ISSN 1213-6972 *Volume 18, Number 1-3, 2010*

Journal of WSCG

An international journal of algorithms, data structures and techniques for computer graphics and visualization, surface meshing and modeling, global illumination, computer vision, image processing and pattern recognition, computational geometry, visual human interaction and virtual reality, animation, multimedia systems and applications in parallel, distributed and mobile environment.

ISSN 1213-6972 *Volume 18, Number 1-3, 2010*

Journal of WSCG

An international journal of algorithms, data structures and techniques for computer graphics and visualization, surface meshing and modeling, global illumination, computer vision, image processing and pattern recognition, computational geometry, visual human interaction and virtual reality, animation, multimedia systems and applications in parallel, distributed and mobile environment.

Vaclav Skala – Union Agency

Journal of WSCG Cumulative edition

Editor-in-Chief: Vaclav Skala
c/o University of West Bohemia, Univerzitni 8

306 14 Plzen
Czech Republic
skala@kiv.zcu.cz

Managing Editor: Vaclav Skala

Published and printed by Printed and Published by:
Vaclav Skala - Union Agency
Na Mazinach 9
CZ 322 00 Plzen
Czech Republic

Hardcopy: *ISSN 1213 – 6972*
CD ROM: *ISSN 1213 – 6980*
On-line: *ISSN 1213 – 6964*

ISBN 978-80-86943-89-3

Journal of WSCG

Editor-in-Chief

Vaclav Skala, University of West Bohemia
Centre for Computer Graphics and Visualization
Univerzitni 8, Box 314
CZ 306 14 Plzen, Czech Republic
skala@kiv.zcu.cz http://herakles.zcu.cz
Journal of WSCG: URL: http://wscg.zcu.cz/jwscg

Direct Tel. +420-37-763-2473
Direct Fax. +420-37-763-2457
Fax Department: +420-37-763-2402

Editorial Advisory Board
MEMBERS

Board of Reviewers

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

Papaioannou,G. (Greece)
Patane,G. (Italy)
Pedrini,H. (Brazil)
Pina,J. (Spain)
Platis,N. (Greece)
Plemenos,D. (France)
Post,F. (Netherlands)
Pratikakis,I. (Greece)
Puig,A. (Spain)
Puppo,E. (Italy)
Purgathofer,W. (Austria)
Renaud,c. (France)
Richardson,J. (United States)
Ripolles,O. (Spain)
Ritschel,T. (Germany)
Rojas-Sola,J. (Spain)
Rosenhahn,B. (Germany)
Rudomin,I. (Mexico)
Sakas,G. (Germany)
Sanna,A. (Italy)
Sbert,M. (Spain)
Segura,R. (Spain)
Sellent,A. (Germany)
Schneider,B. (United States)
Schumann,H. (Germany)
Sirakov,N. (United States)
Sochor,J. (Czech Republic)
Solis,A. (Mexico)
Sousa,A. (Portugal)
Steinicke,F. (Germany)
Stroud,I. (Switzerland)
Svoboda,T. (Czech Republic)
Teschner,M. (Germany)
Theoharis,T. (Greece)
Theußl,T. (Austria)
Tokuta,A. (United States)
Torrens,F. (Spain)
Tytkowski,K. (Poland)
Vanecek,P. (Czech Republic)
Vasa,L. (Czech Republic)
Veiga,L. (Portugal)
Vergeest,J. (Netherlands)
Vitulano,D. (Italy)
Weiss,G. (Germany)
Wu,S. (Brazil)
Yencharis,L. (United States)
Zach,C. (Switzerland)
Zachmann,G. (Germany)
Zalik,B. (Slovenia)
Zara,J. (Czech Republic)
Zemcik,P. (Czech Republic)
Zhu,Y. (United States)
Zitova,B. (Czech Republic)

A Secure Touch-less based Fingerprint Verification System

Hiew Bee Yan[1], Andrew Teoh Beng Jin[2], Ooi Shih Yin[1], Fathin Fakhriah Abdul Aziz[1]

[1]Faculty of Information Science and Technology
Multimedia University, Jalan Ayer Keroh Lama,
75450 Melaka, Malaysia.
{byhiew, syooi, fathin.abd.aziz}@mmu.edu.my

[2]School of Electrical and Electronic Engineering,
Yonsei University, Seoul,
South Korea.
bjteoh@yonsei.ac.kr

ABSTRACT

Touch-less based fingerprint verification systems are free from the problems of image deformation, latent fingerprint issues and so forth that appear in the contemporary touch based fingerprint verification systems. Coupled with template protection mechanism, a touch-less fingerprint verification system is further enhanced. In this paper, a secure end-to-end touch-less fingerprint verification system is presented. The fingerprint image captured with a digital camera is first pre-processed via the proposed pre-processing algorithm. Then, Multiple Random Projections-Support Vector Machine (MRP-SVM) is proposed to secure fingerprint template while improving system performance.

Keywords

Touch-less based Fingerprint Verification, Template Protection, Support Vector Machine.

1. INTRODUCTION

A preponderance of the fingerprint sensors available today use "touch" method. However, the durability of a touch-based fingerprint scanner is weakened if it is used heavily. Additionally, the pressure of the physical contacts degrades the quality of the touch-based fingerprint images. Conversely, touch-less fingerprint acquisition technology is free from these problems [Son04]. Furthermore, touch-less based fingerprint verification systems have great potentials in building secure verification systems provided that they are free from latent fingerprint (the trail of the fingerprint on the contact surface of the sensor) issues which can lead to fraudulent use [Lee06a]. Hence, with the incorporation of cancellable biometrics in these fingerprint features, the security and privacy protection of fingerprint biometric templates are enhanced. Cancellable biometrics is a concept initiated to secure biometric templates. This concept is important because biometric templates are not revocable and their compromise is permanent [Sch99]. A good cancellable biometric formulation must fulfill four requirements [Mal03]: (i) diversity; (ii) reusability; (iii) one way transformation; (iv) performance.

1.1 Related Works

1.1.1 Touch-less Fingerprint Recognition

Digital Descriptor System Inc. (DDSI) [Mai06] produced the world first contact-less fingerprint capture device which can be integrated with Fingerprint Matching Solution [Dig00]. Later, TST Corona-GmbH developed biometric recognition systems using novel touch-less optical fingerprint scanning technology. Song et al. proposed a pre-processing technique for their custom designed touch-less sensor [Son04]. Using a strong view difference image rejection method, [Lee06a] resolved the three-dimensional (3D) to two-dimensional (2D) image mapping problem which appeared in [Son04]. Pre-processing of the fingerprint images captured with mobile camera has been proposed by [Lee06b]. To make 3D touch-less fingerprints interoperable with the current Automated Fingerprint Identification System (AFIS),

Chen et al. [Che06] proposed an unwrapping algorithm that un-folds the 3D touch-less fingerprint images into 2D representations which are comparable with the legacy rolled fingerprints.

