is manipulated locally depending on the current view point to guarantee the visibility of the objects of interest in context of the whole representation. Whereas a cutaway-view hides occluding geometry parts completely, a ghost-view maintains some information about the occluding objects shape (e.g. using semi-transparency, silhouettes or other visual clues).

Both, ghost-views and cutaway-views have been successfully applied in different fields and their automatic generation is an ongoing research topic. In [DWE03] [LRA+07] for example, such illustrative views are generated for complex 3D models and in [HBP+07] [Die05] the output on mobile devices is additionally addressed.

Moreover, ghost-views are also used in the field of volume visualization to make encapsulated volume data visible (e.g. in [BGKG05] [VKG05] [TIP05] [RBG07] [BG07]). In case of such interactive explorative scenarios ghost-views and cutaway-views are mostly calculated view-dependently to guarantee the visibility of objects of interest (e.g. a tumor) for every view point.

The recent publications demonstrate the relevance of those views to illustrative rendering and volume visualization.

In 3D information visualization the same problem exists since objects of interests may not be visible due to occlusion. After complex parametrization steps one can ensure those objects to be visible but even small interactions during the exploration process can make them disappear again. The occlusion will even lead to misinterpretation if the object of interest is the result of a user defined query. A hidden object may be interpreted as an absence of this object. This has to be avoided, since in different application scenarios the existence of specific objects is of high relevance: In a 3D network scenarios for example an important node of interest has to be visible [BTB+99] or a node holding a special kind of data within a hierarchy [TON03]. Moreover, single stocks in a 3D stock market visualization [Wri95] or specific regions in a scenario with spatial data [TSWS05] could by of high interest and should therefore be visible.

In principle, there are two very successful approaches that address this 3D occlusion problem: Information hiding techniques (e.g., in [Fur81] [TSWS05]) and distortion techniques (e.g., in [SCCF96]). Information hiding techniques reduce complexity in the information space to limit the number of opaque visual representatives. The approach of filter fisheye views [Fur81] for example, reduces – dependent of the currently selected object of interest – the number of visible objects by calculating a degree of interest for the surrounding objects. Techniques like this decrease the probability of occlusions for selected objects in a 3D visualization but they are not able to guarantee the visibility. The named approach for example, retains objects with a high degree of interest. These are often visualized close to the selected object and may occlude it (semantic affinity often means spatial affinity). Furthermore, information hiding techniques are hard to apply if the suppressed context is necessary for a meaningful interpretation of the objects of interest (e.g., for comparison).

Solutions that base on view-dependent distortions or arrangements of 3D geometry, operate in presentation space to avoid the occlusion. In [SCCF96] for example, the occluded object of interest and the view point is used to define different scalings and distortions that are applied to the objects of the visualization. The result is a free view at the object of interest. Although this technique shows the whole 3D representation, the visualization may be difficult to interpret if the position of an object of interest itself is an important information in the exploration process.

Besides these principle strategies, there are also technique-specific solutions like the predefined and view-independent use of transparency. In [RG93] [TON03] for example, every enclosing geometry is transparent to enable navigation and interpretation of the visualized hierarchy structure. However, this approach does not guarantee the visibility neither of every visualized nor a single selected hierarchy node.

Although ghost-views are an integral rendering technique in other 3D visualization and rendering scenarios, they are not considered in 3D information visualization yet. Therefore, in the following sections we introduce a simple – visualization independent – approach to generate interactive ghost-views in 3D information visualization and demonstrate this approach with two short examples.

3 A GENERAL APPROACH FOR GHOST-VIEWS IN INFORMATION VISUALIZATION

Inspired by the idea of importance driven rendering presented in [VKG05], which handles volumes of different importance to generate ghost-views and cutaway-views, our approach is based upon a so called *importance map* and a two-pass-rendering. This approach can be summarized as follows:

1. Generate the importance map view-dependently by rendering the objects of interest into a (0/1)-texture.
2. Render all objects using the importance map as a look-up table to determine, whether they are occluding interesting objects or not. In case of occlusion, their transparency is adapted locally.

Therewith, our ghost-view approach works in information and in presentation space: In information space the objects of interest are defined as key elements in the later ghost-view generation. In the presentation space the visual representatives of these elements are used for transparency adaption of occluding geometry.

During the rendering step also the objects of interest are rendered using the importance map to handle the conflict that will appear if two or more objects of interest share their position in the importance map. For this case we propose two solutions:

- Using automatically applied depth sorting always shows the object of interest in front occluding the others.
- Extending the former (0/1)-importance map to multiple importance levels and allowing gradual differences between those objects of interest. In this way, objects of interest that are not of maximum importance at that position are also adapted in their transparency according to their importance (see Fig. 2).

Since this approach manipulates transparencies of occluding geometry, it can be used for both the generation of ghost-views (semitransparent) and cutaway-views (100%-transparency).

In Section 2 we showed that the position of visual representatives may be crucial for interpretation. Therefore this position should be clear to perceive.

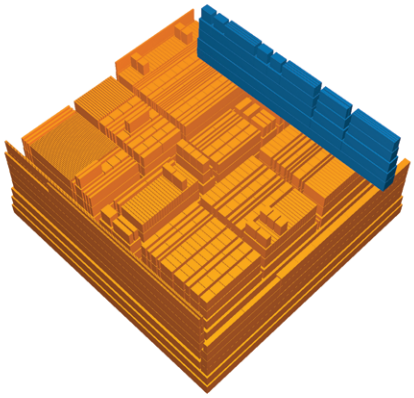


Figure 3: The use of 100% transparency (cutaway-view) without other visual clues disorders depth perception and may lead to misinterpretation (compare to Fig. 1).

Since a cutaway-view may bring the objects of interest visually in front of all other geometry, this perception – especially of depth – may be disordered (see Fig. 3). Therefore, we propagate the use of ghost-views with semitransparent occluding geometries to maintain correct position interpretation.

The application of ghost-views in different fields created also different approaches for their generation. The commonly used approach in volume visualization is a direct manipulation of voxel parameters (e.g. in [VKG05]). Thereby, the *level of sparseness* is mapped onto the transparency of a voxel for example. The rendering step is generally unmodified. Moreover, there are several approaches for ghost-view creation in technical illustrations. In [Die05] different solutions – reaching from simple CSG operations to complex multipass-rendering steps realizing a depth sorting – are demonstrated. A sorting of geometry is usually necessary to achieve *correct* transparencies that are needed in illustrative rendering.

Although our straight-forward approach does not include a view dependent sorting of geometry, it is sufficient to guarantee the visibility of objects of interest in a 3D information visualization.

4 APPLICATION EXAMPLES

In the following sections we will demonstrate our approach with two different examples from 3D information visualization. The first is a structure visualizing technique called treecube (see [TON03]) that is used for administration of 3D multimedia data. The second example comes from the field of information object visualization and is based upon a spring-model (see [TK98]).

 ISBN 978-86943-15-2

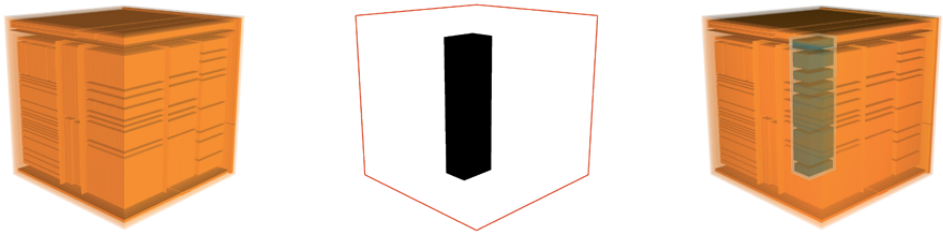


Figure 4: A family of interest (Corellidae) within a phylogenetic dataset is selected and unfortunately hidden in a standard treecube visualization (left). Using an importance map (center) in combination with a GPU-program generates a ghost-view and ensures the objects visibility (right). The red contour in the importance map is only shown for orientation purposes.

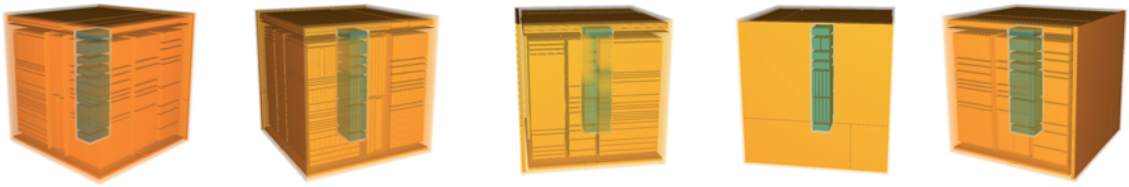


Figure 5: This series shows the view-dependent generation of ghost-views that ensures visibility within this treecube visualization.

4.1 Treecube

A treecube is a 3D extension of the well known hierarchy visualization technique called treemap [Shn92]. A cube is divided along the x,y and z-axis according to the underlying hierarchy (see [TON03] for details). Although the separation planes are semitransparent, it is not guaranteed that every node of the shown hierarchy is visible. Especially nodes that are represented by boxes in the deep interior of the cube are endangered of occlusion.

In [TON03] the treecube holds 3D multimedia data like 3D geometry models. In this case, an object of interest may be a hierarchy node that holds one kind of models (e.g. airplanes) as these models are currently used in a modelling scenario. Moreover, the structural relationship to the surrounding hierarchy is necessary for fast navigation. Therefore, that node and the surrounding structures should be visible for further use or investigation.

We demonstrate the application of ghost-views to the treecube visualization technique by using a small phylogenetic dataset ($\approx$ 580 nodes, part of [FP03]) and one selected subtree of interest (family Corellidae). Even in case of such few visualized hierarchy nodes, the selected subtree is usually not visible (Fig. 4 left).

As described in Section 3 we generate a view-dependent importance map using the selected subtree (Fig. 4, center). Afterwards, this importance map is used as a look-up table in the final rendering step for transparency adaption (Fig. 4, right).

Therewith, the occluded subtree becomes visible for every view point without a continuous context suppression (Fig. 5). This enables an interactive investigation and shows the subtrees context concerning the outer geometry. Nevertheless, other interior structures or the near context of the selected subtree are not visible.

4.2 Spring-model

To demonstrate that ghost-views are also applicable to non-encapsulating 3D visualization techniques we use a simple example, showing numerous information objects as spheres in 3D space. The objects positions are given by their attributes that influence spring-forces towards attracting points distributed on a surrounding sphere (see [TK98]). These techniques are generally used to see the distribution of the whole information space according to its attributes or the development of information objects. Moreover, it enables the comparison of single objects towards the whole volume. Therewith, this technique is generally applied to multivariate data sets (like quality of living parameters of US cities in [TK98]).

In our example we use this technique for the visualization of health data of $\approx$ 230 districts of the federal state Mecklenburg-Vorpommern, Germany in January 2000. Ten different attributes recorded by a health insurance were used to place the visual representatives of the districts. These attributes include the number of influenza cases, the cases of gastro-intestinal diseases, cases of arthropathy and more. Additionally an object holding aggregated data of all districts is visualized to

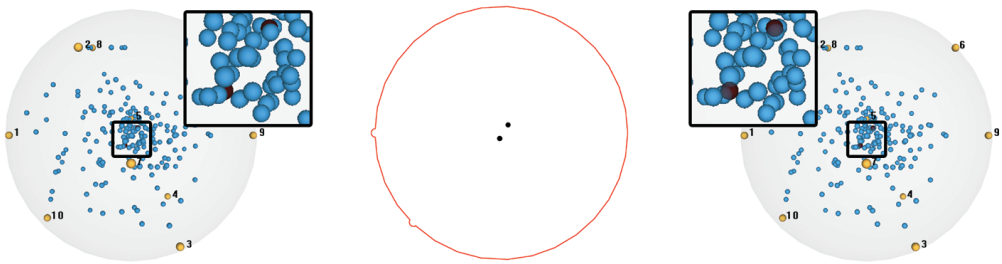


Figure 6: A district of the federal state Mecklenburg-Vorpommern, Germany is compared to an object holding aggregated data of the whole state. Their visual representatives (dark spheres) are difficult to locate in a spring-model visualization (see magnification left). Using the importance map (center) during rendering makes them visible (right). The red contour (center) is only shown for orientation purposes.

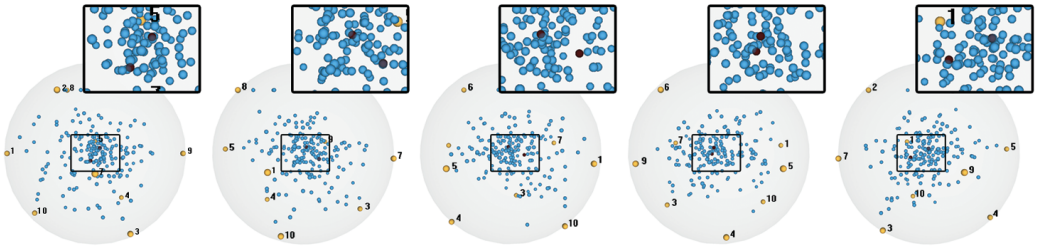


Figure 7: This series shows the view-dependent generation of ghost-views to ensure visibility within a spring model visualization.

show the average. In Figure 6 one of those districts is selected for comparison to this aggregated value (also selected).

Although a relative small data set is visualized, the selected objects are hard to see. To guarantee their visibility, we use our approach of Section 3: Both objects are used to generate the importance map that is afterwards used in the final rendering of the whole data set (see Fig. 6).

Thus, the occluded objects of interest become visible for every view point without unnecessary context suppression or distortions (Fig. 7). Semitransparent occluding objects ensure the depth and shape perception. Moreover, the whole context of the selected objects is visible due to a local transparency adaption and a missing encapsulation within this visualization technique.

5 IMPLEMENTATION

Our approach of importance map based ghost-view generation for 3D information visualization uses GPU programming facilities of recent graphics hardware to keep this approach interactive and applicable to different visualization techniques. The shown examples were implemented prototypal in Java 1.6 using JOGL (JSR-231 1.1) and GLSL (all non-optimized). A fragment shader is used to modify the transparency of the occluding geometry on a per pixel basis. The importance of the currently rendered object is passed to the shader which compares it to the importance map and adapts the pixels transparency if necessary.

The examples run on a Intel Pentium 4, 3.2GHz, 1GB RAM and NVIDIA Quadro NVS 280 PCI-E graphics card at framerates of 11 to 43 fps. This demonstrates that our method works at interactive framerates even on desktop computers that are not optimized for 3D graphics. To additionally speed up the visualization, the importance map can be rendered in lower resolutions resulting in a fringed border between the opaque and transparent parts of an occluding geometry.

The anchors in Figure 6,7 are currently not involved in the ghost-view generation but may at any time. An integration of the simple 2D labeling within our implementation is currently not possible, due to a separate rendering processing. A revised implementation should facilitate this.

6 CONCLUSION

Our introduced approach of applying ghost-views to 3D information visualizations ensures the visibility of important objects in every step of the visual analysis. Hence, it supports the exploration of large data volumes and represents an additional alternative to distortion and information hiding techniques.

The major benefit of our technique is that objects of interest can be perceived at first glance without complex parametrization steps. This visibility is guaranteed during the interactive exploration process.

 ISBN 978-86943-15-2

Furthermore, the use of modern graphics hardware keeps the ghost-view generation interactive and independent of any visualization technique or data structure. Therewith, it is also applicable to other 3D visualization scenarios than the ones described in Section 4. A so called *importance map* is used to determine whether the opacity of an occluding geometry has to be changed. The generation of that importance map is easy to implement and needs only one additional rendering pass. Since generally a lower number of polygons has to be rendered in this generation step, the maximum rendering time is twice of a standard rendering. The map ensures a locally bounded manipulation of transparencies and therewith supports the perception of context information.

Although our approach guarantees visibility without distortions, it reveals only objects of interest. In 3D visualization techniques that produce enclosures, the *near* context of the interesting object is still hidden and may hinder an interpretation. Distortion techniques are usually able to show that context. Another problem arises through the use of semi-transparencies: If attributes of the interesting objects are mapped onto its color, this color changes view-dependently due to different occlusion scenarios (see Fig. 5). This may lead to misinterpretations.

Therefore, further investigations will examine the use of other visual clues than transparency to retain depth and shape perception. Moreover, a combination with information hiding / distortion techniques could solve the problem of hidden near context.

REFERENCES

[BCS04] Thomas Bladh, David A. Carr, and Jeremiah Scholl. Extending tree-maps to three dimensions: A comparative study. In *Proceedings of the Asia-Pacific Conference on Computer-Human Interaction (APCHI'04)*, pages 50–59, 2004.

[BG07] Stefan Bruckner and Eduard Gröller. Style transfer functions for illustrative volume rendering. *Computer Graphics Forum*, 26(3):715–724, 2007.

[BGKG05] Stefan Bruckner, Sören Grimm, Armin Kanitsar, and Eduard Gröller. Illustrative context-preserving volume rendering. In *Proceedings of EUROGRAPHICS - IEEE VGTC Symposium on Visualization (EuroVis'05)*, pages 69–76, 2005.

[BTB+99] Steve Benford, Ian Taylor, David Brailsford, Boriana Koleva, Mike Craven, Mike Fraser, Gail Reynard, and Chris Greenhalgh. Three dimensional visualization of the world wide web. *ACM Computing Surveys*, 31(4):25, 1999.

[CMS99] Stuard Card, Jock Mackinlay, and Ben Shneiderman. *Readings in Information Visualization – Using Vision To Think*, chapter 2.2, pages 57–61. Morgan Kaufmann, 1999.

[Die05] Joachim Diepstraten. *Interactive Visualization Methods for Mobile Device Applications*. PhD thesis, University of Stuttgart, Germany, 2005.

[DWE03] Joachim Diepstraten, Daniel Weiskopf, and Thomas Ertl. Interactive cutaway illustrations. *Computer Graphics Forum*, 22(3):523–532, 2003.

[FP03] Jean-Daniel Fekete and Catherine Plaisant. Visualization and Pair Wise Comparison of Trees, IEEE InfoVis 2003 Contest, 2003.

[Fur81] George W. Furnas. The FISHEYE view: A new look at structured files. Technical Report #81-11221-9, Bell Laboratories, 1981.

[HBP+07] Jingshu Huang, Brian Bue, Avin Pattath, David S. Ebert, and Krystal M. Thomas. Interactive illustrative rendering on mobile devices. *IEEE Computer Graphics and Applications*, 27(3):48–56, 2007.

[LRA+07] Wilmot Li, Lincoln Ritter, Maneesh Agrawala, Brian Curless, and David Salesin. Interactive cutaway illustrations of complex 3d models. *ACM Transactions on Graphics*, 26(3):31, 2007.

[MRC91] Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The perspective wall: Detail and context smoothly integrated. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'91)*, pages 173–179, 1991.

[RBG07] Peter Rautek, Stefan Bruckner, and Eduard Gröller. Semantics driven illustrative rendering. Technical report, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2007.

[RG93] Jun Rekimoto and Mark Green. The information cube: Using transparency in 3d information visualization. In *Proceedings of the Third Annual Workshop on Information Technologies & Systems (WITS'93)*, pages 125–132, 1993.

[SCCF96] M. Sheelagh, T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia. Distortion viewing techniques for 3-dimensional data. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'96)*, pages 46–53, 1996.

[Shn92] Ben Shneiderman. Tree visualization with tree-maps: A 2-d space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.

[TIP05] Christian Tietjen, Tobias Isenberg, and Bernhard Preim. Combining silhouettes, shading, and volume rendering for surgery education and planning. In *Proceedings of the Eurographics / IEEE VGTC Symposium on Visualization (EuroVis'05)*, pages 303–310, 2005.

[TK98] Holger Theisel and Matthias Kreuseler. An enhanced spring model for information visualization. *Computer Graphics Forum*, 17(3):335–344, 1998.

[TON03] Yoichi Tanaka, Yoshihiro Okada, and Koichi Niijima. Treecube: Visualization tool for browsing 3d multimedia data. In *Proceedings of the 7th International Conference on Information Visualization (IV'03)*, pages 427–432, 2003.

[TSWS05] Christian Tominski, Petra Schulze-Wollgast, and Heidrun Schumann. 3d information visualization for time dependent data on maps. In *Proceedings of the 9th International Conference on Information Visualisation (IV'05)*, pages 175–181, 2005.

[vHvW02] Frank van Ham and Jarke J. van Wijk. Beamtrees: Compact visualization of large hierarchies. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, pages 31–39, 2002.

[VKG05] Ivan Viola, Armin Kanitsar, and Eduard Gröller. Importance-driven feature enhancement in volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):408–418, 2005.

[Wri95] William Wright. Research report: Information animation applications in the capital markets. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'95)*, pages 19–25, 1995.

Fluid Dynamic Simulation for Cutting in Virtual Environment

Sugeng Rianto
Medical Imaging and Applied Science
Curtin University of Technology
Perth, Western Australia 6102
Sugeng.rianto@postgrad.curtin.edu.au

Ling Li
Department of Computing
Curtin University of Technology
Perth, Western Australia 6102
L.Li@curtin.edu.au

Bruce Hartley
Department of Exploration Geophysics
Curtin University of Technology
Perth, Western Australia 6102
B.Hartley@curtin.edu.au

ABSTRACT

In this paper, we introduce a 3D fluid dynamics solver for real-time interactions in virtual environment. We approach the solution of differential equations based on the cubic interpolated propagation (CIP) technique on GPU. Since the CIP combine the solution for fluid equations and their interactions with the environment together, the Navier-Stokes equation can be solved efficiently. Furthermore, to achieve high performance results without involving a supercomputer, we take advantage of the parallelism and programmability of the GPU. Simulation is performed on pixels that can be considered to be a grid of cells; therefore processing on multiple vertices and pixels can be done simultaneously in parallel. This strategy is effective enough to simulate fluid dynamic model for real-time virtual cutting in 3D computer graphic. Experimental results demonstrate that the skin cutting followed by blood flowing over the anatomical surface run smoothly in a real-time interaction and realistic visual effect is achieved.

Keywords

CIP, cutting simulation, fluid dynamic solver, GPU, virtual environment

1. INTRODUCTION

Virtual reality (VR) is widely used as a training tool from science to engineering or social sciences [Ros00, Sto00]. The VR techniques provide the potential for a realistic, safe, controllable environment for novice doctors to practice surgical operations, allowing them to make mistakes without serious consequences [Ano01, Hig02]. Virtual reality is also widely known as virtual (or synthetic) environments (VEs). These are three dimensional computer generated environments that the user is not only able to experience interactively but also able to manipulate in real time [Was01]. The way humans interact with their physical environments are artificially imitated in VEs.

VEs provide natural interface between humans and computers in virtual reality. A VE consists of an interfacing system with humans through output of sensory information and input of commands. Several examples of such input and output devices are 3D space mouse, data gloves, 3D navigation devices, and haptic feedback.

The major goal of computer graphics and virtual environment simulation is to provide *realistic* methods to allow an easy creation of digital equivalents for natural phenomena such as fluid dynamic behavior and its interactions. The common problem in a fluid simulation and animation, on the other hand, is how to solve the underlying laws that govern fluids motions. The fluid dynamic simulation is known to be difficult to solve in an efficient way, sometimes even too difficult to be solved at all.

Furthermore, a real time rendering in immersive virtual environment with haptic interaction demand high frequency processing (1000 Hz) and the visual aspects need frame rates in the order of 30 fps.

This study introduces novel framework utilizing the immersive virtual environment for fluid interaction in cutting procedures and haptic rendering. The fluid dynamic computations rely on the Navier-Stokes equation in three-dimension (3D) is solved using cubic interpolated propagation based on GPU programming. The framework is expected to provide

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

efficient computation which is applicable for haptic rendering.

2. PREVIOUS WORK

Several methods have been developed to simulate fluids and its interaction to objects based on the Navier-Stokes equations to exhibit physically realistic fluid behaviours [Carl04, Che95]. In consideration to computation cost some studies attempted to solve the NS equation with simplifications [Fos01, Sta99].

The cubic interpolated propagation (CIP) methods was firstly introduced in [Yab91c] as a universal solver for hyperbolic equation by cubic interpolation. It is then used for solving fluid dynamic equation as a conservative semi-Lagrangian solver for solid, liquid and gas [Yab91b, Yab02]. Some studies prove that this method is efficient enough to solve dynamic fluid equation in all states with motions [Yab04, Yab02] .

Although the CIP has been accepted as an alternative method for fluid solver that fits the need of computer animations, there has been no study that uses this method in immersed virtual environments with a haptic interaction. More advance studies [Liu04, Li03] attempted to conduct fluid dynamic computation on the GPU to get parallelisms for more efficiency computations. The GPU applications for study fluid erosions [Anh07, Ben06, Mei07, Nei05] also become a current research interest in computer graphics and animation. These study show promising results to increase the interactivity visual cue; however, most of them are not involving kinesthetic interactions.

This paper introduce a framework which combines CIP and parallelism in GPU, trying to achieve most efficient solution of fluid dynamic equations for real time rendering in virtual environment with haptic interaction.

The rest of the paper is organized as follows: CIP as the fluid dynamic solvers is described in Section 3; computational fluid dynamics implemented in GPU is described in Section 4; Section 5 described the experimental set-ups and the results; with Section 6 concludes the paper.

3. FLUID DYNAMIC SOLVER

3.1. Cubic interpolated propagation

The Navier-Stokes equation (NSe) for incompressible flow can be described as a *momentum*

$$\frac{\partial u}{\partial t} = -(u \bullet \nabla)u - \frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2 u + F_c$$

and *mass conservation* function

$$\nabla \bullet u = 0 \qquad (2)$$

where u is a velocity field of fluid, p is a pressure, ρ is the density of the fluid, μ is a viscosity coefficient, F_c represents external force (gravity, haptic motion), and ∇ is the differential operator, respectively.

The behaviors of a fluid can be predicted by solving equations (1) and (2). Then, a flow dynamics simulation can be done using staggered grids that define velocity components at cell faces and scalar variables in cell centers. The fraction of volume of fluid in each cell is transported by advection equation as formulated below.

$$\frac{\partial f}{\partial t} = -(u \bullet \nabla) f \qquad (3)$$

where f is the function of volume of fluid fraction (VOF). By using the CIP method [Tak03, Yab02, Yok02] and the advection form ($-(u \bullet \nabla)u$), the equation (3) can be solved.

The interface between liquid and solid is traced by distributing Eqn. (4) into advection phase shown in Eqn. (5) and non-advection in Eqn. (6) as follows:

$$\frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} = G \qquad (4)$$

$$\frac{\partial f}{\partial t} = G \qquad (5)$$

$$\frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} = 0 \qquad (6)$$

The CIP scheme calculates the advection phase by shifting a cubic interpolated profile into space according to the total derivative equations. Since the interpolation profile uses the cubic polynomial and the spatial derivative value at each grid point, it can be derived that:

$$\frac{\partial f'}{\partial t} + u\frac{\partial f'}{\partial x} = G' - f'\frac{\partial u}{\partial x} \qquad (7)$$

$$\frac{\partial f'}{\partial t} = G' - f'\frac{\partial u}{\partial x} \qquad (8)$$

$$\frac{\partial f'}{\partial t} + u\frac{\partial f'}{\partial x} = 0 \qquad (9)$$

The exact solution of Equation (6) is (1)

$$f(x,t) = f(x - ut, 0) \qquad (10)$$

If the velocity is assumed to have a constant value within short time, we get

$$f(x,t+\Delta t) \cong f(x-u\Delta t,t) \qquad (11)$$

Since x-$u\Delta t$ is not always located on grid points, this point is interpolated locally by Hermit polynomials within the calculation grid point as the value for the next time step. Within the discretised function a cubic-Hermite polynomial can be expressed as follow:

$$F_j(x) = [a_j X - b_j)X + \Delta x f_j^{'}]X + f_j \qquad (12)$$

where

$$a_j = \Delta x(f_j^{'} + f_{j+1}^{'}) - 2(f_{j+1} - f_j)$$

$$b_j = \Delta x(f_j^{'} + 2f_{j+1}^{'}) - 3(f_{j+1} - f_j)$$

and

$$X = \frac{x - x_j}{\Delta x}$$

$$\Delta x = x_j - x_{j-1}$$

Furthermore, Equation (1) can also be used to approximate the other fluid states such as a fluid pressure. The pressure equation is solved using the Poisson's equation with assumption that the liquid has no viscosity and has the external force that is determined by the force released from/to the haptic interaction. After the VOF of each cell is obtained in the simulation, a fluid surface is created and it is smoothened by subdivision, if necessary.

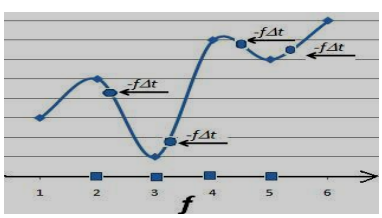


Figure 1. Estimation of the physical value on the departure point of the arbitrary contour in advection phase.

The CIP scheme uses a spatial profile that rely on different interpolations to propagate the solution along the characteristics [Uts91a]. The physical value on f_t is calculated by a Hermite interpolation constructed locally within the calculation cell. Then, an estimate value is directly advected toward the grid point calculation at the next time step value (figure 1).

By employing a cubic-Hermite polynomial within the discretised function (a grid cell) and its approximation in continuous manner of an explicit finite-difference form, and by assuming that both a physical quantity (f) and its spatial derivative (f') satisfy the master advection equations, the CIP method provides a stable result, less diffusive, and has third order accuracy [Uts97, Uts04, Yok02]. Furthermore, the numerical solvers do not need the solution for each interface, because the solver yields the solutions at once.

When using the CIP method as fluid solver, the highly accurate result in coarse grid can be obtained; however, the volume of fluid is not properly conserved; which may be caused by isosurface extraction during rendering and it is tolerable for a short simulation period.

3.2. Fluid interaction and simulation

Although CIP can combine the solution of fluid dynamics and the fluid-solid interaction together in one algorithm, the dynamic interaction between fluid and solid, and between a haptic and fluid, need to be studied in order to achieve realistic visual result. Such interactions can be described as collision between two objects. For haptic rendering, a collision detection and the impulse response are employed using the method introduced in [Gre00].

The interface between fluid and solid are simulated with a volume of solid (VOS) value. The VOS defines the fraction of a solid contained in a fluid cell and approximate the shape of rigid body in the fluid solver. The cells with VOS values less than or equal to 0.5 are considered as fluid otherwise a solid (figure 2).

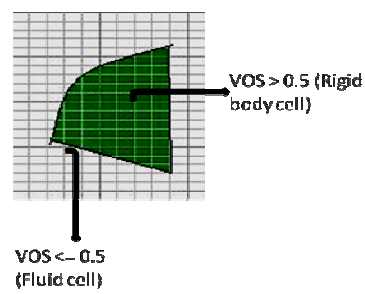


Figure 2. Approximation of fluid and solid cells.

The dynamics of rigid bodies are approximated based on force Fc acting on the centre of mass and the torque Tc from the centre of mass as follow:

$$F_C = Mg + \iint_s (p.n)ndS \qquad (13)$$

$$T_C = \iint_s (r - r_C)(p.n)dS \qquad (14)$$

where g is the gravity, M is mass of the object, p is the pressure of a boundary between a solid cell and a fluid cell, n is the normal vector to the boundary, r is

a point on the boundary, r_c is the object's centre of gravity, and s is the boundary between the fluid and the solid objects.

A spilling fluid from the cut is assumed as a droplet liquid that travel over the mesh. When the dropped fluid flows on a surface, some amount of fluid remains behind because of surface absorption and wetting. This remaining fluid will merges into the other fluid that spill later on.

3.3. Simulation and computation of the fluid dynamic on a GPU

Although, the real-time fluid computation system involving in this simulation can be categorised into the process that is done in a CPU (central processing unit) and in a GPU (graphic processor unit), the fluid simulation presented in the flowchart do not clearly show these divisions (figure 3). Beside controls the hardware (peripherals, haptic and 3D space navigator), the CPU computes the collision detections, detects or/and delivers the forces from/to a user, and maintains the haptic rendering rates. The GPU, on the other hand, responsible to render the image into a display and to compute a fluid dynamic numerical solver.

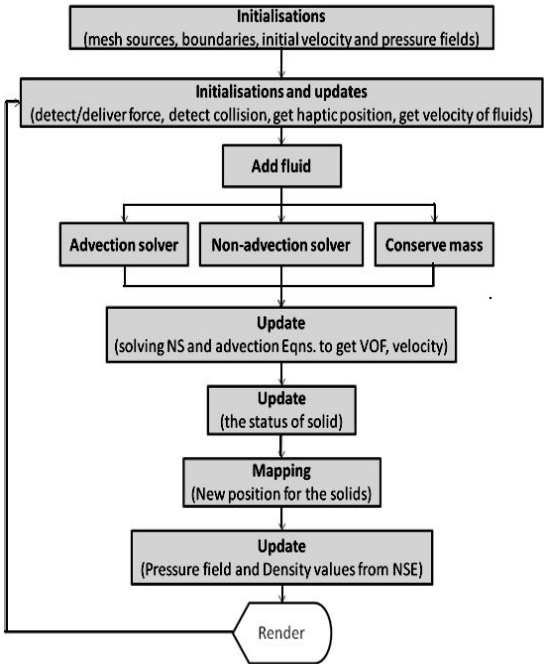


Figure 3. Flowchart of the real-time fluid simulation.

The fluid solver for the simulation that is developed here, is adapted from [Har04] with novelties on the numerical methods and the use of parallelism on the GPU for fluid dynamic computations. As explained above the CIP numerical method to solve the NSe is used. The solver is mainly divided into four fragment programs in the GPU: an advection, a non-advection, divergences and gradients, and boundary conditions.

Besides the fluid solver computations, the GPU is employed as a tree lookup generator in the simulation. The simulation is initialized by creating an octree around the mesh surface in the GPU. The algorithm for the texture mesh is developed based on the octree texture from the fragment program using the tree lookup as suggested in [Lev05]. Then, the density values are updated using a render-to-texture operation during the simulation and stored in 2D texture map. The octree's leafs store the index of a pixel in the form of the three 8-bit values (in RGB Channels). The simulation will also access the density of the neighbors to recode the neighboring information.

All the fragment programs are written in Cg [Fer03, Wil03]. The fragment codes for octree has the call *tex2D(Dmap, I)*, where *I* is the index stored within a leaf *L* of the octree and *Dmap* is the density map. This call returns the density value associated with leaf *L*. The call *tex2D(N ,I)* yields the index within the density map of a 3D space neighbor. To get all neighbor information, we require 26 textures in 3D or 9 textures in 2D. When all textures have been created, the density map can be render on the texture mesh and the location of each density can be retrieve from the octree. The texture density values are updated based on the fluid solver computation, the current VOF, the velocity, and the solid status respectively.

3.4. Force feedback design

Even though force feedback design is not a main discussion in this paper, this section describes briefly our force feedback design to show how to manage the haptic device. This section will also not explain in detail of each force involved in the discussion. This project employ two different threads: the first one is for visual rendering updated in 30Hz and the other for haptic rendering in the rate of about 1000Hz.

A force computation generating feedback into a haptic usually is called as a *force model*. This force has a means of mapping from the haptic device position to a force vector[Ano05]. The force model also deliver constrain or reaction forces that are sensed by the user through a device. Although a force model is updated in the scene-graph loop at about 60 Hz, the current force model will be used extensively by the real-time loop at about 1000 Hz by repeated calls to its evaluate function.

In this paper the goals of force models design are: to fell a skin surface, to create cutting path, and to

prompt bleeding. The formulation of forces can be expressed as is shown in equation (15) as follows:

$$F_t = kF_p - \sum F_{xy} + F_d \qquad (15)$$

where F_t is total forces felled by a user, F_p is pressure and displacement forces released from the user, k is a blade sharpness constant, F_{xy} is force reactions from surface returned through haptic device, and F_d is a damper force added to reduce haptic jerkiness respectively.

4. EXPERIMENTAL SETUP AND IMPLEMENTATION

The proposed framework is implemented on a personal computer (PIV Dual Core 3.2 GHz, 2 GB RAM, 80GB HD), the PHANTOM Haptic premium 1.5 (from Sensable) in Reachin display with stereo LCD shutter glasses (Figure 4). The visual and haptic rendering is controlled using Reachin API 4.2 in combination with Visual Studio.net and Python.

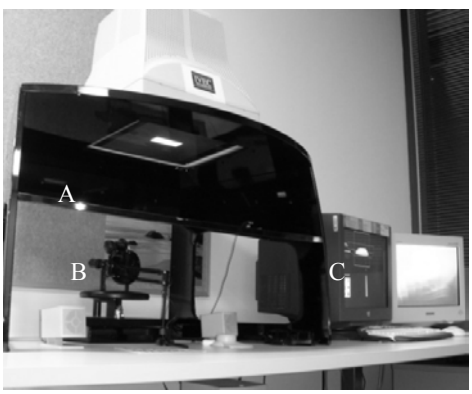
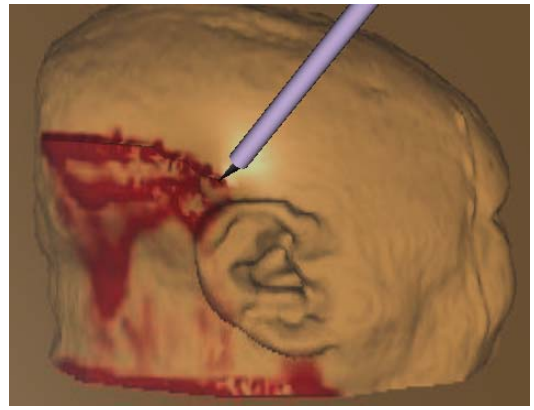


Figure 4. Haptic workbench with Reachin display (A), haptic (B), and work station (C).

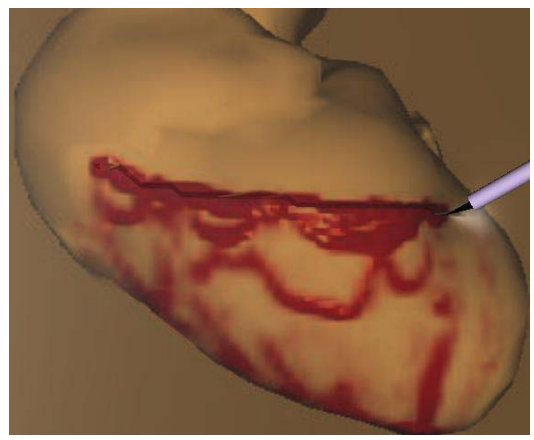
In this study we employ mesh cutting techniques by combining adaptive re-meshing and mesh sub-division for creating surface cutting paths. The cutting paths are created when the following conditions are met: the scalpel collides with a surface, the stylus button is pressed, and the force is applied greater than the surface stiffness.

The force here represents pressure force and the sharpness constant of the blade. The cutting process will be followed by flowing of the fluid. The fluid, representing blood, flows over the surface toward lower parts on the surface or toward the centre of the gravity force.

Several examples of snapshot simulations are shown in figure 4. These examples describe the cutting simulation followed by blood flowing over the surface in the cutting path in a head (figure 5a) and in a heart (figure 5b).



a) A cutting simulation on the head's skin



b) Heart cutting simulation

Figure 5. Interactive cutting over the mesh surface with liquid flowing along the surface paths.

This paper examines several different mesh sizes versus the observed frame rate per second (fps) in the simulation as presented in table (1). As can be seen from the table, the visual/graphic (G) frame rate remain stable for various mesh sizes either with fluid dynamic inclusion (f) or not; however, when a mesh cutting through haptic in combination with fluid dynamic interaction (cf) is included in the simulation the frame rate fall into 18fps for the triangles greater than 29000. We can see that the cutting process reduce the interaction speed. This may be influenced by the re-meshing processes and mesh deformations that need more computation on the CPU. It is also proved from the decreasing frame rate (fr in Hz) on haptic rendering.

The fluid simulation presented here will not go through the detail of bleeding released from the vein or artery when they are cut, since this is beyond the scope of this project. At this time, we only concentrate to control the fluid movement released from the skin when they are cut, regardless the existing vein or artery.

Mesh	Triangles	G_{fps}	$G_{fps(f)}$	$G_{fps(cf)}$	H_{fr}
Full Head	37096	58	58	18	980
Chevy	29304	58	58	18	989
Half Head	23587	58	58	28	989
Bunny	2915	59	59	28	1000
Heart	1619	59	59	28	1000
Torus	576	59	59	28	1000

Table 1. Mesh sizes versus frame rate pers second (fps)

Furthermore, although this paper does not present quantitative data to evaluate the effect of the haptic in user immersion, we verify that the user can communicate kinesthetically in 3D space using this device. Not as in ordinary computer animation that actually only has two and half dimension visual presentation, using haptic device in combination with 3D stereo shutter glasses can also improve user immersions, where the movement can be generated in six degree of freedom with sense of touch and force feedbacks.

5. CONCLUTION AND FUTURE WORK

The computation and simulation of fluid for haptic rendering and cutting procedure can be done at reasonable efficiency using CIP fluid solver that combines the solution of fluid dynamics and its interaction with solid object together. By that using CIP fluid solver and GPU programming, the computation for solving Navier-Stokes equation can be done efficiently without involving supercomputer.

Our results confirm that by combining the CIP solver and GPU programming, the haptic interaction for skin cutting followed by releasing blood from the paths and blood flowing over the anatomical surface can be simulated in a real-time interaction with visually realistic display.

More detail results such as involving two or more fluid and their interactions will be gathered in the next study. This may investigate the fluid dynamic interaction during surgical simulation. The cutting and bleeding involving blood flow from the vein or artery may also be studied in the future.

6. ACKNOWLEDGMENT

The authors wish to thank IVEC (Central TAFE) that provide the haptic equipment, Reachin display, and haptic workbench. The Medical Imaging Department of Curtin University for providing other facilities, and the Exploration Geophysics department that provide spaces for work and equipments. This work is part of the PhD project research in the department of imaging and applied science Curtin University of Technology.

7. REFERENCES

[Anh07],N. H. Anh, A. Sourin and P. Aswani, "Physically based hydraulic erosion simulation on graphics processing unit," in Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia Perth, Australia: ACM New York, NY, USA, 2007.

[Ano01],Anonimous, "Virtual Simulation of Temporal Bone Dissection," in http://www.osc.edu/Biomed/temporal.html (1 of 2) [4/5/2001 8:33:54 AM], 2001.

[Ano05],Anonimous, ReachinAPI 4 Programmer's Guide, Revision No: 27 ed.: Reachin Technology AB, 2005.

[Ben06], B. Benes, V. Tensinsky, J. Hornys and S.K.Bhatia, "Hydraulic erosion," Computer Animation and Virtual Worlds, vol. 17, pp. 99-108, 2006.

[Carl04],M. T. Carlson, "Rigid, Melting, and Flowing Fluid," in College of Computing: Georgia Institute of Technology, 2004.

[Che95], J. X. Chen and N. d. V. Lobo, "Toward Interactive-Rate Simulation of Fluids with Moving Obstacles Using Navier-Stokes Equations," Graphical Models and Image Processing, vol. 57, pp. 107-116, March 1995.

[Fer03],R. Fernando and M. J. Kilgard, The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics: Addison-Wesley Longman Publishing Co., Inc., 2003.

[Fos01], N. Foster and R. Fedkiw, "Practical animation of liquids," Proc. of SIGGRAPH 2001, ACM Press/ ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, pp. 23-25, 2001.

[Gre00], A. Gregory, M. C. Lin, S. Gottschalk and R. Taylor, "Fast and accurate collision detection for haptic interaction using a three degree-of-freedom force-feedback device," Computational Geometry, vol. 15, pp. 69-89, February 2000.

[Har04],M. J. Harris, "Fast Fluid Dynamics Simulation on the GPU," in GPU Gems - Triks for Real-Time Graphics, R. Fernando, Ed.: Addison Wesley, 2004, pp. 637-665.

[Hig02], A. Higgins and S. Koucky, "Mission impossible?," Machine Design, vol. 74, p. 28, Dec 12, 2002.

[Lev05],S. Lefebvre, S. Hornus and N. Fabrice, "Octree Textures on the GPU," in GPU Gems 2 - Programming Techniques for High-Performance Graphics and General-Purpose Computation, M. Pharr, Ed.: Addison Wesley, 2005, pp. 595-613.