1.1.2 Cancellable Biometrics

Ratha et al introduced the first notion of cancellable biometric formulation. The underlying idea is to generate deformed biometrics data by distorting the biometric image in a repeatable but non-reversible manner [Rat01]. Teoh et al. [And04] introduced another cancellable biometrics approach using iterative inner product between tokenized pseudo-random numbers (PRN) and biometric data. But, this formulation suffered from the stolen-token scenario when the genuine token is stolen and utilized by impostor to claim as the genuine user. Savvides et al. [Sav04] encrypted the training images which are used to synthesise the correlation filter for biometrics authentication. They showed that different templates can be obtained from the same biometrics by varying the random convolution kernels. Thus, those templates can be cancelled. Ang et al. [Ang05] generated a cancellable fingerprint template via a key-dependent geometric transformation on a minutiae based finger-print representation. Nevertheless, the matching accuracy is degraded notably in the distorted domain. In [Bou06], the authors introduced the concept of biometric-based tokens that support robust distance computations, which offers cryptographic security such that it can be revoked and replaced by a new one. Teoh and Chong [And07] introduced Multispace Random Projection (MRP) as one of the cancellable biometrics approaches that fulfils good cancellable biometrics requirements as stated above. However, the Equal Error Rate (EER) result is only 30% due to the poor classification capability of the matching metric.

1.2 Our Approach

A secure digital camera based fingerprint verification system is presented in this paper. This system uses "touch-less" method with a digital camera to capture fingerprint images. The problems appeared in the digital camera acquired fingerprint images are: (i) low ridges valleys contrast; (ii) defocus; and (iii) motion blurriness. To reduce these problems, the fingerprint images are pre-processed using the proposed method [Hie06]. To protect the template and improve the system performance, a variant of MRP [And07], named as Multispace Random Projections-Support Vector Machine (MRP-SVM), is proposed. MRP-SVM is performed after pre-processing. In MRP-SVM, MRP, which used normalized dot product, is replaced with a more powerful classifier – Support Vector Machine (SVM) while still retains the properties of MRP.

The outline of the paper is as follows: Section 2 provides the details of the proposed system. Section 3 shows the experiment results. Discussions are presented in Section 4. Finally, conclusions are given in Section 5.

2. TOUCH-LESS FINGERPRINT VERIFICATION ALGORITHM

2.1 Pre-processing

Initially, the captured fingerprint image in Red-Green-Blue (RGB) format is converted to grey scale [0-255]. To reduce the problem of non-uniform lighting in the image, local normalization is adopted to reduce the intensity variations in the image. Local normalization of $o(x_o, y_o)$ is computed as follows [Bio02]: $S(x_o, y_o) = \frac{o(x_o, y_o) \quad m_o(x_o, y_o)}{\sigma_o(x_o, y_o)}$ where $o(x_o, y_o)$ is the original image, $m_o(x_o, y_o)$ denotes an estimation of a local mean of $o(x_o, y_o)$, $\sigma_o(x_o, y_o)$ indicates an estimation of the local standard deviation. Next, the fingerprint region is segmented from the raw image by applying skin color detection, adaptive thresholding, and morphological processing [Hie06]. Then, the local normalized image is multiplied with the fingerprint binary mask to get the segmented image. The resulting image is cropped and enhanced by using the Short Time Fourier Transform (STFT) analysis [Chi05]. Subsequently, the ridge orientation is calculated and the core point is detected from the enhanced image [And03]. The true and false core point detections are obtained by: (number of true core point detected images/total number of fingerprint images) and (number of false core point detected images/total number of fingerprint images) respectively. The detail description can be obtained in [Hie06]. After the core point detection, the pre-processed image is created by cropping the local normalized image into a size of 200 x 200 with the core point as the centre.

2.2 Multispace Random Projections-Support Vector Machine (MRP-SVM)

2.2.1 Brief Review of Multispace Random Projections (MRP)

MRP covers two stages: (i) feature extraction: feature vector, $\boldsymbol{f} \in \Re^d$, with fixed length d is extracted; (ii) random projections: the feature vector is projected onto a random subspace, which is formed by externally derived random matrix, $\boldsymbol{R} \in \Re^{m \times d}$, where $m \leq d$. The $\boldsymbol{R}$ is formed by the independent, zero

mean and unit variance Gaussian distributed random bases. Thus, the user-specific random-projected vector, $\boldsymbol{p} \in \Re^m$ is described as

$$\boldsymbol{p} = 1/\sqrt{m}\boldsymbol{R}\boldsymbol{f} \quad (1)$$

During verification, the extracted feature vector is mixed with the genuine $\boldsymbol{R}$. The resulting vector is compared with the enrolled template using the normalised dot product, a dissimilarity measure, $\gamma = 1 - \boldsymbol{x}^T \boldsymbol{y}$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are the normalised feature vectors. From the performance perspective, three scenarios are considered when MRP is applied: (i) *Legitimate-token*: in which the genuine biometric is mixed with the user-specific token; (ii) *Stolen-token*: wherein an impostor has possessed genuine token and used by the impostor to claim as the genuine user; (iii) *Stolen-biometrics*: where an impostor assesses intercepted biometric data of high possibility to be considered genuine. Through the theoretical and experimental analysis [And07], Figure 1 illustrates the original system performance and performance behaviour of MRP in the legitimate-token, stolen-token and stolen-biometrics scenarios using the genuine-impostor distributions. By referring to Figure 1, MRP's genuine, impostor and genuine-impostor distributions in the legitimate-token, stolen-token and stolen-biometrics scenarios with its respective recognition performance are summarised in Table 1.

2.2.2 Multispace Random Projections-Support Vector Machine (MRP-SVM)

MRP-SVM is applicable to biometrics features represented in feature vector format. This technique consists of three stages: (i) feature extraction, (ii) random projection, and (iii) SVM classification.

2.2.2.1 Feature Extraction

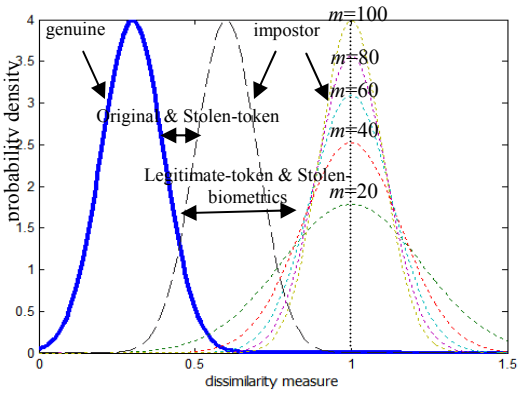


Figure 1. Genuine-impostor distributions of the original system and performance behavior of MRP in the legitimate-token, stolen-token and stolen-biometrics scenarios.