[Liu04],Y. Liu, X. Liu and E. Wu, "Real-time 3D Fluid Simulation on GPU with Complex

Obstacles," in Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on, 2004, pp. 247-256.
[Mei07],X. Mei, P. Decaudin and B.-G. Hu, "Fast Hydraulic Erosion Simulation and Visualization on GPU," in Computer Graphics and Applications, 2007. PG '07. 15th Pacific Conference on, 2007, pp. 47-56.
[Nei05],B. Neidhold, M. Wacker and O. Deussen, "Interactive physically based fluid and erosion simulation," in Proceedings of Eurographics Workshop on Natural Phenomena, 2005, pp. 25-32.
[Ros00], L. Rosenblum and M. Macedonia, "Virtual Orthopedic Surgery Training," IEEE Computer Graphics and Applications, pp. 6-9, 2000.
[Sta99], J. Stam, "Stable Fluids," Proc. of the 26th annual conference on computer graphics and interactive techniques, pp. 121-128, 1999.
[Sto00], R. J. Stone, "Haptic feedback: A potted hystory, from telepresence to virtual reality," Lecturer notes in computer science, vol. 2058/2001, pp. 1-7, 2000.
[Tak03], T. Takahashi, H. Fujii, A. Kunimatsu, K. Hiwada, T. Saito, K. Tanaka and H. Ueki, "Realistic Animation of Fluid with Splash and Foam," Computer Graphics Forum, vol. 22, pp. 391-400, September 2003.
[Uts97], T. Utsumi, T. Kunugi and T. Aoki, "Stability and accuracy of the Cubic Interpolated Propagation scheme," Computer Physics Communications, vol. 101, pp. 9-20, April 1997.
[Uts04], T. Utsumi, T. Yabe, J. Koga, T. Aoki and M. Sekine, "Accurate basis set by the CIP method for the solutions of the Schrodinger equation," Computer Physics Communications, vol. 157, pp. 121-138, February 2004.
[Li03], W.Li, Z.Fan, X.Wei and A.Kaufman, "GPU-Based flow simulation with complex boundaries," Technical Report 031105, Computer Science Department, SUNY at Stony Brook, 2003.
[Was01], T. M. Wasfy and A. K. Noor, "Visualization of CFD results in immersive virtual environments," Advances in Engineering Software, vol. 32, pp. 717-730, September 2001.
[Wil03],R. William, R. Mark, S. Glanville, K. Akeley and M. J. Kilgard, "Cg: A system for programming graphics hardware in a C-like language," in SIGGRAPH, 2003.
[Uts91a], T. Yabe and T. Aoki, "A universal solver for hyperbolic equations by cubic-polynomial interpolation I. One-dimensional solver," Computer Physics Communications, vol. 66, pp. 219-232, 1991.
[Yab91c], T. Yabe, T. Aoki, G. Sakaguchi, P.-Y. Wang and T. Ishikawa, "The compact CIP (cubic-interpolated pseudo-particle) method as a general hyperbolic solver," Computers & Fluids, vol. 19, pp. 421-431, 1991.
[Yab91b], T. Yabe, T. Ishikawa, P. Y. Wang, T. Aoki, Y. Kadota and F. Ikeda, "A universal solver for hyperbolic equations by cubic-polynomial interpolation II. Two- and three-dimensional solvers," Computer Physics Communications, vol. 66, pp. 233-242, 1991.
[Yab04], T. Yabe, H. Mizoe, K. Takizawa, H. Moriki, H.-N. Im and Y. Ogata, "Higher-order schemes with CIP method and adaptive Soroban grid towards mesh-free scheme," Journal of Computational Physics, vol. 194, pp. 57-77, February 2004.
[Yab02], T. Yabe, Y. Ogata, K. Takizawa, T. Kawai, A. Segawa and K. Sakurai, "The next generation CIP as a conservative semi-Lagrangian solver for solid, liquid and gas," Journal of Computational and Applied Mathematics, vol. 149, pp. 267-277, December 2002.
[Yok02], K. Yokoi and F. Xiao, "Mechanism of structure formation in circular hydraulic jumps: numerical studies of strongly deformed free-surface shallow flows," Physica D: Nonlinear Phenomena, vol. 161, pp. 202-219, January 2002

On the use of a palm computer for design review interaction in a virtual room

Pignatel,A.[1] , Farella,E.[2], Brevi,F.[1], Benini,L.[2], Gaiani,M.[1]
[1]INDACO - Dept. of Industrial Design, Arts, Comunication & Fashion, Politecnico di Milano
Via G. Durando, 38/a - 20158 Milano – Italy
{alice.pignatel | fausto.brevi | marco.gaiani}@polimi.it
[2]DEIS - Dept. of Electronics, Computer Science and Systems, Università di Bologna
Viale Risorgimento, 2 - 40136 Bologna – Italy
{elisabetta.farella | luca.benini}@unibo.it

Abstract

Increasingly often the use of photorealistic models for design review processes involves the setup of special rooms, fully dedicated to visualize digital models onto large display, the so called CADwall. In this scenario an effective interaction with virtual prototypes is a basic need, but it is almost always based on traditional input devices, such as mouse and keyboard. The design review process is usually led by a chief designer, but the use of traditional interaction tools requires an additional operator. This operator is guided by the chief designer through vocal inputs in a slow, misleading and inefficient way.

In this paper we propose a solution based on a handheld device and an inertial sensor node, with a graphic interface that allows a direct interaction with 3-D digital models inside virtual rooms with large displays.

Keywords

CADwall; virtual room; interaction; handheld device; inertial sensor node; large display; design review

1. INTRODUCTION

3-D digital models are used in every step of the industrial design product development process. Digital models are used as virtual prototypes for anticipating and evaluating product features, as well as aesthetic appearance. For this reason, in the last few years, in the industrial context, several "virtual theatres" have been built for design review activities. These theatres are wide rooms equipped with projection-based large display, additional devices for stereoscopic visualization of digital models and several seats, so that many people can attend screening at the same time. Projectors are connected to one or more workstations, according to the requested rendering quality, equipped with software dedicated to photorealistic visualization of 3-D digital models. Usually, workstation controls are placed far away from the screen, behind audience seats.

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

Copyright UNION Agency – Science Press, Plzen, Czech Republic.

For this reason the work session needs an additional qualified technician to perform interaction tasks with the digital scene as the review activities leader may ask for, from his/her location near the display device. A visualization software is designed, usually, to be controlled by "traditional" input devices, namely mouse and keyboard, while advanced interaction devices, such as 3-D mice or data gloves used in virtual reality environments, are not yet commonly supported. The main reason is that mouse and keyboard represent the standard computer interface in every working area and users are already trained in their use.

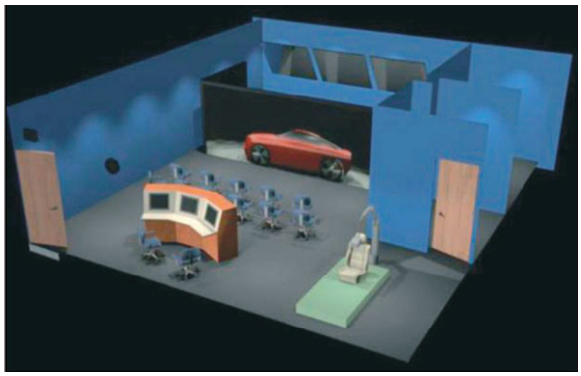


Figure 1. A typical virtual room configuration

Moreover, the software is not used for virtual environments visualization, which requires high

immersion feeling and accurate manipulation devices. On the contrary it is used during design reviews in industrial environment, where these features are less important.
There is another fact that hinders the introduction of virtual environment interaction techniques: the design review activity itself. Design reviews involve a number of people who need to walk freely in front of the screen to interact with the digital model; at the same time, they need not to be isolated from the context and from other people because they have to exchange opinions and advices, being certain they are sharing the same vision of the scene. It is easy to observe that introducing a continuous tracking systems in this environment is very hard: tracking just one person would mean that everyone has to observe the scene from a point of view that moves according to someone else's displacement and position; on the other side, tracking every single person in the room would be possible only with individual display devices, such as HMD, that would isolate people involved in the design review. Mixed-reality techniques are neither very effective in this context, because they need cumbersome devices, such as see-through displays or "head-mounted" cameras, that would trouble the review activity or, at least, would move the attention from the real work focus. To these technical problems we must add that most of the chief designers are reticent in adopting systems too "technology-oriented" because they think these devices are exceptionally complex and difficult to use.
These considerations are true especially when the design review is focused on aesthetic evaluation of the product shape. A survey of the main Italian industrial design companies with a private virtual room, carried out at Politecnico of Milan, showed that design reviews can be very different depending on the product process step and that mainly two different types of design reviews can be observed. There is a first type of design review focused mainly on technical aspects, which main goal is to define product features: it involves, as general rule, only the project team so that all the users share almost the same know-how.
The second type of review is quite different; it resembles mostly to a product presentation and aims to present the product outside the project team. People attending the presentation can be members of the same company of the project team as well as external buyers, but this means, in any case, that users' skill and know-how are very different. These reviews are mainly focused on aesthetic and functional aspects of products and are led from chief designer, the person who coordinated every step of product design process. The survey showed also that, during this second type of review, the need of focusing on different model details involves a continuous dialogue between the chief designer near the screen and the hidden operator far away from him; thus troubles the review activity and makes it less effective.

2. RELATED WORKS

Interest in finding new paradigms for virtual environments interaction has highly increased in the last few years. Specifically, two topics are catching researches' attention: multi-user interaction and multi-modal interaction.
Bierz [Bie07] offers an interesting overview of the current state of interaction technologies and metaphors for large displays.
Bourdot and Touraine [BT02] present a polyvalent display approach, which allows the system to determine the desired focus and the moving intention of the user for virtual navigations.
Kim and Fellner [KF04] as well as Corradini [Cor02] propose gesture recognition based systems, while Ciger et al. [CGVT03] suggest a combination of gesture and speech interaction.
A topic highly connected with multi user interaction is wireless communication systems applicability, see Green et al. [GSVS05], as well as handled or portable device use, see Kukimoto [Kuk05].

3. PROJECT GUIDELINES

The described project aims to build a device that can be used by chief designers for direct interaction with 3D models during design reviews in virtual rooms with large display.
The target user is represented by a chief designer, the project leader who guides the project team and coordinates all the product design stages. His knowledge of industrial product design processes is very good, but usually he is not trained in 3-D digital models creation and manipulation technologies.
The main goal is to allow the chief designer to interact with the virtual scene and to complete some basic tasks without the need of interacting with the operator.
For this reason, the device must have a very user-friendly interface, to allow every user to master it in a very short time and without special training. In addition, the device has to be small and lightweight, to be handled easily and not to disturb interaction with other people during review activity, and it must have a wireless connection so that the user can walk freely in the room.
The proposed system is composed by a pocket PC with a graphic interface that transmits information to the workstation and an accelerometer fixed to the pocket PC measures its position and sends it back to the virtual room workstation.
Project components have been developed at the same

time by Micrel Lab at DEIS Department at University of Bologna and at Virtual Prototyping Lab at Politecnico of Milan.
The DEIS research team designed the software for data transmission and exchange, and tested the connections between wireless devices and workstation, while Virtual Prototyping Lab team defined the general features of the system and developed the graphic user interface implemented onto the pocket PC.

4. GENERAL SYSTEM DESCRIPTION

The general architecture (Figure 2) can be divided in three main components: (i) the input system, which is composed by the combination of a handheld device and a wireless accelerometer node and the software running on both, (ii) the middleware, (iii) the visualization software both for workstation and CADwall.

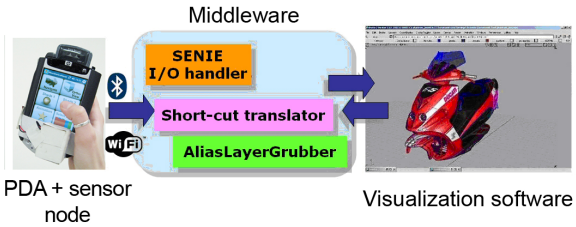


Figure 2. System components

The input system enables the use of graphical short cuts, displayed on the handheld device, namely a PDA, and of tilt based gestures to control the visualization environments. Data streams from wireless devices are received from the middleware, which acts as an intermediate software layer in charge of acquiring and converting data from the handheld device and accelerometer in controls for the visualization software.
The middleware handles the input device, transparently from the visualization software, by mapping input controls in mouse and keyboard controls. This solution is very effective, because it is platform independent and can be implemented on different systems by setting middleware preferences. Moreover, the user can ignore how the specific visualization software works: he/she just needs to select the task to perform on the digital scene.

5. INPUT SYSTEM

Input Devices

Pocket PC has been chosen mainly because of its capability of exchanging information with the workstation, thanks to integrated data transmission protocols and high operating system compatibility. Moreover, it has a wide touch screen, it is a small device, lightweight, easy to use, available at a low price and fairly popular, so that it may result a familiar device.
The model used in this project is the Hewlett-Packard *iPaq h5550* with Microsoft Windows CE 2003 SE, equipped with WiFi card, for connection to wireless LAN networks, and Bluetooth interface. It has a VGA resolution display, storage and processing resources. Moreover, the operative system supports *Microsoft.NET Compact Framework 2.0,* which supports the libraries we developed.
The sensor node used for extending pocket PC input is called *WiMoCa Sensing Node* and has been developed by Micrel Lab at DEIS Department, as a building block of a wireless sensor network, mainly used for gesture recognition and body movement tracking [FPBABR05].

Figure 3. PDA with sensor node

The WiMoCa node is extremely flexible, thanks to its modular architecture to ease fast replacement and update of each functional layer. The main layers of the node are the power supply layer, the microcontroller and sensor layer and the wireless transmission layer. This node has very small size and weight, it is wearable, it has low cost and very limited power consumption.

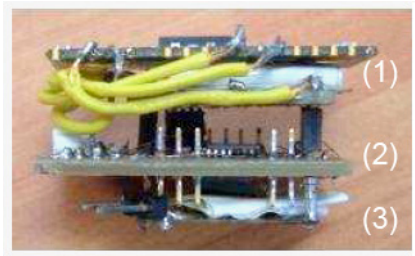


Figure 4. WiMoCa Sensing Node structure: (1) *Communication* – Bluetooth transceiver; (2) *MSC/Sensors* – ATMega8 & Triaxial Accelerometer; (3) *Power Supply*

The main hardware components are ATmega8 AVR RISC architecture, an 8 bit microcontroller with low-cost and low-power features. It has a tri-axial digital MEMS accelerometer, used in this work as inclinometer, thanks to its ability to capture the

gravity acceleration. The communication layer is based on a Bluetooth 2.0 transceiver that operates at 2.4 GHz and supports Serial Port Profile (SPP).

Graphic Interface and Software

The graphic interface implemented onto the pocket PC has been designed specifically for this project by researchers at Politecnico of Milan.

The interface contains selected features that could be useful during design review activities. Features choice has been managed by the already mentioned survey of virtual rooms in industrial environment. Chief designers have suggested features of common use during design reviews, pointing out those tasks that create greater problems in obtaining the right interpretation by the operator at the workstation. These features include mainly digital camera controls for navigation tasks, visualization controls and selection commands; they have been collected in homogeneous groups, to give a better organization to the interface. Keys are designed so that they can be selected by touching the screen with fingers, not only with the PDA stylus, but this means that the display size does not permit to place all the keys in just one window. Considering both screen size and average finger size, the maximum number of keys that can be displayed at one time is of about six or eight. Thus, the interface presents to the user a main page with keys recalling submenu pages. Key position on different pages remains the same and it is planned to easily remembering where a specific key is on the display, speeding navigation inside the graphic interface. A "BACK" key on the bottom of every submenu page brings back the main window.

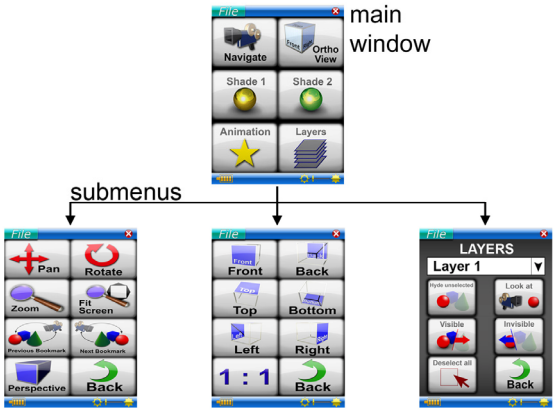


Figure 5. Submenus organization of graphic interface

Icons have been designed referring to a popular graphic symbolic system diffused in almost every software with graphic interface and their look is designed according to basic usability rules; features related to keys are immediately recognizable thanks to both graphic and written feedback. Colours used are bright, with high contrast between background and icons, so that the keys can be seen and recognized even in the dark of the theatre and wearing shutter glasses for active stereoscopic vision, that absorb a high light percentage. Anyway, display brightness can be easily adjusted to fit lighting conditions and user's need.

The sensor node is fastened to the pocket PC, to have the best lever length, and it is enabled by selection on the PDA interface. Accelerometer detects pocket PC pitch when it is rotated by user. Translations on horizontal plane have not been taken into account because they are strongly affected by scale factor due to the human component, which is not predictable and cannot be modified in an interactive way.

On the contrary, wrist rotation in left-right and up-down directions is a natural movement that can be performed without specific training, therefore the system has been designed taking advantage from these movements. Horizontal mouse pointer shifting has been connected to left-right wrist rotation, while vertical shifting has been connected to up-down wrist rotation. Thus, user can see the mouse pointer moving on the large screen according to a simple movement of his hand.

The graphic interface is handled by a C# application running on the palmtop computer, responsible for capturing the event of touching a key and running the correspondent thread for dispatching the control request to the visualization software. Therefore, the application on the palmtop computer (i) captures events generated by the graphic interface; (ii) translate them into a code corresponding to a control command; (iii) access to the TCP/IP channel to send via socket the request to the middleware running on the visualization platform.

6. MIDDLEWARE AND VISUALIZATION SOFTWARE

The middleware running on the workstation is mainly used for three aims. First, we exploited SENIE libraries to handle the I/O streaming of data both coming from the pocket PC and the inertial node. SENIE is a Java-based set of libraries and a visual environment to handle use of heterogeneous input/output platforms and devices and their use for various applications. It is also an environment to easily handle debugging and testing of new applications. In this work we exploited mainly SENIE I/O interfacing features, using the libraries specifically dedicated to handle the COM port and a socket communication. In the first case SENIE connects automatically to the Bluetooth device and extracts the payload from the data received, that are the accelerometer values. The socket connection is used to connect to the pocket PC, to extract controls coming from the graphic interface.

 ISBN 978-86943-15-2

SENIE enables the implementation of new plug-ins to handle the interface with external application.
For the visualization of the 3D models we used AliasStudio v.13.0.2 by Autodesk, state of the art software for industrial design. AliasStudio offers an advanced support for real-time rendering of 3D models, therefore enabling its use during presentation in virtual theaters. We chose AliasStudio for its wide use in companies with internal design reviews virtual room and for the support this tool gives to develop additional features to its work environment. Indeed, the license of AliasStudio includes the access to the programming API, that are the libraries that we used to implement our own application to interface the input devices with AliasStudio functionalities.

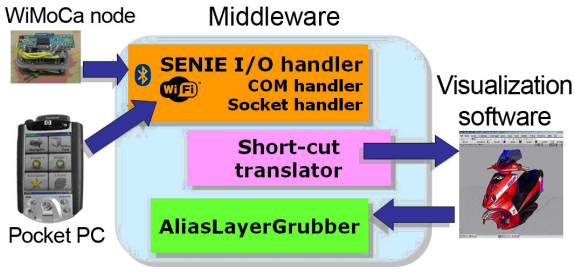


Figure 6. Connection between system components

Therefore, we designed and implemented two components: the short-cut translator and the *AliasLayerGrubber*. The first one is responsible of receiving the tilt derived from the accelerometer data and/or the controls coming from the pocket PC graphical interface and to translate them in short-cuts, that is combination of keys, understandable by the visualization software, as we will explain in the following paragraph. The *AliasLayerGrubber* is a plug-in that provides access to the 3D model layers. In particular, he reads a proprietary Alias file format (which extension is .WIRE) and translates information in an XML file. This plug-in, in practice, enables the selection of the different layers of the model by the pocket PC, providing access also to the features linked to each layers (e.g. it is this plug-in that enables the selection of the details of a model).

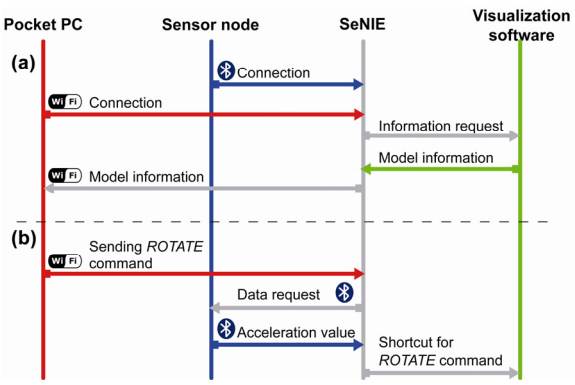


Figure 7. Data stream during (a) the connection procedure and (b) a command activation

We exploited, in particular, the powerful possibilities offered by the graphical interface of this tool. Because all the menu items can be mapped in keyboard short-cuts, by translating control commands coming from our input devices in short-cuts we provided access to all the features we displayed on the pocket PC.
Figure 6 shows the connection between the building blocks of the system. Figure 7 shows the direction of data flows among the pocket PC, the sensor node, the middleware (SENIE) and the visualization software.

7. USABILITY TEST

A first test of the system is now in progress at Virtual Prototyping Lab of Politecnico di Milano. The test is performed with a usability form filled in by groups of users after using the device in a design review simulation. Users have been parted in three different groups, according to their 3-D models skills, to have different feedback steps for a better understanding of the interface usability. A first group of users is composed by virtual room technicians, skilled in 3-D digital models creation, experienced in virtual scene interaction and with advanced product design process experience. A second group is made up of users with minor skills in product design process but experienced in 3-D models creation and interaction. The third group, on the contrary, is composed by target users that are highly skilled in industrial product processes but do not have experience in virtual scene interaction.

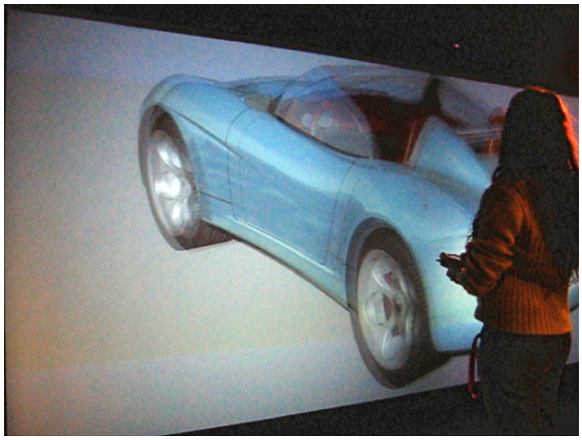


Figure 8. Test in the INDACO virtual room at Politecnico of Milan

The form aims to understand completeness of features implemented and usability of both the single feature and the whole interface. It is divided in two sections: the first part is focused on the graphic interface evaluation and includes questions about feature usability and interface look, while the second part is focused on usability of camera control commands.

Preliminary tests have been carried out with virtual room technicians, corresponding with the first group of users already described. The test provided a first hint about device usability, and it was performed in an advanced device development stage.
The interface implemented on the pocket PC gained a generally high appraisal both for completeness of features and usability (Figure 9).

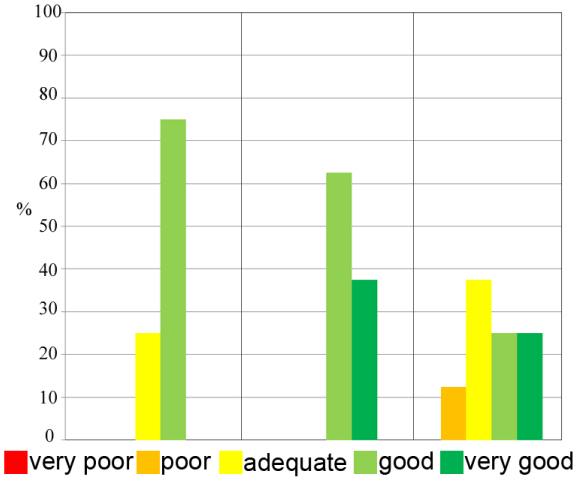


Figure 9. Some diagrams from usability test about: (i) interface look; ii) features identification; iii) interface usability

With regards to navigation tasks, on the contrary (Figure 10), users report some difficulties on using the device, mainly due to high feedback speed of sensor node, which causes the perception of having scarce control of the digital scene. To this, it must be added that appraisal is very different for every single function. Rotating tasks are generally considered easier to understand and perform than other navigation tasks, while zoom and pan commands involve a less natural movement and are harder to accomplish. By the way, the feedback speed is an easy tuning parameter.

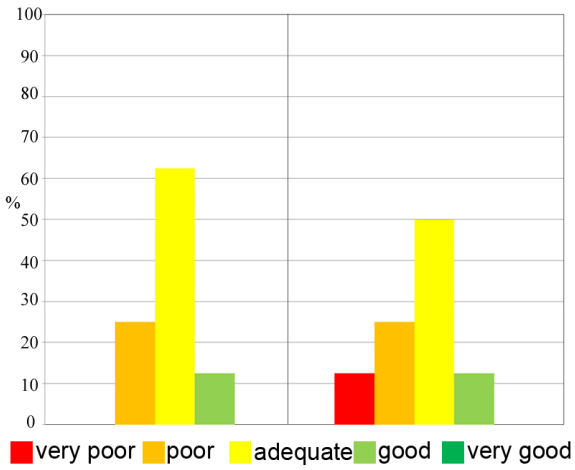


Figure 10. Some diagrams from usability test about: (i) navigation tasks; ii) self-training time

8. CONSIDERATIONS AND FUTURE WORKS

The system designed and implemented is made by state of art technology building blocks. The novelty can be identified in the integration of the building blocks in an effective and working prototype where a great attention has been focused on providing an *ad hoc* solution for the particular needs of the target scenario. In this direction most of the effort has been dedicated to study an effective graphical interface, easy to learn and of immediate use, in covering exhaustively the features used in typical presentations, and in providing mobility and involving interaction method by the use of a mobile device and tilt-based gestures.
During industrial virtual room survey, managers demonstrated high interest in having a direct input device for design reviews. Moreover, a first set of usability test shows that appraisal for the device interaction approach is very high, therefore we decided to improve the device and go on with a second development step.
Problems raised by users during device tests suggest to differentiate the sensor node set-up for navigation tasks, to have the best response for every command. Anyway, users' opinion suggests that the main focus for device improvement is to slow down sensor node feedback speed, to grant higher control of the digital scene.
Once set the second prototype, a new testing step will be performed, involving also the others two groups of less skilled users.

9. REFERENCES

[AR00] Assmann E., Ramezani H., *The use of virtuality in car development*, IEA 2000/HFES 2000: XIVth Triennial Congress of the International Ergonomics Association and 44th Annual Meeting of the Human Factors and Ergonomics Society, Ergonomics for the new Millenium; San Diego, 29 July-4 Aug. 2000, pp. 828-831.

[Ayl05] Aylett, *REMOTE: desk-top Virtual Reality for future command and control?,* Virtualreality n.8, 2005, Virtual Press, Hertfordshire, pp131-146,

[Bie07] Bierz T., *Interaction Technologies for large displays – An overview*, contribution in [KEM07], pp.77-162,

[Bry96] Bryson, S. *Virtual reality in scientific visualization*, Communications of the ACM, vol.39, no.5, 1996, pp.62-71 Association for Computing Machinery, New York

[Bux00] Buxton, W.; Fitzmaurice, G.; Balakrishnan, R. & Kurtenbach, G., *Large displays in automotive design*, Computer Graphics and Applications, IEEE vol.20, no.4, July-Aug.2000, pp.68–75, IEEE Computer Society, Los Alamitos, CA

[Bux01] Buxton, W. Less is More (More or Less), in P. Denning (Ed.), *The Invisible Future: The seamless integration of technology in everyday life,* 2001, New York: McGraw Hill, pp.145 – 179.

[BT02] Bourdot P., Touraine D., *Polyvalent display framework to control virtual navigations by 6DOF tracking*, Proceedings of the IEEE Virtual Reality, 2002, pp.277-278 IEEE Computer Society

[CGVT03] Ciger J., Gutierrez M., Vexo F., Thalmann D., *The magic wand*, 2003, in Proceedings of the 19th Spring Conference on Computer Graphics (Budmerice, Slovakia, April pp.24-26, 2003). L. Szirmay-Kalos, Ed. SCCG '03. ACM, New York, NY,

[Cho04] Choi, *A virtual prototyping system for rapid product development*, Computer aided design, vol.36, no.5, pp.401-402, 2004, Elsevier, Oxford

[Cor02] Corradini A., *Real-Time Gesture Recognition by Means of Hybrid Recognizers*, Gesture and Sign Language in Human-Computer Interaction: International Gesture Workshop, 2002, LNCS, vol.2298, pp.34-46, Springer

[CP03] Cheng K., Pulo K., *Direct interaction with large-scale display systems using infrared laser tracking devices*, 2003, In *Proceedings of the Asia-Pacific Symposium on information Visualisation - Volume 24* (Adelaide, Australia). T. Pattison and B. Thomas, Eds. ACM International Conference Proceeding Series, vol.142. Australian Computer Society, Darlinghurst, Australia, pp.67-74.

[Dub04] Dubois, E., Nedel, P., Freitas, S., and Jacon, L. 2005. *Beyond user experimentation: notational-based systematic evaluation of interaction techniques in virtual reality environments*. Virtual Reality vol.8, no.2 (Jun. 2005), pp.118-128

[FPBABR05] Farella E., Pieracci A., Brunelli D., Acquaviva A., Benini L., Riccó B., *Design and Implementation of WiMoCA Node for a Body Area Wireless Sensor Network* in Proceedings of the 2005 Systems Communications (Icw'05, Ichsn'05, Icmcs'05, Senet'05) – vol. 00 (August 14 - 17, 2005). ICW. IEEE Computer Society, Washington, DC, pp.342-347.

[Fai04] Fairen M. Brunet B. and Techmann T., *MiniVR: a portable virtual reality system*, Computers & Graphics, vol.28, no.2, April 2004, pp.289-296, Pergamon Press, New York

[Goe01] Goebel, M.; Hirose, M.; Rosenblum, L., *Today's VR*, Computer Graphics and Applications, IEEE , vol.21, no.6, pp.22-24, Nov/Dec 2001

[GSVS05] Green D., Stanton N., Walker G., Salmon P., *Using wireless technology to develop a virtual reality command and control centre*, Virtual Reality, vol.8, no.3 (Jun. 2005), pp.147-155

[KEM07] Kerren A., Ebert A., Meyer J., *Human-Centered Visualization Environments*. . Springer-Verlag, 2007.

[KF04] Kim H., Fellner D.W., *Interaction with hand gesture for a back-projection wall*, Computer Graphics International, 2004, IEEE, pp. 395-402

[KK05] Kim G., Kim G.J., *Designing Virtual Reality Systems: The Structured Approach*, 2005, Springer

[Kuk05] Kukimoto N., *Scientific visualization in collaborative virtual environment with PDA-based control and 3D annotation*, JSME international journal. Series B, Fluids and thermal engineering, Vol. 48 (2005) , No. 2 Special Issue on Advanced Fluid InformationJapan Society of Mechanical Engineers, Tokyo, Japan, pp.252-258

[May00] May J., *Perceptual Principles and Computer Graphics*, Comput.Graphics Forum, vol.19, no.4, Dec. 2000 , pp. 271-279(9)

[RCD97] Robertson, G., Czerwinski, M., and van Dantzich, M. 1997. *Immersion in Desktop Virtual Reality*. In Proceedings of the 10th Annual ACM Symposium on User interface Software and Technology (Banff, Alberta, Canada, October 14-17, 1997). UIST '97. ACM, New York, NY, pp.11-19

[SWM04] Sayers, M., Wilson, S., and McNeill, J. *Navigational tools for desktop virtual environment interfaces.* Virtual Reality Vol.7, no.3-4 (Jun. 2004), pp.131-139

[WL04] Wang Q.H., Li J.R., *A desktop VR prototype for industrial training applications*, Virtual Reality, vol.7, no.3-4 (Jun. 2004), pp. 187-197

[WTTC03] Wu S.T. Abrantes M., Tost D., Costa Batagelo H., *Picking and snapping for 3D input devices*, SIBGRAPI 2003, XVI Brazilian Symposium on Computer Graphics and Image Processing 2003, p. 140

From Bag of Categories to Tree of Object Recognition

Haijing Wang Tianwen Zhang
School of Computer Science and Technology
Harbin Institute of Technology, Harbin, China
haijing.wang@yahoo.com

Peihua Li
College of Computer Science and Technology
Heilongjiang University, Harbin, China
peihualj@hotmail.com

ABSTRACT

To recognize different category of objects, multiclass categorization problem is often reduced to multiple binary problems. Traditional approaches require training different classifiers for each category. This can be slow and the performance of learned single classifier is poor for limited training samples. We present a multiclass object recognition tree, in which the leaf node and the non-leaf node correspond to one category and a bag of categories, respectively. Each non-leaf node captures the shared features of a bag of categories. Each node also holds a group of classifiers trained by AdaBoost, to discriminate the categories locating at its left and right child node. Recognition is then a process to find a path from the root to a leaf, which represents a unique category. The very promising result on Caltech 101 dataset shows the robustness of the proposed approach.

Keywords
object recognition, bag of categories, multiclass object recognition tree, AdaBoost

1. INTRODUCTION

Object recognition[1, 2, 3, 4, 5] is a fundamental vision problem: put simply, what's in the image, and where? To recognize different categories of objects in images, people usually reduce multiclass categorization problem to binary problem [6]. The approaches include that each category is compared against all others, or all pairs of categories are compared to each other, or error correcting output codes (ECOC) are used. They all ignore one fact that the subset of categories may exist some common features. And these traditional approaches require training different classifiers for each category. This can be slow and the performance of learned single classifier is poor for limited training samples.

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

We propose a novel concept "*bag of categories*" to show how to share the common features among different categories. Just like in the context of document analysis, *bag-of-words* [7] assumes that the order of words in a document can be neglected. In computer vision, the *bag of keypoints* [8, 9] method is based on vector quantization of affine invariant descriptors of image patches. *Bag-of-features* [10, 11] methods represent an image as an orderless collection of local features. *Bag of patches* and *bag of codewords* [12] are also appeared in the context of vision analysis.

For "*bag of categories*", two questions should be answered:

(Q1) How to combine these bags of categories?

(Q2) How to describe each bag of category?

For the first question, we build a multiclass object recognition tree, in which the leaf node and the non-leaf node correspond to one category and a subset of all categories, respectively. For each non-leaf node, its left and right children node hold two disjoint category subsets of its parent node. We call the category subset at each non-leaf node of tree as "*a bag of categories*". That is, bags of categories are combined by object recognition tree. For the second question, features with classifier is used to describe a bag of category. To improve the performance of each node, a group of classifiers are trained by AdaBoost [13, 14] based on the same dataset and different feature set. In the proposed framework, object recognition turns into a process to find a path from the root to a leaf, which represents a unique category, instead of the common nearest neighbor algorithm.

Our approach is different with probabilistic boosting tree (PBT) [15]. At each layer of the tree, each

category only exists in one node in our approach, however, each category may appear in more than one nodes in PBT. The construction and training of tree are together done by PBT. They are two different phases in our approach. In addition, we take more complicated features than PBT approach. Each node holds more than one strong classifier in our approach comparing one strong classifier in PBT. The recognition rate is 20% for PBT, however, it is 56.3% in our approach under the same dataset Caltech 101 [16].

The whole paper is scheduled as follows. Binary decision tree for multiclass object recognition is presented in Section 2. Details of bag of categories are introduced in Section 3. In Section 4, experiment design and analysis result are given. Finally, the conclusion of the paper is presented.

2. BINARY DECISION TREE FOR COMBINING BAGS OF CATEGORIES

Decision tree employs a hierarchically structured decision function in a sequential fashion. To combine bags of categories, the binary decision tree is constructed. For each non-leaf node, it corresponds to a binary classification. Thus, multiclass object recognition is transferred into many binary classification problems. It is the basis to construct a tree framework. In this section, we first describe the elements of the recognition tree and the splitting rules. Then we present the algorithm of tree construction for combining bags of categories.

2.1. From Multiclass Problem to Binary Classification Problem

Suppose we have m samples, each of which corresponds to an image,

$$\{\mathbf{x}_i | \mathbf{x}_i \in \mathbf{X}, \mathbf{X} \subset \mathbb{R}^n\}_{i=1}^m$$

where n is equal to the number of image pixels. The corresponding class label of each sample is

$$\{y_i | y_i \in \mathbf{Y}, \mathbf{Y} = \{1, \ldots, C\}\}_{i=1}^m$$

where C is the total number of categories.

There are many ways to reduce a multiclass problem to multiple binary classification problems [6], such as one-against-all approach, all-pairs approach, and error correcting output codes (ECOC).

- One-against-all approach is the simplest approach to create one binary problem for each of the C categories. That is, for any category label $r \in \mathbf{Y}$, all samples labeled $y_i = r$ are considered as one class and all other samples are considered as the other class.
- In all-pairs approach, for each distinct label pair $r, s \in \mathbf{Y}$, samples labeled $y_i = r$ are considered as one class, and those labeled $y_i = s$ belong to the other class. All other samples left are simply ignored.
- Error correcting output codes (ECOC) is to associate each class $r \in \mathbf{Y}$ with a row of a "coding matrix" $\mathbf{M} \in \{-1, 0, +1\}^{C \times l}$ for some l, where l is number of classifiers. The binary classification problem is then run once for each column of the matrix.

Unlike the one-against-all, all-pairs, and ECOC approach, we construct the multiclass object recognition tree by subdividing the whole category set into two subsets, as the two sides of the binary classification tree, and recursively to apply this kind of subdivision until all category is distinguished each other. To construct the decision tree, some elements should be considered first.

2.2. Elements of Decision Tree

Binary decision tree is known as a class of nonlinear classifiers. The root of the tree $\mathbf{T}$ is associated with the training set $\mathbf{X}$, which corresponds to all categories to be recognized. Each node, t, represents a specific subset $\mathbf{X}_t$ of the training set $\mathbf{X}$, which corresponds to a subset of all categories. We call categories in one subset as "a bag of categories", since we describe these categories in the subset using the same classifiers with sharing features. Splitting of a node is equivalent to split the subset $\mathbf{X}_t$ into two disjoint descendant subsets, $\mathbf{X}_{tL}, \mathbf{X}_{tR}$. For every split, the following equation (1) and (2) need to be satisfied:

$$\mathbf{X}_{tL} \bigcap \mathbf{X}_{tR} = \emptyset \tag{1}$$

$$\mathbf{X}_{tL} \bigcup \mathbf{X}_{tR} = \mathbf{X}_t \tag{2}$$

Here $\mathbf{X}_{tL}$ and $\mathbf{X}_{tR}$ correspond to two bags of categories, and locate at the left and right children node, respectively.

In order to develop a binary decision tree, the following design elements have to be considered in the constructing phase:

(E1) A splitting criterion should be adopted according to which the best split from the set of candidate one is chosen.

(E2) A stop splitting rule is required to control the growth of the tree and to declare a node as a terminal leaf.

(E3) A rule is required that assigns each leaf to a specific category.

A leaf node is formed when the subset $\mathbf{X}_t$ only includes the samples which belong to a single category. That is, a leaf node corresponds to a single category. And non-leaf node is a bag of categories. The code word from the root to each leaf node represents a specific category. Details on splitting criterion are described in the next subsection.

2.3. Splitting Criterion

The descendant nodes are associated with two new subsets, i.e., $\mathbf{X}_{tL}, \mathbf{X}_{tR}$, respectively. For the tree growing methodology, from the root down to the leaves, every split generates new subsets, in which elements are are more similar each other compared to the ancestor's subset $\mathbf{X}_t$. This means that the training feature vectors in each one of the new subsets show a higher preference for specific categories. The splitting algorithm is described as follows.

Algorithm 1. Splitting algorithm

Input:

- Matrix of feature values $\mathbf{G_t}$ of samples $\mathbf{X_t}$ with corresponding label set $\mathbf{Y_t}$.
- Desired number of clusters L, here $L = 2$ for left and right node of the binary tree.
- Iteration times N of splitting.

Algorithm:

(S1) For $j = 1$ to L

- Initialize arbitrary clustering center $\theta_j(0)$ for the j-th cluster center θ_j.

(S2) Repeat N iterations to find L cluster centers.

(S2a) For $i = 1$ to m_t (m_t is the number of samples to be split.)

- Determine the closest representative, say θ_j, for $\mathbf{g}_i \in \mathbf{G_t}$. Here $\mathbf{g}_i$ is the feature value of sample $\mathbf{x}_i$. The closest representative means that $d(\mathbf{g}_i, \theta_j) < d(\mathbf{g}_i, \theta_k), k = 1, \ldots, L,$ and $k \neq j$, where $d(\cdot, \cdot)$ is a distance measure.
- Set $b(i) = j$ for sample $\mathbf{x}_i$. It shows sample $\mathbf{x}_i$ is allocated to the j-th node of the tree.

(S2b) For $j = 1$ to L

- Determine θ_j as the mean of the vectors $\mathbf{g}_i \in \mathbf{G}$ with $b(i) = j$ to update parameter. That is, the original θ_j is replaced by the mean of the feature values of samples which are allocated to the j-th node of the tree in step (S2a).

(S3) For each category, count the number of samples located in each child node.
For $j = 1$ to L
For $k = 1$ to C_t (C_t is the number of categories to be split.)

- Compute

$$\lambda_{jk} = \sum_{i=1}^{m} \delta(\mathbf{x}_i) \tag{3}$$

where

$$\delta(\mathbf{x}_i) = \begin{cases} 1 & \text{if } \mathbf{y}_i = k \text{ and } b(i) = j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

(S4) Decide the final splitting
For $i = 1$ to m_t

- Set $b(i) = j$, if $\lambda_{jy_i} \geq \lambda_{ky_i}, k = 1, \ldots, L,$ $k \neq j$. It means sample $\mathbf{x}_i$ is split to the j-th node of the tree.■

For binary tree ($L = 2$), the descendant nodes are derived with two new subsets, i.e., $\mathbf{X}_{tL}$ and $\mathbf{X}_{tR}$.

$$\mathbf{X}_{tL} = \{\mathbf{x}_i | b(i) = 1\}, \mathbf{X}_{tR} = \{\mathbf{x}_i | b(i) = 2\}$$

Therefore, the splitting algorithm is also the process to transfer each bag of categories into a smaller bag of categories. The process is continued until the stop rule is satisfied.

2.4. Algorithm for Constructing Binary Decision Tree

After the discussion on the major elements needed for the growth of a decision tree, we are now ready to summarize the basic algorithmic steps for constructing a binary decision tree for bag of categories.

Algorithm 2. Tree constructing algorithm

(S1) Begin with the root, i.e., $\mathbf{X}_t = \mathbf{X}$

(S2) For each new node t

- Learn a group of classifiers with AdaBoost based on sample set $\mathbf{X}_{tL}$ and $\mathbf{X}_{tR}$.

(S3) Do all samples of the subset $\mathbf{X}_t$ belong to a single category?

- If yes, stop splitting and declare node t as a leaf and designate it with a category label. End.
- If not, generate two descendant nodes t_L and t_R with associated subsets $\mathbf{X}_{tL}$ and $\mathbf{X}_{tR}$. Return to (S2).■

In the algorithm, we first classify the whole sample set into two subsets. In the first stages of binary tree, the root includes all categories to be classified. The existed classifiers in the node of tree are used to estimate the image's category label for its children nodes. Next, we classify each subset into two new subsets following Equ.(1) and Equ.(2). Each subset corresponds to a bag of categories. We end up the classification until each category is a sole part. This is a binary decision tree model for multiclass object recognition.

3. BAG OF CATEGORIES FOR VISUAL CATEGORY VOCABULARY

The classifier with selected features in each node of the tree corresponds a visual category vocabulary. To make the visual category vocabulary describe the bag of categories more clearly, we learn a group of classifiers for each node. Since these classifiers are independent, we propose a two-stage voting strategy to help the final decision of each bag of categories. After describing the bag of categories with visual category vocabulary and combing the bag of categories with binary decision tree, the recognition process for a new sample can be finished by the traversal of the tree from root to leaf.

3.1. Visual Category Vocabulary

Each category vocabulary is a group of AdaBoost strong classifiers. Once the samples are grouped into a binary tree, we train a group of classifiers for each node of the binary tree. AdaBoost[13, 14] is used to train each classifier, and original haar-like features proposed by viola and Jones[17] are taken as feature set. AdaBoost algorithm, proposed in the Computational Learning Theory literature, is a method to find a highly accurate hypothesis (a strong classifier) by combining many "weak" hypotheses, each of which is based on the reweighted version of the training data in order to emphasize those which are incorrectly classified by previous weak classifiers, and only moderately accurate. The final strong classifier is a weighted combination of weak classifiers followed by a threshold. The decision of a strong classifier is bias. Thus, we learn a group of independent strong classifiers at each node. The final decision is created with the voting rule described as follows.

3.2. Voting Rule

The combined decision is obtained by a majority vote of the individual classifiers. Majority vote does not assume prior knowledge of the behavior of the individual classifiers.

We assume that there are n classifiers, and each classifier produces a unique decision regarding the identity of the sample. In our approach, the following conditions are satisfied:

(C1) The number of voters is odd.

(C2) Each voter has the same probability of voting one way.

(C3) The individual decision is independent, since each classifier is trained independently and is based on different feature set.

We take two-stage voting strategy.

Algorithm 3. Two-stage voting algorithm

(S1) If $|k_t - \frac{n_t+1}{2}| > \tau$, the sample is assigned to the side when k experts are agreed.

(S2) If $|k_t - \frac{n_t+1}{2}| \leq \tau$, consider two descendant nodes t_L and t_R with associated subsets $\mathbf{X}_{tL}$ and $\mathbf{X}_{tR}$.

- If $|k_{tLL} + k_{tLR} - n_{tL}| > |k_{tRL} + k_{tRR} - n_{tR}|$, the sample is assigned to the left node.
- If $|k_{tLL} + k_{tLR} - n_{tL}| < |k_{tRL} + k_{tRR} - n_{tR}|$, the sample is assigned to the right node.
- If $|k_{tLL} + k_{tLR} - n_{tL}| = |k_{tRL} + k_{tRR} - n_{tR}|$, the sample is assigned to the left or right node randomly.

Here τ is voting factor. k_t, k_{tL}, and k_{tR} are the number of experts (strong classifiers) agreeing on the identity in the parent node, left child, and right child node, respectively. k_{tLL} and k_{tLR} are left and right child of the left child node t_L. And k_{tRL} and k_{tRR} are left and right child of the right child node t_R.

3.3. Category Encoding

In the proposed framework, each node is a set of classifiers. The bag of categories refers to a set of classifiers to describe a group of categories. More precisely, the category recognition is a two-step process.

(a) The local regions (patches) corresponding to the features of each classifier are sampled from image.

(b) The sample is assigned to the left or right node of the binary decision tree.

We repeat (b) until we reach a leaf node classifier to make a global decision about the test image.

Decision tree is a multistage system, in which categories are sequentially rejected until we reach a final accept category. To this end, the feature space is split into unique regions, corresponding to each category, in a sequential manner. Recognition is achieved from the root to the leaf node. Each leaf node corresponds to a code book, and each category can be noted by a binary code. This is a deep first traversal of tree.

4. EXPERIMENTS

We use the Caltech 101 object class dataset [16]. The Caltech 101 dataset[1] contains 9,197 images comprising 101 different object categories, plus a background category, collected via Google image search in September 2003 by Fei-Fei Li, Marco Andreetto, and Marc 'Aurelio Ranzato. Each category contains about 40 to 800 images. Most categories have about 50 images.