The individual's feature matrix is extracted from the pre-processed image through Gabor filter feature extractor. The adopted feature extraction method is similar to the method described in [Lee99]. Initially, the pre-processed images are sampled by Gabor filters. Given that a band of Ω Gabor filters is applied in an experiment, the filtered images are divided into a set of MxM non-overlapping blocks, respectively. The resulting magnitude will be next converted to a scalar number by calculating its standard deviation value. The scalar numbers form the Gabor features of each image. Finally, $N = (200/M) \times (200/M) \times \Omega$ Gabor features are extracted from each image.

Then, PCA is used to compress the feature vectors. An eigenspace is built during the training phase. During the testing phase, each testing Gabor feature vector is projected onto the eigenspace to form eigenGabors, $\boldsymbol{f}$ with feature length d [Tur91]. Before

Scenario	Genuine distribution (intra-class variation)	Impostor distribution (inter-class variation)	Genuine-Impostor Distributions	Recognition Performance
Legitimate-token Stolen-biometrics	Preserved	Impostor distribution is amplified. The mean is 1 whereby the curve is centred at 1; the distribution profile's shrinking rate (standard deviation) follows $1/\sqrt{Y}$.	Clear separation can be attained, hence zero EER, if Y is sufficiently large as depicted in Figure 1.	Significantly improved.
Stolen-token		Preserved	Reverts to its original state in feature vector level and thus the performance is retained as like original state before the random projection is performed.	Reverts to its original state in feature vector before MRP is applied.

Table 1. MRP's genuine, impostor and genuine-impostor distributions in the legitimate-token, stolen-token and stolen-biometrics scenarios with its respective recognition performance

applying Multi-space Random Projection, each eigenGabors is normalised to unit length, $\|\boldsymbol{f}\| = 1$.

2.2.2.2 Random Projection

Subsequently, the unit length feature vector is projected onto a random subspace as described in Section 2.2.1. The user-specific random-projection (RP) vector, $\boldsymbol{p} \in \Re^m$ is produced through the random projection process, which is defined in Eq. 1. The one way transformation property can be assessed by looking at Eq. (1). $\boldsymbol{p}$ can be regarded as a set of underdetermined systems of linear equations (more unknowns than equations) if $m < d$. Thus, it is impossible to find the exact values of all the elements in $\boldsymbol{f}$ by solving an underdetermined linear equation system in $\boldsymbol{p} = 1/\sqrt{m}\boldsymbol{R}\boldsymbol{f}$ if $m < d$, based on the premise that the possible solutions are infinite [And07]. The detail of the analysis can refer to [Kar05]. In the event of compromised, the template could be renewed by just changing $\boldsymbol{R}$ in Eq. (1) of the compromised biometric template. From this way, reusability property of MRP-SVM is fulfilled.

2.2.2.3 Support Vector Machine (SVM) classification

Then, the random projected vector $\boldsymbol{p}$ is fed into SVM [Bur98] to discriminate genuine and impostor. Let a training set be $(\boldsymbol{p}_i, q_i), i = 1, \cdots Q, \boldsymbol{p} \in \Re^m, q \in \{+1, -1\}$, where $\boldsymbol{p}_i$ either belongs to genuine's or impostor's random projection vector, q_i indicates the label (+1 for genuine, -1 for impostor), Q denotes the number of training samples, and m signifies the feature length. During training, SVM looks for an optimal hyper-plane, which takes the form $\boldsymbol{w}^T \boldsymbol{p} + b = 0$ where $\boldsymbol{w}$ is the normal to the plane, b denotes the bias term. The optimal hyper-plane is a separating hyper-plane, which gives the largest margin. Assuming that genuine and impostor samples are linearly separated, the Langrage Dual optimisation problem for maximal margin separation is:

$$\max. L_D = \sum_{i=1}^{Q} \alpha_i - 1/2 \sum_{i=1}^{Q} \sum_{j=1}^{Q} \alpha_i \alpha_j q_i q_j \boldsymbol{p}_i^T \boldsymbol{p}_j \quad (2)$$

subject to $\sum_{i=1}^{Q} \alpha_i q_i = 0$ and $C \geq \alpha_i \geq 0$ where α_i are the Langrage multipliers, C is the regularisation constant.

The extension to non-linear boundaries is determined through projecting each data point to a higher dimensional feature space [Bur98], which may be carried out through the use of kernels: (i) polynomial kernel as (the order of the polynomial) or (ii) radial basis function (RBF) kernel as (the width of the radial basis function). Subject to constructing the optimal hyper-plane for SVM with non-linear kernel involves the following dual:

$$\max. L_D = \sum_{i=1}^{Q} \alpha_i - 1/2 \sum_{i=1}^{Q} \sum_{j=1}^{Q} \alpha_i \alpha_j q_i q_j K(\boldsymbol{p}_i, \boldsymbol{p}_j) \quad (3)$$

During verification, $\boldsymbol{w}$ and b learned from training phase are used. The decision value, G_{dv} of a test sample of vector $\boldsymbol{p}$ for linear SVM and non-linear SVM are calculated by Eq. 4 and Eq. 5 respectively, where $\boldsymbol{p}_j$ are the support vectors.

$$G_{dv} = \boldsymbol{w}^T \boldsymbol{p} + b = \sum_{j=1}^{N_S} \alpha_j q_j \boldsymbol{p}_j^T \boldsymbol{p} + b \quad (4)$$

$$G_{dv} = \boldsymbol{w}^T \boldsymbol{p} + b = \sum_{j=1}^{N_S} \alpha_j q_j K(\boldsymbol{p}, \boldsymbol{p}_j) + b \quad (5)$$

$\boldsymbol{p}_j^T \boldsymbol{p}$ (in Eq.4) and $K(\boldsymbol{p}, \boldsymbol{p}_j)$ (in Eq.5) are a similarity measure comparing $\boldsymbol{p}$ (test sample) and $\boldsymbol{p}_j$ (support vectors) in input space and feature space respectively. G_{dv} is a weighted sum of the similarity between $\boldsymbol{p}$ and $\boldsymbol{p}_j$. Comparison is made between G_{dv} and a threshold, Th_{dv}. The claimed user is accepted if $G_{dv} < Th_{dv}$ and rejected if $G_{dv} \geq Th_{dv}$.