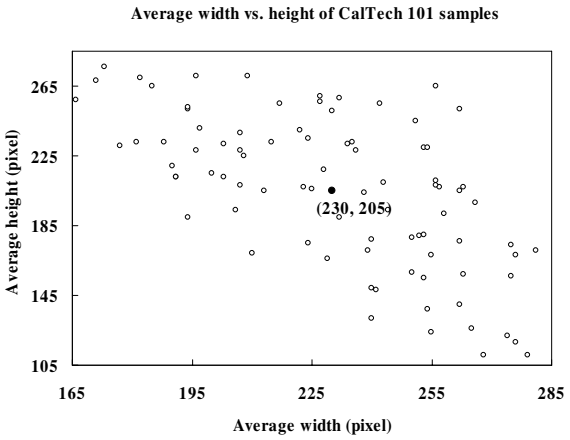


Fig. 1. Average width vs. height of Caltech 101 samples. Each circle represents a pair of width and height of one category. Some small width and height pairs are excluded. The solid circle point with coordinate (230, 205) is the average width and height of samples for all categories.

In our experiment, "Faces" and "Faces_easy" category are combined into one category. Therefore, the total number of categories is 100. Fig. 1 computes the average width and height of Caltech 101 samples. In Fig. 1, fifteen pairs of width and height are excluded because their small values in width, height, or both. In order to accelerate the training speed, we resize the width and height to 100×90 following the scale of average width and height 230×205.

Thirty samples were chosen randomly from each of the 100 object categories, yielding a total 3000 samples. For each class, testing images are ones excluding those used as training samples. Samples are not alignment before training.

Experimental results on Caltech 101 dataset show the robustness of our proposed approach. The results are shown in Table 1 and Fig. 2 for 15, 25 and 30 training samples per category. Results are reported from Fei-Fei, Fergus, & Perona [2], Serre, Wolf, & Poggio [4], Holub, Perona, & Welling [5], Berg, Berg, & Malik [1], Mutch & Lowe [3], Tu [15], and proposed approach.

[1] www.vision.caltech.edu/ImageDatasets/Caltech101/

The final decision tree of our experiment based on Caltech 101 dataset is given in Fig3. Each row corresponds to the coding of each category, and 0 and 1 shows the node locates at the left and right of its father node. For example, the ID of the second row is 23, and the code string "00000010" is the code word of the category 23 named "crab". It exists in the left child node from stage 1 to stage 6, the right child node of the stage 7, and the left child node of the final stage 8. Its substring represents a node of the tree, whose length indicates the number of layer of the tree.

Approach	Samples		
	15	**25**	**30**
Fei-Fei, Fergus, & Perona [2]	16	–	–
Tu [15]	–	20	–
Serre, Wolf, & Poggio [4]	35	–	42
Holub, Perona, & Welling [5]	37	–	43
Berg, Berg, & Malik [1]	48	–	–
Mutch & Lowe [3]	51	–	56
Proposed approach	42	–	56.3

Table 1. Correct rate in percentage with 15, 25, or 30 training samples per category on Caltech 101 dataset.

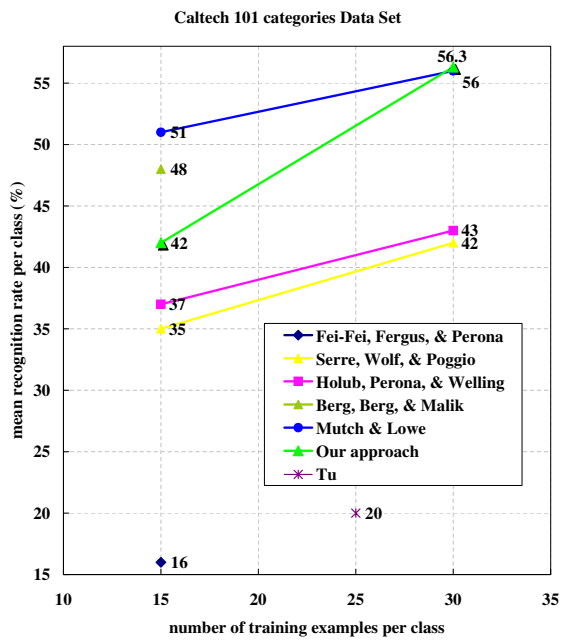


Fig. 2. Mean recognition rate on the Caltech 101 data set. This plot shows the recognition rate using the binary decision tree for multiclass object recognition for 15, 25 and 30 training samples per category. All points on the plot refer to recognition rates that have been normalized according to the number of test samples per category. (The figure is best viewed in color.)

 ISBN 978-86943-15-2

ID	Category Name	STAGE(LEFT) 1	2	3	4	5	6	7	8	9	10
1	accordion	0	0	0	0	0	0	0			
23	crab	0	0	0	0	0	0	1	0		
48	hedgehog	0	0	0	0	0	0	1	1		
20	chandelier	0	0	0	0	0	1	0	0		
55	lamp	0	0	0	0	0	1	0	1	0	
80	schooner	0	0	0	0	0	1	0	1	1	
89	stop_sign	0	0	0	0	0	1	1			
13	butterfly	0	0	0	0	1	0	0			
53	kangaroo	0	0	0	0	1	0	1	0		
83	sea_horse	0	0	0	0	1	0	1	1		
73	pizza	0	0	0	0	1	1	0			
94	umbrella	0	0	0	0	1	1	1			
19	chair	0	0	0	1	0	0				
41	flamingo_he	0	0	0	1	0	1	0			
65	minaret	0	0	0	1	0	1	1			
76	revolver	0	0	0	1	1	0				
95	watch	0	0	0	1	1	1				
2	airplanes	0	0	1	0	0	0	0			
39	ferry	0	0	1	0	0	0	1			
6	bass	0	0	1	0	0	1	0			
46	hawksbill	0	0	1	0	0	1	1			
62	mayfly	0	0	1	0	1	0				
67	nautilus	0	0	1	0	1	1				
12	buddha	0	0	1	1	0	0				
50	ibis	0	0	1	1	0	1				
35	euphonium	0	0	1	1	1	0				
98	wild_cat	0	0	1	1	1	1				
3	anchor	0	1	0	0	0					
25	crocodile	0	1	0	0	1	0				
31	dragonfly	0	1	0	0	1	1				
28	dalmatian	0	1	0	1	0	0	0			
57	Leopards	0	1	0	1	0	0	1			
71	panda	0	1	0	1	0	1				
43	gerenuk	0	1	0	1	1	0				
84	snoopy	0	1	0	1	1	1				
5	barrel	0	1	1	0	0	0	0			
37	Faces	0	1	1	0	0	0	1			
10	brain	0	1	1	0	0	1	0			
54	ketch	0	1	1	0	0	1	1	0		
60	lotus	0	1	1	0	0	1	1	1		
16	car_side	0	1	1	0	1	0	0			
47	headphone	0	1	1	0	1	0	1			
96	water_lilly	0	1	1	0	1	1				
26	crocodile_h	0	1	1	1	0					
40	flamingo	0	1	1	1	1	0				
100	wrench	0	1	1	1	1	1				

ID	Category Name	STAGE(RIGHT) 1	2	3	4	5	6	7	8	9	10
4	ant	1	0	0	0	0	0				
14	camera	1	0	0	0	0	1				
17	ceiling_fan	1	0	0	0	1	0	0	0		
34	emu	1	0	0	0	1	0	0	1	0	
63	menorah	1	0	0	0	1	0	0	1	1	
74	platypus	1	0	0	0	1	0	1			
85	soccer_ball	1	0	0	0	1	1				
7	beaver	1	0	0	1	0	0	0	0	0	
36	ewer	1	0	0	1	0	0	0	0	1	0
52	joshua_tree	1	0	0	1	0	0	0	0	1	1
56	laptop	1	0	0	1	0	0	0	1		
61	mandolin	1	0	0	1	0	0	1			
32	electric_guitar	1	0	0	1	0	1	0			
79	saxophone	1	0	0	1	0	1	1			
88	stegosaurus	1	0	0	1	1					
8	binocular	1	0	1	0	0	0				
33	elephant	1	0	1	0	0	1				
18	cellphone	1	0	1	0	1	0	0			
21	cougar_body	1	0	1	0	1	0	1			
93	trilobite	1	0	1	0	1	1				
44	gramophone	1	0	1	1	0					
66	Motorbikes	1	0	1	1	1	0				
72	pigeon	1	0	1	1	1	1	0	0		
77	rhino	1	0	1	1	1	1	0	1		
99	windsor_chair	1	0	1	1	1	1	1			
9	bonsai	1	1	0	0	0	0	0			
22	cougar_face	1	1	0	0	0	0	1			
81	scissors	1	1	0	0	0	1				
29	dollar_bill	1	1	0	0	1	0				
70	pagoda	1	1	0	0	1	1	0	0		
82	scorpion	1	1	0	0	1	1	0	1		
86	stapler	1	1	0	0	1	1	1			
11	brontosaurus	1	1	0	1	0	0	0			
24	crayfish	1	1	0	1	0	0	1	0		
59	lobster	1	1	0	1	0	0	1	1	0	0
69	okapi	1	1	0	1	0	0	1	1	0	1
78	rooster	1	1	0	1	0	0	1	1	1	
15	cannon	1	1	0	1	0	1	0	0	0	
45	grand_piano	1	1	0	1	0	1	0	0	1	
64	metronome	1	1	0	1	0	1	0	1		
91	sunflower	1	1	0	1	0	1	1	0		
92	tick	1	1	0	1	0	1	1	1		
30	dolphin	1	1	0	1	1	0	0	0		
42	garfield	1	1	0	1	1	0	0	1		
68	octopus	1	1	0	1	1	0	1			
97	wheelchair	1	1	0	1	1	1				
27	cup	1	1	1	0	0					
49	helicopter	1	1	1	0	1					
51	inline_skate	1	1	1	1	0	0	0			
75	pyramid	1	1	1	1	0	0	1	0		
87	starfish	1	1	1	1	0	0	1	1		
90	strawberry	1	1	1	1	0	1				
58	llama	1	1	1	1	1	0				
101	yin_yang	1	1	1	1	1	1				

Fig. 3. Final decision tree for multiclass object recognition of Caltech 101 dataset. The first two columns are category ID and name. From Column three to Column twelve is the code of category.

5. CONCLUSION

We construct the multiclass object recognition tree by subdividing the whole category set into two subsets, as two sides of the binary classification tree, and recursively to apply the subdivision until all category is distinguished each other. A subset corresponds to a bag of categories. Recognition is a process to find a path from root to a leaf, which represents a unique category. Multiclass object recognition is transferred into many binary classification problems by the binary tree.

We answer three issues for object recognition.

1. How do we represent the object? We represent object by code book and bag of categories.
2. Using the representation, how do we learn a particular object category? The binary decision tree is constructed for multiclass object recognition.
3. Finally how do we use the model we have learnt to find further instances in query images? This is a deep first traversal of tree.

More complicated features will be considered in the future work to improve the performance.

6. REFERENCES

[1] A. C. Berg, T. L. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondence," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[2] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," in *CVPR Workshop of Generative Model Based Vision*, 2004.

[3] J. Mutch and D. Lowe, "Multiclass object recognition with sparse, localized features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006, pp. 11–18.

[4] T. Serre, L. Wolf, and T. Poggio, "object recognition with features inspired by visual cortex," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[5] A. D. Holub, M. Welling, and P. Perona, "Combining generative models and fisher kernels for object recognition," in *Proceedings of IEEE International Conference on Computer Vision*, 2005.

[6] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: a unifying approach for margin classifiers," *The Journal of Machine Learning Research*, vol. 1, pp. 113–141, 2001.

[7] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," *The Journal of Machine Learning Research*, vol. 3, pp. 993–1022, January 2003.

[8] G. Csurka, C. Bray, C. Dance, and L. Fan, "Visual categorization with bags of keypoints," in *In Workshop on Statistical Learning in Computer Vision, ECCV*, 2004, pp. 1–22.

[9] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: A comprehensive study," *International Journal of Computer Vision*, vol. 73, no. 2, pp. 213–238, 2007.

[10] M. Marszalek and C. Schmid, "Spatial weighting for bag-of-features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 2118–2125.

[11] E. Nowak, F. Jurie, and B. Triggs, "Sampling strategies for bag-of-features image classification," in *Proceedings of European Conference on Computer Vision*, 2006, pp. 490–503.

[12] L. Fei-Fei and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 524–531.

[13] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proceedings of European Conference on Computational Learning Theory*, 1995, pp. 23–37.

[14] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of International Conference on Machine Learning*, 1996, pp. 148–156.

[15] Z. Tu, "Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering," in *Proceedings of IEEE International Conference on Computer Vision*, vol. 2, Beijing, China, Oct 2005, pp. 1589–1596.

[16] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[17] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

Visual simulation of Civil Engineering activities: Didactic virtual models

Alcínia Zita Sampaio
Technical University of Lisbon
Dep. Civil Engineering and Architecture
Av. Rovisco Pais
Portugal, 1049-001 Lisbon
zita@civil.ist.utl.pt

Pedro Gameiro Henriques
Technical University of Lisbon
Dep. Civil Engineering and Architecture
Av. Rovisco Pais
Portugal, 1049-001 Lisbon
pgameiro@civil.ist.utl.pt

ABSTRACT

Three-dimensional geometric models have been used to present architectural and engineering works, showing their final configuration. But, when the clarification of a detail or the constitution of a construction step in needed, these models are not appropriate because they do not allow the observation of the construction activity. Models that could present dynamically changes of the building geometry are a good support on education in civil engineering domain. Techniques of geometric modeling and virtual reality were used to obtain interactive models that could visually simulate the construction activity. The applications explain the construction work of a cavity wall and a bridge. These models allow the visualization of the physical progression of the work following a planned construction sequence, the observation of details of the form of every component of the works and support the study of the type and method of operation of the equipment applied in the construction. The models present distinct advantage as educational aids in first-degree courses in Civil Engineering. The use of Virtual Reality techniques in the development of educational applications brings new perspectives to the teaching of subjects related to the field of civil construction.

Keywords

Didactic models, Education, Virtual reality, Visual simulation.

1. INTRODUCTION

Normally, three-dimensional (3D) geometric models, which are used to present architectural and engineering works, show only their final form, not allowing the visual simulation of their physical increasing. The models concerning construction needs to be able to produce changes of the geometry of the project. The integration of geometric representations of a building together with scheduling data related to construction planning information is the bases of 4D (3D + time) models [Fis00]. In construction field 4D models combine 3D models with the project timeline [Ret97]. In addition Virtual Reality (VR) technology has been used to turn 4D models more realistic allowing interaction with the environment representing the construction place. VTT Building Technology has been developing and implementing applications based on this technique improving a better communication between the partners in a construction project [Lei03].

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

The use of 4D models just linked with construction planning software or with virtual/interactive capacities, concerns essentially economic and administrative benefits as a way of presenting the visual simulation of the "real situation" of the work in several step of its evolution. Those models are created for each particular project and are mostly manipulated by the principal designer or contractor. In another cases they are used to explain complex or innovative construction processes to subcontractor.

When modeling 3D environments a clear intention of what to show must be planned, because the objects to display and the details of each one must be appropriated to the goal the Engineer want to achieve with the model. For instance, if the objective is to explain the relationship between construction phases and the financial stages, the 4D model must represent the correspondent physical situation according to the

established construction diagram and with the degree of detail appropriated. Developing didactic models for students concerns technical tasks, at a level that could be understood by undergraduate students, but also pedagogical judgments.

In the present study, two engineering construction work models were created, from which it was possible to obtain 3D models corresponding to different states of their shape, simulating distinct stages in construction processes [Sam06]. In order to create models, which could visually simulate the construction work and allowing interact with it, techniques of virtual reality were used. The developed applications make it possible to show the physical evolution of the work, the monitoring of the planned construction sequence, and the visualization of details of the form of every component of each construction. They also assist the study of the type and method of operation of the equipment necessary for these construction methodologies [Sam04].

2. DIDACTIC VIRTUAL MODELS

The aim of the practical application of the virtual models is to provide support in civil engineering education namely in those disciplines relating to bridges and construction process both in classroom-based education and in distance learning based on e-learning technology. Specialist in construction processes and bridge design were consulted and involved in the execution of the models in order to obtain efficient and accurate didactic applications. The selected examples are two elementary situations of construction works: an **external wall** is a basic component of a building and the cantilever method of **bridge deck** construction is applied frequently.

The pedagogic aspect and the technical knowledge are presented on the selection of the quantity and type of elements to show in each virtual model, on the sequence of exhibition to follow, on the relationship established between the components of both type of construction, on the degree of geometric details needed to present and on the technical information that must go with each constructive step. Further details complement, in a positive way, the educational applications bringing to them more utility and efficiency. Namely, the model of the wall shows the information concerning construction activity of interest for students corresponding to the geometric stage displayed in each moment and the bridge construction model shows particularly the movement of the equipment in operation during the progression. So when students go to visit real work places, since the essential details were previously presented and explained in class, they are able to better understand the construction operation they are seeing.

In addition, the use of techniques of virtual reality on the development of these didactic applications is helpful to education improving the efficiency of the models in the way it allows the interactivity with the virtual activity. The virtual model can be manipulated interactively allowing the teacher or student to monitor the physical evolution of the work and the construction activities inherent in its progression. This type of model allows the participant to interact in a intuitive manner with the simulated environment, to repeat the sequence or task until the desired level of proficiency or skill has been achieved and to perform in a safe environment. Therefore, this new concept of VR technology applied to didactic models brings new perspectives to the teaching of subjects in the area of civil engineering.

3. 3D MODEL OF THE WALL

One of the developed applications corresponds to the model of a masonry cavity wall, one of the basic components of a standard construction. To enable the visual simulation of the construction of the wall, the geometric model generated is composed of a set of elements, each representing one component of the construction. The selection of elements and the degree of detail of the 3D model configuration of each component had the support of teachers and specialist in construction. Using the *EON Reality system* [EON03], a system of virtual reality technologies, specific properties were applied to the model of the wall in order to obtain a virtual environment. Through direct interaction with the model, it is possible both to monitor the progress of the construction process of the wall and to access information relating to each element, namely, its composition and the phase of execution or assembly of the actual work, and compare it with the planned schedule. This model had been used to distinct advantage as an educational aid in Civil Engineering degree course modules.

Geometric Modeling of the Elements of the Construction Environment

The definition of the 3D model of an exterior wall of a conventional building comprises the structural elements (foundations, columns and beams), the vertical filler panels and two bay elements (door and window). Every element was modeled using the *AutoCAD system*.

The structural elements of the model were created with parallelepipeds and were connected according to their usual placement in building works. Because this is an educational model, the steel reinforcements were also defined. In the model, the rods of the reinforcements are shown as tubular components with circular cross-section (Figure 1).

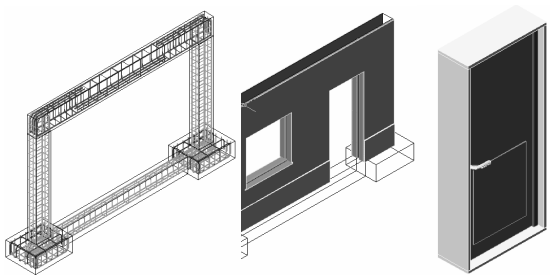


Figure 1: 3D models of the wall components

The type of masonry selected corresponds to an external wall formed by a double panel of breezeblocks, *11 cm*, wide with an air cavity, *6 cm,* wide (Figure 1). Complementary to this, the vertical panels were modeled, comprising: the thermal isolation plate placed between the brick panels; the plaster applied to the external surface of the wall; the stucco applied on the internal surface; two coats of paint both inside and out and the stone slabs placed on the exterior surface. Finally, two usual bay elements, a door and a window, were modeled.

Programming the Virtual Construction

The completed model was then transferred to the virtual reality system *EON* (as a design file with *3ds* extension). In this system, the visual simulation of the building process of the wall, following a realistic plan of the construction progress, was programmed. For this effect, 23 phases of construction were considered (Figure 2).

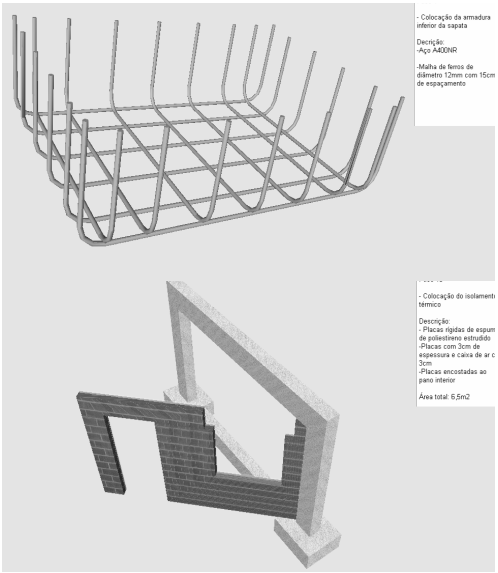


Figure 2: Exhibition of phases in building evolution.

The order in which components are consecutively exhibited and incorporated into the virtual model, represent realistically the physical evolution of the wall under construction.

During the animation, the student can control the length of time that any phase is exhibited and observe the model using the most suitable camera and zoom positions for a correct perception of the details of construction elements. It is possible to highlight the component incorporated at each new phase and to examine it in detail (Figure 3).

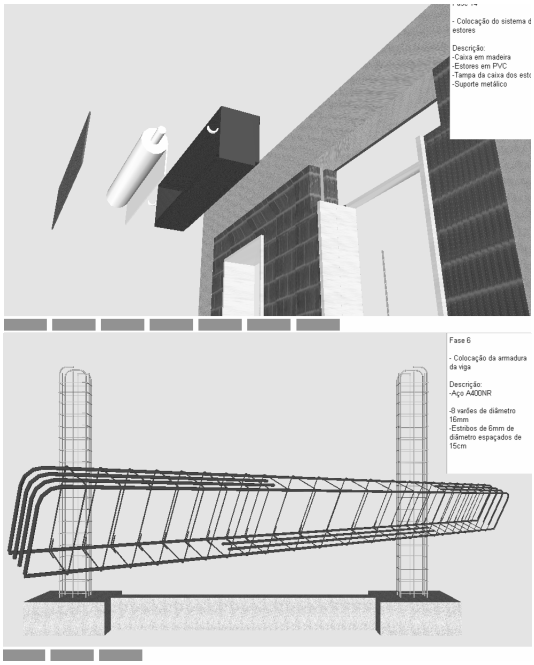


Figure 3: Elements displaced from the global model of the wall.

Included, under the window in which the virtual scene is exhibited, is a bar, which shows the progress of the construction. Throughout the animation, the bar is filled, progressively, with small rectangles symbolizing the percentage built at the time of the viewing of that particular phase, in relation to the completed wall construction (Figure 3). Symbolically, it represents the bar diagrams normally used on construction plans.

Simultaneously, with the visualization of each phase, a text is shown (in the upper right corner of the window, Figure 4), giving data relating to the stage being shown, namely, its position within the construction sequence, the description of the activity and the characterization of the material of the component being incorporated.

The development of the model was supported by engineer specialist in construction activity. So this is a guarantee that the model shows the construction sequence in a correct way and the configuration of each component was defined with accuracy. In this educational application, it is important to include details such as: bar showing the construction progress; text with information concerning the stage

observed; the possibility to highlight elements from the model; the accuracy of the reinforcements and the way then connect inside the structural elements; the details of the configuration of vertical panels and components of the window and the door.

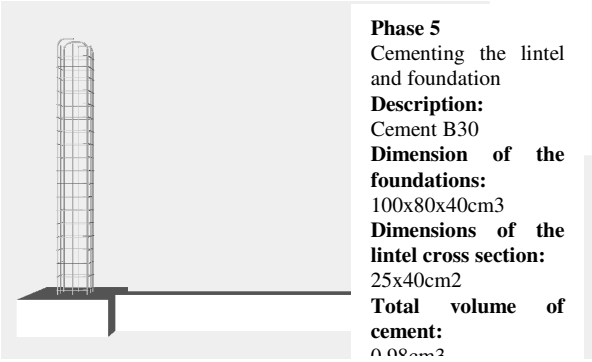


Figure 4: Presentation of text describing the exhibited phase.

4. 3D MODEL OF THE BRIDGE

The second model created allows the visual simulation of the construction of a bridge using the cantilever method. Students are able to interact with the model dictating the rhythm of the process, which allows them to observe details of the advanced equipment and of the elements of the bridge (pillars, deck and abutments). The sequence is defined according to the norms of planning in this type of work.

The North Viaduct of the Bridge Farm, in Madeira, Portugal, was the case selected for representation in the virtual environment [GRI97]. In cross-section, the deck of the viaduct shows a box girder solution and its height varies in a parabolic way along its three spans. The most common construction technique for this typology is the cantilever method of deck construction. This method starts by applying concrete to a first segment on each pillar, the segment being long enough to install on it the work equipment. The construction of the deck proceeds with the symmetrical execution of the segments starting from each pillar. The continuation of the deck, uniting the cantilever spans, is completed with the positioning of the closing segment. The support of specialist in bridge designs was essential to obtain an accurate model, not only on the geometry definition of components of the bridge and devices, but also on the establishment of the progression sequence and of the way the equipment operates.

Geometric Modeling of the Construction Environment

A computer graphic system which enables the geometric modeling of a bridge deck of box girder typology was used to generate, 3D models of deck segments necessary for the visual simulation of the construction of the bridge. Geometric description can be entered directly into the deck-modeling program. To achieve this, the developed interface presents diagrams linked to parameters of the dimensions, so facilitating the description of the geometry established for each concrete case of the deck. Figure 5 shows the interface corresponding to the cross-section of the deck of the example.

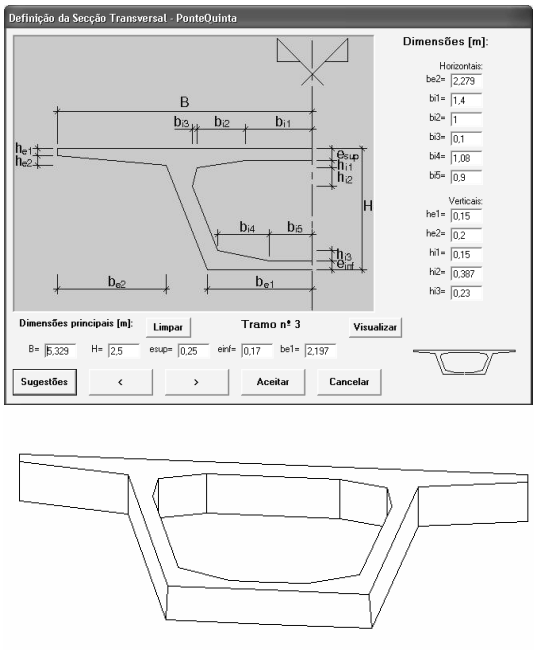


Figure 5: Interface to describe cross-section s and the 3D model of a deck segment.

The description of the longitudinal morphology of the deck and the geometry of the delineation of the service road, serving the zone where the bridge is to be built is carried out in the same way. The configuration and the spatial positioning of each are obtained with a high degree of accuracy. Using the data relating to the generated sections, the system creates drawings and three-dimensional models of the deck. To obtain the definition of the deck segment models, consecutive sections corresponding to the construction joints are used. The configuration presented by the segment models is rigorously exact. Figure 5 shows one of the segments of the deck.

To complete the model of the bridge, the pillars and abutments were modeled using the *AutoCAD* system. Then followed the modeling of the advanced equipment, which is composed not only of the form traveler, but also the formwork adaptable to the size of each segment, the work platforms for each formwork and the rails along which the carriages run (Figure 6).

As, along with the abutments, the deck is concreted with the false work on the ground, the scaffolding for

placement at each end of the deck was also modeled (Figure 6). Terrain suitable for the simulation of the positioning of the bridge on its foundations was also modeled.

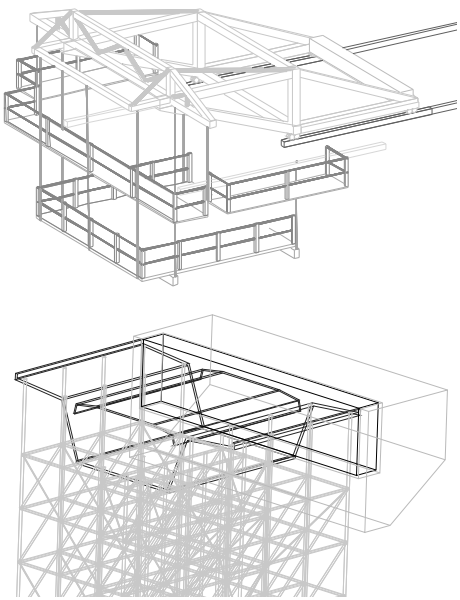


Figure 6: 3D models of the scaffolding and the advanced equipment.

Programming the Construction Animation

The attribution of virtual properties to the model of the bridge was implemented by using the virtual reality system *EON Studio* [EON03]. Once all the 3D models of the construction environment had been generated, they were transposed, in *3ds* extension data file format, to the virtual reality system.

The definition of the construction sequence is based on a counter, which determines the next action when a mouse button is clicked (Figure 7):

- The first action consists of the insertion of the pillars in the first scenario, which is composed solely of the landscape;
- The next step is to place one of the segments on the top of each pillar;
- After this, a form traveler is placed on each segment;
- The construction of the deck is defined symmetrically in relation to each pillar and simultaneously.

For the simulation of the first cantilever segment (in each span), the four form travelers, the corresponding work platforms and the formwork components are included in the scenario. Once the first segments have been concreted, the construction of the cantilevered deck takes place. In each phase, two pairs of segments are defined.

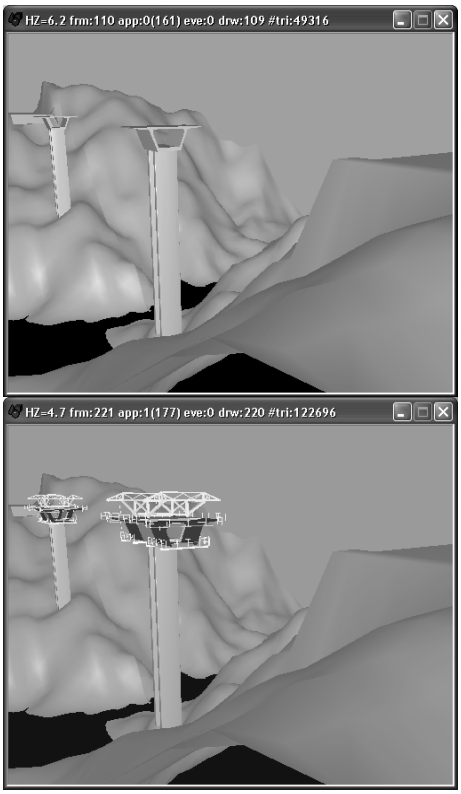


Figure 7: Placing the initial pillars and the advanced equipment.

For each new segment the following steps are established: raising the form traveler; moving the rails in the same direction as the construction (relocating them on the latest segment to have been concreted); moving the form traveler on the rails, positioning it in the zone of the next segment to be made; concrete the segment. Finally, the zone of the deck near the supports is constructed, the false work resting on the ground (Figure 8).

Moving the camera closer to the model of the bridge and applying to it routes around the zone of interest, it is possible to visualize the details of the form of the components involved in the construction process. In this way, the student can interact with the virtual model, following the sequence specifications and observing the details of the configurations of the elements involved.

In a real construction place of a bridge, for security reasons, the student stays far from the local were bridge is under construction, so they can't observe in detail the way of operation and the progression of the construction. Interacting with the model of the bridge in class or using their personal computers they better understand what is going on there in the construction zone.

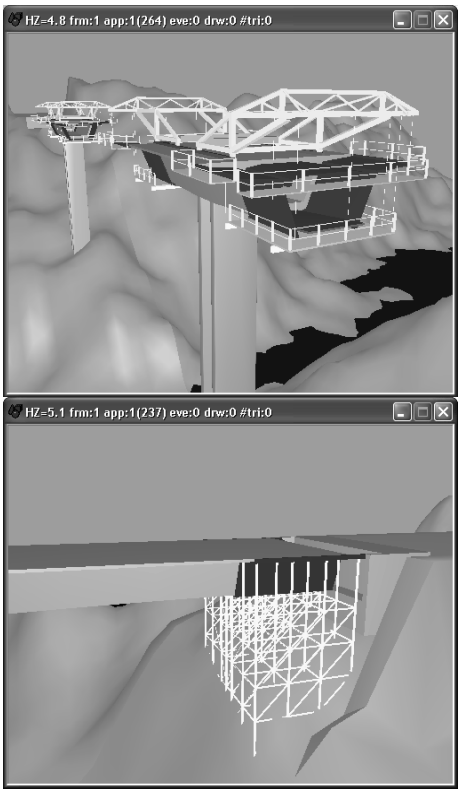


Figure 8: Movement of the advanced equipment and concreting above the false work near the abutment.

5. LEARNING ASPECTS

The models are actually used in face-to-face classes of disciplines of Civil Engineering curriculum: Technical Drawing (1st year), Construction Process (4th year) and Bridges (5th year). The traditional way to present the curricular subjects involved in those virtual models are 2D layouts or pictures. Now, the teacher interacts with the 3D models showing the sequence construction and the constitution of the modeled type of work. Essentially, the models are used to introduce new subjects.

- As in **Technical Drawing**, students have to define and draw structural plants over the architectural layouts, the virtual model of the wall helps to explain the connection between the architectural drawings and the structural solutions needed to support the house configuration. Some indication must be assumed when choosing a structural solution in order to minimize the unpleasant visual appearance in the interior of a house when structural elements (beams, columns, ...) are included in it. The students are 1st year degree, so they have some difficulty to understand the spatial localization of the structural elements and how they must be built and located almost inside the walls. The relationships between the architectural configurations and the structural elements in a building are well explained following the exhibition of the virtual construction of the wall.
- In the discipline of **Construction Process**, in order to prepare students to visit real work places, the teacher shows the construction animation and explains some aspects of the construction process of the wall. Namely, the way the net of irons is defined inside a beam or a column and specially the complexity of the relationship between the distinct types of irons near the zone where the structural elements connect each other (Figure 9). In order to clearly explain this issue related to the structural elements, the iron nets were created as 3D models with distinct colors, and they appear on the virtual scenario following a specific planned schedule. The type, sequence and thickness of each vertical panel that composes a cavity wall are well presented in the virtual model showing step by step the relationship between each other. The configuration detail of each element of a complete wall can be clearly observed manipulating the virtual scenario of the construction.

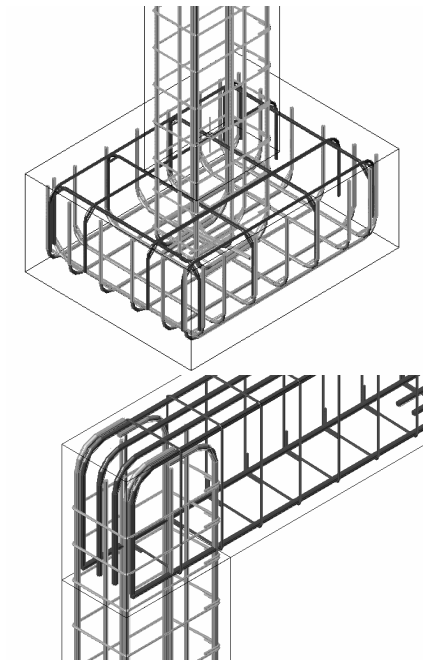


Figure 9: Complex relationship between reinforcements in the join zones of the structural elements.

- The construction model of a **bridge** particularly shows the complexity associated to the concrete work of the deck bridge that is done in a symmetric way. The model also shows in detail the movement of the advanced equipment. In class, the professor must explain way the

process must follow that sequence of steps and the way the equipment devices operates. When the student, of the 5th year, goes to the work place he can observe the complexity and the sequence of construction previously explained.

The students can also interact with those models. For that, the models were posted on the Internet pages of undergraduate courses in Civil Engineering. The student will be able to interact with the application *EonX,* which can be accessed at: http://download.eonreality.com.

6. CONCLUSIONS

It has been demonstrated, through the examples presented here, how the technology of virtual reality can be used in the elaboration of teaching material of educational interest in the area of construction processes. The pedagogical aspects and the technical concepts are attended on the elaboration of both models.

The applications generated represent two standard situations of constructions. The student can interact with the virtual model in such a way that he can set in motion the construction sequence demanded by actual construction work, observe the methodology applied, analyze in detail every component of the work and the equipment needed to support the construction process and observe how the different pieces of a construction element mesh with each other and become incorporated into the model.

These models are used in disciplines involving construction in courses in Civil Engineering and Architecture. They can be used in classroom-based education and in distance learning supported by e-learning technology.

7. REFERENCES

[EON03] Introduction to working in EON Studio, EON Reality, Inc., 2003.

[Fis00] Fischer, M., 4D CAD - 3D Models Incorporated with Time Schedule, In CIFE in Finland, VTT-TEKES, Helsinki, Finland, 2000.

[GRI95] GRID – Panning office, Graphical documentation of the design of North Viaduct of the Bridge Farm, 1st phase, Lisbon, 1995.

[Lei03] Leinonen J., Kähkönen, K. and Retik A., New construction management practice based on the virtual reality technology. In: Raja R.A., Flood I, William J, O'Brien (Ed.) 4D CAD and Visualization in Construction: Developments and Applications, A.A. Balkema Publishers, pp. 75-100, 2003.

[Ret97] Retik, A., Planning and monitoring of construction projects using virtual reality projects, In Project Management Journal, 3(97), pp. 28–32, 1997.

[Sam04] Sampaio, A.Z., Henriques, P.G. and Studer, P., The application of virtual reality technology in housing projects, in proc. XXXII IAHS World Congress – Sustainability of the Housing Projects, Trento (Italy), paper in CDROM (6 pgs), abstract pp. 26, 2004.

[Sam06] Sampaio, A.Z., Henriques, P.G., Studer, P. and Luizi, R.P., Learning and planning in virtual scenarios, in proc. ECT 2006, 5th Int. Conf. on Engineering Computational Technology, Las Palmas de Gran Canaria (Spain), paper ECT 113, abstract pp. 249-250, 2006.

New method to optimize Force-directed layouts of large graphs

Meva DODO dodo@irit.fr
Fenohery ANDRIAMANAMPISOA fandriam@irit.fr
Patrice TORGUET torguet@irit.fr
Jean Pierre JESSEL jessel@irit.fr

Institut de Recherche en Informatique de Toulouse (IRIT)
118, Route de Narbonne, 31062, Toulouse, France

ABSTRACT

This paper describes a novel method to optimize the force-directed placement algorithm for 3D drawing of large graphs. The main idea behind our approach consists in optimizing the layout by equitably distributing vertices in space. We consider the largest sphere inscribed in the 3D space and the vertices are then assigned random initial positions that are improved by force-directed placement. In order to ensure the effectiveness of the algorithm, we propose a new energy function minimization which uses the conjugated gradient of Fletcher-Reeves. Our algorithm is not only addressed to general undirected graphs but it also produces good layouts for large trees. This work is motivated by our need to offer 3D visualization tools for large computing networks but this first phase will be focused on the graph representation.

Keywords

Algorithm, 3D graph drawing, Force-directed placement.

1. INTRODUCTION

Graph algorithms are used in most applications and tools that visualize complex systems or large databases. Their effectiveness comes from the fact that they highlight the structure of a system by presenting each element as a node and their relations as links. This facilitates the perception process required to understand the structure of complex systems, identify the centre of interest and detect possible anomalies.

Several research projects have been carried out on the graph-drawing algorithms these two last decades. [Bat98a] summarizes the most important works on the graphs that can be classified according to their applicability and required aesthetics.

Generally graph drawing algorithms can be classified in two categories: hierarchical algorithms that distribute nodes according to their hierarchy in the graph, and algorithms based on physical models.

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

The hierarchical algorithms are often applied on trees and acyclic directed graphs. These algorithms try to identify a suitable pre-tree from the given graph. Thus the vertex set is partitioned in several subsets so that the smallest subset contains the root node and the other nodes are partitioned into branches. Nodes are placed successively from the root to leaf nodes.

In contrast, algorithms based on physical models focus on undirected general graphs. The family of these algorithms, generally called force-directed placement, involves transforming vertices and edges into a system of forces and finding the minimum energy state of the system. This state is found by running a simulation of the forces or by resolving differential equations.

The algorithm that we present in this paper falls into the second category of graph drawing techniques. Indeed, undirected graphs can be considered as the most general class of graphs because they allow the representation of any type of information and are generally used in the representation of complex systems.

Representing visually hundreds or thousands of vertices in a small area however remains a principal challenge so that several techniques have been associated with the force-directed methods in order to reduce the poor running time and to produce optimal drawings.

This work presents new techniques that optimize the layout by distributing the vertices equitably, which

therefore increasingly reduce edge crossing. Our algorithm is addressed to 3D graph-drawing and its basic ideas consist in using the largest sphere inscribed in space and recursively distributing this sphere to the various vertices. The final positions of vertices are found after several steps and at each step the position of each vertex is improved by force-directed placement.

The second feature of our algorithm is the implementation of a numerical optimization to compute the final position of each vertex. Indeed, the effectiveness of the force-directed placement algorithm resides in the choice of the energy function and the method to minimize this energy.

The conjugated gradient of Fletcher-Reeves is then implemented to perform the numerical optimization. The conjugated gradient is the most efficient method of optimization [Fle64a], [Gil90a], [Wan05a] and its main advantage is that it minimizes a function with several variables in a relatively short time.

This paper is structured as follows. Section 2 reviews earlier force-directed placement algorithms. Section 3 details the contribution of this work, i.e., the vertices distribution approach, the definition of attractive and repulsive forces and the new energy function. We then describe our force-directed placement algorithm at the end of this section. Section 4 presents initial results and shows the capacity of our algorithm to draw large undirected graphs and trees. Finally we conclude and outline perspectives in section 5.

2. RELATED WORK

Most graph-drawing algorithms are based on the physical model that represents vertices as steel rings and edges as springs. Vertices are initially placed at random positions and the system is released so that springs modify the coordinates of vertices until stability is reached (i.e. when the generated total energy is minimal).

This concept was initially developed by Eades [Ead84a] and several projects have been successively carried out. [Kam89a] improved Eades' work by introducing the concept of ideal distance between vertices that are not neighbors. This distance is proportional to the length of the shortest path between them. They are also the first who formulated the total energy of a graph and found that the searching for the vertices' final positions that minimize edge crossings is a process reducing the total energy of a physical system of rings and springs. [Fru91a] developed a new variant of Eades' algorithm but, instead of solving differential equations to minimize the total energy of the system, they used a simulation of repulsive and attractive forces to find the final position of each vertex. Most recent force-directed algorithms are based on the approach of Fruchterman & Reingold. Although they produce aesthetic results for medium graphs, these algorithms have two major drawbacks. The first is that they do not take into account space optimization. Consequently, the vertices are not effectively distributed in space. The second drawback is time consumption. Indeed, these algorithms are based on the principle that every pair of vertices exerts repulsive forces but also that attractive forces are only calculated between neighbors. Before obtaining the final state of the system, several iterations are computed and the positions of the vertices are modified. Given that each iteration is computed in a time of the order of $O(n^2+m)$, n being the total number of vertices and m the total number of edges, this poses a major problem in drawing complex graphs formed by thousands of vertices.

In order to improve quality of drawing and to reduce the total execution time for large graphs, several methods were combined with force-directed algorithms. [Wal03a] presented a multilevel technique based on the partitioning of a graph into many sub-graphs. Thus the algorithm starts with the construction of the smallest sub-graph, improves the positions of vertices with the force-directed placement and uses the obtained result in the construction of the next sub-graph. Similarly, [Gaj04a] has recently combined a new technique called MIS (Maximal Independent Set) with the force-directed placement to draw large graphs. MIS consists in finding a partition of vertices set in many independent subsets. A subset is independent if no pair of elements is connected by an edge.

The major advantage of these techniques is the reduction of execution time since the force-directed algorithm is only applied to the improvement of vertices' positions in a subset at any given time. We only quote here a few techniques but the idea of partitioning large graphs in many sub-graphs had already been introduced by Fruchterman & Reingold with their multi-grid technique. In the same way the idea of partitioning into multi-layer was also used by [Dav96a] and [Har02a].

Initially, the majority of force-directed placement algorithms are applied in two dimensional layout but several studies have proved that three dimensional layouts are visually better and have several advantages. [Dwy01a] confirmed this result in his study on 3D visualization of UML class diagrams using force-directed placement. [Chu01a] propose virtual worlds which can allow users to comprehend large graphs. They have developed a platform for experimenting with 3D force-directed placement algorithms.

3. OUR ALGORITHM

3.1 Overview

The algorithm that we propose in this paper is addressed to 3D general undirected graph-drawing. We will later show that it is also suited to the drawing of large trees. Our algorithm is based on the approach of Kamada & Kawai that consists in solving differential equations to minimize total energy. Indeed, the process for finding the final positions of vertices by simulating attractive and repulsive forces can rarely minimize total energy. This is why each algorithm based on this method gives its own energy function that is never expressed explicitly. Instead of using the partitioning method like recent algorithms, we are interested rather in optimizing the layout by considering each graph as complete, i.e. each vertex is related to other vertices by n-1 links, n being the total number of vertices. This produces a better representation of less complex graphs. Indeed, the idea of partitioning a graph into independent subsets is not very reliable in the sense that it is not always applicable to complete graphs that are impossible to subdivide into independent subsets. The optimization approach that we propose consists in subdividing the largest sphere inscribed in space into several smaller spheres of the same dimension and containing one vertex. The positions of vertices are improved by force-directed placement in order to minimize the unused space in the initial sphere.

The main features of the algorithm are then:

- an equitable distribution of vertices,
- no collision between vertices,
- a minimization of unused space,
- an effective energy function.

3.2 Vertices distribution approach

The key feature of this approach is to build the largest sphere inscribed in the 3D space and to equitably divide it for the various vertices. Let S_0 be the initial sphere, n vertices are distributed inside according to the following conditions:

- each vertex i is at the centre of the partition S_i, where S_i is a sphere,
- for all vertex i and vertex j, the partition S_i has the same size as the partition S_j, i.e. volume(S_i)=volume(S_j).