3. EXPERIMENT EVALUATION

3.1 Pre-processing Experiment Results

Pre-processing experiments are conducted by using all digital camera acquired fingerprint images (1938 images) described in [Hie06]. The results are assessed subjectively by visual inspection. The proposed pre-processing provides admirable results as shown in Figure 2. By counting the number of false core point detection, it reveals that the proposed pre-processing algorithm can achieve an accuracy of 95.44% of core point detection whereas the false core point detection is only 4.56%. Deep wrinkle, motion blurriness and defocus problems are the causes of the core point detection failure [Hie06].

Figure 3 depicts the comparison of the proposed algorithm to those of existing enhancement methods in the literature by using cropped local normalised images of our independent database. As illustrated by the region rounded by circles shown in Figure 3, some portions of the images are not enhanced fittingly by Hong enhancement and root filtering. It can be witnessed that the proposed algorithm performs more competent than Hong's and root filtering. It signifies that image cropping is needed to eliminate the gratuitous noisy background; local normalisation succours in reducing the non-uniform lighting effect.

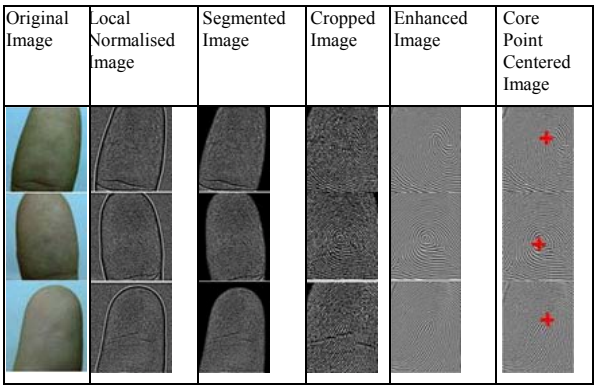


Figure 2. Results for the proposed pre-processing

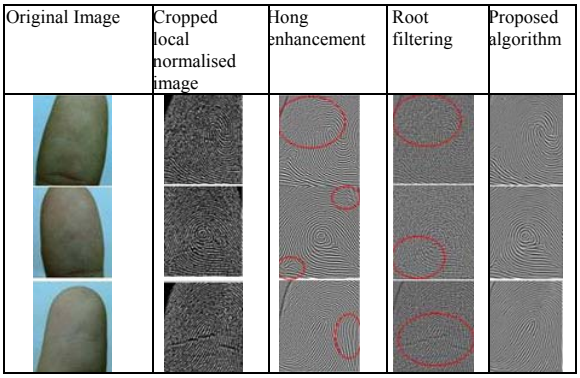


Figure 3. Comparative results for different enhancement algorithms

3.2 Verification Experiment Results without MRP

In lieu of using all of the raw images described in [Hie06] directly, a total of 1030 core point centred local normalised fingerprint images are used in performing the verification experiments. These fingerprint images originate from 103 different fingers with 10 images for each finger. The system performance is determined by using EER, i.e. False Accept Rate (FAR) = FRR. The lower the EER, the more accurate the system is considered to be.

The parameters of the Gabor filter feature extractor are set as: (i) number of Gabor filters (Ω)=6; (ii) frequency (f_g)=10; (iii) variance ($\sigma_{x_g}^2$ and $\sigma_{y_g}^2$)=128; (iv) number of non-overlapping blocks (M)=8. To get eigenGabors, the features derived from the Gabor filter based feature extractor are compressed using PCA. One Gabor feature vector from each finger class is used for training, while the rest will be used as testing data. The eigenGabors are normalised to be of unit length before verification test is conducted. The experiment setting is fixed and shown in Table 2. To enhance the reliability of the assessment, we perform ten runs for each of *Tr* samples with different random partitions between training and testing images, and the results are averaged.

The verification accuracy is tested using linear SVM, polynomial SVM and RBF SVM. To fix the feature length, the eigenvectors are extracted from the Gabor feature vectors with the dimensionality of 100.

Setting	No. of Training Image, *Tr*	No. of Testing Image, 9- *Tr*	No. of Client	No. of Impostor
1	1	8	8x103 =824	8x103x102 =84048
2	2	7	7x103 =721	7x103x102 =73542
3	3	6	6x103 =618	6x103x102 =63036
4	4	5	5x103 =515	5x103x102 =52530

Table 2. Configuration for the experiments using SVM on eigenGabors

Before comparing the performance of linear, polynomial and RBF SVM, the optimal parameter values for polynomial (degree) and RBF SVM (gamma and C) are investigated. This investigation is necessary as these parameter values will affect the results. In this study, the polynomial of degree between 2 and 5 are tested. Besides, different parameter values of RBF SVM (gamma=1, C=10; gamma=10, C=1; gamma=1, C=1; gamma=10, C=10) for both uncompressed Gabor features and eigenGabors are investigated. After the parameter value tuning, the best parameter values of polynomial SVM (degree=2) and RBF SVM (gamma=1 and C=10) are fixed for further evaluation tests. Table 3 shows the EER (%) results for linear, polynomial and RBF kernel with different number of training image, *Tr*. For *Tr*=1…4, it can be seen that the EER results of Linear SVM kernel is the worst while the EER results of RBF SVM kernel is the best. The best result (EER=1.23%) is obtained when *Tr*=4 and RBF SVM kernel is adopted as indicated in the table below.

Tr / SVM kernel type	1	2	3	4
Linear	18.89	9.47	6.48	4.21
Polynomial (degree=2)	14.57	6.26	3.25	1.76
RBF(gamma=1, C=10)	5.09	2.75	1.83	1.23

Table 3. EER Results (%) of linear SVM, polynomial SVM and RBF SVM using eigenGabors (dimensionality = 100) with different number of training image, *Tr*

3.3 Multispace Random Projection (MRP) Experiments

3.3.1 Performance Results

To evaluate the performance of MRP on the best performing SVM classifier, we conceal the eigenGabors through MRP before feature matching. During the experiment, setting 4 of Table 2 is adopted. Again, we perform ten runs for each of 4 samples with different random partitions between training and testing images. Then, the results are averaged. egsvm, mrpsvm-m, mrpsvm-m(stolen-token), mrpsvm-m(stolen-biometrics) denote eigenGabors, legitimate-token, stolen-token and stolen-biometrics scenarios respectively. The feature length of eigenGabors (egsvm) are fixed as $d = 100$. Besides, the eigenGabors is normalised to unit length before MRP. To infer the recognition performance, three different scenarios (as described in Section 2.2.1) are considered, i.e. legitimate-token, stolen-token and stolen-biometrics scenarios. For these three scenarios, the eigenvectors are extracted from the Gabor feature vectors with the dimensionality varied from 10 to 100 in intervals of 10. Later on, the normalized eigenGabors are used in these three different scenarios.