For this, we have to maximize the volume of each sphere S_i, i.e. to minimize the unused space. Maximizing the volume of each sphere S_i comes down to finding the final position of each vertex minimizing the total energy of the corresponding spring system.

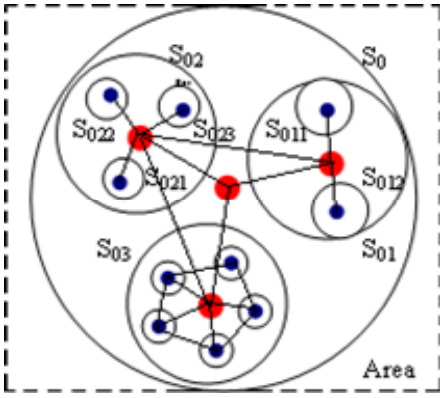


Figure 1: Vertices distribution.

The figure above shows the process of vertices distribution. Colored objects placed in the centre of spheres represent vertices. This is an example of a graph that mixes both a tree and a general undirected graph. Positioning vertices of undirected graphs is easier because the initial sphere is directly subdivided into the various vertices. The only condition to satisfy is that all spheres must have the same volume.

Space optimization is the main interest of the recent algorithm developed by [Ngu02a] for the 2D representation of large hierarchical systems. Their algorithm is very effective because, on a small surface, it lays out thousands of nodes structured in a tree. Unfortunately, it does not apply to topologies structured differently such as complete or general undirected graphs.

3.3 Attractive forces, repulsive forces and total energy

In order to minimize edge crossings and refine the drawing, the principle of force-directed placement is slightly deviated from the physical reality and two forces are introduced: repulsive forces are applied to every pair of vertices and attractive forces that are only applied to adjacent vertices so that they are closer to each other. Thus, the final state of the system is found when the sum of the forces applied is null or minimized down to a chosen value.

Our algorithm is always based on this principle but the main difference resides in the expression of attractive forces. Indeed, in order to optimize the available embedding area and to avoid positioning vertices outside a current sphere, the attractive forces are applied to bring the elements towards the centre. The problem is then modeled as follows:

- the vertices exert repulsive forces towards each other,
- each vertex is attracted by the centre of the sphere (S).

 ISBN 978-86943-15-2

In this way, the repulsive force generated by two vertices is inversely proportional to the distance between them, i.e. the closer to each other two vertices are, the more important repulsive forces they exert. Contrarily, the attractive force is proportional to the distance between a vertex and the centre of the current sphere so it is more important when a vertex is further away from the centre. The expression of the forces applied to a vertex is:

$$\boldsymbol{F}_i = \boldsymbol{F}_i^r + \boldsymbol{F}_i^a \qquad (1)$$

with

$$\boldsymbol{F}_i^r = \sum_{j=1 and\, j\neq i}^{n} \left(Radius - Distance\left(P_i, P_j \right) \right)$$

and

$$\boldsymbol{F}_i^a = n * Distance\left(P_i, Centre\right)$$

n being the total number of vertices, Centre the centre of the sphere (S), $\boldsymbol{F}_i^r$ and $\boldsymbol{F}_i^a$ respectively repulsive and attractive forces applied to a vertex i. P_i and P_j are respectively the positions of vertex i and vertex j. Distance(P_i, P_j) is the Euclidian distance between P_i and P_j.

The final state of the system is found when

$$\sum_{i=1..n} \boldsymbol{F}_i = \sum_{i=1..n} (\boldsymbol{F}_i^r + \boldsymbol{F}_i^a) = 0 \qquad (2)$$

According to the Kamada & Kawai approach on which our algorithm is based, the total energy of the system is expressed by the following sum:

$$\boldsymbol{E} = \frac{1}{2} \sum_{i=1}^{n} \left(\begin{array}{l} \left(\sum_{j=1 and\, j\neq i}^{n} \left(Radius - Distance(P_i, P_j) \right)^2 \right) + \\ n\left(Distance(P_i, Centre) \right)^2 \end{array} \right) \qquad (3)$$

In terms of xyz-coordinates, the expression (3) is given by the formula (4):

$$\boldsymbol{E} = \frac{1}{2} \sum_{i=1}^{n} \left(\begin{array}{l} \sum_{j=1 and\, j\neq i}^{n} \left(\begin{array}{l} Radius^2 + (x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 - \\ 2Radius\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \end{array} \right) \\ + n(x_i - cx)^2 + n(y_i - cy)^2 + n(z_i - cz)^2 \end{array} \right)$$

$P_i(x_i, y_i, z_i)$ and Centre(cx, cy, cz) respectively being the position of vertex i and the centre of the largest sphere inscribed in space.

To find the final state of the system, we need to find the local minima of the energy E at each vertex. This means that we need to take partial derivatives of E and solve them in order to find local minima for each vertex. Let $P_i(x_i, y_i, z_i)$ be the initial position of vertex i. Therefore we need the new position $P'_i(x'_i, y'_i, z'_i)$ which locally minimizes energy E, where $x'_i = x_i+tx_i$, $y'_i = y_i+ty_i$, $z'_i = z_i+tz_i$. In other words, we need the translation $T_i(tx_i, ty_i, tz_i)$ of each vertex i which minimizes energy E. By introducing the expression of (x'_i, y'_i, z'_i) we get the formula (5):

$$\boldsymbol{E} = \frac{1}{2} \sum_{i=1}^{n} \left(\begin{array}{l} \sum_{j=1 and\, j\neq i}^{n} \left(\begin{array}{l} Radius^2 + (x'_i - x'_j)^2 + (y'_i - y'_j)^2 + (z'_i - z'_j)^2 \\ -2Radius\sqrt{(x'_i - x'_j)^2 + (y'_i - y'_j)^2 + (z'_i - z'_j)^2} \end{array} \right) \\ + n(x'_i - cx)^2 + n(y'_i - cy)^2 + n(z'_i - cz)^2 \end{array} \right)$$

3.4 Minimizing the total energy

The common problem of force-directed placement algorithms is to find an efficient energy function *(cost function*) to reduce the total energy of the system. Thus, we need to find for each vertex the position that stabilizes the system. Our method consists in finding the minimum energy using the conjugated gradient of Fletcher-Reeves optimization method. This method has been already used in a Java-based experimental platform [Dan98a] but repulsion forces are computed in O(n log n) using Barnes-Hut tree-code [Bar86].

The gradient of the sum E (5) applied to a vertex i is given by the following expression:

$$\left\{ \begin{array}{l} \frac{\partial E}{\partial tx_i} = \sum_{j=1 and\, j\neq i}^{n} \left((X_i - X_j) - Radius \frac{X_i - X_j}{\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2 + (Z_i - Z_j)^2}} \right) \\ + n(X_i - X_j) \\ \frac{\partial E}{\partial ty_i} = \sum_{j=1 and\, j\neq i}^{n} \left((Y_i - Y_j) - Radius \frac{Y_i - Y_j}{\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2 + (Z_i - Z_j)^2}} \right) \\ + n(Y_i - Y_j) \\ \frac{\partial E}{\partial tz_i} = \sum_{j=1 and\, j\neq i}^{n} \left((Z_i - Z_j) - Radius \frac{Z_i - Z_j}{\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2 + (Z_i - Z_j)^2}} \right) \\ + n(Z_i - Z_j) \end{array} \right.$$

Where $X_i = x_i+tx_i$, $X_j = x_j+tx_j$, $Y_i = y_i+ty_i$, $Y_j= y_j+ty_j$, $Z_i = z_i+tz_i$, $Z_j = z_j+tz_j$

As we deal with n vertices, the sum E (5) results in 2n non-linear and non-independent equations. Therefore, we find the vertex in the system with the highest energy and move it to a location minimizing

its energy, i.e. $\frac{\partial E}{\partial tx_i} = \frac{\partial E}{\partial ty_i} = \frac{\partial E}{\partial tz_i} = 0$. This process is repeated until the local energy for each vertex stops decreasing. This is summarized as follows:

- compute the gradient of E for all tx_1, ty_1, tz_1, tx_2, ty_2, tz_2, …, tx_n, ty_n, tz_n
- compute the norms of translations $\Delta_i = \sqrt{\left(\frac{\partial E}{\partial tx_i}\right)^2 + \left(\frac{\partial E}{\partial ty_i}\right)^2 + \left(\frac{\partial E}{\partial tz_i}\right)^2}$,
- identify the vertex m with the highest norm of translation ($\Delta_m \geq \Delta_i \; i = 1..n$),
- minimize local energy E for this vertex through the conjugated gradient with tx_m, ty_m, tz_m as parameters and keep other vertices at their current positions,
- after minimization, vertex m has $T_m(tx_m, ty_m, tz_m)$ as translation, is moved to its new position $P_m(x_m+tx_m, y_m+ty_m, z_m+tz_m)$, and the conjugated gradients of E for all tx_1, ty_1, tz_1, tx_2, ty_2, tz_2, …, tx_n, ty_n, tz_n are recalculated. These steps are stopped when all norms of translations are lower than a value: $\Delta_i < \varepsilon \; for \; i = 1..n$, i.e. the final state of the system is found.

In order to improve the aesthetic of the layout, we have introduced in the expression of the energy the graph distance between two vertices which is the shortest path between them in the graph. This allows closer positioning of two neighboring vertices. Let d_{ij} be the graph distance between two vertices O_i and O_j. Therefore, the expression of energy E (5) is given by the following sum (6):

$$E = \frac{1}{2}\left(\sum_{i=1}^{n}\left(\sum_{j=1 \, and \, j\neq i}^{n}\begin{pmatrix}(d_{ij} * Radius)^2 + X_{ij}^{\,2} + Y_{ij}^{\,2} + Z_{ij}^{\,2} \\ -2 * Radius * d_{ij}\sqrt{X_{ij}^{\,2} + Y_{ij}^{\,2} + Z_{ij}^{\,2}} \\ + n(x_i - cx)^2 + n(y_i - cy)^2 + n(z_i - cz)^2\end{pmatrix}\right)\right)$$

where $X_{ij} = x_i - x_j$, $Y_{ij} = y_i - y_j$, $Z_{ij} = z_i - z_j$.

The algorithm is then summarized by the following pseudo-code:

Minimization algorithm of total energy E

```
Assign random placement of vertices:
initialize (xi, yi, zi), i=1…n
Initialize the translations to zero
(txi=0, tyi=0, tzi=0), i=1...n
Compute the gradients of E for all (txi,
tyi, tzi) and compute Δi with i=1...n
Find the vertex m with Δm ≥ Δi , i=1...n
While Δm > ε
```

$$grad \leftarrow \left(\frac{\partial E}{\partial tx_m}, \frac{\partial E}{\partial ty_m}, \frac{\partial E}{\partial tz_m}\right)$$

```
Assign the descent direction:
u(ux, uy, uz) ← -grad
While ‖u‖ > ε
```

$$s \leftarrow \|grad\|^2$$

```
Compute the optimal step p in the
direction u, while observing the
different conditions (applying the
translation vector, vertex m must
not be outside the available area
(S) and it must not be in
collision with the other vertices)
Update the translation vector with
(txm, tym, tzm) ← (txm+p*ux, tym+p*uy,
tzm+p*uz).
Compute the gradient of E for txm,
tym, tzm,
```

$$grad \leftarrow \left(\frac{\partial E}{\partial tx_m}, \frac{\partial E}{\partial ty_m}, \frac{\partial E}{\partial tz_m}\right)$$

$$s \leftarrow \frac{\|grad\|^2}{s}$$

$$u(u_x, u_y, u_z) \leftarrow -grad + s * u(u_x, u_y, u_z)$$

```
End While
Compute the gradient of E for all
```

tx_i, ty_i, tz_i and the Δ_i with i=1...n

```
Find vertex m where Δm ≥ Δi , i=1...n
End While
```

To compute the optimal step p, it is possible to use minimization by "quadratic interpolation" or by section in "Golden Search" or by the "one-dimensional Newton method".

4. APPLICATIONS AND RESULTS

We have developed a prototype tool using the Java programming language and Java3D class library.

The following screenshots show the capacity of our algorithm to identify the main interest of a complex system by highlighting the global view and to draw the hierarchical structure.

Our algorithm has the same scalability problem as the other force-directed layout techniques. An improvement will be added in the future to make the algorithm more scalable.

In practice, a complex system is rarely structured as a complete graph. It is not a pure tree either. Therefore it is sometimes possible to create a partition of the set of vertices according to their weights. Here, we mean by weight the number of edges incident from a vertex. We simply use the partition of the graph when it is possible and necessary to show the structure and to improve the layout, but it is not the main interest of our method.

By grouping the vertices according to their weight, the graph can be subdivided into several hierarchies as follows: H_1:{O_1, O_2, O_3,…,O_n}, H_2: {{O_{11},O_{12},…},{O_{21},O_{22},…},…,{O_{h1}, O_{h2},…}}, H_3, H_4, …

Once vertices are hierarchically grouped, the first hierarchy, here H_1, is then given an initial layout. Its elements are distributed in the initial sphere (S) and their positions are refined using the force-directed method. After this step, each vertex O_i of H_1 is placed at the centre of a sphere S_i. The next step starts by distributing in the sphere S_i the vertices (O_{i1}, O_{i2}, ...) considered as the sub-hierarchy of O_i and the force-directed method is then applied. This process is recursively applied through all the hierarchies until positioning all vertices.

In the case where it is impossible to subdivide the graph, vertices are directly distributed in the largest sphere inscribed in the embedding area.

This process applies easily to trees because we just need to place the root vertex at the centre of the initial sphere and to recursively subdivide this sphere through the remaining hierarchies until the leaf vertices.

The six following figures represent the general structures of large systems, i.e. the combination of a tree and a general graph. Figure 1 shows how fifteen vertices are placed equitably in the area and figure 2 represents them with maximum links, i.e. a vertex is linked to every other ones. Figure 3, 4 and 5 show the typical case of a mixture of trees and general graphs. Figure 6 specifically represents a pure tree rooted from the vertex placed at the centre of the area. The embedding area is represented by the xyz axes and colored in grey. We have assigned different colors to vertices in order to highlight hierarchical levels in the graph.

We have developed a simple user interface as a test bed for our algorithm. The number of nodes on each level of a graph can be specified to test the rapidity of the drawing. Thus it is easy to build all types of graph and analyze the strength of the algorithm. The three standard ways of navigation are also added to allow the study of a graph. All the details of a complex graph can then be shown by zooming on in, rotating or translating the display.

Our algorithm does not take into account the effects of edges; they are just used to calculate the weight of each vertex and only plotted after positioning all vertices.

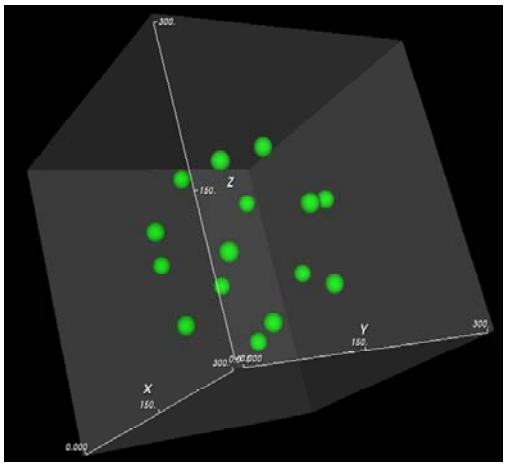


Figure 1: 15 vertices

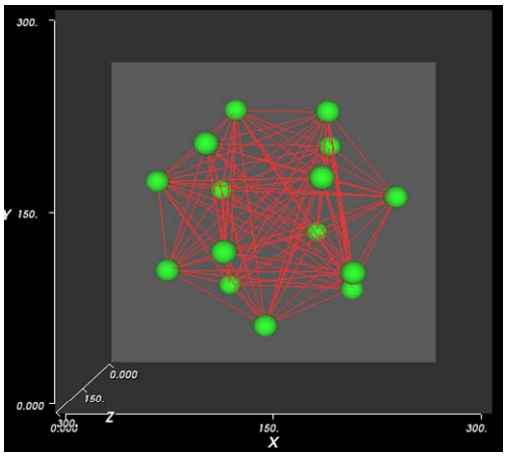


Figure 2: 15 linked vertices

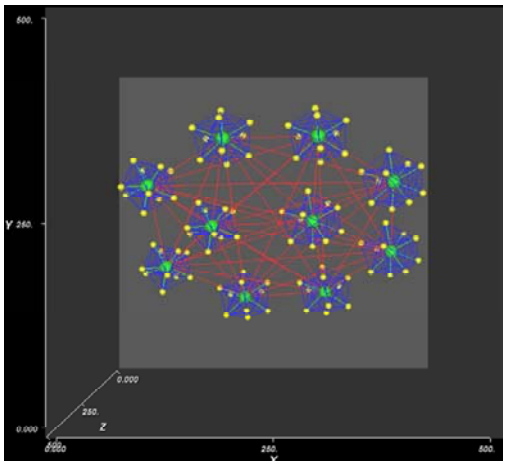


Figure 3: 110 vertices

 ISBN 978-86943-15-2

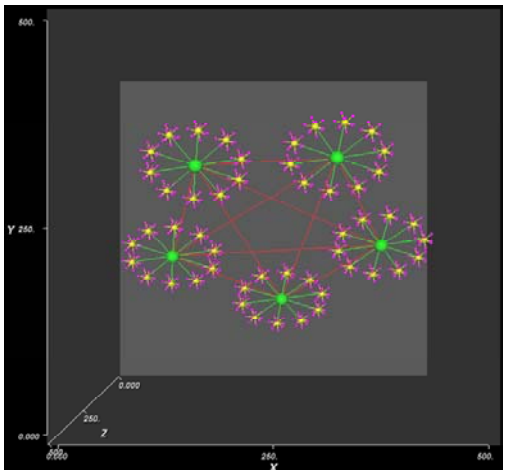


Figure 4: 550 vertices

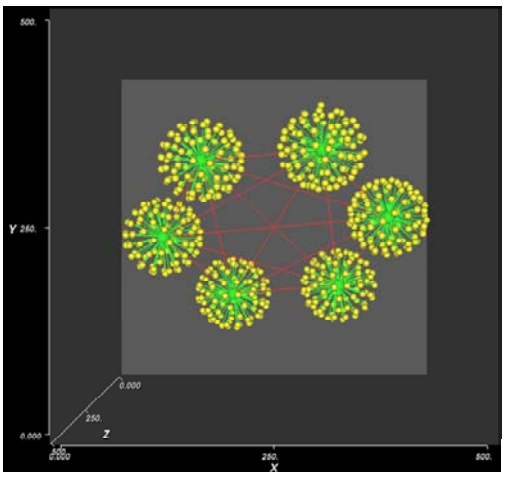


Figure 5: 606 vertices

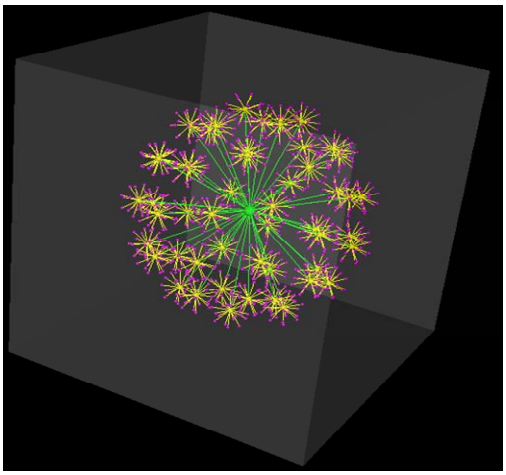


Figure 6: 1051 vertices

Figures 3, 4 and 5 show the typical case of a complete graph. They respectively contain 110, 550 and 606 nodes and the drawing took respectively 0.30, 1.30 and 2.0 seconds on a laptop with a 1Gh AMD processor.

Generating the layout of a tree is very fast because the initial sphere containing the route node is successively subdivided into different branches. Figure 6 for example represents 1051 nodes and is computed in around 2.0 seconds.

Compared with different force-directed algorithms our algorithm can draw a large graph in a reasonable time because it is based on the natural mechanism that consists in associating with a hierarchical of graphs and the drawing starts by the smallest graph in the hierarchy and drawing larger and larger graphs. The second advantage of our method is the use of a numerical optimization while computing the displacement of a vertex.

5. CONCLUSIONS

We have presented a new method to optimize the force-directed placement algorithm. The conjugated gradient of Fletcher-Reeves is implemented to perform the numerical optimization. Our method may not be the fastest amongst the force-directed placement algorithms but its main advantage is its effectiveness in minimizing total energy while most algorithms based on this method fail and do not clearly express their cost function. Our algorithm is addressed to three dimensional drawings of general undirected graphs and trees.

To avoid collisions between vertices we consider the largest sphere inscribed in space and subdivide it into smaller spheres, each containing a vertex. The unused space is minimized by improving the position of each vertex with the force-directed method.

According to the structure of a graph it is possible to associate a partition method in order to improve the layout.

Our next goal consists in improving the presentation by introducing the notion of weighting when assigning space to vertices. Indeed, some vertices have more descendents than others. The size of the area assigned to each vertex must then be proportional to the number of its descendents. Therefore, the presentation will be much more interesting when introducing the vertices' weight into the energy function.

Our algorithm can be used in many computer science fields such as large-scale computing network, telecommunication networks' display, but we are going to apply it to our project of large computing networks 3D visualization.

6. REFERENCES

[Bar86] J.Barnes and T.Hut, A Hierarchical O(N log N) force-calculation algorithm, Nature, vol.324, pp.446-449, 1986.

[Bat98a] G. Di Battista, P. Eades, R.Tamassia and I. G. Tollis, 1998. Graph Drawing: Algorithms for the Visualization of Graphs, Prentice-Hall, Englewood Cliffs.

[Chu01a] N. Churcher, Alan Creek, 2001. Building Virtual Worlds with the Bing-Bang Model. *Australian Symposium on Information Visualization*, Sydney.

[Dan98a] Daniel Tunkelang, 1998. JIGGLE: Java Interactive Graph Layout Environment. In Whitesides, Sue H., Eds. In *Proceedings Graph Drawing*, pages pp. 413-422, Montréal, Canada

[Dav96a] R. Davidson, D. Harel, 1996. Graph Drawing Nicely using simulated Annealing. *ACM Trans. Graphics*, pp. 301-331.

[Dwy01a] T. Dwyer, 2001. Three Dimensional UML Using Force Directed Layout. *Australian Symposium on Information Visualization*, Sydney.

[Ead84a] P. Eades, 1984. A Heuristic for Graph Drawing. *Congressus Numerantium*, pp. 149-160.

[Fle64a] R. Fletcher and C.M Reeves, 1964. Function minimization by conjugate gradients. In *Computer Journal 7,* pp. 149-154.N.

[Fru91a] T. Fruchterman, E. Reingold, 1991. Graph drawing by force-directed placement. *Software. – Practice and Experience*, pp. 1129–1164.

[Gaj04a] P. Gajer, M. T. Goodrich, S. G. Kobourov, 2004. A multi-dimensional approach to force-directed layouts of large graphs. *Computational Geometry, Theory and Applications*, pp. 3-18.

[Gil90a] J. C. Gilbert, Nocedal J, 1990. Global convergence properties of Conjugated Gradient methods for optimization. *Research Report*. National Institute of Computer Science and Automatic (INRIA).

[Har02a] D. Harel, Y. Koren, 2002. A Fast Multi-Scale Algorithm for Drawing Large Graphs. *Journal of Graph Algorithms and Applications*, pp. 179-202.

[Kam89a] T. Kamada, S. Kawai, 1989. An algorithm for drawing general undirected graphs. In *Information Processing Letters*, 31, pp. 7–15.

[Ngu02a] Q. V. Nguyen, M. L. Huang, 2002. Using space- optimized tree visualization for web site-mapping. *International Conference on Internet Computing* (IC'02), Las Vegas, Nevada, USA, pp. 622-628.

[Wal03a] C. Walshaw, 2003. A Multilevel Algorithm for Force-Directed Graph Drawing. *Journal of Graph Algorithms and Applications*, http://jgaa.info/vol. 7 no. 3, pp. 253-285.

[Wan05a] C.Y Wang, M.X. Li, 2005. Convergence property of the Fletcher-Reeves Conjugated Gradient method with errors. In *Journal of Industrial and management optimization*, Volume 1, Number 2.

Comparative Evaluation of Random Forest and Fern Classifiers for Real-Time Feature Matching

Iñigo Barandiaran[1], Charlote Cottez[1],Céline Paloc[1], Manuel Graña[2]

[1]VICOMTech Paseo Mikeletegi, 57
20009, San Sebastian, Spain
{ibarandiaran, cottez,cpaloc} @vicomtech.org

[2] University of Basque Country
Computer Science School, Pº. Manuel de Lardizabal, 1
20009, San Sebastián, Spain
ccpgrrom@si.ehu.es

ABSTRACT

Feature or keypoint matching is a critical task in many computer vision applications, such as optical 3D reconstruction or optical markerless tracking. These applications demand very accurate and fast matching techniques. We present an evaluation and comparison of two keypoint matching strategies based on supervised classification for markerless tracking of planar surfaces. We have applied these approaches on an augmented reality prototype for indoor and outdoor design review.

Keywords

Feature matching, Tracking by Detection, Augmented Reality.

.

1. INTRODUCTION

The main goal of the Augmented Reality (AR) technology is to add computer-generated information (2D/3D) to a real video sequence in such a manner that real and virtual objects coexist in the same world. In order to get a realistic illusion, the registration problem must be addressed. Real and virtual objects must be properly aligned with respect to each other. In this way, the position-orientation (pose) of the camera relative to a reference frame must be accurately estimated or updated over time. One of the key task for the registration problem to be solved is the keypoint or feature matching. Optimal markerless tracking uses natural features such as edges or corners extracted from images. In this work, we address the registration problem for interactive AR applications, using tracking by detection techniques based on supervised classification techniques (see Fig.2).

Our approach to solve the registration problem is based on tracking of plane surfaces. In indoor or outdoor scenarios, planes are commonly present. The ground, the building facades or walls can be seen as planes. These 3D world planes and its projection in the image are related by homography transformation. By recovering this transformation it is possible to estimate the position and orientation (pose) of the camera.

Keypoint matching is a critical task in many computer vision applications, such as optical 3D reconstruction or optical markerless tracking. As described in [Lep06], we propose to treat wide base-line matching of features points as a classification problem. We have implemented a Random Forest classifier [Bre01] and a semi-Naïve Bayes classifier [Özu07], and carried out an evaluation of both in the context of optical markerless tracking for Augmented Reality applications.

The article is structured as follows. Section 2 gives an overview of current optical tracking techniques and methods in augmented reality applications. Section 3 describes two approaches based on supervised classification. Section 4 gives an overview of the behavior of both approaches. In Section 5 a case study augmented reality application using our implementations is described. Section 6 summarizes some conclusions and future work.

2. RELATED WORK

Although the real-time registration problem using computer vision techniques has received a lot of attention during recent years, it is still far from being solved. Ideally, an augmented reality application should work without the need of landmarks or references for the object or the environment to be tracked. This issue is known as markerless tracking.

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

We can divide optical markerless tracking technology in two main groups: recursive techniques or model-based techniques. Recursive techniques start the tracking process from an initial guess or a rough estimation, and then refine or update it over time. They are called recursive because they use the previous estimation to propagate or calculate the next estimation. During the estimation process several errors may occur, such as wrong point matching or ill conditioned data that can degenerate the estimation. Due to the recursive nature of this kind of tracking, they are highly prone to error accumulation. The error accumulation over time may induce a tracking failure that requires a re-initialization of the tracking process, which can be cumbersome and not feasible in practical applications.

Other approaches are known as tracking by detection or model-based tracking. In this kind of techniques some information of the environment or the object to be tracked is known a priori. They are also known as model-based tracking because the identification of some features in the images (texture patches or corners) corresponding to a known model are used to recognize such objects. This kind of tracking does not suffer from error accumulation (drift) because, in general, does not rely on the past. Furthermore, they are able to recover from a tracking failure since they are based on a frame by frame detection not depending on the past. They can handle problems such as matching errors or partial occlusion, being able to recover from tracking failure without intervention [Wil07].

Tracking by detection needs information data about the object or objects to be tracked prior to the tracking process itself. This data can be in the form of a list of 3D edges (CAD model) [Vac04], color features, texture patches or point descriptors [Low01]. A good comparison about different point descriptors can be found in [Mik05]. The tracker is trained with a priori data, to be able to recognize the object from different points of view. A good survey about different model-based tracking approaches can be found in [Lep05].

Some authors propose the use of machine learning techniques to solve the problem of wide baseline keypoint matching [Özu06]. Supervised classification systems require a previous pre-processing, in which a system "is trained" with a determined set of known examples (training set) that present variations in all their independent variables. Once the process is finished, the system is trained and ready to classify new examples. Some of the most widely used supervised classifiers are for example, k-Nearest Neighbors, Support Vector Machine or decision trees.

While k-Nearest Neighbors or Support Vector Machine can achieve good classification results, they are still too slow and therefore not suitable for real-time operation [Lep04]. Recently the approach based on decision trees has been successfully applied on tracking by detection during feature point matching task (see Fig.1), by training the classifier to establish correspondences between detected features in a training image and those in input frames [Özu06]. This approach has been also applied to object recognition in [Moo06].

Figure 1: Feature Points recognition and matching.

Figure 2: Example of an outdoor augmented reality scene (San Telmo square in San Sebastian, Spain.)

Based on this recent progress in the field, we integrate two supervised classifiers in the implementation of a tracking module and carry out some evaluation studies.

3. DESCRIPTION

In this section we briefly describe two approaches for feature point recognition based on supervised classifiers that have been recently applied in some real-time implementations [Bof06, Özu07, Wil07]. In these approaches the problem of wide-baseline matching problem is treated as a classification problem.

Random Forest

Random Forest classifiers were firstly introduced by [Ami97]. Recently, such classifier has been applied in optical tracking for interest point matching and recognition [Lep06]. It is able to detect key-point occurrences even in the presence of image noise, variations in scale, orientation and illumination changes. This classifier is a specific variation of a decision tree[Bre01].

When the tree is constructed and trained, it classifies a given data (example) by pushing it down the tree. While the data is going down the tree, a discriminant criteria at every node is deciding to which child the example has to go.

A Random Tree is called random because instead of doing exhaustive search for the best combination of features to be tested at each node, only some random combinations are used. When the number of classes to be recognized and the size of the class descriptor are high, an exhaustive analysis is not feasible. In addition, the examples that are going to be used during the training process are selected at random from the available ones. The combination of some random trees forms a multi-classifier known as Random Forest. One of the advantages of the Random Forest is their combinational behavior. Even when a random tree is poor with a low recognition rate, the combination with other classifiers can generate an efficient one.

3.1.1 Random Forest Training

In supervised classification each class must be defined and labeled before the training process itself. In order to define the classes, we use a point extractor [Rosten06] to get the candidate points and their surrounding patches. Then the classifier assigns a class number to each patch, and their class descriptor is defined. The descriptor of each class is constructed as the intensity values of the pixels that forms the extracted patch centered at interest point *p* (see Fig.3).

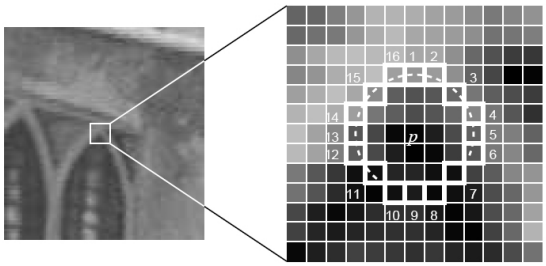


Figure 3: (left) Interest point. (Right) Pixels surrounding interest point *p*. [Ros06].

Once the classes to be recognized by the classifier are defined, the training set must be generated.

As described in [Lepe06], we can exploit the assumption that the patches belong to a planar surface, so we can synthesize many new views of the patches using warping techniques as affine deformations(see Fig.4).These affine transformations are used to allow the classifier to identify or recognize the same class but seen from different points of view and at different scales. This step is particularly important for tracking, where the camera will be freely moving and rotating in space.

Synthetically generated views will allow to build up new training and test data sets, that will be used during classifier training and testing steps.

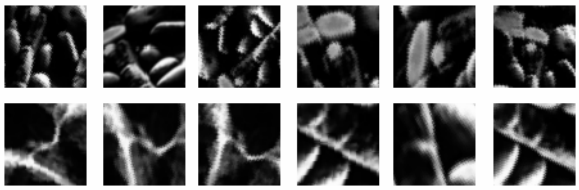


Figure 4: Randomly generated training examples of four classes by applying affine transformations.

Once the training set is ready, the training task can be performed. During this task, a number of examples are randomly selected from the available ones. These examples are pushed down in the trees. In order to decrease the correlation between trees, and therefore increase the strength of the classifier, different examples from the training set must be pushed down in each tree. This randomness injection favors the minimization of trees correlation.

While building up the tree, each node of the tree is treated as follows:

- N training examples from the training set are in the node.
- S random sets of *n* pixels are selected.
- For each set, its information gain [Brei01] is calculated.
- The variable set with the greatest information gain value is selected.
- The examples are tested with the selected set of pixels. Depending on the result of this test, they are pushed down to their corresponding child node.
- The above process is recursively done for the children nodes, whether until there is only one example, or only one class is represented in the remaining examples or the maximal predefined depth is reached (see Fig.4).

Once the descriptors reach the bottom (maximal depth) of the tree, it is said that they reached a leaf node and the recursion stops. In leaf nodes the class posterior distributions are stored. These distributions represent the number of class examples from the

training set that has reached that node. Once an example of a given class reaches a leaf node, the posterior probability distribution stored in that node must be updated accordingly.

The tests to be performed in each node can be simply binary tests, based on the comparison of the intensity values of two pixels, as:

$$T = \begin{cases} 1 & if\left(v(p_1) - v(p_2)\right) \geq r \\ 0 & otherwise \end{cases}$$

Where $v(p_1)$ and $v(p_2)$ represent the intensity values of two pixels located at positions p_1 and p_2 respectively. The values of p_1, p_2, i.e, the pixel locations to be tested are randomly selected during the training step. The value of *r* represents a threshold that was also randomly selected while training. In our experiments, we select a random value for *r* between 0 and 25. We have also experiment that, given the weakness of the tests, smoothing every patch before training and classification, significantly increases the final reliability of the classification.

Figure 4: Random Tree construction example.

3.1.2 Classification

Once the classifier is built, i.e, the pixels to be tested in each node and the class posterior distributions are calculated, it is ready to classify new examples different from the ones in the training set. During the classification task any new example is dropped down in every tree that constitutes the forest. These examples will reach a leaf node depending on the results of the tests obtained in the previous nodes they visit (see Fig. 3). The posterior distributions stored in leaf nodes are used to assign a class probability value to the examples that reach that node, $P(Y = c \mid T_1 = T_l, n = \eta)$ where T_l is a given tree of the forest and η is the reached node by the example (patch) Y and c is the assigned class label.

As any multi-classifier, the random forest needs to combine the independently generated output by each tree in the forest, in order to assign a final class label to the examples to be classified.

Depending on the number of classes to be trained, the output of the trees during classification can reach a very low value. When combining these outputs among the trees, the final value can be close to zero because of floating point precision (underflow) [Alk00]. Therefore, instead of using the addition or the product rules for classifier combination, we apply the addition of the logarithms of the posteriors to generate the final class label.

Ferns

Random Ferns like Random Forest is also a multi-classifier, compound of a determined number of entities or classifiers. This approach was firstly introduced in [Özu07], and it was also independently proposed in [Wil07]. In opposite to Random Forest, Ferns is a non-hierarchical structure where each entity that constitutes the multi-classifier is basically a set of tests. In Random Forest the test set of each tree is the collection of different tests that are distributed along the nodes that forms the tree. Due to the flat structure of every entity in a Ferns classifier the test set is a simple ordered list of positions of pixels to be evaluated.

3.1.3 Training

As in Random Forest, we use the interest point extractor proposed in [Rosten06] to get the candidate points and their surrounding patches. The extracted patches will be the classes to be recognized by the classifier.

Due to the non-hierarchical structure of the classifier, during the training step not information gain is calculated. Therefore, in contrast to Random Forest where only the data that falls in a child node is taken into account in the test, in Ferns classifiers the test set (the pixels to be evaluated in each example) is applied to the whole training data set.

Instead of saving the posterior distributions over the classes in leaf nodes, Ferns classifiers build up a look-up table where those distributions are stored [Özu07]. This look-up table is a *N*x*M* matrix where *N* represents the number of different classes that are going to be classified by the classifier and *M* the number of possible outputs given a test set. The number of columns, *M*, is calculated as 2^p where *p* is the number of tests that forms a test set. The result of each individual test and the ordering on the set defines a binary code that, interpreted as a decimal number, can be used as an index for the look-up table to get the corresponding posterior probability distribution of the class. Access to a position in the look-up table is similar to reach a leave node in the Random Forest approach. The class posterior distributions are computed as in Random Forest.

The process for building up the training data set remains the same as the one proposed for Random Forest.

3.1.4 Classification

During classification, the examples are tested by the whole entities that form the classifier. In every entity the complete test set is evaluated given an example. This test generates the binary code that allows the classifier to access a position in the table (column) and recover the posterior probability stored for each class (row).

As Random Forest does, the outputs of each entity that forms the Ferns classifier must be combined to obtain the final class label for the example. During combination numerical errors (underflow) may occur, so we use the addition of logarithms, instead of addition or product combination rules.

4. EVALUATION

We have implemented our own API to evaluate the influence of different factors on the behavior of Random Forest and Ferns classifiers during training period as well as during execution period. Depending on different factors such as, number of different classes, the number of classifiers, or the size of the training set, the point classification rate may vary.

Due to the random selection of features, all tests were carried out ten times and the results were averaged. The entire evaluation was conducted by using feature points extracted from well-textured images. Due to the weakness of tests performed in each step of both classifiers, well textured and non-symmetric textured images are needed to obtain accurate results (see Fig.5).

Figure 5: Image used for the evaluation

We have evaluated the performance of Random Forest and Ferns classifiers by using the same number of structures in both approaches. The same number of trees is tested against the same number of entities that constitutes a Ferns classifier. Moreover, the number of tests to be evaluated in each entity in the Ferns classifier is equal to the maximal allowable depth of every tree in the forest.

The following tests were all carried out by using the same training data set for both classifiers, in order to obtain consistent and reliable results.

Rotation Range

We are interested in the evaluation of the behavior of both approaches depending on the allowable rotation range to be applied to the patch examples during the training phase. During the generation of the training set each patch representing a class is transformed by applying it an affine transformation that approximates the homography. This affine transformation can be decomposed as a rotation matrix R and a scaling matrix S with parameters $[\lambda_1, \lambda_2]$. These affine transformations are generated by extracting values at random from uniformly distributed intervals for rotation and scale parameters.

Obtaining robustness against variation in rotation and scale is particularly important for tracking, where the camera will be freely moving around the object. In this way, the classifier must be trained to be able to recognize the classes (points) from different orientations and scales.

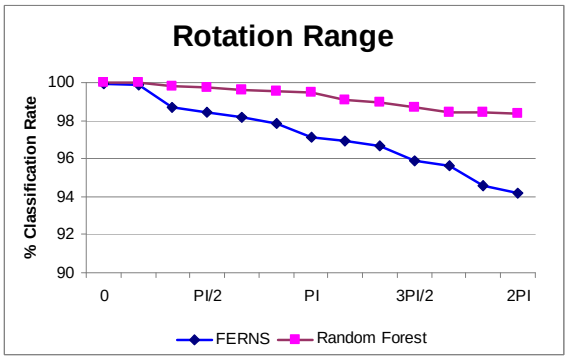


Figure 6: Rotation Range evaluation results.

Figure 6 shows the classification rate of Random Forest and Ferns classifiers trained with 225 different classes, no variation in scale and 325 examples per class. Both classifiers show similar behavior. In our test Random Forest classifier performs slightly better.

As expected, by increasing allowable rotation range with a fixed trained set size, the classification rate decreases.

Scale Range

As the rotation range test, the scale range test in intended to evaluate the robustness of the classifier against changes in scale. In this test we trained 225 different classes, with no variation in orientation and 325 examples per class. We define the scale range as follows: a value v of 1 means no variations in scale, a value (v <1) means a maximum possible reduction of scale of (1-v) percent, while value (v>1) means a maximum increase of scale of (1+v) percent. The following results were obtained by applying an

isotropic scaling to the examples in the training set, where the scaling matrix S of the affine transformation matrix is $\lambda_1 = \lambda_2$, with a randomly generated value, given a predefine range.

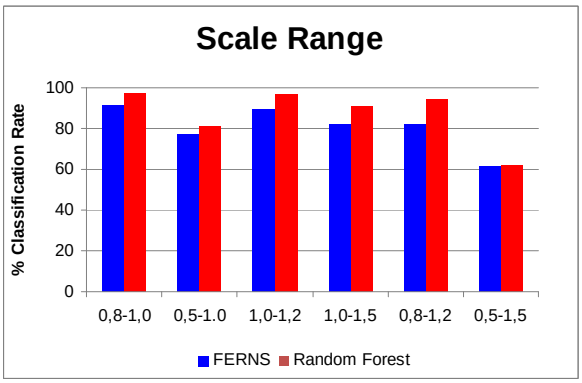


Figure 7: Scale range evaluation results.

We can conclude that the accuracy of the classifiers is drastically affected when severe changes in scale occur (see Fig.7). Also, while the range of scales is increased, the possible variations in appearance of every class also increase, spanning bigger subspaces. In order to be able to handle such subspaces the training data set must be increased accordingly.

Training Set

The training set size is a key factor during the training step of a supervised classifier. These classifiers are very dependent of the quality and size of the training set. The training set has a real influence in the final accuracy of the classifier. We have evaluated the behavior of Random Forest and Fern classifier by varying the size of the training set, i.e, the number of examples, while the number of classes remains constant.

The training sets are built with 225 different classes, where the examples were generated by applying random affine transformations with $[0,2\pi]$ allowable rotation range, and [0.5,1.5] scale range. We have evaluated the behavior of both classifiers by modifying the number of different examples per class.

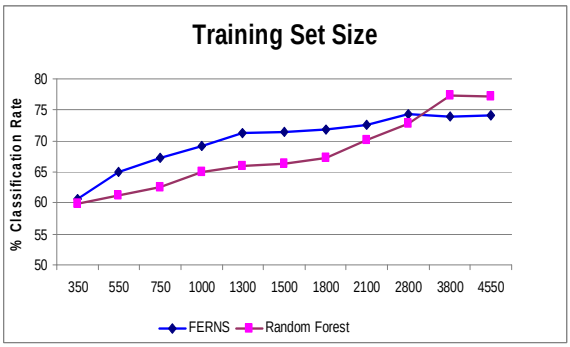


Figure 8: Training set size tests result.

Both classifiers get convergence when a number of examples per class are about 3500. At this value Random Forest classifier gets slightly better results reaching 78% of well recognized examples (see Fig.8).

Number of Classes

Feature or keypoint matching is a critical task in many computer vision applications, such as optical 3D reconstruction or optical markerless tracking. Related to tracking by detection techniques, is important that the tracker can cope with a large number of different points in order to get robustness against factors such as partial object occlusion or noise. In this way, we measured how the classifiers handle different number of classes, while the size of the training set remains fixed. More precisely, the following results were obtained by using 1000 examples per class, $[0,2\pi]$ allowable rotation range and [0.8,1.2] scale range.

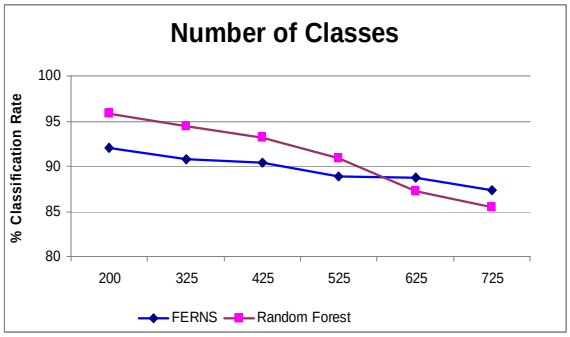


Figure 9: Number of different classes test result.

As expected, (see Fig.9) as the number of different classes increases the accuracy of both classifiers decrease. Our results show that Ferns classifier performs better than Random Trees being able to cope with a bigger number of different classes.

Discussion

Both classifiers perform well when the range of scale values is about [0.7,1.4]. We estimate that this range is sufficient to handle many views when tracking operation. When this range increases, the classification rate starts to decrease rapidly. Both approaches are robust against rotation changes when the training set size is large enough according to the number of supported classes.

When the classification rate is low, the number of miss-matched points (outliers) increases. Once the population of outliers is high, posterior processes related with tracking such as outlier filtering with RANSAC or non-linear minimization methods like Levenberg-Marquardt are seriously afected due to the number of iterations they need to reach convergence.

5. APPLICATION

The approaches described previously were applied within a prototype using head mounted displays (HMD) for collaborative mobile mixed reality design reviews. Figure 11 shows the markerless tracking module by using a camera attached to a laptop working outodoors, while figure 12 shows the same module but working indoors.

Our tracking module uses natural features to estimate the position and orientation of the camera, mounted on the HMD. Once this transformation is computed, the virtual object can be registered and viewed through the HMD as part of the real world. During the tracking process, the transformation must be updated over time.

By using automatically extracted natural features (see Fig.10), the use of artifacts such as reflective markers is avoided, allowing the system to be more flexible and being able to work in non-well controlled conditions, such as outdoor environments.

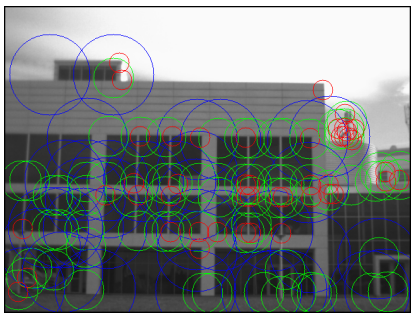


Figure 10: Feature or interest point extracted from a building facade to train a random forest classifier.

The internal camera parameters estimation task is performed only once, when the camera is to be used for the first time.

As described earlier, tracking by detection techniques requires an off-line process where the classifier is trained. During this period, one image of a highly textured plane, such as a building facade or a picture over a table, must be acquired. After the acquisition, some features points and their surrounding texture patches are extracted from the image [Ros06], and synthetic views of the plane are generated.

Based on the results described previously, the integrated classifier in the tracking module of the prototype is trained to be able to recognize about 200-250 different classes (interest points). The forest or Fern classifiers are constructed with 20 entities and a training set compound of 1000 synthetically generated examples for each class in less than 10 minutes. This size of the training set is a good compromise between training time and final accuracy of the classifier. Training time is a very important factor in practical situations such as outdoor setup preparation time. Once the training set is ready, the system is ready for tracking.

The obtained frame rate is about 20-25 frames per second (near real-time) on a 1.6Ghz dual core CPU. This frame rate may vary depending on the accuracy of the tracker, i.e, depending on the number of different points to be recognized. The drift and jitter are well controlled, so no severe movements of the objects occur. On a lower CPU (1Ghz) the obtained frame rate is only 5 frames per second.

Independently on the robustness of the classifier, the wrong classified points (outliers) are removed by using RANSAC. The final estimation is refined by using Levenberg-Marquardt non-linear minimization method using all the inlier points, starting from the estimation obtained by RANSAC. This final minimization is very useful to avoid virtual object jittering, what is an uncomfortable behavior during augmented reality scene visualization.

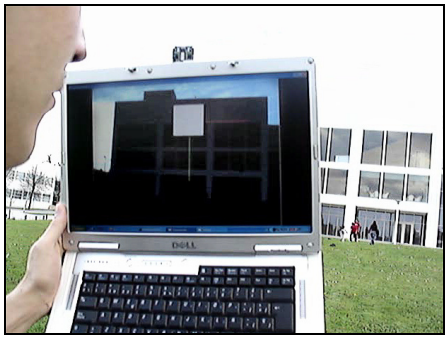


Figure 11: Outdoor Tracking of a building facade.

The tracking by detection approach based on feature point classification and matching allows the tracker to be robust against partial plane occlusion, or fast camera movement. The tracker can run indefinitely without requiring re-initialization.

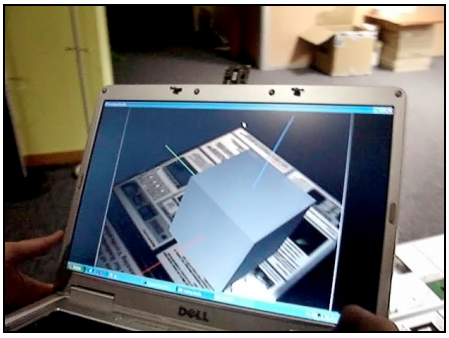


Figure 12: Indoor tracking of a textured floor.

6. CONCLUSION AND FUTURE WORK

In this work we have presented an approach of tracking by detection for plane homography estimation using the Random Forest based classifier

and Ferns classifier for interest point matching. A comparative evaluation of both approaches was carried out by analyzing their behavior while modifying important parameters, such as number of classes or training set size. Our results show that both approaches are very similar with no clear advantage of one approach over the other.

Also, an augmented reality prototype using these classifiers was described. The prototype is able to robustly track a plane even if partial plane occlusion occurs, at real-time frame rate.

We want to extend our work to support on-line training classification [Wil07]. On-line training allows the tracking to update the model with new feature points not present in the original training set. On-line training can be exploited in several frameworks such as Simultaneous Localization and Mapping (SLAM).

7. ACKNOWLEDGMENTS

This work has been partially funded under the 6th Framework Programme of the European Union within the IST project "IMPROVE" (IST FP6-004785, http://www.improve-eu.info/).

8. REFERENCES

[Alk00] Alkoot,F.M.,and Kittler, J. Improving the Performance of the Product Fusion Strategy. *In Proc. International Conference on Pattern Recognition*. Vol.(2), pages: 164-167, 2000.

[Ami97] Amit, Y. and Geman, D. Shape quantiation and recognition with randomized trees. *Neural Computation*, Vol.(9) pages: 1545-1588, 1997.

[Bof06] Boffy, Aurélien, Tsin, Yanghai and Genc, Yakup. Real-Time Feature Matching using Adaptative and Spatially Distributed Classification Trees. *In Proc. British Machine vision Conference*, Vol.(2) pages: 529-539, 2006.

[Bre01] Breiman, L. Random Forests. *Machine Learning Journal*, Vol. (45), pages: 5-32. ISSN 0885-6125, 2001.

[Lep04] Lepetit, V., Pilet, and J., Fua, P. Point Matching as a Classification Problem for Fast and Robust Object Pose Estimation. *In Proc. Conference on Computer Vision and Pattern Recognition*. ISBN: 0-7695-2158-4, 2004.

[Lep05] Lepetit, V., and Fua, P. Monocular model-based 3D object tracking of rigid objects: A survey. Foundations and Trends® in Computer Graphics and Vision., Vol.(1), pages 1–89, 2005.

[Lep06] Lepetit, V., and Fua, P. Keypoint Recognition Using Randomized Trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28(9), pages: 1465-1479. ISSN: 0162-8828, 2006.

[Low04] Lowe, D. Distinctive Image Features from Scale Invariants Keypoints. *International Journal of Computer Vision*. Vol. 20(2), pages: 91-110, 2004.

[Mik05] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Gool, L. V. A Comparison of Affine Region Detectors. *Int. Journal of Computer Vision.* Vol. 65(1-2), pages: 43-72. ISSN:0920-5691, 2005.

[Moo06] Moosmann, F., Triggs, B., and Jurie, F. Fast discriminative visual codebooks using randomized clustering forest, *NIPS*, 2006.

[Ros06] Rosten, E., and Drummond, T. Machine Learning for High-Speed Corner Detection. *In Proc. European Conference on Computer Vision*. Pages: 430- 443. ISBN 3540338322, 2006.

[Özu06] Özuysal, M., Fua, P., and Lepetit, V. Feature Harvesting for Tracking-By-Detection. In *Proc. European Conference on Computer Vision*, pages 592-605. ISBN:3-540-33836-5, 2006.

[Özu07] Özuylan, M, Fua. P, and Lepetit, V. Fast keypoint recognition in ten lines of code. *Computer Vision and Pattern Recognition*, pages: 1-8, ISBN-1-4244-1180-7, 2007.

[Vac04] Vacchetti, L., Lepetit, V., Fua, P. Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking. In *Proc. IEEE and AM International Symposium on Mixed and Augmented Reality*. Vol. (4), pages: 48-57. ISBN:0-7695-2191-6, 2004.

[Wil07] Williams, B., Klein, G. and Reid, I. Real-time SLAM Relocalisation. In *Proc. IEEE International Conference on Computer Vision*, 2007.

Human Skeletal and Muscle Deformation Animation Using Motion Capture Data

Ali Orkan Bayer
Department of Computer Engineering, Middle East Technical University
06531 Ankara, Turkey
orkan@ceng.metu.edu.tr

Ayça Müge Sevinç
Tubitak Space Techonologies Research Institute,
06531 Ankara, Turkey
muge.sevinc@ceng.metu.edu.tr

Tolga Can
Department of Computer Engineering, Middle East Technical University
06531 Ankara, Turkey
tcan@ceng.metu.edu.tr

ABSTRACT

Human character animation has problems of unrealistic motion and unrealistic skin deformation. In this study, we present an application of motion capture data for generating realistic human character animation using anatomically correct skeleton and muscle models. For this purpose, we attach a skeleton model to the motion data and then construct and attach a set of muscle models to the underlying skeleton. Deformation under isotonic contractions are modeled using a number of geometric primitives. As a result, we have implemented a toolkit where animations can be viewed and recorded as a video file. In order to demonstrate the use of the toolkit, we have produced a number of short movies for entertainment purposes. The generated muscle deformations are realistic. As future work, we plan to apply skin to our model and generate realistic looking skin deformations.

Keywords: Character animation, motion capture, anatomical based modeling of the human musculature, muscle deformation.

1 INTRODUCTION

Ever since the beginning of animation, people are interested in human character animation. However, we do not see many human characters in computer generated animation movies, since animation of human characters is still a challenging task.

One of the major problems in animating human characters is the difficulty of creating realistic motion. Human body has many degrees of freedom and human motion includes many parameters. Even when the constructed motion is very close to the real one; since we observe humans all the time, satisfying the audience is a very challenging task. One of the most widely applied solution to this problem is the use of motion capture technology in which the movements of real humans are recorded and then applied to the virtual characters. In this study, we present yet another application of motion capture data by developing an application that maps generic motion capture data to a skeleton model with attached muscles. Our goal is to present the audience a realistic motion.

The second problem in human character animation is to create body deformations. Human skin is affected by and deforms with motion. A character that does not have any skin deformation does not look realistic.

In this study, we present a skeletal character animation using motion capture data. Also, a subset of the muscles of the upper body is produced to model the deformation of the body in motion.

We present our study in two main parts. The first part is for entertainment purposes and includes the generation short movies of a moving skeleton (See Figure 1). The second part presents muscle deformations during body movements. The main purpose of this paper is to present the know-how of the application of generic motion capture data to a specific human character.

2 BACKGROUND AND PREVIOUS WORK

2.1 Motion Capture

Motion capture is the procedure of locating some sensors on human subjects; and then with the help of tracker devices recording the motion of the sensors during the movement of the human body.

In this study, we have used motion capture data to obtain realistic motion. We have used the Carnegie Melon University's motion capture database [2]. In the database, there are many skeleton models for different human subjects and their recorded motions.

Although there are many different formats for representing motion capture data, all of these formats represent very similar information. In the CMU database .asf file format is used for the skeletal information, and .amc file format is used for the description of the move-

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

Copyright UNION Agency – Science Press, Plzen, Czech Republic.

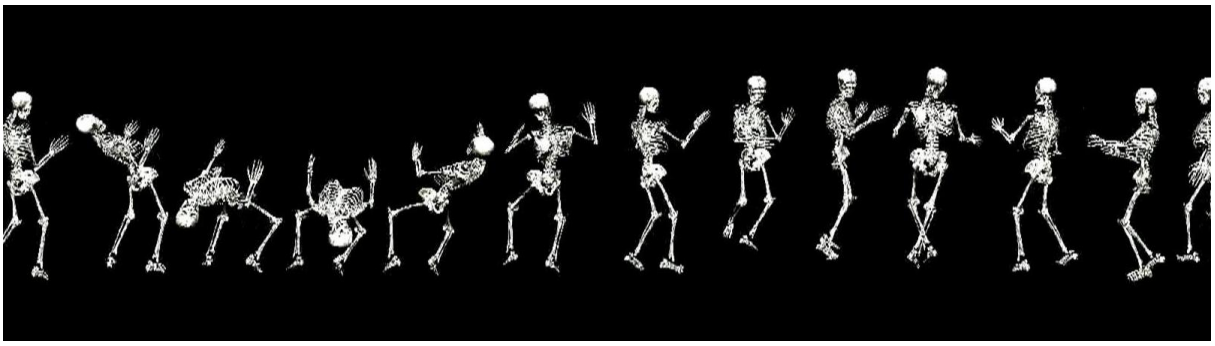


Figure 1: Animation of a skeleton with salsa dance motion capture data.

ment. The skeleton file (.asf file) supplies information about the skeleton of the human subject.

Figure 2 shows an example .asf file which consists of three parts containing information about the root of the skeleton, each joint, and the hierarchy of the skeleton. The part about the root of the skeleton contains the information such as the order of the channels (translation and rotation axes) that the motion would be applied to the root, the rotation order (eg., XYZ or YZX), the starting position, and the orientation of the root. For each joint, the unique id and the name of the joint, the orientation vector in the world coordinate system, the length of joint, the rotation of the local coordinate system for the joint with respect to the world coordinate system, and how many degrees of freedom this joint has are given in blocks between the keywords 'begin' and 'end'. In the last part of this file, the parent-children relation of these joints are given. Figure 3 shows how the joints of the skeleton are represented hierarchically.

```
:root
    order TX TY TZ RX RY RZ
    axis XYZ
    position 0 0 0
    orientation 0 0 0
:bonedata
    begin
        id 1
        name lhipjoint
        direction 0.655637 -0.713449 0.247245
        length 2.52691
        axis 0 0 0 XYZ
    end
    begin
        id 2
        name lfemur
        direction 0.34202 -0.939693 0
        length 7.59371
        axis 0 0 20 XYZ
        dof rx ry rz
        limits(-160.0 20.0)
            (-70.0 70.0)
            (-60.0 70.0)
    end
    ...
:hierarchy
    begin
        root lhipjoint rhipjoint lowerback
        lhipjoint lfemur
        ...
    end
```

Figure 2: An example .ASF file (Taken from [2])

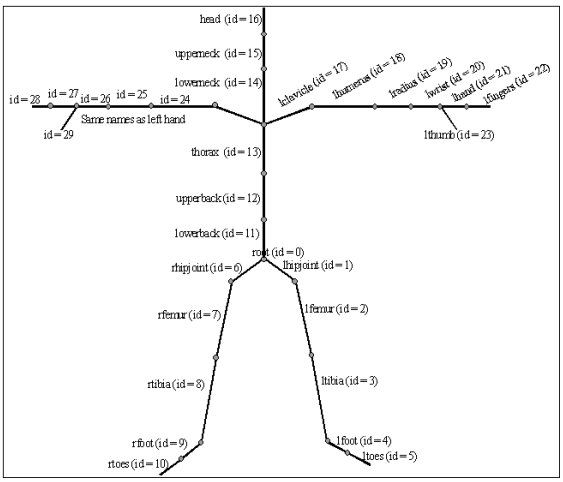


Figure 3: Skeleton hierarchy defined in .asf file (Taken from [7])

The motion file (.amc file) presents information about the transformations of the joints in their local coordinate systems for each frame.

2.2 Skin Deformation

The realistic looking motion should be supported with realistic skin deformation to obtain a successful human character animation. There are basically two approaches in the literature.

The first approach is known as smooth skinning or skeleton-subspace deformation [5]. In this approach, the shape of the skin is estimated by the transformations of the underlying joints. This approach is very efficient; however, it yields unrealistic results in some situations. For more information on the approach and the problems, the reader is referred to [5].

The second approach is a layered approach [1]. The layers include the skeleton, underlying muscles, and finally the fatty tissue and the skin. The skin actually sits on the top of the volume that is consumed by the skeleton and the muscle layer. Thus, skin deformation can be obtained by the underlying layers. The layered approach can be used successfully if the muscle and muscle deformations can be modeled correctly. Hence, studies on anatomical modeling emerged to target that

problem. Scheepers [8] uses artistic anatomy and models the skeleton and the muscles as close to as reality. To simplify computations, the layers are modeled using geometric primitives. Realistic results are observed, however, it is very hard to model muscles and insert them to the skeleton manually. To overcome the difficulties of manual modeling, in [3], computer aided tomography data are used to model muscles and their behavior correctly.

One recent study, [6] uses large amount of optical markers that are placed on the skin of the subjects with high resolution. In this method, they try to capture skin deformations with the motion of these markers. The results are very satisfactory; however, there are limitations due to large number of markers.

In this study, we present anatomical modeling that is inspired from [8]. However, skin deformation is not in the scope of the study. We have only performed modeling and deformations of a small portion of the musculature. We have used a publicly available skeleton model and attached the model to motion capture data. This process was very time consuming since the rest position of the .asf file and our skeleton model were different. The rest position of the skeleton model and the mapping of that position to the motion capture data are shown in Figure 4. We have followed a similar approach to [8] and try to model muscles by using geometric primitives. The details of our study are given in the next section.

3 IMPLEMENTATION DETAILS

The application is implemented in C language using the openGL library. Also, ffmpeg library is used for producing the accompanying videos. We have tested our system on the Linux operating system.

The developed application consists of many modules. The first module parses the skeleton and motion data files into an appropriate hierarchical data structure. The second module determines the position of the joints by traversing the data structure in a hierarchical manner. The third module is the skeleton module which draws the bones. The final module, deals with the construction of the muscles. In addition to these modules, the last module, video module, saves the produced animations as video files. The following sections give the details of these modules.

3.1 Data Module

As mentioned above, this module is responsible for constructing the required data structures from skeleton and motion files. It reads the skeleton file and stores the skeletal information considering the parent and child relationship of the joints. The motion file is also read and each frame information is stored in the memory.

3.2 Motion Capture Engine Module

The motion capture engine module is the underlying module that generates motion. Each frame is generated by traversing the skeleton in a breadth first manner and applying the given transformations.

The transformations for each joint are given in the joint's local coordinate system. In that respect, the transformation matrices that give the global coordinate system given the local coordinate system are constructed. The .asf specification presents the rotations that maps the local coordinate system to the global coordinate system as a series of rotations around the global x, y, and z axes one by one. Briefly, the order of operations applied to each of the joints for each frame is given as follows:

- Translation is performed using the direction vector and the length of the joint.
- The rotation that maps the global coordinate system to local coordinate system is applied.
- The rotations of the joint are applied.
- The rotation that maps the local coordinate system to global coordinate system is applied.
- The transformation of the parent joint is applied.

3.3 Skeletal Module

The main purpose of the skeletal module is to draw the skeleton. For this purpose, the module applies the necessary transformations to each bone to attach them to the joints. As we have mentioned in the previous sections, the rest poses of the .asf data and the skeleton model do not fit. In that respect, the necessary transformations for attachment is determined manually by trial and error. Thus, this is a very time consuming process. We have actually supplied the correct transformations for only one .asf data (79.asf in the CMU database), the rest can be found by using the ratio of the length of the corresponding joints between the supported one and the targeted one, however, the application we have developed does not support this feature.[1]

This module also supports smooth transformations for some of the bones that consist of a bone structure rather than just a single bone, e.g., the spine. The .asf file presents 6 segments between the root and the head, where the spine fits, however, in reality the spine contains around 30 segments. If we segment the spine into 6 segments and rotate them using the motion data,

[1] Since in the .amc file, the motion is defined in the local coordinate system of the joints, the motion data for other skeletons can be applied to the supported one. It seems to work fine but there may be some problems in the synthesized motion due to the small differences in the calibration stage of the motion capture equipment for each subject. Thus, we plan to add support for other skeletons as future work.

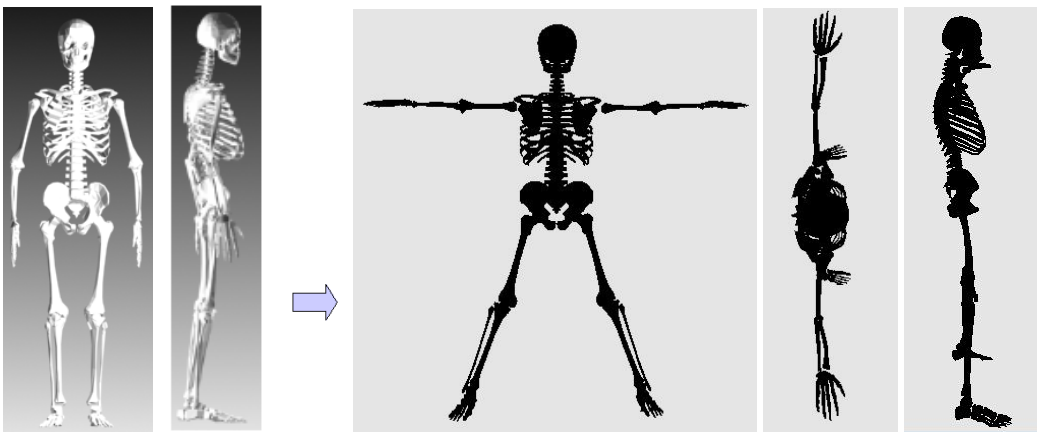


Figure 4: Attaching the skeleton model to the skeleton hierarchy in asf files requires mapping between the rest poses

the spine looks broken at the connection points. To avoid this problem we have applied the rotations to each vertex of each segment proportional to its local y-coordinate. Thus, the bottom vertex is not rotated, the upper vertex is fully rotated and the rotations of intermediate vertices are interpolated linearly.

Another purpose of this module is to determine the global coordinates of the attachment points of the muscles. This is needed because each end of the muscle is attached to a different bone. The attachment point of each muscle to each bone is determined manually by using the coordinates of the vertex of the bone that the muscle attaches (Weighted average of multiple vertices is also used.). Hence, for each frame the global coordinates of the attachment point must be calculated. This is done by applying the transformations of the corresponding joint and the bone to the coordinates of the attachment point. Later the muscle module draws the muscles that originates from these points.

3.4 Muscle Module

The muscle module is responsible for drawing the muscles between attachment points determined by the skeleton module. In addition, the module performs deformations that are caused by isotonic contraction.[2] The deformations caused by isometric contractions[3] are not implemented because the motion capture data do not provide us with the forces applied by each joint.

We have modeled only a small portion of the human musculature. Instead of using real anatomical data, the shapes and the attachment points of the muscles are modeled manually by trial and error using an anatomical atlas. This is also a time consuming process and with the help of digital anatomical data one can produce more realistic results. We have modeled 5 groups of muscles for both sides of the upper body. Brachialis muscle, biceps brachii muscle, triceps brachii muscle, deltoid muscle, and pectoralis muscle are modeled. The muscles we have modeled can be seen in Figure 5. We have generally used three modeling strategies for each of the groups, which are presented in detail below.

The muscle module generally uses some of the approaches given in [8]. It also allows tendons to be inserted. The details of the approaches we have used are given in the next sections.

Fusiform Muscle Brachialis muscle, biceps brachii muscle, and triceps brachii muscle are modeled using the "fusiform muscle approach"[8]. This approach constructs muscles using ellipsoids. This type of muscle takes the volume and ratio of two radii as parameters, also calculates the muscle length using the distance between the attachment points. Then the three radii of the ellipsoid can be found.

The fusiform muscle model that has certain length, width, and height uses the following parameters:

$$a = \frac{height}{2}$$
$$b = \frac{width}{2}$$
$$c = \frac{length}{2}$$

Thus, the volume, v and the ratio of two radii, r are given as:

$$v = \frac{4\pi abc}{3}$$
$$r = \frac{a}{b}$$

[2] In isotonic contraction, the volume of the muscle is preserved and the shape of the muscle deforms only as a result of the change in the length of the muscle.

[3] In isometric contraction the volume of the muscle is increased when it is tensed and is decreased when it is relaxed.

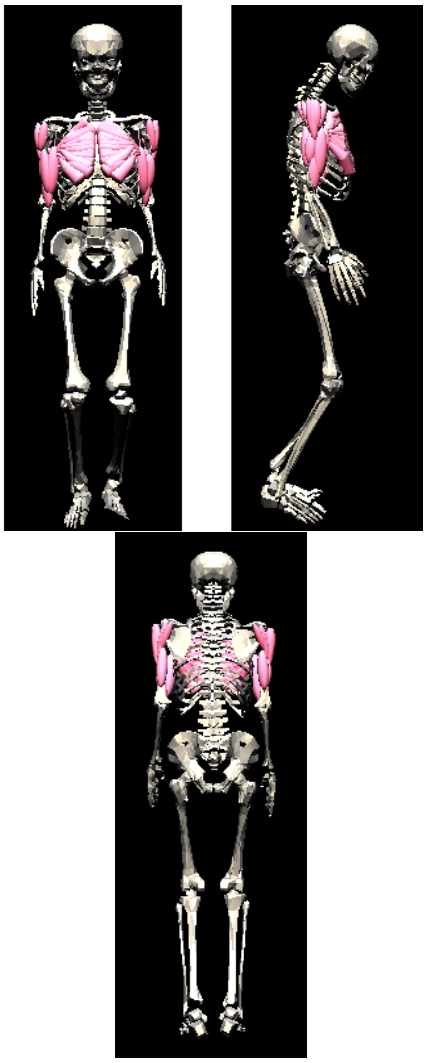


Figure 5: Anterior, lateral, and posterior view of the model.

When the length of a muscle changes, the new values of a, b, and c can be derived as follows considering the volume is preserved:

$$c' = \frac{l'}{2}$$
$$b' = \sqrt{\frac{3v}{4\pi r c'}}$$
$$a' = b' r$$

For the correct shape to be constructed, the local coordinate system of the muscle should be found. The z-axis of the muscle is determined to be the vector from the first attachment point to the other. The x-axis is found by taking the cross product of the z-axis with the vector from the first attachment point to the joint coordinate of the bone that the point is on. The y-axis is given as the cross product of x-axis and z-axis. The transformation matrix that maps the local coordinate system to the global coordinate system is constructed using these axes.

The fusiform model may also contain tendons. A tendon is a structure that attaches a muscle to a bone and its length does not change. Thus, tendons that have certain lengths can be given as parameters to the muscle model for having muscles with tendons. If tendons are used when calculating the length of the muscle, length of tendons should be subtracted from the distance between the two attachment points.

When modeling triceps brachii muscle, the distance between attachment points cannot be used as the muscle length, since the tendon of that muscle group passes from the back of the elbow. Thus, to estimate the correct length for triceps brachii muscle, an additional control point is added at the elbow.

Multi-Belly Muscle The multi-belly muscle model, inspired from [8], is used to model the deltoid muscle. This muscle consists of several segments and attachment points for each segment are given separately. Each segment of multi-belly muscle is actually a fusiform muscle. One way to insert a fusiform muscle for each segment is to use the technique described in the previous section, but using such an approach would yield sharp orientations for each segment. Instead of finding the orientations of each segment separately, this model uses the attachment points of the neighboring segments to find the orientation of the each segment. The local z-axis of each segment is the vector from the first attachment point to the second. The y-axis can be found by the cross-product of two vectors from the first points of the neighboring segments to the second point of the current segment. If the current segment is a boundary segment, one of the vectors is the z-axis of the segment. The x-axis can be found by the cross product of the z-axis and the y-axis.

The segments presented in this section behave similar to the fusiform muscle described above. The tendons for each of the segments are also added as described in the previous section.

Multi-Attachment Muscle The multi-attachment muscle is used to model pectoralis muscle. The pectoralis muscle sits on the ribs and one end is connected to sternum (the bone that connects the ribs at the front) and the other end to humerus (the bone that is between the shoulder and the elbow). To place the muscle on the ribs, we determine two additional control points that the muscle connects other than the attachment points. The cross-section area of the muscle gets smaller as moved from sternum to humerus. Thus, we have used cones as an approximation of this shape.

The deformation of the muscle is approximated by the change of the base radius of the cone as the length

of the muscle changes. The new radius ,r, of the cone with volume v and length l is given as:

$$r = \sqrt{\frac{3v}{l\pi}}$$

The length of the muscle is determined as the sum of the Euclidean distances between the consecutive control points and the attachment points. The muscle is drawn as segments between these points starting from the first attachment point.

The orientation of this muscle can be found similarly as in fusiform muscles. However, the orientation of each segment is found separately. The z-axis of each segment is found using two attachment points, the x-axis is found by the cross product of the z-axis with the vector from the point of the joint that the muscle connects and its parent joint. The y-axis is similarly determined as the cross product of x and z axes. Actually, the orientations of x and y axes have no effect on the shape of the muscle, since the cross section of the muscle on z-axis is a circle.

3.5 Video Module

The video module is used to record the animation to a video file. We have used ffmpeg [4] library for recording each frame to a video file. We have only support for .mpg extension and 30 frames per second or below. The framebuffer is read from the memory for each frame and recorded as a video frame using the library.

4 THE APPLICATION OUTCOMES

The application we have developed has three main outcomes.

The first outcome is the animation toolkit that can be used either to view animation of motion capture data or to record the animation as a video file. The user can select to view or record only the markers, the skeleton, or the skeleton with muscles. The user can also select viewing directions by arrow keys (this property is very limited only to four degrees of freedom and should be developed further).

The second outcome is related to entertainment. We have observed that the motion of a human skeleton is very entertaining and decide to produce short movies using motion capture data. The production of movies started from the recording of the desired animation using the toolkit. Later, we have used a commercial movie editor to produce credits and titles, and to insert music or sound into the animation files.

The final outcome, is the muscle deformations. We have produced realistic looking muscle deformations. The pictures of the muscles are given in Figures 6 and 7. As the lengths of the biceps brachii and triceps brachii muscles change, their deformations can be easily seen. In this study, we have provided only small portion of the muscles. However, these muscles can be used to generate realistic deformation with a skin attached on top of them.

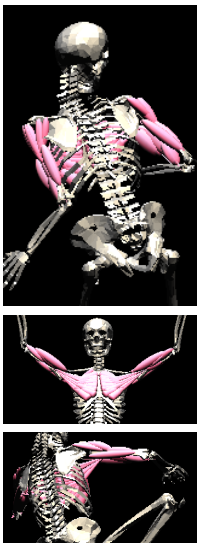


Figure 6: Muscle deformation with some postures of the model.

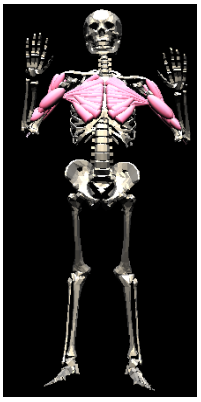


Figure 7: Muscle behavior in a posture.

5 CONCLUSIONS

Realistic modeling of the human body and motion is an interesting subject in computer animation. It is also very challenging because of many parameters that the human motion involves. In this study, we have produced human motion animation using motion capture data, modeling muscles, and simulating muscle deformations. We have constructed some of the human upper body muscles using some geometric primitives and a skeleton model which supports the muscles.

The produced deformations of the muscles can be used to find the skin surface deformations by attaching a skin over the volume that is formed by the skeleton and the muscles.

Although motion capture data yield very realistic results, it has some limitations. The markers for capturing

motion is placed on the surface of the body. Therefore, when these marker points are attached to a skeleton, a number of problems will arise. One of the problems is the segmented rotation of the spine. We have solved that problem by applying rotation proportional to the y-coordinate of the bone and made the rotations smoother. The motion data for hands were also problematic. The hand definitions in general .asf files do not generally fit on a skeleton model; therefore we had to model the hands as rigid objects and could not present the motion of the fingers. Actually, the motion of hands requires high resolution markers and they are usually not regarded as an important component for full body capture.

Muscle modeling and deformation are done for only a small subset of the muscles. This is a tedious work because we did not employ any digital anatomical data. We have looked up the positions and the shapes of the muscles from an anatomy atlas and tried to obtain a similar shape at similar locations on the model. Results are realistic and can be used for the surface skin deformation. Whole body muscles can be modeled in a similar way.

In conclusion, we have tried to animate a skeleton character using motion capture data. Although we have produced movies at the skeletal level, we have obtained successful results for entertainment purposes. In addition to that, we have tried to model muscle deformations using geometric primitives. A small portion of the muscles is modeled and realistic results are obtained. This approach can be used for generating surface skin deformation, however, all the human musculature should be modeled.

ACKNOWLEDGMENTS

The data used in this study were obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217. We are greatly thankful to the people at Graphics Laboratory at the Carnegie Mellon University for publishing their motion capture database on the Internet.

REFERENCES

[1] J. E. Chadwick, D. R. Haumann, and R. E. Parent. Layered construction for deformable animated characters. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 243–252, New York, NY, USA, 1989. ACM Press.

[2] CMU. CMU Graphics Lab Motion Capture Database. http://mocap.cs.cmu.edu, 2001.

[3] F. Dong, G. J. Clapworthy, M. A. Krokos, and J. Yao. An anatomy-based approach to human muscle modeling and deformation. *IEEE Transactions on Visualization and Computer Graphics*, 8(2):154–170, 2002.

[4] FFMPEG. Ffmpeg. http://ffmpeg.mplayerhq.hu/, 2007.

[5] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 165–172, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[6] Sang Il Park and Jessica K. Hodgins. Capturing and animating skin deformation in human motion. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 881–889, New York, NY, USA, 2006. ACM Press.

[7] Nancy Pollard. Description of motion capture file format. http://graphics.cs.cmu.edu/nsp/course/15-464/Fall05/assignments/StartupCodeDescription.html, 2005.

[8] Coenraad Frederik Scheepers. *Anatomy-base Surface Generation For Artivulated Models of Human Figures*. PhD thesis, The Ohio State University, 1996.

Spatially constrained model for mean shift

M.J. Lucena
Departamento de Informática
University of Jaén
Campus Las Lagunillas Edif. A3
23071 - Jaén, Spain
mlucena@ujaen.es

J.M. Fuertes
Departamento de Informática
University of Jaén
Campus Las Lagunillas Edif. A3
23071 - Jaén, Spain
jmf@ujaen.es

N. Pérez de la Blanca
Departamento de Ciencias de la
Computación e I.A.
University of Granada
Daniel Saucedo Aranda s/n
18071 - Granada, Spain
nicolas@ugr.es

ABSTRACT

This paper presents a multiple model real-time tracking technique based on the mean-shift algorithm. The proposed approach incorporates spatial information from several connected regions into the histogram-based representation model of the target, and enables multiple models to be used to represent the same object. The use of several regions to capture the color spatial information into a single *combined* model, allow us to increase the object tracking efficiency. We use a model selection function that takes into account both the similarity of the model with the information present in the image, and the target dynamics. In the tracking experiments presented, our method successfully coped with lighting changes, occlusion, and clutter.

Keywords: Non-rigid object tracking, target representation and localization.

1 INTRODUCTION

Object tracking has been studied and applied to numerous computer vision problems which include vehicle tracking, surveillance, medical diagnosis, actor animation, tracking multiple people, and face detection and animation. In this paper we are mainly interested in people tracking although our approach is general. The most recent survey of the state of the art on this topic is given in [8].

The goal of a tracking process is to estimate the state of the object in a time t, represented by a vector $\mathbf{X}_t$, given the set $\mathbf{Z}$ of measurements taken from the sequence of images in times $t-1, t-2, \ldots$

A straightforward way to derive a distribution model $p(\mathbf{X}_t|\mathbf{Z})$, is by using histogram analysis [2, 6, 4]. To do this, the current frame is searched for a region, a fixed-shape variable-size window, whose content best matches a reference color model. Comaniciu et al. [4] proposed a tracking algorithm in which a reference target model is represented by a h-bin histogram that approximates the probability density function (pdf) in the feature space. Maximization is then performed by a mean-shift procedure. Collins [10] improves the algorithm using two different kernels, one for scale and another for motion, what allows more stable tracking on targets with fast motion from or towards the camera.

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

When the target moves outdoors, noise, shadows and lighting changes can appear which significantly alter the color distributions in the image sequence. In order to deal with these difficulties, Porikli&Tuzel[9] introduce a mean-shift based model update technique with an adaptive change detection method. We take a different approximation, modelling all the appearance information in terms of probability distributions. In this case, a single pdf will be insufficient for modeling and tracking the object reliably. We suggest the use of a set of models which are switched according to a probabilistic rule [7].

The other important point in the mean shift algorithm is the use of the spatial information in the appearance model. The classic formulation [4] encodes spatial information using radially symmetric kernels and therefore it becomes easy for the tracker to get confused with other objects having the same feature distribution but different spatial configurations of features. Several contributions have been done to overcome this problem [1, 11]. In these cases the spatial information is encoded in complex statistical appearance models (spatiograms) using the pixels of the tracking region. In our case the focus is different, we use a more complex tracking region defined as the union of several connected regions, overlapping or not, each one having an appearance model defined by its color histogram. This choice looks more suitable for the tracking objects defined by several regions of homogeneous color.

The aim of this research is to develop a technique for real-time object tracking under variable lighting conditions in a cluttered scene. This paper proposes a generalization of the classical mean-shift tracking model to enable the handling of spatial restrictions, us-

ing several histograms associated to different parts of the tracking region, instead of using only one summarizing all the color information.

The paper is organized as follows: Section 2 introduces the combined model which is used to incorporate spatial information into the histogram-based model; experimental results are shown in Section 3; and Section 4 presents our conclusions.

2 PROPOSED METHOD

2.1 Tracking

The pdfs that characterize the target model and the target candidate are given by h-bin histograms, extracted from ellipsoidal regions of the image.

$$\begin{array}{rll} \text{target model:} & \hat{\mathbf{q}} = \{\hat{q}_u\}_{u=1\ldots h} & \sum_{u=1}^{h} \hat{q}_u = 1 \\ \text{target candidate:} & \hat{\mathbf{p}}(\mathbf{y}) = \{\hat{p}_u(\mathbf{y})\}_{u=1\ldots h} & \sum_{u=1}^{h} \hat{p}_u = 1 \end{array}$$

Where $\mathbf{y}$ is the spatial location of the target candidate. A similarity function $\hat{\rho}(\mathbf{y})$ is defined, whose local maxima in the image indicate the presence of objects having representations similar to $\hat{\mathbf{q}}$. The tracking process is then defined as a search procedure for those maxima. In order to avoid the computational cost associated with a gradient-based optimization, Comaniciu et al. regularize the similarity function by masking the region of interest with an isotropic Epanechnikov kernel in the spatial domain [3] and apply a mean-shift technique. Previously, the region of interest which defines the target is normalized to a unit circle, by independently rescaling the horizontal and vertical dimensions of the original ellipsoid.

The similarity function defines a distance between the target model and the candidates. The distance between two discrete distributions is defined as:

$$d(\mathbf{y}) = \sqrt{1 - \rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}]} \tag{1}$$

where the similarity function will be denoted by:

$$\hat{\rho}(\mathbf{y}) \equiv \rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}] = \sum_{u=1}^{h} \sqrt{\hat{p}_u(\mathbf{y})\hat{q}_u} \tag{2}$$

which is the sample estimate of the Bhattacharyya coefficient between $\mathbf{p}$ and $\mathbf{q}$ [5].

2.2 Adding Spatial Information

One of the main drawbacks presented by histogram-based models is the absence of any spatial information. However, it is sometimes interesting to be able to include spatial restrictions in the object model. For example, when attempting to track a person walking, it is possible to identify two areas with typically different histograms, corresponding to the target's torso and head, with a specific spatial relationship between both of these. In this section, we extend the target model in order to incorporate this kind of information.

Our combined model Q is a set comprising m regions, overlapping or not, each of which is characterized by a distribution $\hat{\mathbf{q}}^i$, together with the offset value $\Delta\mathbf{y}^i$ of each region with respect to the location where the model is centered, $\mathbf{y}$. The new target candidate $P(\mathbf{y})$ is given by an analogous expression:

$$\begin{array}{rl} \text{Combined model:} & Q = \{\hat{\mathbf{q}}^i, \Delta\mathbf{y}^i\}_{i=1\ldots m} \\ \text{Combined candidate:} & P(\mathbf{y}) = \{\hat{\mathbf{p}}^i(\mathbf{y} + \Delta\mathbf{y}^i)\}_{i=1\ldots m} \end{array} \tag{3}$$

The *Bhattacharyya coefficient* corresponding to Q for a given location $\mathbf{y}$ of the image is therefore given by the following expression:

$$\rho[P(\mathbf{y}), Q] = \frac{1}{m}\sum_{i=1}^{m} \rho[\hat{\mathbf{p}}(\mathbf{y} + \Delta\mathbf{y}^i), \hat{\mathbf{q}}^i] \tag{4}$$

The following algorithm is used to estimate, in the new frame, the location $\mathbf{y}_1$ of the maximum value of the *Bhattacharyya coefficient*, starting from the location $\mathbf{y}_0$ estimated for the previous frame:

Algorithm: Given the combined target model Q, comprised by m regions, and its location $\mathbf{y}_0$ in the previous frame:

1. For each i from 1 to m:

   (a) Compute the weights $\{\omega_j^i\}_{j=1\ldots n_i}$ [4]:

   $$\omega_j^i = \sum_{u=1}^{h} \sqrt{\frac{\hat{q}_u^i}{\hat{p}_u^i(\mathbf{y}_0 + \Delta\mathbf{y}^i)}}\,\delta[\beta(\mathbf{x}_j^i) - u] \tag{5}$$

   (b) Obtain $\mathbf{y}_1^i$ from:

   $$\mathbf{y}_1^i = \frac{\sum_{j=1}^{n_i} \mathbf{x}_j^i \omega_j^i k'(\mathbf{x}_j^i)}{\sum_{j=1}^{n_i} \omega_j^i k'(\mathbf{x}_j^i)} \tag{6}$$

2. The next location of the target is computed as

   $$\mathbf{y}_1 = \arg\min_{\mathbf{y} \in \{\mathbf{y}_1^1, \ldots \mathbf{y}_1^m\}} \{\rho[P(\mathbf{y}), Q]\} \tag{7}$$

3. If $||\mathbf{y}_1 - \mathbf{y}_0|| < \varepsilon$, return $\mathbf{y}_1$ and stop.
   Otherwise, set $\mathbf{y}_0 \leftarrow \mathbf{y}_1$ and go to Step 1.

Where n_i is the number of pixels of the i–th region, $\mathbf{x}_j^i$ are the locations of their pixels, normalized to the unit circle. $\beta(\mathbf{x})$ represents the associated bin at the pixel located on $\mathbf{x}$, h is the histogram size, and δ is the

Kronecker delta function. $k(\mathbf{x})$ is the Epanechnikov kernel function, and $k'(\mathbf{x})$ represents its derivative.

Values $\hat{q}_u^i$ and $\hat{p}_u^i$ are defined by the expressions:

$$\hat{q}_u^i = C_q \sum_{j=1}^{n_i} k(\mathbf{x}_j^i)\delta[\beta(\mathbf{x}_j^i) - u] \tag{8}$$

$$\hat{p}_u^i(\mathbf{y}) = C_p \sum_{j=1}^{n_i} k(\mathbf{x}_j^i - \mathbf{y})\delta[\beta(\mathbf{x}_j^i) - u] \tag{9}$$

where C_q and C_p are normalization constants obtained by imposing the conditions $\sum_u \hat{q}_u^i = 1$ and $\sum_u \hat{p}_u^i = 1$ for each i.

When the number of regions $m = 1$, our combined model is reduced to the model proposed by Comaniciu et al.

2.3 Scale Selection

Different approaches for dealing with scale changes have been proposed. These techniques define a scale factor σ, which allows to adjust the bandwidth of the kernel profile, i.e. the size of the ellipsoid where the target candidate histogram is computed.

The approach proposed in [4] works as follows: Given the scale σ_{prev} of the previous frame, we run the target localization algorithm three times, with $\sigma_i = \sigma_{prev} + i \cdot \Delta\sigma$, for $i \in \{-1, 0, 1\}$. The new scale σ_{new} is then computed as

$$\sigma_{new} = \gamma \cdot \sigma_{opt} + (1 - \gamma) \cdot \sigma_{prev}, \tag{10}$$

where σ_{opt} is the value of σ_i which gives the best Battacharyya coefficient. In our experiments, we have chosen the default values proposed in [4] $\gamma = 0.1$ and $\Delta\sigma = 0.1\sigma_{prev}$.

In order to employ the scale selection technique proposed in [10], some modifications to the tracking algorithm are needed:

- The kernel function $k(x)$ is replaced by a Difference-of-Gaussian filter $H_x(x, s)$, where s is a scale parameter, as defined in [10].
- Expression (6) must be replaced by:

$$\mathbf{y}_1^i = \frac{\sum_s H_s(s) \sum_{j=1}^{n_i} \mathbf{x}_j^i \omega_j^i H_x'(\mathbf{x}_j^i, s)}{\sum_s H_s(s) \sum_{j=1}^{n_i} |\omega_j^i H_x'(\mathbf{x}_j^i, s)|} \tag{11}$$

  where $-n \leq s \leq n$ defines a range of scales σ_s, where $\sigma_s = \sigma_{prev} \cdot b^s$, with $n = 2$, $b = 1.1$, and $H_s(s) = 1 - (s/n)^2$.

- A mean-shift step is applied to estimate scale for every region, using the following equation:

$$s_i' = \frac{\sum_s \sum_{j=1}^{n_i} H_x(\mathbf{x}_j^i, s)\omega_j^i s}{\sum_s \sum_{j=1}^{n_i} H_x(\mathbf{x}_j^i, s)\omega_j^i} \tag{12}$$

  Being s' the scale value s_i' which gives the best Bhattacharyya coefficient, the scale for the next step is computed as $\sigma_{new} = \sigma_{prev} \cdot b^{s'}$.

2.4 Model Selection

In order to prevent loss of the target due to changes in object orientation and lighting conditions, we will extend the approach proposed in [7] to deal with scale changes. We define $\mathbf{M}$ as a multiple model, comprising a set of n combined models, corresponding to several histograms extracted from images generated by the object under different orientations and lighting conditions:

$$\mathbf{M} = \{Q_0, Q_1, \ldots, Q_{n-1}\} \tag{13}$$

We can thus run the target localization algorithm for each Q_i, proposed in Sections 2.2 and 2.3, and obtain a set $\mathbf{Y}$ of image locations and scales:

$$\mathbf{Y} = \{(\mathbf{y}_0, \sigma_0), (\mathbf{y}_1, \sigma_1), \ldots, (\mathbf{y}_{n-1}, \sigma_{n-1})\} \tag{14}$$

and a set $\mathbf{B}$ of coefficients:

$$\mathbf{B} = \{b_0, b_1, \ldots, b_{n-1}\}, \tag{15}$$

where each ($\mathbf{y}_i$, σ_i) represent the location and scale where the model Q_i maximizes the Bhattacharyya coefficient (4), and each $b_i = \rho[P(\mathbf{y}_i), Q_i]$, computed at scale σ_i, represents the actual value of the coefficient itself.

For each frame, we then need to select the combined model in $\mathbf{M}$ which best fits the observed image. Selecting the Q_i with the largest Bhattacharyya coefficient may increase the risk of distractions with image regions having similar histograms to the ones present in our model. In order to avoid this, we can weight the estimated values, giving more importance to the ones with the greatest coherence with the dynamics observed for the target so far. With this goal in mind, we will define a probability distribution based on the displacement between location $\mathbf{y}_i$ and scale σ_i estimated by the tracker, and predicted location $\bar{\mathbf{y}}$ and scale $\bar{\sigma}$ for the target, given by some dynamic model for the object to be tracked. In addition, we will consider that each of the combined models $Q_i \in \mathbf{M}$ presents an *a priori* probability $p(Q_i)$ of appearing in each image of the sequence.