Table 4 shows the performance comparison of eigenGabors, legitimate-token, stolen token and stolen biometrics scenarios. From Table 4, for d=10…100, it is clearly shown that legitimate-token's EER results outperform the eigenGabors's EER results (egsvm in this context). Legitimate-token and stolen-biometrics scenario attain better EER results than the eigenGabors in the range of $40 \le m \le 100$. On the other hand, the EER of eigenGabors and stolen-token scenarios are equal when $m \approx d$. In other words, stolen-token scenario reverts the system to its original state when $m \approx d$.

For legitimate-token and stolen-biometrics scenarios, the MRP performance can be boosted through a classifier with better separation whilst the classifier quality will determine the performance of MRP in stolen-token scenario. However, it would not be poorer than its original method i.e. classification using eigenGabors without MRP. This is favourable in practical application whereby MRP survives either in stolen-token or in stolen-biometrics attack and also capable to offer significant improvement in verification setting with better classifier.

From Table 5, it can be observed that the impostor distribution's mean and standard deviation for both eigenGabors and stolen-token scenario are the same when $m \approx d$. Similar result can be seen in the genuine distribution where the mean and standard deviation of stolen-token scenario are equivalent to the mean and standard deviation of eigenGabors when $m \approx d$. This indicates that the preservation of genuine-impostor distribution is maintained under SVM framework when $m \approx d$. It is also depicted in Figure 4 where the separation of genuine-impostor class distribution for eigenGabors and stolen-token scenario is almost identical. The preservation of the intra-class variations (genuine distribution) as well as inter-class variations (impostor distribution) asserts that SVM statistical properties are preserved under the MRP framework.

3.3.2 Diversity Test

In order to fulfil the diversity requirement of cancellable biometrics, Pairwise Independent Test is conducted to inspect whether the MRP template with PRN A and MRP template with PRN B (both with same fingerprint feature) are associated. The same fingerprint feature is mixed with different PRNs. We observe that the scores generation procedure has followed the impostor distribution as described above. Figure 5 exhibits the Pairwise Independent Test of MRP. As indicated in Figure 5, the mean and standard deviation are 1.0177 and 0.2982, respectively. It can be concluded that MRP is Pairwise independent as the histogram of Figure 5 approaches the independent and similarly distributed (i.i.d) random variables drawn from Gaussian distribution, **N**(-1610.2, 407.3). This means that there is almost no correlation between the refreshed MRP template and old MRP template. Therefore, random number refreshment is equivalent to issue a new template to the user. In the real application, every user is assigned a unique random number. Hence, only the respective template is renewed in the event of compromise.

scenario \ *d*	10	20	30	40	50	60	70	80	90	100
mrpsvm-m	36.69	10.38	1.84	0.63	0.37	0.36	0.31	0.30	0.37	0.48
mrpsvm-m (stolen-biometrics)	35.98	11.06	2.23	0.87	0.70	0.56	0.43	0.42	0.49	0.57
mrpsvm-m (stolen-token)	39.75	14.58	4.80	2.31	1.58	1.51	1.16	1.12	1.19	1.23
egsvm										1.23

Table 4. EER results (%) of egsvm, mrpsvm-m, mrpsvm-m(stolen token) and mrpsvm-m(stolen biometrics) for RBF SVM with different feature length, *d*

scenario	m	$\mu_{genuine}$	$\mu_{imposter}$	$\sigma_{genuine}$	$\sigma_{imposter}$
egsvm	-	-0.6445	1.0668	0.4428	0.3188
mrpsvm-m (stolen-token)	100	-0.6445	1.0668	0.4428	0.3188
	90	-0.6582	1.1212	0.4410	0.3463
	80	-0.6742	1.1880	0.4380	0.3776
	70	-0.6867	1.2799	0.4363	0.4298
	60	-0.6935	1.3971	0.4549	0.5094
	50	-0.6897	1.5195	0.4517	0.5755
	40	-0.6440	1.6600	0.5040	0.6513
	30	-0.3766	1.7365	0.5865	0.6831
	20	0.4513	1.3946	0.3854	0.5092
	10	0.9244	1.0647	0.2172	0.3227

Table 5. Statistics measurement of egsvm and mrpsvm-m(stolen-token) for RBF SVM

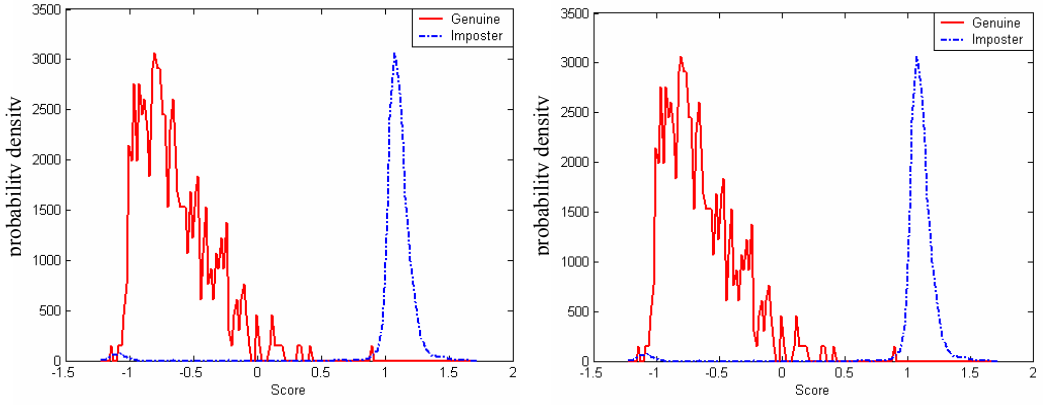


Figure 4. Genuine and impostor class distribution for: (a) egsvm; (b) mrpsvm-m (stolen- token).

4. DISCUSSIONS

MRP maps the data onto a random subspace while preserving the pair-wise distance that is quantified by dot product. Hence, MRP can be generalised for Linear SVM as the original data points are quantified by the dot product, $\boldsymbol{p}_i^T \boldsymbol{p}_j$ in Lagrange dual optimisation problem as shown in Eq. 2. Furthermore, a dot product $\boldsymbol{p}_j^T \boldsymbol{p}$ also appears in decision stage as exhibited in Eq. 4. Therefore, SVM inherited the MRP characteristics that described in [And07]. As stated in Section 2.2.2.3, to extend the idea to non-linearly separable data, a non-linear kernel is needed. The dot product of $\boldsymbol{p}_i^T \boldsymbol{p}_j$ in input space is substituted with $K(\boldsymbol{p}_i, \boldsymbol{p}_j)$ in the Langrage dual optimisation problem as displayed in Eq. 3. On the other hand, the dot product of $\boldsymbol{p}_j^T \boldsymbol{p}$ in input space is substituted with $K(\boldsymbol{p}, \boldsymbol{p}_j)$ in deciding the decision value as shown in Eq. 5. The non-linear kernel measures the similarity between two feature vectors. Typically, it represents the dot product between two representations in the transformed space in which the data are linearly separable.