We will define the probability of each Q_i, given $\mathbf{B}$, as:

$$p(Q_i/\mathbf{B}) = \frac{b_i \cdot p(Q_i)}{\sum_j \left(b_j \cdot p(Q_j)\right)} \tag{16}$$

and the probability of each Q_i, given $\mathbf{Y}$, as:

$$p(Q_i/\mathbf{Y}) = \frac{p(\bar{\mathbf{y}} - \mathbf{y}_i, \bar{\sigma} - \sigma_i) \cdot p(Q_i)}{\sum_j \Big(p(\bar{\mathbf{y}} - \mathbf{y}_j, \bar{\sigma} - \sigma_j) \cdot p(Q_j) \Big)} \quad (17)$$

In our case, we suppose that the $(\bar{\mathbf{y}} - \mathbf{y}_i)$ and $(\bar{\sigma} - \sigma_i)$ values follow a multivariate zero-mean Gaussian distribution, i.e. $p(\bar{\mathbf{y}} - \mathbf{y}_i, \bar{\sigma} - \sigma_i) \sim N(0; \sigma_x, \sigma_s)$. According to Bayes' rule,

$$p(\mathbf{Y}/Q_i) = \frac{p(Q_i/\mathbf{Y}) \cdot p(\mathbf{Y})}{p(Q_i)} \quad (18)$$

and

$$p(\mathbf{B}/Q_i) = \frac{p(Q_i/\mathbf{B}) \cdot p(\mathbf{B})}{p(Q_i)} \quad (19)$$

and also

$$p(\mathbf{B}, \mathbf{Y}/Q_i) = \frac{p(Q_i/\mathbf{B}, \mathbf{Y}) \cdot p(\mathbf{B}, \mathbf{Y})}{p(Q_i)} \quad (20)$$

We assume that $\mathbf{B}$ and $\mathbf{Y}$ are statistically independent given Q_i, i.e.

$$p(\mathbf{B}, \mathbf{Y}/Q_i) = p(\mathbf{B}/Q_i) \cdot p(\mathbf{Y}/Q_i) \quad (21)$$

Replacing expressions (18), (19) and (20) in (21), we obtain the following equation:

$$p(Q_i/\mathbf{B}, \mathbf{Y}) = \frac{p(Q_i/\mathbf{B}) \cdot p(Q_i/\mathbf{Y})}{p(Q_i)} \cdot C \quad (22)$$

where $C = \frac{p(\mathbf{B}) \cdot p(\mathbf{Y})}{p(\mathbf{B}, \mathbf{Y})}$ is a constant term.

Finally, substituting $p(Q_i/\mathbf{B})$ and $p(Q_i/\mathbf{Y})$ for expressions (16) and (17) in Equation (22), and discarding the constant C, we obtain the following expression for $S(i)$:

$$\left[\frac{b_i \cdot p(\bar{\mathbf{y}} - \mathbf{y}_i, \bar{\sigma} - \sigma_i) \cdot p(Q_i)}{\sum_j \Big(b_j \cdot p(Q_j) \Big) \cdot \sum_k \Big(p(\bar{\mathbf{y}} - \mathbf{y}_k, \bar{\sigma} - \sigma_k) \cdot p(Q_k) \Big)} \right] \quad (23)$$

used to select the best model Q^* for each frame, which is the model Q_i with highest $S(i)$ value.

In practice, the proposed framework consists of two phases: first, the *mean shift* algorithm is applied to each Q_i comprising the multiple model. Next, the specific model which best matches the observation is selected.

In order to avoid processing all the models for each frame, specially when their number is high, we can represent the multiple model by means of a graph, where each node corresponds with a simple model Q_i. Two nodes will be connected if they can appear in two consecutive frames of a typical sequence. As we deal with video sequences, we can assume that the model changes must be gradual, and so only those nodes which are sufficiently similar in the graph will be connected. Therefore, it will be only necessary to examine (in each frame) the neighbors of the actual model Q^*, enabling us to restrain the computational requirements of the technique.

Sequence	*Frames*	*Resolution*	*Ground truth*
eps	208	320×240	Yes
paddle	501	320×240	Yes
player	142	352×288	No

Table 1: Sequences used in the experiments.

3 EXPERIMENTAL RESULTS

In order to evaluate his performance, the method proposed in this article has been applied to various sequences. In this section, some representative results are presented on mpeg-compressed sequences (Table 1), for some of which we have ground truth data. The RGB color space was taken as the feature space, quantized into $8 \times 8 \times 8$ bins. For scale selection, we have employed the technique proposed in [4] (see Section 2.3) to all the sequences, except for the *paddle* sequence, in which we have used the Collins approach [10].

In the images showing the tracking results, small squares are superimposed in the upper-left corner of each frame indicating the active simple model in the corresponding multiple model. All of these are shown in white, except for the one corresponding to the model chosen in each frame, which is displayed in gray.

3.1 Dynamical Model

In order to estimate the expected location $\bar{\mathbf{y}}^{t+1}$ and scale $\bar{\sigma}^{t+1}$ of the target in frame $t+1$ needed to compute $p(Q_i/\mathbf{B}, \mathbf{Y})$, we have defined a simple dynamical model, that suffices to our purposes:

$$\begin{aligned} \bar{\mathbf{y}}^{t+1} &= \mathbf{y}^t + \mathbf{d}^t \\ \mathbf{d}^t &= \lambda \cdot (\mathbf{y}^t - \mathbf{y}^{t-1}) + (1-\lambda) \cdot \mathbf{d}^{t-1} \\ \bar{\sigma}^{t+1} &= \sigma^t \end{aligned} \quad (24)$$

where $\mathbf{y}^t$, σ^t represent the location and scale of the target estimated by the tracking algorithm at time t, and $\mathbf{d}^0$=0. In our experiments, we have used a value for λ of 0.5. In the case of a specially adapted tracker for a certain object, the dynamical model can be replaced by a more suitable one, or learnt from examples.

3.2 Performance Metrics

In order to evaluate our method we have employed several localization metrics to the results obtained from the sequences from which we have ground

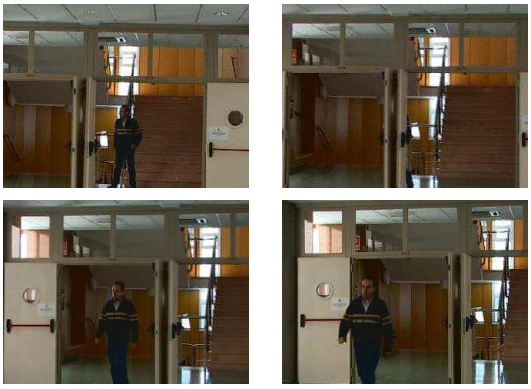


Figure 1: *Eps* sequence: Frames 41, 81, 121 and 161.

	OAR	ADC	BAP	OTE
Non Combined	28.36	34.24	30.51	90.50
Combined	67.44	77.49	70.47	10.10

Table 2: Performance results obtained for the *eps* sequence using combined and non-combined models.

truth data. Being $\mathbf{g} = (g_x, g_y)$ the real location of the object, $\mathbf{e} = (e_x, e_y)$ the target location estimated by the tracker, A_C the area corresponding to the object, and A_T the area estimated by the tracker:

- Object Area Recall: $OAR = 100 \cdot \frac{|A_T \cap A_C|}{|A_T|}$
- Box Area Precision: $BAP = 100 \cdot \frac{|A_T \cap A_C|}{|A_C|}$
- Area Dice Coefficient: $ADC = 100 \cdot \frac{2 * |A_T \cap A_C|}{|A_T| + |A_C|}$
- Object Tracking Error: $OTE = ||\mathbf{e} - \mathbf{g}||$

All of the mentioned metrics are averaged over the whole sequence.

3.3 Experiments

The *eps* sequence (Figure 1) was taken with a home video camera, showing a person coming down some stairs and walking along in poor lighting conditions. In this example, the target goes behind a door at around frame 80, and is occluded for several frames. As we can see, the non combined model can't overcome this occlusion. Figure 4 shows the Bhattacharyya coefficient between combined and non-combined models and the target estimated location along the sequence. When the target disappears, combined model coefficient decreases, while the non-combined one remains stable. This suggests that the first one falls into a local maximum.

As can be seen in Table 2, the best results are obtained by the combined models. The other tracking result is significantly worse, because the target is missed.

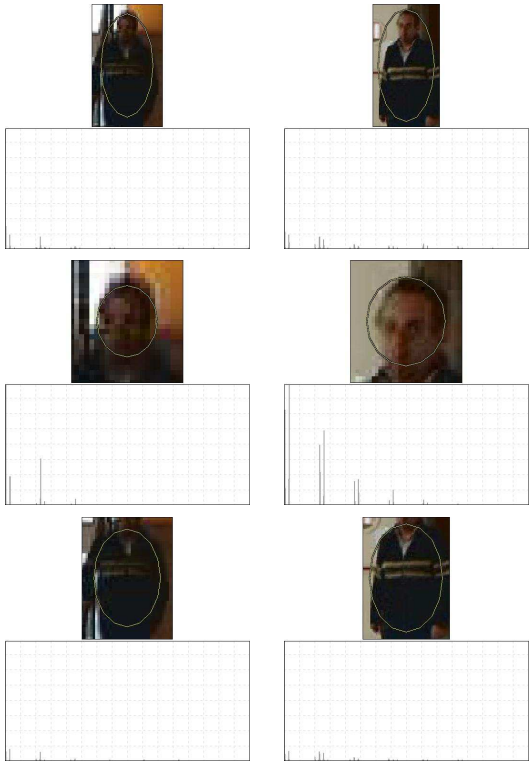


Figure 2: Models used for the *eps* sequence, with their corresponding weighted histograms.

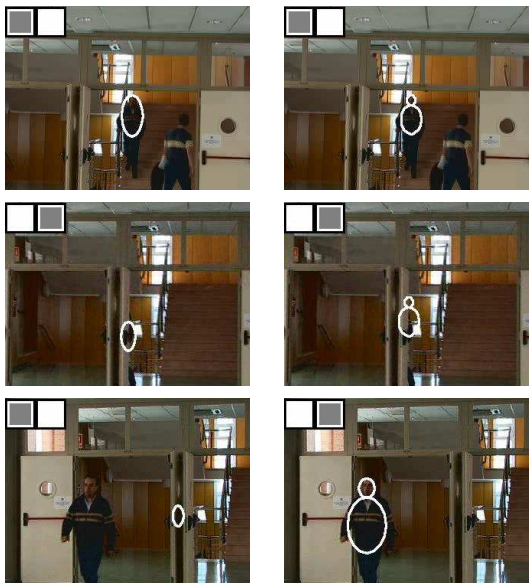


Figure 3: Results obtained with the *eps* sequence, using non-combined models (left) and combined models (right). In the upper-left corner, the active model is shown in gray.

 ISBN 978-86943-15-2

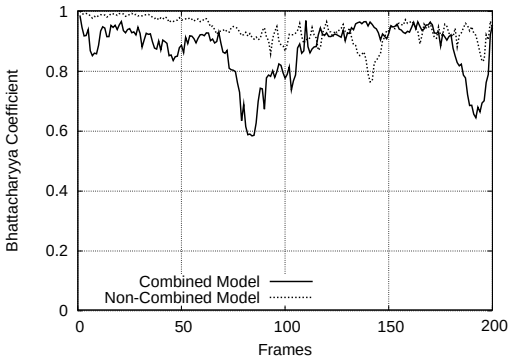


Figure 4: Bhattacharyya coefficient between the combined and non-combined models and the object's estimated location in the *eps* sequence.

Figure 5: *Paddle* sequence: Frames 86, 171, 256 and 341.

	OAR	ADC	BAP	OTE
Non Combined	29.39	80.70	40.94	22.02
Combined	55.20	97.15	69.45	6.77

Table 3: Performance results obtained for the *paddle* sequence using combined and non-combined models.

The *paddle* sequence (Figure 5) lasts 20 seconds, and shows a person playing paddle, moving, changing position and moving out of the shade. In this experiment, we have employed the Collins approach (see Section 2.3) to estimate target location and scale. As we can see, almost all of the models track correctly the target, with the multiple combined model behaving best (even when the target is inclined).

The performance results for the *paddle* sequence are shown in Table 3. As we can see, the combined models perform better than the non-combined ones.

The *player* sequence (Figure 8) represents several soccer players. It was taken from a television news broadcast and shows various players from the same team, moving quickly and with sudden zoom and panning movements. In this case, three simple models have been employed, three combined models, and the associated multiple models. The results are shown

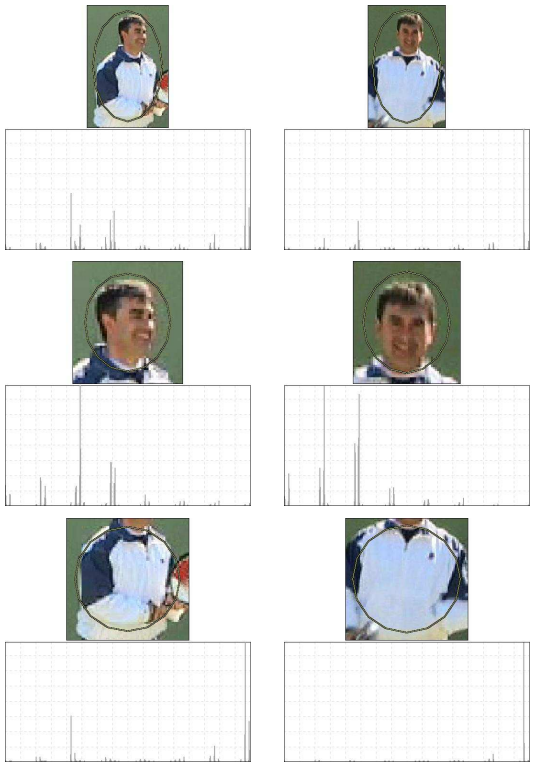


Figure 6: Models used for the *paddle* sequence, with their corresponding weighted histograms.

Figure 7: Results obtained with the *paddle* sequence, using non-combined models (left) and combined models (right). In the upper-left corner, the active model is shown in gray.

Figure 8: *Player* sequence: Frames 26, 51, 76 and 101.

Figure 9: Models used for the *player* sequence, with their corresponding weighted histograms.

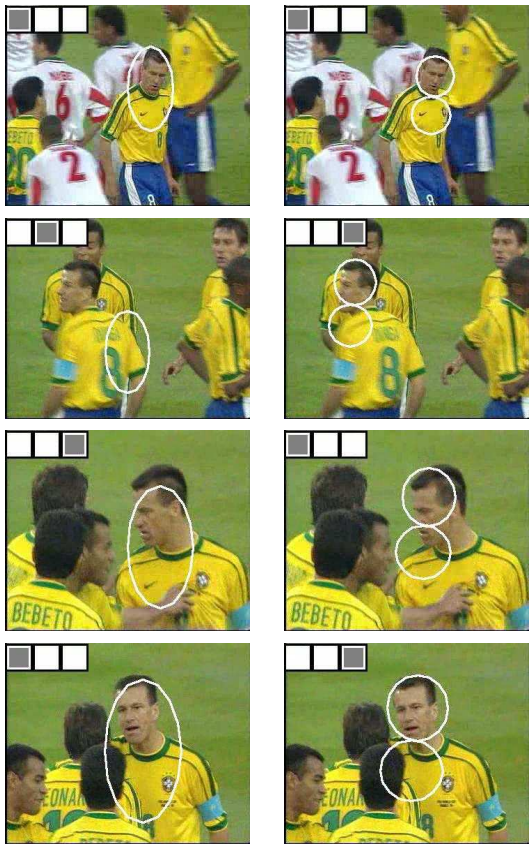


Figure 10: Results obtained with the *player* sequence, using non-combined models (left) and combined models (right). In the upper-left corner, the active model is shown in gray.

in Figure 10, and reveal once again how the combined multiple model behaves best, enabling the target to be tracked even when there are partial occlusions or when similar objects appear on the scene. Figure 11 shows that, like in the *eps* experiment, the Bhattacharyya coefficient of the combined model is generally lower than the non-combined model one. This suggests again that the combined model is less prone to fall into local maxima.

The experiments shown in this paper run at over 20 frames/second (40 in some cases) on a Pentium IV 3GHz desktop computer, except the ones that use the Collins scale selection technique (see Section 2.3), which run at 10 frames/second, due to the higher computational requirements of that approach.

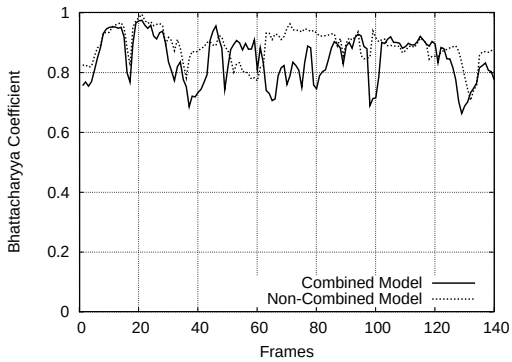


Figure 11: Bhattacharyya coefficient between the combined and non-combined models and the object's estimated location in the *player* sequence.

4 CONCLUSIONS

A combined tracking model has been presented, which enables spatial information to be incorporated into the histogram-based models, increasing the robustness of

 ISBN 978-86943-15-2

the tracking process. On this, a distance measure has been defined which can be used in a minimization process based on the mean-shift algorithm.

The proposed combined models provided better results in our experiments than the classic approaches. Although the proposed technique involve a higher computational cost, this is low enough for them to be used in real time.

Our tracking scheme can be extended in several ways in order to be more flexible and applicable in real world situations: extending the combined model to deal with rotations and articulated motion, integrating into the tracking scheme more information sources, such as edges or optical flow.

5 ACKNOWLEDGMENTS

This work has been financed by Grant TIC-2005-1665 from the Spanish Ministry of Science and Technology.

REFERENCES

[1] S. Birchfield and S. Rangarajan. Spatiograms versus histograms for region-based tracking. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 1158–1163, 2005.

[2] G. R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, 2, 1998.

[3] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[4] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.

[5] T. Kailath. The divergence and Bhatacharyya distance measures in signal selection. *IEEE Transactions on Comm. Technology*, 15:52–60, 1967.

[6] T. L. Liu and H. T. Chen. Real-time tracking using trust-region methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):397–402, 2004.

[7] M. Lucena, J. M. Fuertes, and N. Perez de la Blanca. Real-time tracking using multiple target models. *Lecture Notes in Computer Science*, 3522:20–27, 2005.

[8] T. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding.*, 104:90–126, 2006.

[9] Faith Porikli and Oncel Tuzel. Human Body Tracking by Adaptive Background Models and Mean-Shift Analysis. Technical Report TR-2003-036, Mitsubishi Electric Research Laboratories, July 2003.

[10] R. T. Collins. Mean-shift Blob Tracking through Scale Space. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 234–240, 2003.

[11] Q. Zhao and H. Tao. Object tracking using color correlogram. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 263–270, 2005.

A Generalized Light-Field API and Management System

Joan Blasco
Dto. Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de vera s/n
46022 Valencia, Spain
joablaib@inf.upv.es

Miguel Escrivà
Dto. Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de vera s/n
46022 Valencia, Spain
mescriva@dsic.upv.es

Franciso Abad
Dto. Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de vera s/n
46022 Valencia, Spain
fjabad@dsic.upv.es

Ricardo Quirós
Dto. Lenguajes y Sistemas Informáticos
Universidad Jaime I
Avenida Sos Banyat s/n
28071 Castellón, Spain
quiros@lsi.uji.es

Emilio Camahort
Dto. Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de vera s/n
46022 Valencia, Spain
camahort@dsic.upv.es

Roberto Vivó
Dto. Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de vera s/n
46022 Valencia, Spain
rvivo@dsic.upv.es

ABSTRACT

Light fields are a computer graphics modeling technique that represents objects using radiance samples instead of geometry. Radiance samples may be stored as sets of images or 4D arrays of pixel values. Light fields have various advantages: their rendering complexity only depends on the output image's, they can represent sophisticated illumination effects, and they are well-suited for display using autostereoscopic devices.
To study different light-field representations, as well as their capture, generation, resampling, storage, composition, rendering and display, we have developed a light-field based API and graphics system. The system supports models based on different light-field parameterizations, as well as different generation and capture methods, rendering methods, data caching algorithms and output displays. The API will also support hybrid light-field and geometry based models, as well as light-field resampling and composition.

Keywords: Image-based models, the light field, graphics APIs, modeling and rendering systems.

1 INTRODUCTION

Advances in imaging and graphics hardware in the past ten years have motivated the appearance of novel rendering techniques based on image data. Image-based rendering methods build up a 3D representation of an object using radiance data obtained from 2D image samples of the object. These methods can synthesize non-existent image samples from the information of the existent ones, granting the observer a perspective-correct view from either sampled or unsampled viewpoints.

Image-based rendering techniques offer simple acquisition capabilities coupled with realistic representation of complex lighting conditions. Furthermore, in contrast to geometry based techniques, purely image-based models provide two additional advantages. First, rendering complexity of image-based models depends only on the complexity of the output images. Second, most compression and simplification algorithms perform better on image data than geometric data. Additionally, both geometry-based models and image-based models can be merged to produce hybrid models using information originating from both geometry and images.

Light fields are a Computer Graphics modelling technique that relies on radiance or image data to represent complex geometric objects. These objects are inefficient to store and render using geometry, but radiance data can be organized in different arrangements of images or 4D pixel arrays to represent and display the objects [Lev06].nswered 58 12/19/2007 Roberto V

We introduce a light-field modelling system to efficiently build, store, combine, resample, retrieve and render light fields based on different representations. Our goal is to compare the representations, combine them, and display them on autostereoscopic displays [DEA+06].

We want to achieve interactive rendering framerates together with an intuitive API for light-field capture and generation. This paper describes the design of our light-field representations, our API and the architecture of the underlying graphics system. We also report work in

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 2008 conference proceedings
WSCG'2008, February 4 – February 7, 2008
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

progress in multiple light-field rendering and light-field models augmented with geometry.

Light fields can be applied to many computer graphics related areas. These include rendering, scientific visualization, advertisement and virtual reality applications. Our main interest is driving autostereoscopic displays and other volumetric displays. These displays typically output 3D or 4D light-field data. They are the future of graphics devices because they provide multiple users with correct perspectives for both eyes without using intrusive devices, like goggles.

Our paper starts with a review of previous work in light-field modelling and rendering systems. Then, we formalize light-field representations and review light-field parameterizations. We introduce then the functionality of our system and describe thoroughly its goals. Section 4 describes our API, Section 5 our implementation, and Section 5.6 our results. Finally, we present some conclusions and directions for future work.

2 BACKGROUND

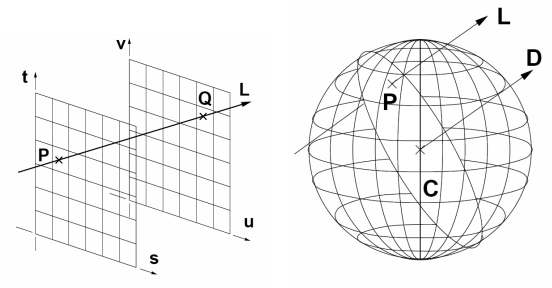
Among the different image-based rendering techniques, light fields are one of the newest and most complex ones. Light fields were introduced in the papers by Levoy and Hanrahan [LH96] and Gortler et al. [GGSC96], where they were called lumigraphs.

Formally, the light field or plenoptic function is defined as a 5D function that describes the radiance flowing through all the points in a scene in all possible directions [AB91]. A static light field can be represented as a 5D scalar function that can be simplified to 4D in the absence of occluders, as shown in [GGSC96, LH96].

In order to render a light field, the radiance function is sampled at points positioned on a sampling pattern, determined by the kind of parameterization used in the model. There are planar anisotropic [LH96] and spherical quasi-isotropic [CLF98] parameterizations. They result in different parameter arrangements and light-field configurations. Additionally, unstructured light-fiel models can also be built [Tak05].

The most important examples of planar parameterizations are Levoy and Hanrahan's "light slab" fields [LH96] and the Lumigraph [GGSC96]. Both use a 2 plane parameterization. They select a planar grid of camera positions and take an image for each camera. They require multiple pairs of planes, called slabs, to represent entire objects.

Schirmacher [SHS99] uses the same representation, but adds a simple geometric model (called proxy) similar to the lumigraph's. The entire light-field model is then built using an adaptive sampling method starts with a simple set of views, then attempts to add a new image from a different view. In order to determine which point of view to use for the new image, several candidates are considered. Candidate views are then prioritized using a cost function that takes into account disocclusion artifacts and a radiance error measure.



(a) Two-plane parameterizations. (b) Direction-and-point parameterizations.

Figure 1: Two related light-field parameterizations.

In the case of spherical arrangements, cameras are positioned throughout a sphere's surface, pointing towards its center. Images are thus sampled from the sphere's surface as described in [CLF98]. This way the sphere acts as a simple approximation to the convex hull of an object centered inside the sphere.

Using this approach, camera positions are determined by the sampling directions needed, and sampling direction distribution is determined by the specific set of directions selected. Examples of these arrangements are those used in [CLF98, IPL97], where the chosen distribution of the samples follows quasi-uniform triangulations of the sphere's surface [EDA+06].

In unstructured light fields, however, camera positions need not to be positioned following any specific arrangement. Among these methods, it is possible to make a difference between those in which correct renderings are based on camera positions but also on a priori knowledge about geometry information from the scene, and those that use other approaches.

For example in [Tak05], where an algorithm for light-field rendering that does not make use of any geometry information is presented. The technique is based on a 3-step algorithm. In this algorithm a series of depth layers are created and pixels from the captured images are positioned on those layers according to an estimation of their depth based on an *in-focus* measurement.

In the case of geometric information support, the use of geometric proxies to enhance object representations has been researched since the birth of light-field modeling [GGSC96, PPS, CBCG02, WAA+00, BBM+01] and greatly resembles view-dependent texture mapping[DTM96] in the use of transformed captured images as textures for the geometric proxies.

For instance, in [BBM+01] the authors use approximate geometry models to build a map containing the weight of the images obtained from each position for every pixel. A subsequent sampling of the image plane uses these weights to render images from cameras with non-zero weights at every sample point used in the representation. Each sample is rendered m times, being m

the number of cameras. These samples are also taken based on geometry information, and are determined by projecting the geometric approximation of the scene on the image plane.

We want to support these different types of light-field representations: planar, spherical and unstructured. Our work thus builds on the results of [LH96], [GGSC96], [?], [SHS99] and [Tak05] among others. But our focus is on the systems and implementation issues of building, managing and rendering multiple types of light-field models. In this paper we focus on a management system for general light fields and the associated API to access the functionality (features) of our system.

3 FUNCTIONALITY

Providing support for light-field modelling and rendering requires a fairly sophisticated hardware and software system. We may need to support different models and camera arrangements, different robot configurations for image capture, several storage strategies, and multiple rendering algorithms and display devices. Additionally, we may want to compare different sampling, resampling, composition and compression algorithms. In this Section we describe which of these features we want to support in our light-field system.

3.1 Light-Field Representations

We want a system capable of drawing and capturing different light-field representations. With this in mind, we have designed software that can be easily configured and extended to support different types of light-field representations. The flexibility of our API also supports models that include geometric information like, for example, depth information. Our architecture supports adding new representations by configuring light-field plugins in a simple way.

We have developed a plugin based on spherical light fields. It is based on the direction-and-point parameterization [EDA+06]. A planar light-field plugin is also being currently developed. This added functionality, is useful to combine and resample different types of light fields.

3.2 Generation and Capture

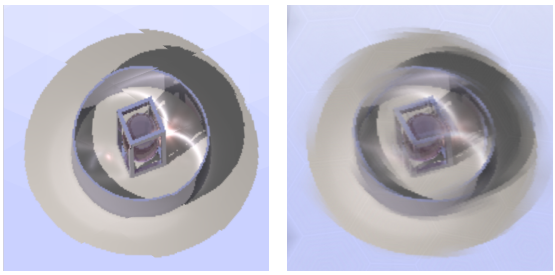
Light-field models can not exist without generating or capturing them. Generation produces models of synthetic object, while capture samples radiance data from real-world objects. Light-field generation is performed using methods that are specific to each type of representation. Typically, the methods are camera iterators that produce the camera parameters needed to properly obtain the images of the light-field model. To generate synthetic models, we support rendering using OpenGL, Blender, POV or YafRay.

Figure 2: The robot arm is controlled by a PC. The arm is moved to different locations and takes snapshots of the object to capture.

Our software also allows the capture of light fields from real-world objects providing a list of camera parameters for each of the images needed. This camera parameters are also produced by iterators and they allow us to create snapshots from the real world objects. This capture of real-object light-fields is performed using a camera mounted on a robotic arm(see figure 2).

3.3 Spherical Light-Field Plugin

Light fields have a major drawback. Obtaining high-quality renderings needs a huge amount of images, up to 20000, which uncompressed result in approximately 3.7 GBytes of storage. Use of fewer images produces artifacts due to discretization errors. These errors produce seams visible across triangle boundaries when rendering is done without interpolation (this kind of artifact can be noticed in figure 11). Use of interpolation however turns this seams into ghosting (see figure 3(b)).



(a) With constant kernel. (b) With quadralinear kernel.

Figure 3: A light-field model with caustics. The model was rendered using two algorithms with different interpolation kernels. .

An alternative to higher-resolution light fields is the use of depth information and lower directional resolutions. Gortler et al. implemented depth-corrected

 ISBN 978-86943-15-2

light-field rendering using a representation extended with a low-resolution geometric proxy of the target object [GGSC96]. Shade et al. introduced layered depth images, images extended with a depth value associated to each pixel [SGHS98].

Using depth information rendering can be improved to compute accurate texture coordinates. With this technique the amount of storage required by a light field's radiance data can be reduced by a factor of $1/4$ or $1/16$.

3.4 Light-field Composition

Another of our goals is the composition of different light-field models and addition to external geometric information such as labels, tags and regular objects into a single scene. This goal can be accomplished with the use of depth maps, multiple images per directional sample, and improved rendering algorithms.

Handling of multiple light fields is managed by storing multiple images per directional sample. Images are sorted according to their position in 3D space along each direction and then multiple images for each visible triangle are reprojected. Drawing of these images in proper back-to-front order must be ensured for obtaining of correct renderings.

Geometry and light-field rendering can also be accomplished using depth information associated to the light-field data. Again, texture-mapped triangles are generated for each directional sample and each light-field model. They are drawn in 3D space at roughly the same location as the surface of the light fields' original geometric objects. Geometric objects are drawn subsequently.

4 API DESIGN

Light-field capture and generation have been simplified to provide an intuitive programmatical interface in our light-field software. Acquisition can be accomplished by following four simple steps. First, the light-field parameters are initialized. Second, a list of the camera positions, which may vary depending on the parameterization plugin used, is provided. Images from the different positions are then obtained sequentially, and once all of them have been processed, the light field is finally stored. The whole process is depicted in figure 4.

The rendering process has been simplified in a similar way. In this case, after the initialization of the light-field plugin of choice, the user needs to select and load the algorithm responsible for its rendering, and assign its concerning properties in case they exist. After that, the renderer and the image cache need to be created before starting the actual rendering.

Our software uses multiple threads, so that rendering and image caching can run separately (see figure 5).

Control of the image retrieval thread is done by the rendering thread, which decides which images to transfer depending on the time available for each frame. The

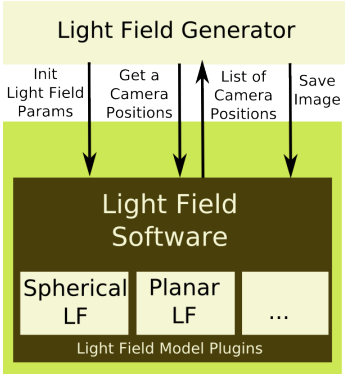


Figure 4: Capture system architecture.

rendering thread also decides the size of the working set based on the number of light-field images that the GPU can display in 1/25 seconds.

The whole set of light-field data is difficult to fit into main memory concurrently, so we use out-of-core storage techniques. Our system uses a texture least-recently-used (LRU) cache that manages the light-field images so that only new images need to be loaded from disk at any time.

Using a two-level cache architecture we support rendering of light-field models requiring up to several gigabytes of storage. The architecture transfers image data from secondary storage to main memory, and from main memory to GPU memory. The size of a light-field model requires storing it in a hard drive, but loading the image data from a hard drive produces really poor performance, so we need a faster way of loading the images. The fastest way stores the images in GPU memory and loads them directly when needed for a visible triangle. However, GPU memory is not enough to hold all the image data, so we have added an intermediate cache system in main memory to provide a better capacity/response time ratio as seen in figure 8)

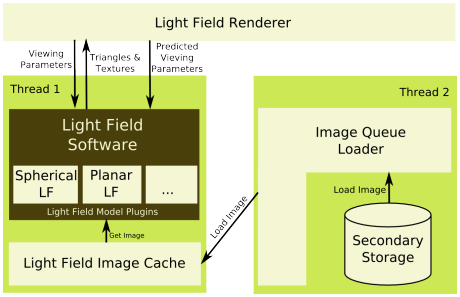


Figure 5: Render system architecture.

5 IMPLEMENTATION

In this section, we discuss the implementation of the above features. For the implementation we use C++, because it provides the expressiveness and performance needed for both the image processing and the rendering

of the light-field data. We start with an overview of our class architecture.

5.1 Light-Field Interface

A light-field class represents an abstract interface to all of the different light-field representations. It is in charge of managing the light-field model and also possesses knowledge about the organization of the images stored in secondary storage.

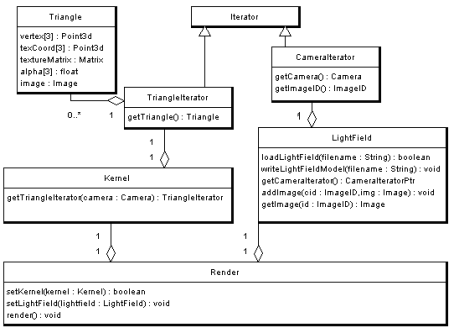


Figure 6: Light–Field and Kernel interface diagram

The light-field class knows how to load, save and generate its light-field representation, but knows nothing about the rendering of the light-field (see figure 6). The class responsible for the task of rendering the light field is the *Kernel* class.

5.2 Kernel Interface

A light-field parameterization can be rendered using different algorithms providing different results (see figure 15 for an example). Control of the different algorithms used in the rendering stage has been modelled with the use of *Kernels*. In this system, a *Kernel* is the class in charge of the rendering of a light-field type (see figure 6).

A light-field representation is able to use different kernels to render the light field. As an example of this fact, rendering of a direction-and-point parameterization (DPP) based light-field can be performed by using various kernels. At the present moment these consist of a constant reconstruction kernel, a quadratic reconstruction kernel and a kernel using depth-maps for geometric information supported rendering.

Examples of the use of such kernels with the DPP light-field plugin can be observed in figures 3(a), 3(b) and 15. The different results obtained with the use of the various kernels are clearly visible.

5.3 Plugin Architecture

Our system uses a plugin architecture in order to support multiple light-field parameterizations. In this way, different light-field parameterizations can be implemented in plugins that the *base* library can load on demand.

We support two kinds of plugins: LightFields and Kernel modules (see figure 7). Our plugin system allows us to check which plugins are registered in the system and inspect which kind of plugin the loaded plugins represent, as well as which properties the plugins support. All of this processing can be performed in execution time.

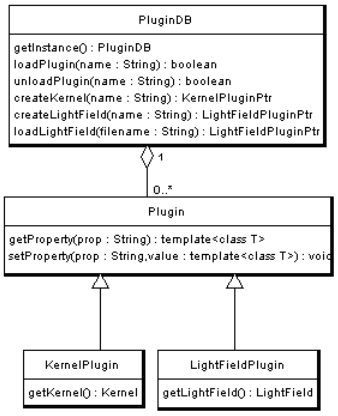


Figure 7: Classes that implement the plugin functionality.

Each LightField or Kernel may have different properties to configure. For example, the DPP LightField implementation has a property related to the level of the icosahedron subdivision, but PDP LightField has a property related to the number of horizontal and vertical plane subdivisions. This properties can be accessed directly by their name, and they are fully configurable and independent of the rest of the architecture.

5.4 Cache Architecture

As it was said in section 4, our system uses a two-level cache architecture, that allows us to render light-field models requiring a huge storage capacity.

As new images are needed by the renderer, the system caches the corresponding images, first obtaining them from disk and fitting them into main memory (when they are not directly available in there) and then from main memory into GPU memory (see figure 8).

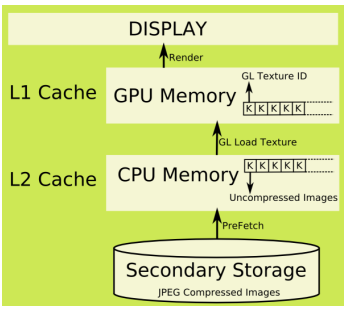


Figure 8: Two-level cache architecture: images are addressed using unique triangle identifiers (CPU memory and secondary storage) and texture identifiers (GPU memory).

 ISBN 978-86943-15-2

The caching algorithm has also been designed to steer image retrieval following camera movement. During rendering, the most probable new camera position and orientation is computed using the last camera movement. This prediction provides an estimate of the next frame's viewing frustum. The frustum is then used to determine the set of images that are most likely to be needed soon.

5.5 Implementing a new Light-field type

In order to add a new light field, a series of steps are to be performed. First of all, a new class inheriting from the *LightField* interface and implementing its methods must be created. Class *DPP_LightField* is an example implementation, as figure 9 shows.

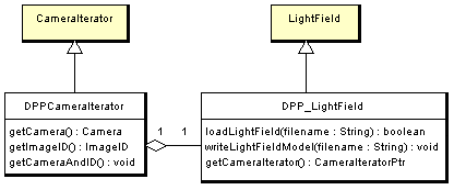


Figure 9: DPP light-field implementation diagram

The most important methods needed to implement in this class are *loadLightField*, *writeLightFieldModel* and *getCameraIterator*. The *loadLightField* method is responsible for the setup of the light-field information, while the *writeLightFieldModel* method will be in charge of the information dump to a massive storage device at the end of the acquisition process. Finally, *getCameraIterator* is the most important method regarding the obtaining of the camera position arrangements of the plugin in use (see figure 9).

Once the light-field class for the new model has been implemented, the user must adapt, if needed, a new *CameraIterator* for it. This iterator will provide the new LightField class with the desired camera arrangement. At this stage, the system is already capable of acquiring light fields based on the new model, see figure 9.

In order to render the acquired light fields, the user must also implement the rendering algorithm to do so. This is accomplished by implementing one or more adapted classes inheriting the *Kernel* interface. This class will be the placeholder of the rendering algorithm, and will work jointly with a corresponding triangle iterator, that should be implemented too if needed. An example of this class architecture is shown in figure 10.

5.6 Results

The DPP light-field plugin for our system has been tested on several datasets. Figures 11, 12 and 13 show renderings from different models at different levels of detail. The models used were large polygon meshes, with polygon counts ranging from 226k to 478k polygons. In figure 14, complex lighting effects like caustics

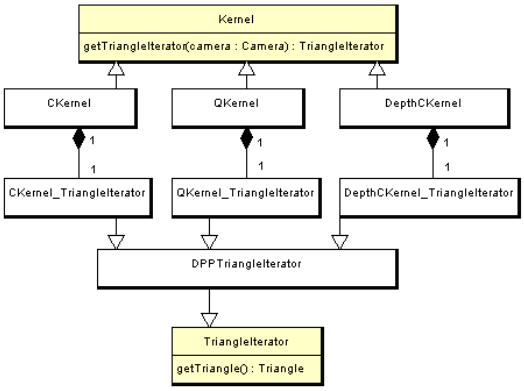


Figure 10: Kernel module architecture example

were also added to the scene. All of these light fields were obtained using the Blender and Yafray renderers.

Rendering of the original geometric models took 60 seconds for Models 1 and 2, 90 seconds for Model 3 and roughly 3.5 minutes for Model 4. The light-field models, however, were rendered at 50–60 frames per second using the constant and quadralinear kernels, and two seconds per frame using depth correction. This greatly improves on the rendering times of the geometric models.

Figure 11: Model 1: The UMA model was scanned with our 3D scanner. We captured the light field using Blender. The image shows the render of model at level 4.

In figure 15 two different renderings of the Stanford dragon model can be observed, showing the differences in image quality obtained when depth information is used for the rendering of the light fields. In this case the light field was obtained using OpenGL.

Figure 12: Model 2: A gargoyle model rendered al level 4. The light field was captured using Blender and the model geometry was acquired with our 3D scanner.

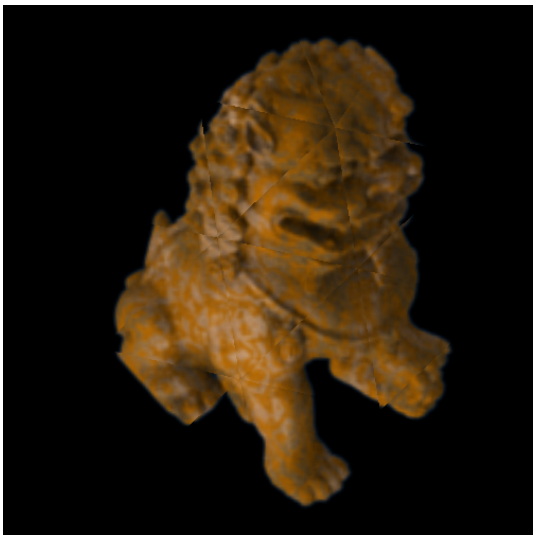


Figure 13: Model 3: Chinese lion light field rendered at level 3. Model is provided courtesy of INRIA by the AIM@SHAPE Shape Repository.

6 CONCLUSIONS AND FUTURE WORK

In this paper we address the issue of light-field modeling and rendering in a generalized way, through the creation of a system which allows abstraction from the different features that different light-field parameterizations have.

We presented a system that creates an abstract framework for light-field modelling and rendering, allowing a quicker development of novel techniques of this kind, which may use different parameterizations, or be supported by information coming from varied sources like radiance, depth or others. The system has been implemented in standard C++, also granting its flexibility and portability.

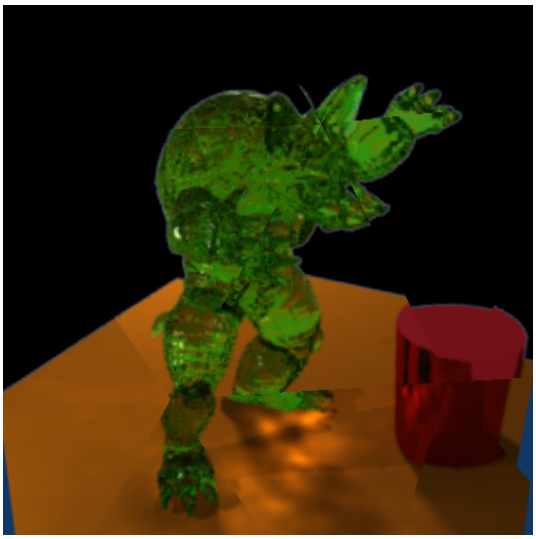


Figure 14: Model 4: The Stanford armadillo model with a green glass material. The light field was captured with YafRay raytracer to obtain high quality images with effects like caustics and reflections. Rendered at level 2.

Current work includes creation of a spherical light field plugin for the system, while implementation of additional plugins like a planar light field plugin are still under development.

The reader is referred to the webpage `http://www.sig.upv.es/ALF/papers/wscg2008/` for some videos produced with our system.

ACKNOWLEDGEMENTS

This work was partially supported by grants TIN2004-203303-E and TIN2005-08863-C03-01 of the Spanish Ministry of Education and Science and by the GAMETOOLS STREP project IST-004363 of the 6th Framework Program of the European Union.

REFERENCES

[AB91] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. In *In Computational Models of Visual Processing*, page Chapter 1, Cambridge, MA, 1991. MIT Press.

[BBM+01] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 425–432, New York, NY, USA, 2001. ACM Press.

[CBCG02] Wei-Chao Chen, Jean-Yves Bouguet, Michael H. Chu, and Radek Grzeszczuk.

(a) Rendered without depth maps.

(b) Rendered using the depth maps.

Figure 15: The Stanford dragon model. This figure shows how the use of depth maps improves the output image quality.

Light field mapping: efficient representation and hardware rendering of surface light fields. *ACM Trans. Graph.*, 21(3):447–456, 2002.

[CLF98] Emilio Camahort, Apostolos Lerios, and Donald Fussell. Uniformly sampled light fields. In *Proc. Eurographics Rendering Workshop '98)*, pages 117–130, 1998.

[DEA+06] A. Domingo, M. Escrivá, F.J. Abad, R.Vivó, and E.Camahort. Introducing extended and augmented light fields for autostereoscopic displays. In *In Proceedings of 3rd Ibero-American Symposium in Computer Graphics*, pages 64–67, 2006.

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 11–20, New York, NY, USA, 1996. ACM Press.

[EDA+06] M. Escrivá, A. Domingo, F. Abad, R. Vivó, and E. Camahort. Modeling and rendering of dpp-based light fields. In *In Proceedings of GMAI'2006*, London, UK, 2006. IEEE Computer Society Press.

[GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Proc. SIGGRAPH '96*, pages 43–54, 1996.

[IPL97] Insung Ihm, Sanghoon Park, and Rae Kyoung Lee. Rendering of spherical light fields. In *PG '97: Proceedings of the 5th Pacific Conference on Computer Graphics and Applications*, page 59, Washington, DC, USA, 1997. IEEE Computer Society.

[Lev06] Marc Levoy. Light fields and computational imaging. *Computer*, 39(8):46–55, 2006.

[LH96] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proc. SIGGRAPH '96*, pages 31–42, 1996.

[PPS] Steven J. Gortler Peter-Pike Sloan, Michael F. Cohen. Time critical lumigraph rendering. In *In Proc. 1997 Symposium on Interactive 3D Graphics*.

[SGHS98] Jonathan W. Shade, Steven J. Gortler, Li-Wei He, and Richard Szeliski. Layered depth images. In *Proceedings of ACM SIGGRAPH'98*, pages 231–242, July 1998.

[SHS99] H. Schirmacher, W. Heidrich, and H.-P. Seidel. Adaptive acquisition of lumigraphs from synthetic scenes. In *In EUROGRAPHICS'99*, pages 151–159, 1999.

[Tak05] T. Takahashi, K.; Naemura. Unstructured light field rendering using on-the-fly focus measurement. *IEEE International Conference on Multimedia and Expo*, 2005.

[WAA+00] Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. Surface light fields for 3D photography. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 287–296. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

Acoustic Modeling of Reverberation using Smoothed Particle Hydrodynamics

Charles Thomas Wolfe
Department of Computer Science
University of Colorado
Colorado Springs, CO
USA, 80933-7150
Charles.t.wolfe@gmail.com

Sudhanshu Kumar Semwal
Department of Computer Science
University of Colorado
Colorado Springs, CO
USA, 80933-7150
semwal@eas.uccs.edu

ABSTRACT

Current methods for digital acoustic modeling of reverberation can be categorized into two main areas: DSP algorithms and sound tracing algorithms. DSP algorithms are usually chosen for their speed and ease of use. DSP methods are generally fast, but do not accurately represent complex environments. Sound tracing algorithms, such as beam tracing, accurately model static environments but have difficulty with complex and/or moving geometry. A Computation Fluid Dynamics (CFD) method called Smoothed Particle Hydrodynamics (SPH) has recently gained interest as a method for accurately simulating fluid flow using discrete particles. This paper extends SPH by adding the ability to model sound wave propagation through fluids. A generalized method that integrates sound generation and reception is presented in this paper. This method provides a basis for acoustic sound effects, such as reverberation.

Keywords

Visualization; Acoustic Field Simulation; smoothed particle hydrodynamics, physical modeling.

1. INTRODUCTION

Reverberation is a natural property of enclosed spaces in which a sound continues to echo after the sound source has been removed. Part of the sound is absorbed and converted to thermal energy, while part of the sound is refracted and continues to travel through the obstacle material. The vast majority of the sound energy, however, is reflected. When a large number of reflections return back to the listener, the effect is called reverberation. The music industry in particular has been interested in artificial reverberation since the early 1900's. With the development of electronic technology, DSP methods are preferred due to their ease of use and easy portability. No longer does a musician or sound engineer has to set up a bulky reverb effect or record in a specialized reverb chamber. The sound is processed electronically after the original sound has been recorded. Other acoustic modeling methods, such as sound tracing, have experienced some success due to their ability to accurately simulate complex environments. Although DSP methods are the current preferred method, they are by no means perfect. The primary restriction on these methods is that they rely on sound engineers recording the reverberation pattern of a given environment. This means for a given reverberation pattern, a room must be constructed, recorded and transformed into a DSP model. The algorithms presented in this paper attempts to eliminate the need for this kind of setup. Using a recently developed fluid mechanics simulation algorithm --- Smoothed Particle Hydrodynamics (SPH) -- complex acoustic environments can be simulated in software. The new algorithms were tested on two sounds -- an acoustic engineering type, the impulse; and a short musical phrase sound.

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

2. RELATED WORK -- SPH

In Computational Fluid Dynamics, most current methods simulate the flow of fluids using an Eulerian, or grid-based, method [LI64]. SPH, however, is particle-based and therefore utilizes a Lagrangian approach. In an Eulerian simulation, field values are defined only on regularly spaced grid points. The flow of fluid is observed flowing past fixed points in space. This method is well suited for simulation where flow past a point must be observed, such as the end of a pipe. The grid-based method is especially well-suited using data structures such as cellular automata, but the method does come with drawbacks. First, the fluid is confined to the grid and

the flow is undefined outside of the grid. Second, if the grid is large, the data set may only use a small portion of the grid and memory is wasted if the fluid never flows into other portions of the grid. Third, if the fluid particles are to interact with solid objects, the grid resolution may need to be increased to achieve the desired fluid motion. An alternative to the grid-based method above is to use discrete fluid elements, or particles, to represent the fluid [LUCY77]. This gives the benefit of only having to calculate interactions between a finite set of elements and no memory space is wasted on storing field data in locations the fluid may never visit. One downside to this approach is the inherent error in creating a vector field from a weighted contribution of a finite set of particles. Increasing the number of particles can reduce the error, but this also increases the computation time and memory requirements. SPH was created independently in 1977 by Lucy and Monaghan to model astrophysical phenomena [MONA77]. However, the method is general enough to apply to any area of fluid dynamics and has been also used to simulate highly deformable elastic bodies [DESB96]. The central concept of SPH is the use of a smoothing kernel, a function that "smoothes out" the field variables of the particles, and increases the spatial extent of the particles. This kernel is applied to the Navier-Stokes momentum equation, the continuity equation and a state equation. Thus, a field variable of a particle is calculated from a weighted contribution of all particles surrounding it. The smoothing length, h, determines the spatial extent of the kernel and therefore what particles are included in the contribution. To evaluate the value of a field variable, such as density or pressure in SPH, a smoothed average is calculated according to the following integral:

$$f(\mathbf{r}) \approx \langle f(\mathbf{r}) \rangle = \int f(\mathbf{r}')\, W(\mathbf{r}-\mathbf{r}';h)\, d^3\mathbf{r}' \qquad (1)$$

where $f(\mathbf{r})$ is the field variable; W is the smoothing kernel; $\langle f(\mathbf{r}) \rangle$ is the smoothed average [CASH02].

The smoothing length, h, determines the particles that are used for the interpolation. In discrete simulations, the smoothed average of the field variable can be approximated by summations. SPH uses the concept of a Monte Carlo summation, where points are randomly distributed throughout the fluid. Thus, the probability of finding one of the points in the volume element dV centered around the point $\mathbf{r}$ is proportional to $\rho(\mathbf{r})dV$. Then, the smoothed average can be approximated by the summation:

$$\langle f(\mathbf{r}) \rangle \approx \sum_{j=1}^{N} f_j \frac{m_j}{\rho_j} W(\mathbf{r}-\mathbf{r}_j;h) \qquad (2)$$

where m_j is the mass of the jth point, ρ_j is the density at the position of the jth point, f_j is the value of the field variable being approximated at j. Because of the page constraints we are going to eliminate equations from our discussion. These equations can be found in [Wolf07]. One of the advantages of this method is the ease in calculating the spatial derivatives of field variables, such as gradients and Laplacian. To simulate the flow of a fluid, equations that describe the motion of the fluid must be evaluated. The equation that determines the force acting on given volume of incompressible fluid is the Navier-Stokes equation for incompressible fluids. SPH formulations have been defined to calculate the force on a particle i due to the pressure gradient. However, the forces between the particles are not symmetrical. To alleviate this problem, the average pressure between the particles is estimated [LIU03]. Equations are also defined to calculate the viscous force using the general viscous term. In addition, another form of the viscous force equation has been suggested by Müller [MÜLL03]. The external forces, such as gravity, do not depend on the SPH formulation and are just added to the pressure and viscous forces. In order to calculate the force equations, the density must be calculated at every time step. Unfortunately, for free surface flows, the density will be incorrectly low for surface particles, since there are no particles beyond the surface to be included in the density calculation. Monaghan suggests a method for density calculations [MONA94]. First, all particles are assigned an initial density (usually 1000 kg/m^3 for water) and the density only changes when the particles are in relative motion. The rate of change of density for a particle can then be calculated. An equation of state is also used to calculate the pressure of each particle during each time step. The simulation of the breaking dam by [ROY95] uses this method.

There are two criteria that smoothing kernels must meet [MONA92]:

$$\int W(\mathbf{r}-\mathbf{r}';h)d\mathbf{r}' = 1 \text{ and } \lim_{h\to 0} W(\mathbf{r}-\mathbf{r}';h) = \delta(\mathbf{r}-\mathbf{r}').$$

This allows many different kernels can be created to satisfy certain conditions needed in a given simulation. By far the most commonly used kernel is the Beta-spline kernel in [MONA92]. Desbrun suggests a "spiky" kernel for use in computing pressure forces, as the Beta-spline kernel above tends to cause particles to cluster since the gradient of the kernel approaches zero as particles move very close together [DESB96]. Using the standard kernel for the viscosity force calculation can cause the simulation to become unstable if low numbers of particles are used, as they are in real-time

applications. To overcome this, Müller has suggested a different kernel for the viscosity force calculation [MÜLL03]. Although the theory behind SPH is straightforward, there are many details to consider for a successful implementation of a general SPH solver, such as fixed and variable smoothing lengths, symmetrization of particle interactions, adaptable time steps and treatment of boundaries. The smoothing length, h, has a direct correlation to a successful implementation of SPH. If h is too small, there may not be enough neighbors in the support domain to exert forces on the particle of interest. However, if h is too large, local properties of the fluid at the particle of interest may be overly smoothed out, resulting in a low accuracy. The ideal number of neighboring particles (including the particle itself) has been determined to be 5, 21 and 57 respectively for 1, 2 and 3 dimensions [LIU03]. There are many ways to adapt the smoothing length to keep the number of neighboring particles relatively constant. According to [LIU03], the simplest way to evolve h over the simulation is by using the average and initial density. If the smoothing length is forced to evolve in both time and space, interactions between neighboring particles will not necessarily conform to Newton's Third Law, as each particle will have its own smoothing length. The following situation could exist: particle i includes particle j in its sphere of influence, but not vice versa (therefore, particle j would exert a force on i, but not i on j, in violation of Newton's Third Law). To overcome this inconsistency, several different methods have been introduced, such as averaging the smoothing length of each of the particles, and also taking the geometric mean. Time integration plays an important role in providing both a stable and fast simulation. The time step must be small enough to ensure that the Courant-Friedrichs-Levy (CFL) condition is met; while not so small that the simulation is computationally infeasible. The discrete SPH equations can be integrated using the familiar Leap-Frog, predictor-corrector and Runge-Kutta methods. For each of these methods, the time step must adhere to the CFL conditions above. To ensure this, the time step is modified to be proportional to the smallest spatial particle resolution, which in SPH is represented by the smallest smoothing length:

$$\Delta t = \min\left(\frac{h_i}{c}\right) \qquad (3)$$

where h_i is the smoothing length for particle i, and c is the speed of sound through the fluid. The treatment of boundaries has been of great interest since the development of SPH. The symmetric nature of the smoothing kernel produces a particle deficiency at the fluid boundary, as there are no particles on the other side of the boundary to include in the density summation. Therefore, several methods exist that address this problem, and the related issue of boundary penetration. There are generally three ways to prohibit boundary penetration by the particles: collision detection and response, force fields and virtual particles. Standard collision detection techniques, such as point-plane intersections work to keep the fluid particles from penetrating the boundary, but they are problematic. For example, if a particle is reflected off of a boundary, the simulation must also perform collision detection against all other boundaries after the collision response. This can lead to costly, sometimes unnecessary computations. The collision detection/response method also does not address the issue of particle deficiency at the boundary. Another method that seems to work better is to use a force field at the boundary [AMA05]. During each time step, each particle in the fluid receives a force component from every boundary. The force equation successfully prevents the particle from penetrating the boundary. The use of stiffness and dampening in the force equation allows for simulation of different boundary materials, such as concrete or foam. The force field method still suffers from particle deficiency at the boundary, but the ability to simulate different materials at the boundary is a useful tradeoff. Monaghan [MON94] suggested a popular solution to this problem. A grid of "virtual particles" is placed on the boundary. The virtual particles have the same density as the initial density of the fluid. As a fluid particle nears the boundary, the virtual particles exert a force on the real particle that resembles the Lennard-Jones potential. The use of stiffness and dampening in the force equation allows for simulation of different boundary materials, such as concrete or foam. The force field method still suffers from particle deficiency at the boundary, but the ability to simulate different materials at the boundary is a useful tradeoff.

3. SPH ACOUSTIC MODELING

Initialization

Initialization of the simulation occurs when an SPH scene file is loaded. This scene file specifies configuration for the following: global settings, fluid parameters, boundary conditions, the sound emitter and the sound receiver. The global settings configuration specifies the simulation time step and any external global forces, such as gravity. The fluid settings specify a cuboid shaped lattice of SPH particles defined by a position, width, height and length. The initial density, equation of state stiffness constant and viscosity constant are also defined here. For each boundary, the position, orientation, size and energy absorption properties are defined. The

boundary can be one of three different shapes: rectangle, disc and cylindrical surface. There can be more than one boundary in the scene file and complex structures such as rooms, halls and cathedrals can be constructed from individual boundary objects. The sound emitter specifies the input WAV file (the source sound), position and size. The emitter is based on a disc type boundary. The sound receiver specifies the output WAV file (the source sound with any reverberation from the environment) and a listening position

Simulation – Stage 1

Before a source sound can be processed through the simulation, the fluid must first reach an equilibrium state. The system is said to be in equilibrium when boundary forces acting on the fluid are equivalent to forces due to the pressure gradient internal to the fluid itself. To reach this state, the simulation must proceed from initialization without any sound source inducing energy into the system. The system runs in this state until a satisfactorily low energy is reached:

$$E_{\text{system}} = \frac{1}{2}\sum_{i=1}^{n} m_i v_i^{\,2} < L \qquad (4)$$

where L is a sufficiently small number (we use 0.00001 J). When equilibrium is reached, the simulation then proceeds to stage 2 below.

Simulation – Stage 2

After equilibrium is reached, the emitter is engaged, which propagates the source sound through the fluid. At the end of each time step, the sound receiver calculates the density at its location. The density at the receiver is compared with the reference density of the fluid. Any excess density will be recorded as a positive deflection in the output sound file, whereas low density will be recorded as a negative deflection.

New Sound Emitter Algorithm

The sound emitter is an object that acts as a boundary in the simulation, but is allowed to move according to a positive or negative deflection from the DC value in the input sound file. A positive deflection will result in a positive movement in the emitter along the x-axis. A negative deflection in the input sound file will result in a negative movement of the emitter along the x-axis. The magnitude of the movement along the x-axis is dependent upon a user-definable deflection factor that is set in the scene file. The magnitude of the x-axis movement is determined by the following equation:

$$x_t = \frac{s_t k}{32768} \qquad (5)$$

where s is the input sample (a 16-bit signed integer), t is the current sample and k is the deflection factor. In effect, this states that the emitter cannot be deflected more than k units on either the positive or negative x-axis. The emitter propagates the sound wave through the fluid by interacting with the fluid particles that are its neighbors. As the emitter moves, the Lennard-Jones potential induces a force on the fluid particles, causing the fluid to move in sync with the emitter, and thus the source sound itself.

New Sound Receiver Algorithm

The purpose of the sound receiver is to capture the vibrations in the fluid as an acoustic wave. When the simulation first enters stage 2, before the sound emitter starts to move, the fluid density is calculated at the receiver location. This density is stored as the receiver's reference density and is used in subsequent time steps to determine the positive or negative deflection from the reference. This deflection is then stored as a sample in the output sound file according to the following equation:

$$s_t = (\rho_0 - \rho_t) \cdot 32768 \qquad (6)$$

where s_t is the tth sample, ρ_0 is the receiver's reference density and ρ_t is the density of the fluid at the receiver's location. The magic number of 32768 is the sample conversion equation represents the maximum deflection of the pressure wave and is directly related to the size of the 16-bit integer, since we are converting fluid density values in the range of -1.0 <= pressure <= 1.0 to 16-bit sound samples. As in the input sound file, the sample is stored as a 16-bit signed integer. To ensure that background noise is filtered out, the output sound file's DC component is determined, the entire sound file is adjusted to make the DC component zero, and then the sound file is normalized.

4. IMPLEMENTATION

Since the SPH approach to acoustic modeling has not been attempted before to the best of our knowledge, a test plan was created to find the variable settings that produced the most desirable results. The simulation has the ability to load individual scene files, which are comprised of the simulation variables and an enclosed space generated by SPH fluid boundaries. The simulation variables that are exposed via the scene file are: timestep and fluid parameters (# of particles, initial volume, mass, viscosity and stiffness constants). In all simulation runs, we ignore temperature changes in the fluid. Acoustic energy is converted to heat by specifying different values for the boundary dampening constant. In addition to the simulation variables, complex environments can be created by assembling individual boundaries (walls) into rooms or halls. For this paper, three different types of boundaries

were created: rectangular, cylindrical and disc-shaped. In all test environments generated, a rectangular or box-shaped room was assembled from six rectangular boundaries. For a boundary, the size, orientation, stiffness and dampening can be adjusted to simulate materials with different acoustic properties, such as sound absorption (low stiffness, high dampening) or sound reflection (high stiffness, low dampening). The sound emitter can be of any size, but is restricted to a disc-shaped object. Since the emitter is also a boundary, stiffness and dampening can also be specified, but is usually set to high stiffness and low dampening so the sound can be propagated through the system efficiently. A deflection constant can also be specified for the emitter, guaranteeing that the emitter will never move beyond a certain distance from its rest position. The sound receiver specifies a single output WAV file and a world position vector.

The system uses C++ with performance in mind. To visualize the particle system in 3D, OpenGL was employed as the graphics engine. The SPH acoustic simulation developed for this paper is based on the core of a water splash simulation that was written by the first author. For the water splash animation, an SPH solver simulated water as a compressible fluid and used as an artificial equation of state to keep the fluid particles at a somewhat constant density. For the acoustic simulation, the bulk of the SPH calculations remained the same, with the equation of state for an ideal gas being substituted in place of the artificial equation of state for water. The following processing methods are provided by SPHSimulator class --

(a) Load() – Reads in a scene file and sets up the system to simulate the environment specified in the file. Particle creation and initialization is done during this step.

(b)StepSimulation – Steps the simulation forward one time step (the duration of the time step is defined in the scene file). The following pseudocode outlines this process:

(b1) For each particle, assign neighboring particles to neighbor list.

(b2) For each particle, compute its density according to Equation 11 in [Wolf07].

(b3) For each particle, compute its pressure using Equation 27 in [Wolf07].

(b4) For each particle, compute the force acting on the particle due to the pressure gradient (Equation 8 in [Wolf07] and the viscous force (Equation 28 in [Wolf07]. Integrate the equations of motion to find the new velocity and position according to the Leap-Frog method.

(b5) For each sound emitter, read the next sample in the WAV file and move the emitter accordingly (see SPHEmitter section below).

Class SPHKernel and Derivatives

The SPHKernel class is an abstract class the enables the simulation to use drop-in kernel equations. Three different kernel types are supported: Monaghan's B-spline kernel, Desbrun's spiky kernel and Müller's viscosity kernel.

Classes SPHGrid and SPHGridCell

When the SPHSimulator class performs an AssignNeighbors() action, a grid is superimposed on the particle collection. This grid is used as a spatial data structure, with cubic buckets having length h. The buckets store references to SPHParticle objects, which are used to perform a nearest-neighbor search algorithm. After the AssignNeighbors() action is performed, each particle will have a list of its neighbors that it uses for the rest of the SPH computations for the current time step.

SPHSoundEmitter

The sound emitter class is derived from one of the concrete classes based on SPHBoundary. In effect, it is a boundary that moves according to input samples from a WAV sound file. The emitter is centered on a rest position, and then vibrates according to the sound file defined for the emitter. A configurable maximum deflection is used to provide amplitude control for the produced sound wave. For each time step that is processed, the emitter reads the sample corresponding to the time step, and moves itself along its local x-axis by an amount defined by:

$$x = x_{\text{rest}} + \left(k s_t\right) \tag{7}$$

where x_{rest} is the rest position, k is the maximum deflection and s_t is the sample at time t such that $-1 \le s_t \le 1$.

SPHSoundReceiver

The SPHSoundReceiver class simulates a sound receiver, such as a microphone. When the SPH simulation starts, the receiver calculates the density of the fluid at its location based upon the weighted sum of the particle masses surrounding it It stores this initial value as a reference density. For each time step afterwards, the density is again measured by the same method and compared against the reference density. This deflection (negative or positive) is stored as a double-precision floating point value between -1 and 1. In addition, an output WAV file is created at the end of each time step that represents the sound wave the receiver detected up to and including that time step.

Class Camera

To visualize the particle system, a camera with the ability to move in all three dimensions as well as rotate around all three axes was created. A Camera object is created for each OpenGL window used in the simulation (currently only one view is used). A reference to this camera is then passed into the SPHSimulator, and is then passed again to the SPHParticle and SPHBoundary classes during the rendering phase.

5. SIMULATION RESULTS

For the simulation runs, two different WAV files were created. The first was designed to produce a very short impulse and very quickly decay to zero after the burst (Figures 1-2). The impulse is an attempt to recreate the traditional method of determining an environment's reverberation properties, such as firing a pistol filled with blanks. The second WAV file is a music application of the simulation and consists of a brief Latin piano melody with a shaker accompaniment (Figures 3-4). To compare the output from the simulator to a professional reverberation effect, an additional WAV file was created by processing the piano sample with the "Hall 1" reverberation effect from a Yamaha S90 ES synthesizer and recording the result. In Figures 1 through 4, the time domain and frequency domain graphs for both the impulse and piano melody are displayed.

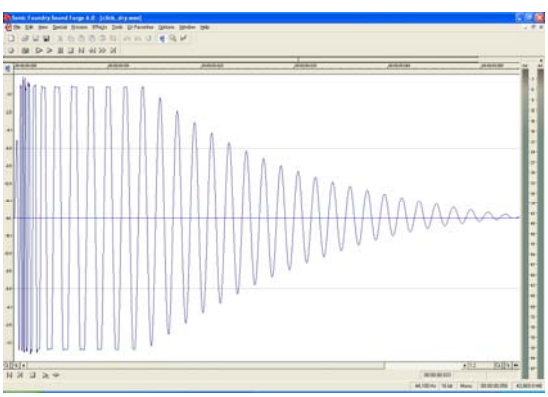


Figure 1: Time and domain graph of impulse.

Simulation time step	2.268×10^{-5} s
Fluid stiffness constant	119.215
Fluid viscosity constant	1.0×10^{-4}
Smoothing length	2.130×10^{-2} m
Mass per particle	1.935×10^{-6} kg
Particle rest density	1.29 kg/m^3
Environment volume	5.488×10^{-2} m^3
Boundary stiffness constant	1.0×10^{9}
Boundary dampening constant	0.1
Emitter radius	2.5×10^{-2} m
Emitter stiffness constant	3.0×10^{8}
Emitter dampening constant	0.1
Emitter deflection factor	0.001 m

Table 1: Samples of values used in simulation.

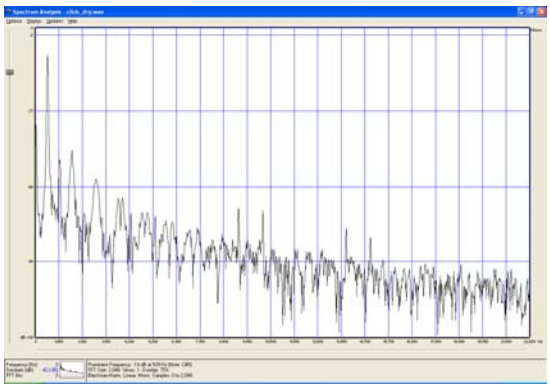


Figure 2: Frequency domain graph of impulse.

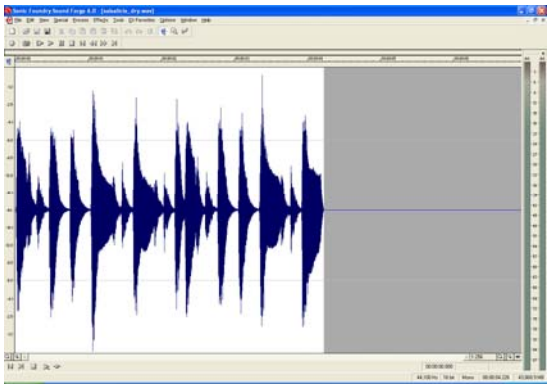


Figure 3: Time domain graph of piano melody.

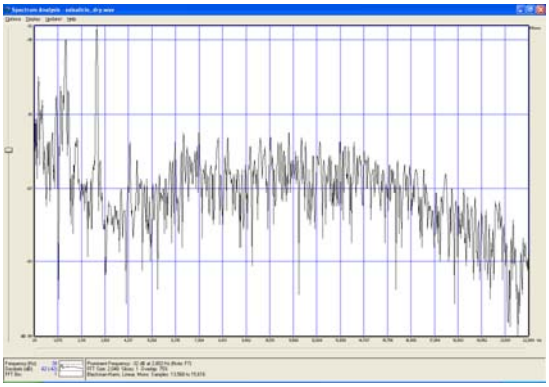


Figure 4: Frequency domain graph of piano melody.

The first simulation scene was created by inputting a "best guess" value for each parameter in the simulation. The time step is set at 1/44100, or the sampling frequency of the input WAV file. The fluid stiffness constant was generated by taking the Ideal Gas Equation and solving it for the appropriate constant *k* for air for a known pressure and density. The viscosity constant was set as low as possible without sacrificing stability in the SPH equations for the specified time step duration. The boundary stiffness and dampening were set so as to generate a high reflection factor with minimal acoustic energy loss. The emitter size and deflection factor were set to appropriate values to simulate a loudspeaker in the environment (Table 1, more details in [Wolf07]). Although the simulation did produce reverberation, the frequency spectrum in Figure 6 shows that the output impulse had a power gain around 4000 Hz. This data agrees with the author's observation of hearing the reverberation as a hollow, metallic sound. The second simulation run using scene file #1 used the piano melody as the input WAV file. As with the impulse WAV, a hollow, metallic reverberation was observed. One interesting artifact with this simulation is observed around 5500 Hz: due to the smoothing effect of the SPH interpolation, the sound was effectively low-pass filtered starting around 5500 Hz. This produced a muddy sound quality in the output WAV. Using more fluid particles may produce better results, but the resulting computation time would be too great to be of any great value. We performed several (six) similar experiments with different scene files which have been omitted due to the page size constraints, but can be found in [Wolf07].

The results from the simulation of scene file #1 show promise. Using the impulse WAV as input to the simulation, the resulting output is similar in sound. Reverberation can be heard as the impulse decays, and has a hollow, metallic quality. In the time domain graph shown in Figure 3, the sound receiver's initial density is incorrectly high, most likely because of minor fluctuations in density at the receiver position due to the chaotic movement of the fluid particles.

6. FUTURE DIRECTIONS

Before further application possibilities are considered, the accuracy of the reverberation results could be improved. The equation of state is the most obvious place to start to improve the simulation. In this paper, the equation of state used was based on the Ideal Gas Law. The stiffness constant that was required bordered on being too high for the simulation to remain stable at reasonable timesteps. In addition, we did not simulate the thermal properties of the fluid, as we assumed the temperature would be more or less constant and any thermal conversion of the acoustic energy would be simulated in the boundary dampening factor.

Another possibility for improvement is using different smoothing kernels for the density, pressure gradient and viscosity equations. Smoothing kernels have been known to produce odd behavior if not properly adapted to the application [MÜLL03]. Liu and Liu have devoted an entire chapter of their SPH book to constructing custom smoothing kernels [LIU03]. For the pressure gradient and viscosity equations, care must be taken to ensure that the kernel derivative and Laplacian exist.

Although this paper focused solely on simulation of reverberation, another possible application of the simulation, once properly adapted, is to perform general acoustic modeling. This type of simulation is useful in many industries, including building design, automobile, aircraft and marine design and music production. To this end, an equation of state that includes thermal energy must be used, as well as the simulation of the acoustic absorption and refraction effects of different materials. For the equation of state, the standard Ideal Gas Law would still apply for simulation of air, if simulated in the well known $PV = nRT$ form. However, a new boundary design would have to be considered that allowed the sound to propagate through the boundary material. This could be accomplished by treating the walls and other obstacles not as boundaries, but as solid masses of particles that the fluid interacts with. The particles could be connected by a damped mass-spring system that would allow sound wave propagation and thermal energy transfer. With this type of a system, the simulation could easily handle a very dynamic environment, where the boundaries and other obstacles are able to move in time.

7. CONCLUSION

This paper has presented the history and theory of acoustic modeling of reverberation, as well as the history of the Smoothed Particle Hydrodynamics method of fluid mechanics simulation. The lack of a general solution to the acoustic modeling problem when dealing with complex environments was a drawback of earlier methods. New algorithms were developed for sound emission and reception. Using the SPH fluid mechanics algorithm as a base, An SPH simulation application utilizing C++ and OpenGL was developed that allowed the testing of the new algorithms. We have provided a first step to a more robust solution which opens door towards further research using our formulation.

The system as implemented is still far from being a solution to the general acoustic modeling problem. Still our simulation results simulate the reverberation to a small degree of accuracy. This shows that the SPH algorithm is a tool for acoustic modeling simulation of complex environments. When a faster solution is obtained, the use of SPH as a tool in acoustic modeling would prove valuable for music studio design, architecture and acoustic dampening for the automotive, and applications for aircraft and marine industries.

8. REFERENCES

[Cash02] Cash, J. 2002. Modeling the accretion stream in polars. Ph.D. Dissertation, Department of Physics and Astronomy, University of Wyoming.

[Desb93] Desbrun, M. and Gascuel, M. 1996. Xue, L. 2004. A new paradigm for animating highly deformable bodies. Computer Animation and Simulation '96. 61-76.

[Ever01] Everest, F., Master handbook of acoustics, Fourth edition, McGraw Hill, 2001.

[Funk98] Funkhouser, T., et al. 1998. A beam tracing approach to acoustic modeling for interactive virtual environments. *Computer Graphics (SIGGRAPH '98)* 21-3.

[Li64] Li, W and Lam S., Principles of fluid mechanics, Addison-Wesley, 1964.

[Liu03] Liu G and Liu M, Smoothed particle hydrodynamics: a meshfree particle method. World Scientific, 2003.

[Lucy77] Lucy L, 1977. A numerical approach to testing of the fission. The astronomical Journal, Vol 82, no. 12.Li, W and Lam S., Principles of fluid mechanics, Addison-Wesley, 1964.

[Mona77] Monaghan J. Gingold R, 1977. Smoothed Particle Hydrodynamics theory and applications to non=spherical stars, Montly notices of the Royal Astronomical Society 181(2):375-389.

[Mona92] Monaghan J 1992. Smoothed Particle Hydrodynamics. Review of Astronomy and Astrophysics, 543-574.

[Mona94] Monaghan J et a, 1994, Simulation of free surface flows with SPH. ASME Sympsium on Computation Methods in Fluid Dynamics.

[Mull03] Muller M et al 2003. Particle based fluid simulation for interactive applications. Eurographics/SigGraph Symposium on Computer Animation.

[Roy 95] Roy T. 1995. Physically based fluid modeling using smoothed particle hydrodynamics. MS University of Illinois at Chicago, USA.

[Whit01] White P. 2001. Advanced Reverberation Part 1. Sound on Sound, October 2001.

[Whit01] White P. 2001. Advanced Reverberation Part 2. Sound on Sound, November 2001.

[Wolf07] Charles Thomas Wolfe, Acoustic Modeling of reverberation using smoothed particle hydrodynamics, MS Thesis, University of Colorado, Colorado Springs, CO, pp.1-63, 2007.

.

Geometric Feature Deletion Through Freeform Feature Recognition

Thomas R. Langerak
Delft University of Technology
Landbergstraat 15
2628 CE Delft
Netherlands

T.R.Langerak@tudelft.nl

ABSTRACT

The deletion of geometric features is a much used operation in the process of shape modeling; by deleting geometric features, the smoothness of a surface can be increased. In this paper, a new method is presented for the deletion of geometric features, which is based on a morphological understanding of the feature. The method first parameterizes the feature and then deletes it using a parameter-based shape manipulation. The advantage of this new method is that a geometric feature can be deleted while maintaining the semantics of both the feature and its embedding shape. To be able to parameterize the feature, an improved version of an existing evolutionary feature recognition procedure is developed, which constructs a feature template that matches a feature on the target model. Application examples are given and the robustness of the method is discussed.

Keywords

Freeform features. Feature recognition, Feature deletion, Evolutionary computation

1. INTRODUCTION

When thinking of a surface with distinct geometric patterns, it is often natural to think of this surface as a base surface with features attached to it. For example, if one would view a landscape with rock formations, then it is only natural to see the rock formations as being *placed* on the landscape (see Figure 1). When a piece of skin swells up after an insect bite, then the bump that originates is seen as 'additional' shape and the original skin surface is still perceived, regardless of the fact that, of course, no additional material has been created.

The geometric patterns of the rock formations and the bump in these examples are *features*. Features can be defined as connected regions of the surface that can be easily separated from the rest of the surface [Rib01]. Geometric features are often used as a modeling tool by which distinct characteristics can be added to an otherwise smooth surface. Features can even be further formalized by adding parameters to them that have a functional relation to the feature shape.

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

Figure 1: Rock formations in Monument Valley, Arizona

In other words, parameter values can be used to modify the shape in a pre-defined and predictable way. This way features can be used to significantly improve the efficiency of shape manipulations. In addition, the recognition of the features can be used or is sometimes even necessary in a process of reverse engineering. However, geometric features can also occur as noise or byproducts of other geometric modeling operations, and in this case can be classified as unpredicted, unnecessary and even unwanted regions of a surface. If this is the case, then a logical wish is to remove them from the surface and to (re)create a smooth surface. Smooth surfaces can be more efficiently represented and as a result the

computation time for many operations on the model is significantly decreased.

Current methods for dealing with geometric features are based on an geometric approach to the problem of feature deletion, where features are considered to be a geometric anomaly. However, in this case, an 'understanding' of the feature is lacking, and consequently the result of the feature removal procedure is imprecise and requires post-processing of the geometric data. In this paper, a new approach to feature deletion is proposed that first recognizes a target feature. The parameterization of the feature that can be derived from this recognition can then be used to delete the feature using a parameter-based shape manipulation.

2. EXISTING LITERATURE

To find approaches to remove features, one does not have to look far. Different approaches exist, and within each approach different techniques can be used. Ribelles et al. [Rib01] propose a method that combines several of these techniques for the removal of features from polyhedral data. In their work, they first split a polyhedral model into several subsections using an intersection plane. They then classify and cluster triangles in the polyhedral model corresponding to their position relative to the intersection plane. In the clustered data, they identify features and separate these from the rest of the polyhedral model. Finally, they use a hole filling algorithm to repair the holes that are left behind by the removal of the feature. Although the method neatly combines several techniques, it is only demonstrated for use with regular polyhedral models, i.e. models with only surfaces that are either perpendicular or parallel to the other surfaces in the model. This implies that the technique cannot be used in freeform models.

Another approach that targets regular features is that of Venkataraman et al. [Ven02a]. The authors detect features by finding a closed boundary loop, i.e. a closed loop of edges along the boundary of the feature. They then use extension and shrinking of the faces that border on this boundary loop to remove the features. In addition to the fact that their method only targets regular features, it is also unclear to what degree the approach can be automated. In [Ven02b] the authors present a similar method that specifically targets blend features.

Lee et al. [Lee05] propose a feature suppression method that uses a cellular modeling approach to identify features and then collapses the geometric elements of a feature such that it can be removed safely from the shape model. They store every collapse operation, in order to later be able to unsuppress the feature. Again, this method mainly targets regular features and suffers from all the disadvantages of history-based modeling.

3. PROBLEM DEFINITION

In this paper we assume that freeform surfaces can be described by a discrete set of elements, which we call *shape configuration elements*. A B-spline surface can be described by a set of control points; polyhedral meshes can be described as a collection of points, vertices and faces. This means that a surface that can be described with n elements, be it a polyhedral mesh or a B-spline surface, can be formalized as $S = \{s_1, \ldots, s_n\}$. Note that the shape configuration elements are not part of the shape themselves. For each representation type, it is assumed that an *evaluation function* exists, which translates the shape configuration element to actual geometry, e.g. for a B-spline representation the shape evaluation function is the well-known de Casteljau's algorithm.

In our approach, features are embedded in a freeform target surface. If the discrete set of elements that represents the feature shape, denoted E, is a subset of that of a surface S, then S is called the *base surface* of E. The deletion of the feature can now be defined as the transformation $S^{original} \rightarrow S^{deleted}$. Note that for each element s_i it holds that if $s_i^{original} \neq s_i^{deleted}$, then it holds that $s_i \in E$. Hence, to delete a feature from a surface, an operation on the feature is needed that transforms the feature shape (and with it the shape of its base surface) from its original state to the 'deleted' state. A large advantage of this approach compared to that of Lee et al. [Lee05] is that the feature information is not removed from the model. The only difference between the 'deleted' state of the feature and the original state of the feature is that its parameter values have been set to a state that corresponds to $S^{deleted}$. Therefore, to reinstate the feature, all is needed is another parameter change.

Features in this paper are taken to be parametrically controlled shapes. This means that, besides the shape information, a feature also contains parameter information and, more importantly, information on how the parameters influence the shape. We define a feature as follows:

A ***feature*** $F\left(E^0, P, \mu\right) = E$ *is a function that, given a basic shape configuration* E^0*, a set of parameter values* P *and a parameter mapping* μ*, results in a shape configuration* E*, where:*

- $E^0 = \{e_1^0, \ldots, e_n^0\}$ *is the set of shape configuration elements of the basic shape configuration*
- $P = (p^1, \ldots p^m)$ *is a vector of parameter values*
- $\mu(E,P) = \bigcup_{i=1}^{n} \bigcup_{l=1}^{m} \{\mu_i^l (e_i \in E, p^l \in P)\}$ *is a parameter mapping*
- $E = \{e_1, \ldots, e_n\}$ *is the resulting shape configuration*

The basic shape configuration E^0 can be any valid shape configuration of the feature. Without loss of generality we therefore assume that the basic shape configuration occurs when the vector of parameter values $P^0 = (0, \ldots, 0)$. This definition implies that a feature can have multiple states, each of which can be derived from an original state (the basic shape configuration of the feature), a vector of parameter values and the parameter mapping.

Information on how the parameters influence the shape is contained in the parameter mapping, which is a set of *nm* functions μ_i^l that describe the influence of parameter p^l on shape configuration element e_i. Although we have implemented more sophisticated parameter mapping functions, in this paper it suffices to assume that a parameter mapping function μ_i^l has the form of a translation vector T_i^l, such that $\mu_i^l(e_i, p^l) = p^l T_i^l e_i$.

Combining this with what has been said earlier in this section, we can pose the following problem statement:

Feature deletion is the problem of, given an original surface $S^{original}$ *that contains one or more features* $E \in S^{original}$*, to find an alternative state* $S^{deleted}$ *of the surface, for which it holds that* $E^0 \in S^{deleted}$.

Note that this problem only requires us to find the feature $F(E^0, P, \mu) = E$, for once E, P and μ are known, E^0 can be backwards computed as $E^0 = F(E, -P, \mu)$. Hence, the problem of feature deletion can be partly solved by using feature recognition.

4. FEATURE RECOGNITION

Feature recognition has been a popular research topic in the last decades of the previous century. However, although many approaches to the recognition of regular features have been proposed, very little work has been done on the recognition of freeform features.

Song et al. [Son05] and Vergeest et al. [Ver01] have proposed a *template matching* approach, in which an instance of a pre-defined feature type, a feature *template*, is iteratively compared with a target surface. They construct a function that takes the parameter values of the template feature as input variables and results in a measurement of the geometric distance between the shape of the template feature and that of the target surface. By then applying a function minimization routine, parameter values can be computed for which the geometric distance between template and target surface is minimal (see Figure 2). The template that is used in the procedure is an instance of a user-selected feature type from a collection of pre-defined features, the *feature library.*

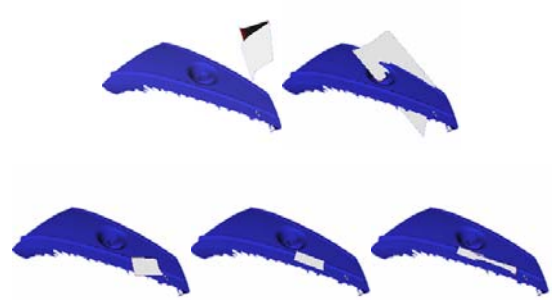


Figure 2: Consecutive steps in a template matching procedure, in which a template is iteratively matched to a shape model

Although the template matching method is successful in finding relatively simple features, the usefulness is limited by the size of the feature library. Before the template matching method can be used to recognize a feature on a target surface, a feature type that resembles this feature must first be defined and added to the library. Part of the recognition process therefore lies with whoever designs the feature types in the feature library.

An improvement on the template matching method has been proposed by the author [Lan07]. In this method, an evolutionary approach was used to find a minimal configuration for the template feature. In this paper, a more elaborate version of this method will be given and it will be discussed how this method can be used for the problem of feature deletion.

Evolutionary computation is a technique that has often been used in the past and that has its roots in the field of biology. Evolutionary methods mimic the

biological concept of natural selection, which is based on the principle of 'survival of the fittest'. In nature, some organisms have higher chances of survival and procreation because they are better adapted to their environment. The fact that they are better adapted to their environment is due to the fact that their genetic structure, their 'DNA', contains specific genes. Because their chance of procreation is bigger than that for other individuals, they are likely to pass on these genes to a next generation. As a result, organisms in this offspring generation are even better adapted to their environment. Due to this mechanism, populations adapt to their environments and 'improve' with each generation. In an evolutionary computation method, possible solutions to a problem are viewed as organisms, of which the genetic elements are the variables that play a role in finding the solution to the problem. In an evolutionary computation method, natural mechanisms that play a role in evolution must be implemented. The most important of these mechanisms are: genetic structure, crossover, mutation, fitness and selection.

Genetic structure

In the feature recognition problem, the 'problem' under consideration is that of shape matching. The organisms in an evolutionary freeform feature recognition problem are, logically, feature instances, and the genetic structure therefore contains all the variables that are needed to compute the shape of a feature: the parameter values of the feature, but also its parameter mapping functions. We define the gene sequence $\gamma = \left\{\gamma_1, \ldots, \gamma_{(n+1)m}\right\}$, such that $\gamma_i = P_i$ for $i \le m$ and $\gamma_i = T_{\mathrm{mod}(i-m,n)}^{\lceil (i-m)/n \rceil}$ for $i > m$.

Crossover and mutation

Crossover and mutation determine how the inheritance of genes from one generation to the other takes place. The gene sequences of two individuals, the parents, are combined such that the gene sequence of each of their offspring is a (different) combination of parts of the gene sequences of their parents. Sometimes, during the process of inheritance, the information in the gene sequence is distorted. This process is called mutation and ensures that the 'gene pool', the collection of all genes that are available in a population, is constantly in motion. Mutation introduces new properties into the gene pool, both good and bad. However, only the good properties survive the natural selection process. Mutation plays a particularly important role in evolutionary computation methods. Because the available computation time and memory is limited, populations are often simulated with less-than-natural sizes. To counter the effect of inbreeding, mutation rates are typically higher than in nature.

Fitness and selection

The extent to which an organism is adapted to its environment is expressed in the *fitness*. In the case of feature recognition, the fitness is implemented as the geometric distance between feature shape and target shape. As a measurement of this distance, the Hausdorff distance is used, which is defined as $H(S_1, S_2) = \max_{s_1 \in S_1}\left(\min_{s_2 \in S_2}\left(dist\left(s_1, s_2\right)\right)\right)$, where $dist$ is the Euclidean distance between two points. The higher the fitness of an organism, the higher the probability that it will generate offspring.

Several variables play a large role in any evolutionary computation:

- The *population size* Π indicates the number of organisms in a population. Each individual in the population signifies a single probe in the search space. Therefore, the larger the population size, the faster (in terms of number of generations needed) the procedure converges to an optimal solution to the feature recognition problem. However, a large population size also means a higher computation time.
- The *selection size* σ indicates the extent to which the fitness has an influence on the chance of selection. If σ is set to a high value, then even the less successful features in a population have a chance of generating offspring in the next generation. In this case, the *genetic diversity* (i.e. the number of different genes) of an offspring feature population remains high, but the selective power of the evolutionary mechanism decreases.
- The mutation probability χ indicates how likely it is that mutation occurs during the creation of an offspring population. If χ is high, then a higher number of new genes is introduced in each generation. This can either be an advantage or a disadvantage: the higher the number of new genes, the less likely the feature recognition is to get stuck in a local minimum; however, too many new genes may lead to an overload of new search directions to be investigated and promising search directions may therefore unjustly be discarded.
- The mutation rate φ determines the amount of mutation that occurs. The higher the mutation

rate, the larger the difference between a mutated gene and the parent genes that contributed to it.

Considering what has been said in this section, an evolutionary feature recognition algorithm can be given as follows:

1) An initial feature population $gen_1 = \{F^1, \ldots, F^\Pi\}$ is generated, where each F is an instance of a pre-defined type from the feature library. A random value is assigned to the parameter values.
2) The fitness of each feature is computed and features are ranked according to their increasing fitness value $f(\ )$, such that $f(F^1) \leq \ldots \leq f(F^\Pi)$
3) A new population gen_{i+1} is generated as follows. For each feature instance in gen_{i+1}:
   - Two features, F^{mother} *and* F^{father}, are selected from gen_i with probability $P(x) = \frac{2}{\sigma\sqrt{2\pi}} e^{\frac{-x}{2\sigma^2}}$ which is a one-sided Gaussian distribution of the fitness over the domain $x \in [0, \infty)$ with standard deviation σ.
   - Each gene of a new individual F^{new} is copied either from F^{mother} *or* F^{father}. Both have an equal chance of being selected to carry on a gene. The chance of mutation is computed using a uniform distribution, and if mutation occurs, then the value of a gene is multiplied by a factor that is computed using a Gaussian distribution with a mean of 1 and a standard deviation χ.

The procedure terminates when one of the following conditions is met:

- The fitness no longer increases over generations, or the increase in fitness is slow, i.e the difference between $f(F^1)$ in gen_{i-1} and $f(F^1)$ in gen_i is smaller than a user-given threshold ε_1.
- The fittest feature in a population is an acceptable solution to the feature recognition problem, i.e. $f(F^1)$ in gen_i is smaller than a user-given threshold ε_2.

Using the evolutionary algorithm, features can be recognized providing the variables of the algorithm are set to a correct value. However, if both parameter mapping functions and parameter values are included in the genetic structure of a feature, then the amount of genes is large. As a result, the population size must be set to a very large value, which in turn leads to a large computation time. To counter this problem, we divide an evolutionary feature recognition procedure into two steps:

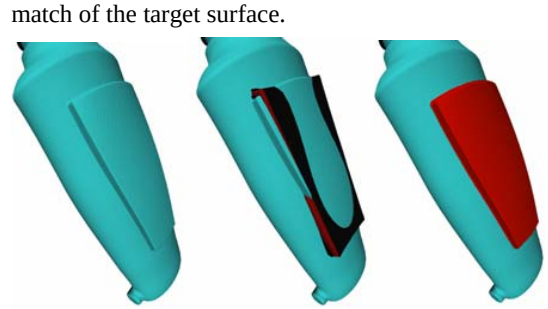
Step 1

During the inheritance of genes, only the genes that represent the parameter values are subjected to mutation. As a result, the genes that represent the parameter mapping functions remain stationary and the population size can be kept small. The result of this step is a configuration of a feature template that roughly matches the target shape, similar to that of the template matching method. Although the match is not perfect, the end result of this step is a good starting point for step 2.