The inheritance of the MRP characteristics (e.g. reusability and diversity properties) by MRP-SVM is concretised by the experiment results given in Section 3.3.2. The experiment results shown in Table 5 and Figure 4 deliver our assertion that SVM statistical properties are preserved under the MRP framework.

In short, SVM is another classifier that can be used in MRP as it contains the property of dot product and non-linear kernel. SVM is still preserved under MRP framework. Moreover, it has better discrimination power compared with normalised matching metric.

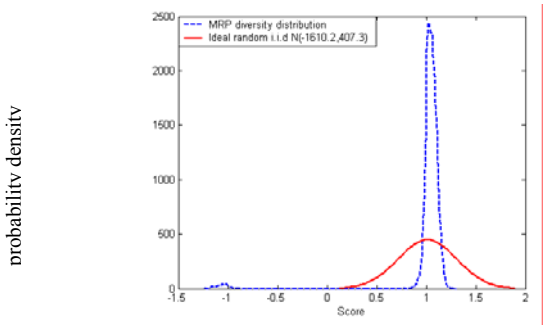


Figure 5. Pairwise Independent Test of MRP using RBF SVM

5. CONCLUSIONS

A complete secure digital camera based fingerprint verification system is proposed. This system uses "touch-less" based method. The proposed pre-processing which encompasses of skin colour detection, local normalisation, fingerprint segmentation, image enhancement, and core point detection resolves the problems exist in its images. After pre-processing, MRP-SVM which comprises Gabor feature extraction, PCA, MRP and SVM verification is performed. The employment of MRP-SVM protects the template whilst improving system performance. In MRP-SVM, SVM with high discrimination power is preserved under MRP framework as it has the property of dot product and non-linear kernel. Experiments show that MRP-SVM contributes high recognition performance and functions well without compromising the verification performance in the event of compromised token. MRP-SVM is proven to fulfil other cancellable biometrics properties, i.e. diversity property and non-reversible property into the bargain.

6. REFERENCES

[And03] Andrew Teoh, B. J., Ong, T. S., David Ngo, C. L. and Sek, Y. W.. Automatic Fingerprint Center Point Determination by Using Modified Directional Field and Morphology. in *AI 2003: Advances in Artificial Intelligence: 16th Australian Conference on AI*, 2003.

[And04] Andrew Teoh, B.J., David Ngo, C.L and Alywn Goh. Personalised Cryptographic Key Generation Based on FaceHashing. *Computers and Security J.*, No. 23, pp. 606-614, 2004.

[And07] Andrew Teoh, B.J. and Chong, T.Y. Cancellable Biometrics Realization with Multispace Random Projections. *IEEE Transaction on Systems, Man and Cybernetics Part B - Special Issue on Recent Advances in Biometrics Systems* 37, No. 5, pp.1096-1106, 2007.

[Ang05] Ang, R., Rei, S. N. and McAven, L. Cancelable Key-Based Fingerprint Templates. in *Proceedings of The 10th Australasian Conference on Information Security and Privacy*, pp. 242-252, 2005.

[Bio02] Biomedical Imaging Group. Local Normalization. [Online] http://bigwww.epfl.ch/demo/jlocalnormalization/, 11 February 2002.

[Bou06] Boult, T. Robust Distance Measures for Face-recognition supporting revocable biometrics token. in *7th International Conference Automatic Face and Gesture Recognition* 2006.

[Bur98] Burges, C. J. C. A Tutorial on Support Vector Machines for Pattern Recognition. *Knowledge Discovery and Data Mining* 2, No.(2), pp. 121-167, 1998.

[Che06] Chen, Y., Parziale, G., Diaz-Santana, E. and Jain, A. 3D Touchless Fingerprints: Compatibility with Legacy Rolled Images. in *Proceedings of Biometric Symposium, Biometric Consortium Conference*, 2006.

[Chi05] Chikkerur, S., Cartwright, A. and Govindaraju, V. Fingerprint Image Enhancement Using STFT Analysis. in *Pattern Recognition*, pp. 198-211, 2005.

[Dig00] Digital Descriptor Systems Inc. Amendment to Registration of Securities of a Small-Business Issuer. [Online]http://www.secinfo.com/dsvrb.52N5.htm, 2000.

[Hie06] Hiew, B.Y., Andrew Teoh, B.J. and David Ngo, C.L. Preprocessing of Fingerprint Images Captured with a Digital Camera. in *9th International Conference on Control, Automation, Robotics and Vision (ICARCV 2006)*, 2006.

[Kar05] Kargupta, H., Datta, S., Wang, Q., Sivakumar, K. Random-data perturbation techniques and privacy-preserving data mining. *Knowledge and Information Systems* 7, No.4, pp. 387-414, 2005.

[Lee99] Lee, C. J. and Wang, S. D. Fingerprint Feature Extraction using Gabor filters. *Electronics Letters* 35, No. 4, pp. 288-290, 1999.

[Lee06a] Lee, C., Lee, S. and Kim, J. A Study of Touchless Fingerprint Recognition System. in *Proceedings of Joint IAPR International Workshops, SSPR 2006 and SPR 2006*, pp.358-365, 2006.

[Lee06b] Lee, C., Lee, S., Kim, J. and Kim, S. Preprocessing of a Fingerprint Image Captured with a Mobile Camera. in *Proceedings of International Conference on Biometrics*, pp. 348-355, 2006.

[Mai06] Mainguet, J. F. Fingerprint Sensing Techniques. [Online] http://perso.orange.fr/fingerchip/biometrics/types/fingerprint_sensors_physics.htm, 2006.

[Mal03] Maltoni, D., Maio, D., Jain, A. K. and Prabhakar, S. *Handbook of Fingerprint Recognition*, New York: Springer-Verlag, 2003.

[Rat01] Ratha, N., Connell, J. and Bolle, R.. Enhancing Security and Privacy in Biometrics-based Authentication Systems. *IBM System Journal* 40, No. 3, pp. 614–634, 2001.

[Sav04] Savvides, M., Vijaya Kumar, B. V. K. and Khosla, P. K. Cancelable Biometrics Filters for Face Recognition. *Int. Conf. of Pattern Recognition*, No. 3, pp.922-925, 2004.

[Sch99] Schneider, J. Biometrics: Uses and Abuses. *Commun. ACM.*, No. 42, pp.136, 1999.