Step 2

Starting from the configuration that was found in step 1, a new evolutionary procedure is started. Because this procedure starts from a reasonable approximation of the optimal configuration of a template feature, the population size can again be kept small. This time, the genes that represent parameter values are fixed and the genes for the parameter mapping are included in the inheritance. During this step, a much more accurate match is found. Because the parameter mapping functions are now subjected to mutation, the resulting feature is no longer an instance of a feature that is available in the library. The shape of the feature has been adapted to that of the target surface, while maintaining the functional relation between parameters and shape.

Examples of the result of the two steps are shown in Figure 3. The main advantage of dividing the feature recognition process into two steps is that the population size can be kept small and hence the computation time is reduced. In step 1, only a small amount of genes is used; in step 2, the initial configuration of a feature is already a reasonable match of the target surface.



(a) (b) (c)

Figure 3: Results of the two steps of the evolutionary feature recognition algorithm, (a) the target shape model, (b) a rough match of a feature template to the shape and (c) an improved match, in which the feature template is adapted to the target shape model.

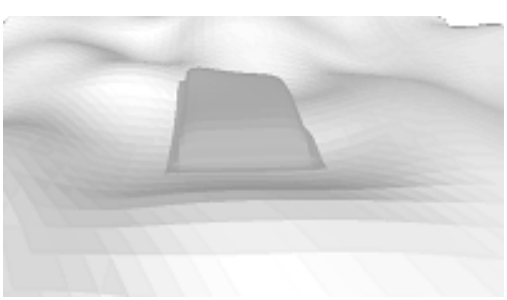
 ISBN 978-86943-15-2

Evolutionary feature recognition is an improvement on existing feature recognition techniques in that it does not require a rigid similarity between a feature on the target surface and a pre-defined feature in the feature library. Although at the starting point of the recognition procedure, instances from pre-defined feature types are used, during the procedure these become adapted to the target surface and the feature definition is slowly changed. However, although the recognition procedure results in an accurate match between a feature template and a target surface, the result is still a template, not a portion of the target surface. In other words, the shape configuration of the feature template E is not a subsection of S. This means that we are still not able to use the feature template that results from the recognition process to delete the feature on the target surface. In the next section we therefore describe a technique to find the base surface of the feature, using the recognized feature as a starting point.

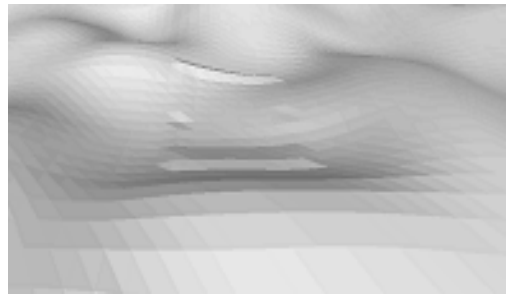
5. BASE SURFACE DETECTION AND FEATURE DELETION

To be able to use the feature template that was found in the feature recognition procedure, the observation must be made that although the shape configurations of the feature template $F'\left(E'^{0}, P', \mu'\right)$ and the target feature $F''\left(E''^{0}, P'', \mu''\right) = E^{target} \subseteq S$ are not identical, this does hold to a high degree for the parameter mapping functions, such that $F'\left(E^{target}, -P', \mu'\right) \approx E''^{0}$. Remember that E''^{0} is the surface that remains after the feature F'' has been deleted. This observation allows us to find an estimate for the base surface of the target feature.

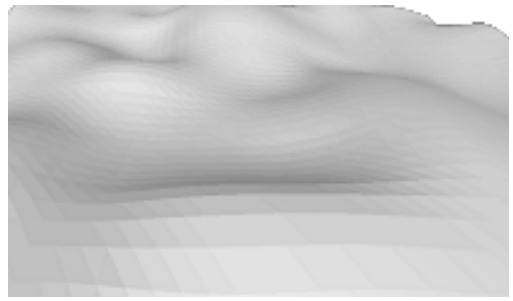
Song et al. [Son05] show how a matched template can be used to deform the feature that it has been matched to. They use a variant of lattice-based deformation that was proposed by Sederberg and Parry [Sed86].



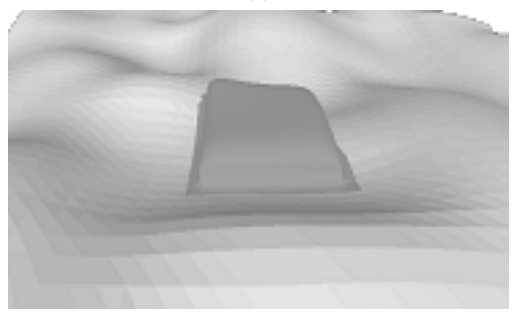
(a)



(b)



(c)



(d)

Figure 4: Steps in the base surface detection: (a) the original surface, (b) after deletion of the feature, with some residual effects, (c) after smoothing the surface (d) after reconstruction of the feature.

By using this approach, the recognized feature can be deleted by setting the parameter values of the template to 0, but because the match between the template and the target surface is not perfect, there is some residual effect (see figure 4b).We have implemented an approach to improve on this result, so that a feature can be removed without residual effect:

1. An evolutionary feature recognition is used to find the configuration of a feature template such that it has a minimal distance to the target surface. This step equals the two steps mentioned earlier.
2. The parameter values for the template feature are set to 0 and the correspondence between template and target surface is used

to deform the target surface using a lattice-based approach.

3. The resulting surface is smoothed
4. A new feature recognition procedure is started, in which the basic shape configuration of the feature is a subset of the surface that results from step 3.

For the smoothing in step 3, any known smoothing algorithm can be used. There is not a 'correct' solution to the base surface detection problem, and therefore the exact result of the smoothing procedure is irrelevant, as long as it leads a smooth surface. We used a simple Laplacian smoothing algorithm, which in practical cases provided a sufficiently smooth base surface.

Note that the feature recognition in step 4 can be done much faster and with much more efficiency than the recognition procedure in step 1, because:

- The configuration of the feature is constrained by its position on the target surface.
- The exact feature type of the target feature is known, as it was constructed during step 1.
- Approximate parameter values for the feature are known.

As a result, the population size for the second feature recognition procedure can be kept very small, and the computation time is small as well.

In the end result of step 3 , the targeted geometric feature has been deleted, leaving a smooth surface. However, earlier it was stated that the preservation of the semantics of the deleted feature is important, for example if one wants to undo the deletion. For this reason, the surface that results from step 3 is assumed to be the base surface of the original feature; in step 4 the feature is reconstructed using this base surface. Although the reconstructed feature that is the result of step 4 is not an exact match of the original target feature, we found that the distance between the two was very small and could not be visually discerned (see figure 4a vs. figure 4d).

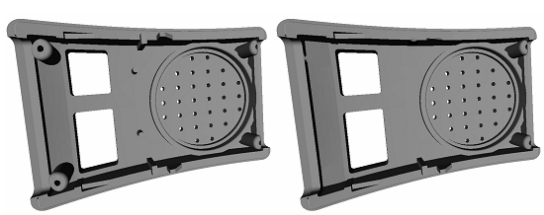


Figure 5: Test models (left) and the result of the feature deletion procedure (right).

To test the proposed method, it was subjected to several test models, of which three are shown in figure 5. The models were in the form of polyhedral meshes containing in between 45000 and 76000 polygons. To be able to recognize and delete the features on these models, first feature type definitions were created; to guard the objectivity of this definition, i.e., to guarantee that the definitions were not an a priori match to the features on the test models, the definitions were created by a colleague on the basis of a verbal description of the feature in question. Then, to guarantee that the correct feature was deleted, first regions of interest were selected around the targeted features on the test models. Finally, these regions of interest were subjected to the proposed method, the result of which can be seen in figure 5.

From the results of the tests, it was concluded that the proposed method can successfully be used to delete geometric features. However, some situations were found in which a feature could not be deleted, due to the fact that (a) the feature interfered with other features or parts of the shape, (b) the feature was attached to multiple base surfaces, and (c) the feature was nested, i.e. its base surface could be considered to be another feature (see Figure 6).

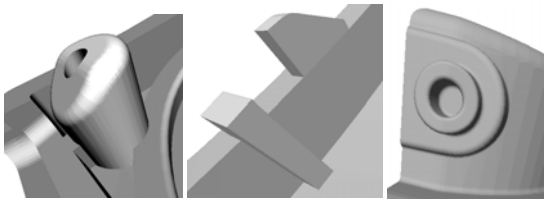


Figure 6: Three cases in which the proposed method failed: (a) interference with other shape parts, (b) multiple base surfaces and (c) nested features

6. RESULTS AND DISCUSSION

A method was proposed that deletes a feature from a target surface. The method first recognizes the feature by matching a feature template to the target shape, and then uses the correspondence with the template feature to reduce the feature to its basic shape configuration. The resulting surface is smoothed and in a final step the original feature is reconstructed. The method was implemented for use on polyhedral meshes, but can be generalized to B-spline surfaces because in the theory no specific assumptions were made on the shape representation type. To implement the method for B-spline surfaces, only the smoothing of a surface must be addressed.

It was found that the feature deletion method is successful in deleting features from polyhedral meshes. In the testing experiments, several problems occurred, some of which were shown in figure 6. First, it was found that the feature deletion method does not fully work in the case where two or more features interfere. Dealing with feature interference is a known problem of feature recognition and it can be expected that if improved feature recognition methods are developed, then the deletion of these features also improves. However, the difficulty of dealing with features that have multiple base surfaces is a minor but systematic shortcoming of the proposed method, which should be addressed in future research. Nested features cannot be unambiguously deleted without additional user input on what feature exactly has to be deleted.

Another problem with the feature deletion method presented in this paper is that it is unable to deal with eliminative features, i.e. holes. This is a consequence of a shortcoming of the proposed feature recognition method, not with the methodology that was proposed for feature deletion. A different approach to feature recognition is necessary for this type of features, which is also left to future research.

7. ACKNOWLEDGMENTS

The research project 06240 is supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Ministry of Economic Affairs, The Netherlands.

8. REFERENCES

[Lan07] Langerak, T.R., Vergeest, J.S.M, An Evolutionary Strategy for Free Form Feature Identification in 3D CAD Models, Proceedings of the WSCG conference, Plzen, 2007

[Lee05] Lee, K.Y., Armstrong, C.G., Price, M.A., Lamont, J.H., A small feature suppression/ unsuppression system for preparing B-rep models for analysis, Proceedings of the 2005 ACM symposium on Solid and physical modeling, June 13-15, Cambridge, USA, 2005

[Rib01] Ribelles, J, Heckbert, P.S., Garland, M., Stahovich, T., Srivastava, S., Finding and removing features from polyhedra, Proceedings of the ASME DETC conference, September 9-12, Pittsburgh, USA, 2001.

[Sed86] Sederberg, T. W., Parry, S. R., Freeform Deformations of Solid Geometric Models, Computer Graphics, Vol. 20, No. 4, pp. 151–160, 1986

[Son05] Song, Y., Vergeest, J.S.M., Bronsvoort, W., Fitting and manipulating freeform shapes using templates, Journal of Computing and Information Science in Engineering, Vol. 5, No. 2, pp. 86-94, 2005.

[Ven02a] Venkataraman, S., Sohoni, M., Reconstruction of feature volumes and feature suppression, Proceedings of the 7th ACM symposium on Solid modeling and applications, June 17-21, Saarbrucken, Germany, 2002

[Ven02b] Venkataraman, S., Sohoni, M., Rajadhyaksha, R., Removal of blends from boundary representation models, Proceedings of the 7th ACM symposium on Solid modeling and applications, June 17-21, Saarbrucken, Germany, 2002

[Ver01] Vergeest, J.S.M., Spanjaard, S., Horváth, I, Jelier, J.J.O., Fitting Freeform Shape Patterns to Scanned 3D Objects, Journal of Computing and Information Science in Engineering, Vol. 1, No. 3, pp. 218-224, 2001

Spectral Modeling and Virtual Restoration on a Polychrome Medieval Sculpture

Sylvain Dumazet
Applied Mathematics and Systems Lab., Ecole Centrale Paris, grande voie des vignes, Châtenay-Malabry, France
sylvain.dumazet@ecp.fr

Patrick Callet
Applied Mathematics and Systems Lab., Ecole Centrale Paris, grande voie des vignes, Châtenay-Malabry, France
patrick.callet@ecp.fr

Ariane Genty
INOLAM, 26 bis rue Kléber, Montreuil, France
arianegenty@wanadoo.fr

ABSTRACT

The presented work is led in the framework of a general collaboration between three academic labs, industrial partners and Cultural institutions (Centre des Monuments Nationaux, Louvre museum). Such a pluridisciplinary work always in progress at Ecole Centrale Paris deals with 3D digitization, simulation, rapid prototyping, virtual restoration applied on a french medieval sculpture. The main purpose is to virtually represent a polychrome statue of the XIIIth century in high quality spectral rendering, to simulate its visual and original appearance at that period. The complete process used throughout all the phases of the project mainly involves optical devices that ensure no physical contact with the museum object. This article describes the complete chain of engineering resources and the main models we used for accomplishing our objective. From 3D capture without contact to plaster replica, the complete process will be described and illustrated with images and objects during the conference. Some sequences extracted from the didactic and scientific movies produced will also be presented.

Keywords Ray-tracing ; spectrophotometry ; 4-fluxes model ; sculpture ; virtual restoration ; gilding ; painting.

Figure 1: The recumbent statue of Philippe Dagobert in Saint-Denis Basilica (13th century AC).

1 INTRODUCTION

A collaboration with the "Centre des Monuments Nationaux" (CMN) helps us to elaborate a new project : the study of a polychrome medieval statue, the recumbent statue of Philippe Dagobert de France (circa 1222 - 1232 AC) in the Saint-Denis Basilica (Fig. 1), the royal necropolis in the north of Paris.

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

Copyright UNION Agency – Science Press, Plzen, Czech Republic.

The presented work takes place in a previously defined general framework known "OCRE method" (Optical Constants for Rendering Evaluation) [CALLET, 1998], developed by Patrick Callet and describing the optical behaviour of materials on the basis of their fundamental properties.

For metals and alloys we naturally used their complex indices of refraction, either computed or carefully measured, but, in any case always validated by measurements. These indices were measured or validated by spectroscopic ellipsometry. These previous important results are very useful for the rendering of the golden parts of the statue.

Moreover, the last study about the visual influence of the under-laying paint, called "bole", on a gilded surface has shown the importance of the transparency of a thin metallic gold film and of the associated small cracks due to the application process (called "burnishing")[DUMAZET et al., 2007].

We deal with modelling and representing the paint materials using spectrophotometry and extrinsic physical parameters such as the paint film thickness and the concentration of each species of pigments. To achieve this goal, we used an extension of the Kubelka and Munk theory : a four-fluxes approach of the multiple scattering of light for the simulation.

We were helped by the "Centre des Monuments Nationaux" (National Monuments Centre) and its knowledge of the painted stones during the middle-age in order to study the pigments and binders that were probably used by the artist.

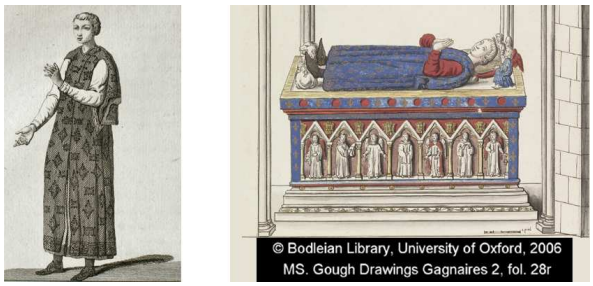


Figure 2: Philippe Dagobert - (left) Stained glass window drawing, (right) Drawing of the tomb with its recess.

2 HISTORICAL CONTEXT

This recumbent statue has been chosen for its remaining polychrome traces (Fig. 1) and also because it was representative of the fine middle XIIIth century funeral sculpture. It concerns the Philippe-Dagobert's tomb, a young Saint-Louis's brother born in 1222 and dead in 1232 and buried in Royaumont (North of Paris).

This statue has encountered several dramatics events. With the french revolution, it has been transferred in Saint-Denis in 1791, then damaged in 1793-94 and deserted until 1796 when it has been preserved in the *Musée des monuments français*. In 1816 it returns to Saint-Denis. In 1820, the architect François Debré order a restoration campaign but, as Baron de Guilhermy has published [de Guilhermy, 1848], this works was already severely damaged by the time in 1848. Then the famous architect Viollet-le-Duc, between 1860 and 1867, inspired by the original pieces, rebuilt entirely the sarcophagus.

Luckily, Millin could study and draw the Philippe-Dagobert tomb in Royaumont as it was in 1790, just before it was transferred to Saint-Denis [Millin, 1791]. Most of his observations were confirmed by a drawing made in 1694 for Roger de Gaignières, which represents a missing stained glass window in the Royaumont abbey-church with the young prince up and wearing the same clothes than his recumbent statue (left figure 2). There is also an other reproduction of the original coloured drawing conserved in the Bodleian Library in Oxford (right figure 2).

As attested on the picture (Fig. 1) there are still some traces of paint and gilding on the stone at the contrary of the other graves gathered in the basilica. First we have tried to retrieve the used pigments with a method based on spectrophotometry without any contact. Then it was decided to take some samples for discriminant chemical analyses.

3 DATA ACQUISITION

Two kinds of data are necessary for the visual reconstruction of the statue. Shape and spectrophotometric informations are required. The first one is more frequently used in computer graphics while the second is generally replaced by a trichromatic description of the actual colors.

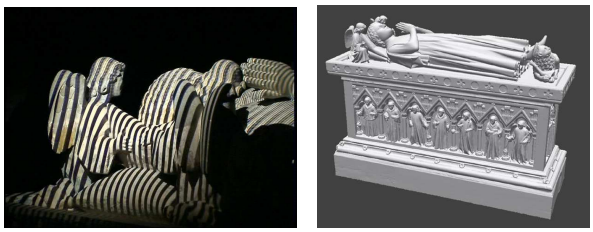


Figure 3: (left) 3D digitization using structured natural white light. (right) 3D colorless model displayed with Catia software (Dassault System).

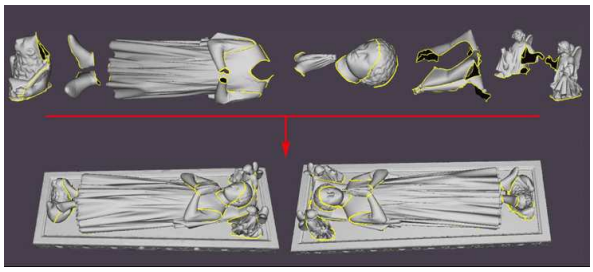


Figure 4: 3D model cut up not completely decomposed in color parts.

3.1 3D shape acquisition and virtual reconstruction

Like previous projects, the 3D digitization has been performed *in situ* (in Saint-Denis Basilica) after public hours, without contact using an optical system based on structured light projector and a camera (Fig. 3) from Breukmann corp. A regular light and shadow grid is projected onto the object surface and the distortion of the boundaries between light and shadows on the surface is recorded by the camera and analysed, in order to extract a cloud of 3D points with a good accuracy. This gave us 167 clouds of 3D points used for reconstructing the surface with RapidForm, a software from Inus Technology corp. The final 3D shape of the tomb (recumbent statue on the sarcophagus) in full resolution represents about 7,5 Million triangles (Fig. 3).

3.2 Spectrophotometric data acquisition

The portable spectrophotometer (USB 2000 series from Ocean Optics corp.) consists in a set of optical fibers guiding a calibrated light source (a halogen lamp for CIED65 illuminant) surrounding the backscattered light guiding fiber. The latest is connected to a spectrometer where a 1024 photodiodes array transforms the received light in a digital signal. We then obtain the diffuse reflectance factor of the paint according to the visible wavelengths domain (from 380*nm* to 780*nm*). Also *in situ*, we captured spectral information on the remaining traces of paint thanks to a spectrophotometer with optical fiber useful for further analyses and material characterization.

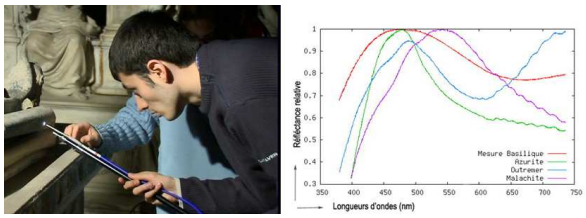


Figure 5: Spectral data acquisition *in situ*, reflectance factors layout and analysis.

Figure 6: Medieval paint samples preparation.

The portable spectrophotometer and the recorded refectance factors are shown in Fig. 5.

As we need to compare and validate permanently our choices and models, we also used pigments and binder samples prepared in the laboratory. We also made our spectral measurements on these samples (Fig. 5).

3.3 Painting materials

During the Middle Age and more particularly during the thirteenth century, we already know, [Escalopier, 2004] what pigments are the most commonly used and also how the paints were applied on the sculpture.

First, the artists start with two or three layers of ceruse, made out of water and white lead. This step enables to waterproof the stone but also enables the painters to rectify the surface defects of the stone by coating. Then they apply several layers of paint, consisting of pigments embedded in a binder like egg (tempera technique), animal protein or gum-resin. The most frequently encountered pigments are:

- White : White Lead, White Chalk, Lime;
- Black : Wood Smoke, Black Vine;
- Blue : Azurite, Lapis-Lazuli, Indigo;
- Red : Vermilion, Red Lead, Red Ochre;
- Yellow : Yellow Ochre, Massicot;
- Green : Green Clay, Malachite, Verdigris.

Which of them were used in the specific case of Philippe Dagobert's recombent statue? The recorded diffuse reflectance factors, obtained by the only possible analysis involving a non-destructive method allow us to compare them to the results obtained by the chemical analysis for validation and simulation.

We also need to specify which part of the global reconstructed shape would be associated with each kind of materials (pigments and binder or/and gilts) (Fig. 4).

4 THE KUBELKA-MUNK MODEL EXTENDED TO FOUR-FLUXES THEORY

Many works are related to spectral representation of colours ([Sun et al., 2000], [Rougeron and Peroche, 1997], [Devlin et al., 2002], [Gondek et al., 1994]) and pigmented materials in computer graphics ([Haase and Meyer, 1992], [Baxter et al., 2004]) using the Kubelka-Munk theory. Our first concern here was, to use an extended Kubelka-Munk model to four fluxes to better fit a modeling of paintings and to realize our calculations based on 81 wavelengths bands of 5nm over the visible spectrum [380;780]nm. Our second concern was in our entire process to constantly go back and forth between our theoretical models and our measurements on real materials. And above all, helped with our collaborations to apply these concepts in an archeological, and historical approach. The Kubelka-Munk model is almost well suited for the description of pigmented materials. These are described as scattering media, laying on a scattering background. The system is illuminated by a diffuse orthotropic incident light. It can be demonstrated ([Volz and Teague, 2001, CALLET and ZYMLA, 2004]), that an incident orthotropic flux of light on such a film of thickness h can be considered equivalent to a collimated directionnal and normal incident flux on a paint film of thickness $2h$. This model gives the reflected and the transmitted fluxes from an incident light, normal to the layers across a paint film of thickness h laid on a substrate. The plain Kubelka-Munk theory [KUBELKA and MUNK, 1931] is then reduced to an equivalent 2-fluxes theory involving two directional fluxes of light and, according to the previous remark above, one going downward L^+ and the other upward L^-. The layer of paint is a macroscopic scattering and absorbing medium so we consider these two coefficients: S and K, respectively the scattering and absorption ones. These last coefficients depend on the pigments diameters and consequently on the granulometry distribution function. A fine grinding involves an important multiple scattering inside the paint film and gives a desaturated colour obtained without any addition of white pigments. This is a

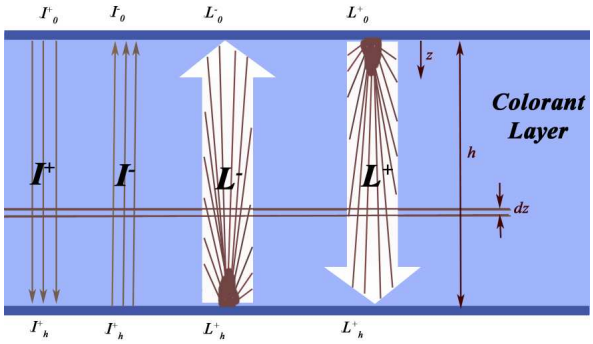


Figure 7: Four-Fluxes theory: the directional (l^+,l^-) and diffuse (L^+,L^-) fluxes, together with the boundary conditions.

physical whitening only. Further works will use the micro-stratigraphy images for extracting a best estimation of the granulometry distribution function. We saw that the 2-fluxes model work on a scattering layer deposited on a scattering background but there are cases where we have a paint layer on a gold leaf. That's why we need the extension of the model to 4-fluxes and, conveniently we use 2 additional, directional, fluxes of light l^- and l^+. These two fluxes are added to the model with the two previous diffuse fluxes, one downward L^+ and the other upward L^- (as described in [Volz and Teague, 2001]).

The figure 7 shows the principle of this model. The incident light is then decomposed in a remaining directional reflected light and an additional scattered light due to volume and surface scattering from the substrate. Two specular components are then added to the classical Kubelka-Munk model.

4.1 Getting the model parameters

The model need five parameters, K, S, k', s^+ and s^-, respectively the absorption and the scattering coefficient for the diffuse light, the absorption, forward scattering and the backward scattering coefficients for directional light. These parameters are fixed after a local radiative balance by the four following equations. Note that all the involved terms are wavelength dependent.

$$dl^+ = -(k+s^+ +s^-)l^+dz \quad (1)$$

$$-dl^- = -(k+s^+ +s^-)l^-dz \quad (2)$$

$$dL^+ = s^+l^+dz + s^-l^-dz - (K+S)L^+dz + SL^-dz \quad (3)$$

$$-dL^- = s^-l^+dz + s^+l^-dz - (K+S)L^-dz + SL^+dz \quad (4)$$

To get the parameters we start by computing the K and s parameters by setting the fluxes l^+ and l^- equal to zero. Then the solution of the above system of equations, leads to the plain 2-fluxes Kubelka and Munk expressions. At this step we need some measurements on some samples of the material we want to characterize. To get them, we need to prepare three samples:

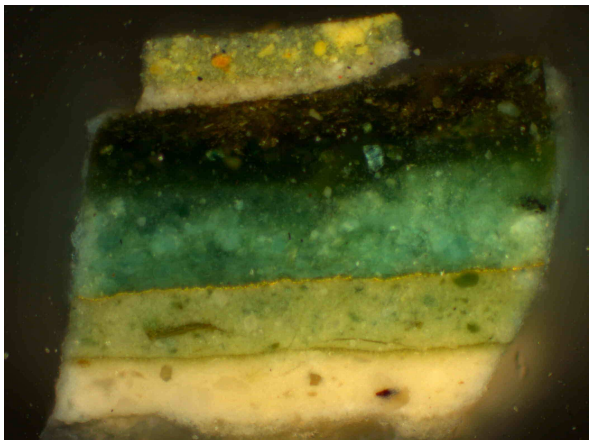


Figure 8: Microstratigraphy of a paint scale taken on the green bottom cushion. Each successive layer gives information about its mean thickness and the granulometric distribution of the embedded pigments. Magnification x20.

- a sample with the substrate only. It's reflectance will be noted R_g ;
- a sample with a totally opaque layer of the considered paint. It's reflectance will be noted R_∞ ;
- a sample of a non-opaque layer of pigments on the substrate with reflectance R and thickness h.

These measurements are made with a spectrophotometer for the reflectances and we get the thickness h with micro-stratigraphic images (Fig. 8).

Next, we deduce the reflectance R_0 of a layer with a thickness h on a perfect black background:

$$R_0 = \frac{R_\infty(R_g - R)}{R_g - R_\infty(1 - R_gR_\infty + R_gR)} \quad (5)$$

Then we get S, K the macroscopic scattering and absorption coefficients:

$$S = \frac{1}{h}\frac{R_\infty}{1-R_\infty^2}\ln\frac{R_\infty(1-R_0R_\infty)}{R_\infty - R_0} \quad (6)$$

$$K = \frac{1}{2h}\frac{1-R_\infty}{1+R_\infty}\ln\frac{R_\infty(1-R_0R_\infty)}{R_\infty - R_0} \quad (7)$$

The four-fluxes model requires the specification of all the terms k, s_i, s_j from the optical coefficients. Let:

$$a = 1 + \frac{K}{S}, \qquad b = \sqrt{\frac{K}{S}\left(\frac{K}{S}+2\right)}$$

$$x = bhS, \qquad A = asinh(x) + bcosh(x)$$

from which we deduce the following parameters

$$\tau = \frac{b}{A}, \qquad \rho_1 = a - b, \qquad \rho_2 = \frac{sinh(x)}{A}$$

We then obtain from the previous relationships:

$$T_r = \left(\frac{R - R_0}{R_g}\right)^{\frac{1}{2h}}$$

and:

$$p = \frac{R_\infty(1-\tau T_r) - R_0}{\rho_1(1-\tau T_r) - \rho_2}$$

$$q = \rho_1 p - R_\infty$$

Next, inverting p and q equations leads to the scattering coefficients:

$$\left\{ \begin{array}{l} s_i = p(\mu - aS) + qS \\ s_j = pS - q(\mu + aS) \end{array} \right\}$$

where $\mu = \frac{1}{h}\ln\frac{1}{T_r}$

Finally we can deduce the absorption coefficient for a very thin layer:

$$k = \mu - (s_i + s_j)$$

4.2 Computing the reflectance

With the five parameters we can compute the reflectance of a paint layer with any thickness h and over any background if we have the substrate reflectance R_g. Let $R_{gs}(\theta_i, \theta_r)$, the substrate specular reflectance function and R_g, the substrate diffuse reflectance. θ_i and θ_r are respectively the incident and the reflection angles. The computation of the diffuse part, R_d, uses the Kubelka and Munk formulas. We start by the calculation of the R_∞ term:

$$R_\infty = 1 + \frac{K}{S} - \sqrt{\frac{K^2}{S^2} + 2\frac{K}{S}} \quad (8)$$

Then we can compute R_d:

$$R_d = \frac{(1/R_\infty)(R_g - R_\infty) - R_\infty(R_g - 1/R_\infty)e^{Sh(1/R_\infty - R_\infty)}}{(R_g - R_\infty) - (R_g - 1/R_\infty)e^{Sh(1/R_\infty - R_\infty)}} \quad (9)$$

For the directional part, R_s, we have to compute the fluxes l^+ and l^-. The forward part l^+ is computed with (1). We get:

$$dl^+(z)/dz + (k + s^+ + s^-)l^+(z) = 0 \quad (10)$$

and the solution to this differential equation is:

$$l^+(z) = l^+(0)\, e^{-(k+s^+ + s^-)\, z} \quad (11)$$

We obtain a similar formula for l^- from the equation (2). The absorption term $A(x)$ is:

$$A(x) = e^{-(k+s^+ + s^-)\, x} \quad (12)$$

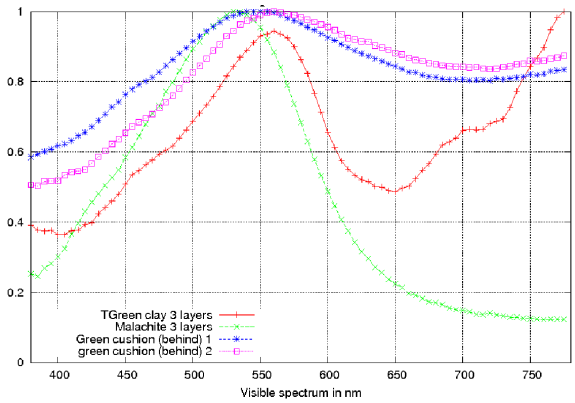


Figure 9: Comparison of the *in situ* measured diffuse reflectance factors with those of their corresponding handcrafted samples.

where x is the effective geometrical thickness of the paint layer (equal to h at normal incidence). We can deduce R_s:

$$R_s(\theta_i, \theta_r) = A\left(\frac{h}{\cos\theta_i}\right) A\left(\frac{h}{\cos\theta_r}\right) R_{gs}(\theta_i, \theta_r) \quad (13)$$

Finally we can compute the final reflectance:

$$R(\theta_i, \theta_r) = R_d * \cos(\theta_i) + R_s(\theta_i, \theta_r) \quad (14)$$

We use here a lambertian model for the diffuse contribution.

5 SPECTRAL SIMULATION AND VIRTUAL RESTORATION

To achieve our goal, the optical properties of the paint compounds are required. As we cannot alter the painting in any way, we have to get the parameters from methods that don't require any contact. We used a methodology that rely on spectrophotometry and spectral sample matching. So we create some samples (Fig. 6) of paint layers and tried to match their reflectance factor with the ones measured on the statue. These samples were created with pigments known to be used when the statue was created. For example, the spectral measurements helped in differentiating a lapis-lazuli from an azurite film (Fig. 10). The first one has two peaks, one in the blue wavelengths near 455*nm* and the other in the red and the near IR wavelengths region (780*nm*). Only one peak appears for the azurite pigments. The Lapis-lazuli and its subtle reddish component is known as natural ultra-marine blue, very efficient for reflecting infra-red radiations (ideal paint used for the shutters in the mediterranean countries). We can also notice that the two pigments have different maxima in the blue region of the visible spectrum. One exhibits a relatively narrow peak while an other has a more flat and extended peak tending to cover the green shades region. Sometimes, a small amount of malachite

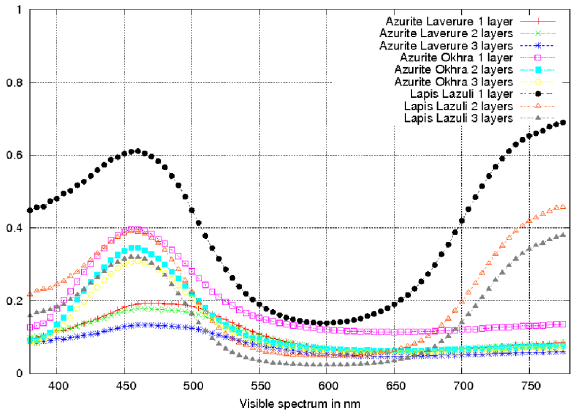


Figure 10: Reflectance factors of the blue pigments, azurite and lapis-lazuli depending on the paint film thickness.

in natural azurite can explain the reflectance curve aspect and also the greenish color observed on the sample. Now that we have the reflectance factors of the pure pigments, we want to compare them to the ones we measured in the basilica on the recumbent statue itself. We have identified some of the pigments used on the statue. It seems that the pigment used by the artist was identified as green clay, according to the reflectance curves. We have made this kind of comparison for each *in situ* reflectance factor. The blue areas seem to have been painted with azurite. The reflectance factor of the red cushion approaches that of the red lead. We can notice that the recorded spectra in the basilica have a weaker amplitude between their maxima and minima than those of the samples. This phenomenon expresses the fact that the colors of the sculpture are now very desaturated. There are several reasons for that whitening. The paint layers are old and dirty so that the deposited dust on the surface increases the whitening by surface scattering of all incident light. More studies were necessary to exactly determine the composition of each paint used to achieve this sculpture. Many samples of different pigments have been elaborated. Therefore we virtually created mixtures of pigments and compared them to the *in situ* recorded reflectance factors, according to the robust color matching methods.

5.1 Samples measurements

The samples give also the parameters used by the Kubelka and Munk extended to four fluxes. Indeed, for several pigments (green clay, malachite, black vine, vermilion, and lemon ochre), the reflectance factors for two and three layers tends to be quite similar, so we will consider that the diffuse reflectance factor for the three layers samples gives the term R_∞ to determine the coefficients K and S of these pigments. We measure also the thickness of the layers using some micro-stratigraphic images of these samples.

Figure 11: First results obtained with only spectrophotometric data and physical samples preparation. (left) Philippe-Dagobert's *cappa magna* rendered in lapis-lazuli blue pigment by the radiosity software Candelux. (right) Philippe-Dagobert's sleeves - vermilion pigment rendered with Candelux, opaque gold leaf covered lion rendered in ray-tracing with Virtuelium. CIE D65 illuminants.

5.2 Material compositing

The main goal of this model is to permit the computation of multi-layered materials. As we can see on figure 8 a lot of paint layers often 3 or 4 are encountered. The overall reflectance is obtained by computing the reflectance bottom-up, by using the previous reflectance as substrate reflectance. Associated with the method presented in [DUMAZET et al., 2007] we can compute any multi-layered paint system where we dispose of the K, S, k, s^+ and s^- parameters. Moreover, an approximation can be used for pigments mixing in the Kubelka and Munk theory. The approximation of the resulting K/S of a such mixing can be computed by :

$$\frac{K}{S} = \frac{\sum_i C_i K_i}{\sum_i C_i S_i} \tag{15}$$

where C_i is the volumetric concentration of the i-th component in the mixing. Therefore, we haven't done yet the studies on the validity of using a such composition law on the k, s^+ and s^- parameters.

5.3 The metallic film influence

Some part of the recumbent statue are "just" gilded. So to make good simulations we have to be able to render metallic materials. For this we use the real part and the imaginary part of the complex indices of refraction of the metals or alloys. Then we introduce them into the Fresnel formulas in the rendering pass. These indices can be measured on real polished plates by *spectroscopic ellipsometry*, an optical method based on Fresnel formulas and the analysis of the amount of

the reflected and polarized light at large incidence on a smooth surface. The whole theory of reflection of light by a metallic surface is available in optics books such as Born and Wolf's [BORN and WOLF, 1975] and for computer graphics and lighting in [CALLET, 2006, CALLET, 2007] including (rare) quantitative data.

5.4 Spectral simulations

With a correct characterisation of all materials and their state of surface (waviness, roughness, paint-film thickness), the lighting conditions and a standard colorimetric observer (CIE 1964 10°), we can compute the restored visual aspect of the museum artifacts[Pitzalis et al., 2007]. In order to produce the rendering we use our multithreaded raytracer, *Virtuelium*. It is a multispectral raytracing software using generally 81 spectral bands of 5 nm width, ranging from 380 to 780 nm. It can also handle light polarisation and rely on CIE standard illuminants and colorimetric observers. All the materials are not define by radiative properties but by there intrinsic properties. So we use the complex indices of refraction for the metals or the dielectric materials. We use four fluxes parameters for paints. The algorithm, itself, used here is a simple ray tracing with just direct and specular illumination. We use an octree to speed up the computation and take benefit from multi-processor/core computer. According to the physical parameters extracted from microstratigraphy and spectrophotometric measurements, we have computed with the 4-fluxes model and Virtuelium, the image presented in Fig. 12. Thus, the successive layers with their thickness and concentration as plausible for the XIIIth century can appear. We have made many simulations with different standard illuminants and with characterized real light sources. Some parts of the sculpture as the prince's face or the angel head were probably painted with a cinnabar and vermilion mixture. No visible traces permitted to confirm that point and we decided to render the corresponding parts with the only white lead in a totally opaque layer.

We computed the virtual restored aspect of the medieval sculpture with the following materials :

- angel tunic: lapis-lazuli layer on azurite ;
- angel wings: 50 % red lead and 50 % cinnabar ;
- Philippe Dagobert hair: 200 nm thick gold leaf on yellow ochre substrate ;
- upper cushion: 50 % red lead and 50 % cinnabar ;
- lower cushion: not completely opaque layer of malachite pigments.

Figure 12: The actual most plausible colors of the medieval polychrome recumbent statue of Philippe Dagobert. Rendering in spectral ray-tracing by Virtuelium on an INTEL Pentium4 3GHz dual processors computer in 8 hours for a 1200x1200 resolution, 1Gb RAM and running under the operating system GNU/Linux - Fedora 5.

6 CONCLUSIONS

We went one step ahead in our knowledge and know-how of cultural heritage engineering and physical models for materials rendering in optical simulation. Till now we worked with standard illuminants such as CIE A, D65 or E illuminants. We shall need for these studies a better knowledge on natural lighting by stained glass windows and all other anthropogenic lightings used in the medieval era. For also improving the rendering of the complete sculpture, the canopy and all the gilded ornaments will be added to the 3D model along with photon mapping and texturing. The second part of that project will gather all the acquired new results and will also produce a new video movie translated in many languages.

ACKNOWLEDGEMENTS

In this pluridisciplinary project the authors were greatly helped by the following colleagues, in Ecole Centrale Paris by François-Xavier de Contencin for 3D digitization, Anna Zymla for material analysis, Philippe Denizet and Marie-France Monanges for video assisted by many students. Colleagues in Laboratoire de Recherche des Monuments Historiques, Vincent Detalle, Olivier Rolland and Annick Texier worked on microstratigraphy, paint analysis and gilding. The Centre des Monuments Nationaux and the Saint-Denis basilica scientific and history team with Jacqueline Maillé, Georges Puchal, Robert Lequément, Alain Erlande Brandenbourg, Françoise Perrot and Serge

Santos. In Louvre museum, Pierre-Yves Le Pogam gave access to an essential and original colored piece while Nicolas Hueber at Ecole Nationale Supérieure d'Arts et Métiers realized the physical replica assisted by the sculptor Brigitte Bonnet. Gilles Raffier from AXIATEC facilitated the physical realization and sponsorship.

REFERENCES

[Baxter et al., 2004] Baxter, W., Wendt, J., and Lin, M. C. (2004). Impasto: A realistic, interactive model for paint. In *In Proc. NPAR2004,the 3rd Intern. Symposium on Non-photorealistic Animation and Rendering*, pages 45–56.

[BORN and WOLF, 1975] BORN, M. and WOLF, E. (1975). *Principles of Optics–Electromagnetic Theory of Propagation, Interference and Diffraction of Light–*. Pergamon Press, Oxford.

[CALLET, 2006] CALLET, P. (2006). Metals, alloys and lighting: optical properties and spectral simulation in computer graphics. In CIE, editor, *Proceedings of the CIE Expert Symposium on Visual Appearance*, CIE x032:2007. CIE. Held in Muséum National d'Histoire Naturelle, october 19-20th, Paris.

[CALLET, 2007] CALLET, P. (2007). Couleur et apparence visuelle ii : l'aspect métallique. In *Sciences fondamentales*, number AF3253 in AF, page 23. Techniques de l'Ingénieur.

[CALLET, 1998] CALLET, P. (mars 1998). *Couleur-lumière, couleur-matière – Interaction lumière-matière et synthèse d'images*. Collection Sciences en actes, ISBN : 2-84-352-087-8. Diderot Editeur, Arts et Sciences, Paris, New-York. CDROM d'images inclus, 320p.

[CALLET and ZYMLA, 2004] CALLET, P. and ZYMLA, A. (2004). Rendering of binary alloys - example and validation with bronze. In team, I., editor, *Proceedings of ICCVG 04*. ICCVG.

[de Guilhermy, 1848] de Guilhermy, B. F. (1848). *Monographie de l'église royale de Saint-Denis. Tombeaux et figures historiques, Paris, p.240-244*. " ".

[Devlin et al., 2002] Devlin, K., Chalmers, A., Wilkie, A., and Purgathofer, W. (2002). Tone reproduction and physically based spectral rendering. In *EUROGRAPHICS 2002, STAR - State of The Art Report*.

[DUMAZET et al., 2007] DUMAZET, S., Ariane GENTY, a. A. ZYMLA., de CONTENCIN, F.-X., TEXIER, A., RUSCASSIER, N., BONNET, B., and CALLET, P. (2007). Influence of the substrate colour on the visual appearance of gilded sculptures. In Ogleby, C., editor, *Proceedings of the XXIth CIPA conference, Athens*. ISPRS.

[Escalopier, 2004] Escalopier, C. (2004). *Essai sur divers arts. Théophilus moine artisan du XIIe siècle. Texte intégral traduit du latin par Charles Escalopier*. Paleo.

[Gondek et al., 1994] Gondek, J. S., Meyer, G. W., and Newman, J. G. (1994). Wavelength dependent reflectance functions. In *Proceedings of SIGGRAPH'94*, pages 213–220.

[Haase and Meyer, 1992] Haase, C. S. and Meyer, G. W. (1992). Modeling pigmented materials for realistic image synthesis. In *ACM Transactions on Graphics*, pages 305–335.

[KUBELKA and MUNK, 1931] KUBELKA, P. and MUNK, F. (1931). Ein beitrag zur optik der farbanstriche. *Zeitschrift fÃ$\frac{1}{4}$r Technische Physik*, 12:593–601.

[Millin, 1791] Millin, A. L. (1791). *Antiquités nationales, ou Recueil de monuments pour servir à l'histoire, tome 2, Paris, p.10-12,pl.4,fig.2 et pl.5, fig.1.* Antiquités nationales.

[Pitzalis et al., 2007] Pitzalis, D., Lahanier, C., Aitken, G., Pillay, R., rodriguez Echavarria, K., and Arnold, D. B. (2007). 3d techniques to create interactive virtual museums: the state of the art in the epoch noe. In José Braz, P.-P. V. and ao Madeiras Pereira, J., editors, *GRAPP 2007 – Second Conference on Computer Graphics Theory and Applications*, pages 197–203. INSTICC Press.

[Rougeron and Peroche, 1997] Rougeron, G. and Peroche, B. (1997). An adaptive representation of spectral data for reflectance computations. In *Proceedings of the 8th Eurographics*.

[Sun et al., 2000] Sun, Y., Fracchia, F. D., and Drew, M. S. (2000). A composite spectral model and its applications, scottsdale, arizona. In *Eighth Color Imaging Conference*, pages 102–107.

[Volz and Teague, 2001] Volz, H. G. and Teague, B. (2001). *Industrial Color Testing: Fundamentals and Techniques (2nd edition) ISBN: 3527304363*. Vch Verlagsgesellschaft Mbh.