[Son04] Song, Y., Lee , C. and Kim, J. A New Scheme for Touchless Fingerprint Recognition System. in *Proceedings of International Symposium on Intelligent Signal Processing and Communication Systems*, pp.524- 527, 2004.

[Tur91] Turk, M. and Pentland, A. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 1991.

Visualization of the continental drift in real-time

Lorenz Rogge
TU Braunschweig
rogge@cg.tu-bs.de

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

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

Abstract

We present an approach to simulate and visualize the paleontological change of the terrestrial globe through time. As input, maps of the Earth at different points in time, each showing a specific stage in the Earth's development, are used. To display the continuous change of Earth through time, smooth transitions between consecutive input maps are computed by manual correspondence matching. Our system makes it is possible to display the Earth's paleogeographic development as an interactive globe in real-time.

Keywords: plate tectonics, paleography, interactive real-time visualization, 3D modelling, morphing

1 INTRODUCTION

Paleogeography is an important as well as popular topic in geography education, especially when considering the high interest in the fields of climate change and fossil fuels. Historical views of the Earth's configuration have been printed in geological atlases for decades, but still a simple and interactive technique to display the changing appearance of Earth is missing. While there are scientific simulations that create highly correct data in order to support research in geology and geophysics, they are not suitable for a interactive real-time display of the tectonic movement through time. Looking at the *PLATES Project* of the University of Texas [6], the data generated in simulations is used to create animated video sequences. These, however, are not interactive and only allow to watch the continents movements from a fixed point of view. Alternatively, Artistic animations of the continents' movement must be produced in a very labor-intense manner. Our goal is thus to create a system to display the change of the Earth's surface as correct as possible and interactively in real-time. It is designed to be an educational tool that offers a much richer experience than static maps or globes.

A set of texture maps, each representing the Earth's surface at a specific point in time, is used as initial input data and processed by manual correspondence matching to create geometrical data for each continent. The shape and movement of the continents are then interpolated in real-time. (see Fig.1). We explicitly allow for a manual processing of the data. In order to keep human labor at a minimum, we employ a tectonic morphing model that enables the user to create these transitions swiftly. The presented system consists of two parts. The first part is an editor, which allows the user to process the initial textures and export the data into a designated format. This data consists of several meshes representing the continents and their movement and transformation over time. The second part is the interactive viewer, which gives the user the opportunity to view the continental drift at any time from any viewing angle at any point in time.

In Section 2, related work of paleogeographic visualization system and also of morphing techniques are presented. Section 3 gives a brief introduction to plate tectonics in general. Our tectonic morphing model is introduced and discussed in Section 4. The manual creation of meshes and their transformations with our editor application is described in Section 5. Finally, we present results in Section 6 before we conclude in Section 7.

WSCG 2010 conference proceedings, ISBN 97-97797-7-7
WSCG'2010, February 1 – 4, 2010
Plzen, Czech Republic.

2 RELATED WORK

2.1 Tectonics simulation and animation

Various systems exist that produce animations of the Earth's lithosphere through time. These can be categorized into two major groups of tectonic animation systems.

The first group is directly connected to tectonic simulations. An example for this software is the *PLATES Project* [6]. Animations of continental drift are a by-product of actual tectonic simulations. These reconstructions are of high precision, however, each visualization output requires a remarkable amount of preprocessing that inhibits real-time computation. Furthermore, only raw geographic information is displayed.

The second group focuses on the visual aspects. Taking the raw geological data of actual simulations as a basis, they enrich the data by taking information about vegetation and climate into account. The resulting data sets usually consist of hand-painted texture maps at discrete time steps, e.g. [4], [13]. Thus, content creation is usually labor-intensive and is only feasible for a few

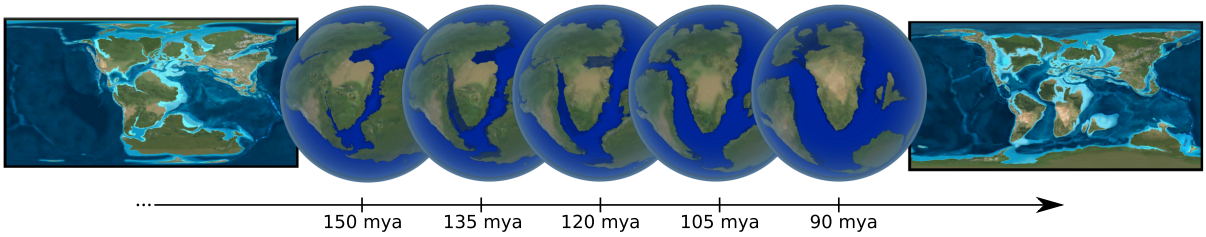


Figure 1: Morphing of continental structures between two time slices. The two texture maps (left and right) serve as input. A three-dimensional depiction of intermediate points in time is obtained by morphing.

dozen time steps. Animations are created by morphing between these images. Again, artists have to create transitions manually. Two prominent examples are [5] and [11], where pre-rendered animations are projected onto a spherical display.

As our system is also designed to give a visually pleasing experience, we designed our system to work with artistically enhanced texture maps. Compared to existing approaches, we contribute two major advantages. First, we keep the creation of transitions as simple and fast as possible by using a manual correspondence matching technique specifically tailored for paleogeographic data sets. Second, we do not need to prerender the animation, so that the Earth's surface can be rendered on-the-fly for any viewpoint at a given point in time.

2.2 Morphing Techniques

In order to investigate the possibility of creating transitions between time instants automatically, we evaluate state-of-the-art morphing techniques. At first thought, two dimensional morphing techniques seem adequate to solve the problem. The principle of image morphing in 2D space is very common in computer graphics. Many approaches and solutions have been developed to automatically transform one image into another. This is often done by computing the optical flow between a pair of images, a well-established technique [2]. A survey of recent approaches can be found at [3]. Image based morphing is able to smoothly transform both shapes and colors of objects in these images. The problem remains that an adequate 2D parametrization of the surface has to be found. Cylindrical, azimuthal or conical projections are commonly used for this task. All these projections encounter the one common problem of mapping singularities of the poles. Conformal mapping [10] allows to choose the locations of the poles on the globe's surface. However, there are no two opposite locations on Earth that are covered by sea at all paleogeographic stages. In addition, distortions appear in all regions that are not in the equatorial zone.

Therefore, morphing 3D geometry is a more appropriate approach to generate interpolated views of the terrestrial globe. In fact, transforming two meshes to an intermediate spherical representation is a common approach to the general problem of polyhedral morphing [9], a thorough survey can be found at [1]. Avoiding the pitfalls of 2D morphing, the remaining problem with 3D mesh morphing is to find correspondences between two such meshes. Since the differences in shape and appearance between two states of the Earth can be quite large and may often be ambiguous, an automatic feature extraction and matching is not feasible in our case. Whole islands or even continents disappear between two consecutive depictions of the Earth (see Fig.12). Two continents with distinctive shorelines may also form a continuous super-continent (e.g., Pangea or Gondwana) in another time slice (see Fig.1). To summarize, automatic 2D or 3D morphing techniques are neither robust nor accurate enough to cope with the available and coarse data sets. Instead, we rely on a manual approach that is closely linked to the general plate tectonics model and produces visually convincing results with only little user input.

3 BACKGROUND

At the beginning of the 20th century geologists and physicists developed a new theory of the physical behavior of the Earth's composition and its development over time. Prior to this new theory the Earth was assumed to be rigid. The new theory assumed a dynamical system of continental plates drifting over the Earth's liquid interior, the almost matching coastlines on either side of the Atlantic Ocean initiated the development of this theory of continental drift. Alfred Wegener pointed out, that the congruency of the coastlines of South America and Africa never could have been caused by rising water levels or sinking landmass [14]. Thus, the theory of diverging landmasses, which were touching at these coastlines in the past, was more likely. J. Tuzo Wilson also proved the fact of diverging and moving continental plates using the example of the Hawaiian Islands forming a nearly straight line of

volcanic islands [15]. Observations in submarine geology, change in magnetism and volcanic activities triggered further research in this field. Since then, geologists have evaluated data from geological structures distributed over the present continents to generate historical maps, which display the configuration of the Earth's surface in the past. The cause for the drift is assumed to be convection of magma within the mantle of the Earth. Upwards flowing magma pushes onto the planet's crust and pushes it to either side. New crust is being created at the region where the continental plates drift away from each other by the uprising magma which hardens as it reaches the surface. According to [13], evidence of the continents' past position is given by paleomagnetism, linear magnetic anomalies, paleobiogeography, paleoclimatology, and geologic history. Due to the individual motion of the continental plates, three basic zones can be found when two continents are adjacent: divergence zone, convergence zone and transform zone (see Fig.2).

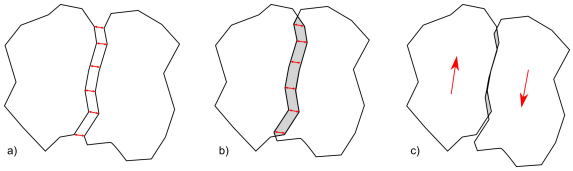


Figure 2: Three typical zones of continental drift: a) divergence zone, b) convergence zone and c) transform zone

The motion of each plate can be described as a motion along circles on the spherical surface. Every point of a plate moves along its own circle around the sphere. The center point of these concentric circles is called the *Euler pole* [8]. Every point of a plate is rotated by the same angle around this Euler pole as the plate moves over the surface. In order to describe a motion for a single continental plate only three values, the longitude and latitude of the Euler pole and the angle of rotation around it, have to be defined. For introductions into paleogeography, we refer to [7], [12].

4 A TECTONIC MORPHING MODEL

As stated in section 2.2, using an image based morphing technique causes problems near the poles. E.g., a continent that covers the south pole at one point of time and moves away towards the equator at another point of time changes its projected shape significantly, even if its actual shape remains unchanged (see Fig.3).

As far as we know, there is no parametrization of a spherical surface that does assign only a single and unique set of parameters to a sphere's pole. These singularities in 2D parametrization seem to be inevitable. Therefore, morphing in the projected texture space given by the parametrization of the sphere can be very hard for some configurations. Another problem is that

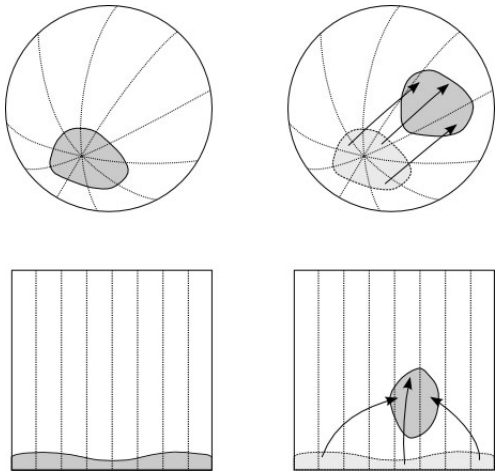


Figure 3: Morphing of a continent situated around the polar region in spherical space (top) and in texture space (bottom). A simple linear translation of the continent in spherical space with no change in shape (upper right) results in a complex distortion in its shape in texture space (lower right).

automated morphing in texture space using optical flow requires to identify and match corresponding features in a pair of texture maps. When having two converging continental plates, that merge from one time slice to another, features in the merged region cannot be identified properly and thus, no plausible flow in that region can be computed. Also, the maps available are often not very similar, since the time difference between two maps may be several million years. In this period of time the face of the Earth may change dramatically and it is be hard to identify corresponding regions even for the human eye. The solution to this problem is to choose another morphing approach for this application. Instead of morphing textures it is more appropriate to morph the geometry itself. Continents may be modeled as single patches of geometry emphfloating on the globe's surface. Using the Euler pole notation a continuous motion of a continent can be described by interpolating the given values between time slices. Given a geometry patch as a polygonal mesh, every vertex can be moved separately by interpolating the vertices' positions between consecutive time slices. Thus, the shape of a geometry patch can be altered locally as well in addition to the basic movement. Together this yields a technique of moving and morphing continents on the globe's surface. Neglecting tectonic movement below sea levels and elevation change in general, this model still bears much resemblance to the original plate tectonics model. Changes in shape and elevation can both be taken care of by applying local deformation, which leads to the same visual effect. The problem of merging and separating continents is also solved, since

a new set of geometry can be used for each time slice. In the case of two merging continents one can use two geometry patches for each continent while they are still separate. Upon aggregation of the two continents, a single geometry patch is used to represent this new continent in the future.

4.1 Continental Meshes

We developed an appropriate model called *Continental Mesh* which is capable of transforming a patch of geometry belonging to one point in time to how the patch would look like in the succeeding point in time. Ideally, one continental mesh corresponds to a single continental plate or a coherently moving group of islands.

A mesh may consist of several faces which are stored in a designated list. The faces' vertices are stored separately and referenced by the faces for memory optimization. Each vertex has an assigned transformation vector which allows to alter the mesh's shape during transformation. Furthermore, angles and rotational axes are defined for each Continental Mesh on the globe's surface. In order to morph and render the continents over time, it is necessary to blend texture colors smoothly. This may happen when a region changes its vegetation or a mountain emerges at a region of two colliding continents. For this purpose, a Continental Mesh contains two lists of static texture coordinates - one for the texture map of the first of two succeeding time slices and another one for the second. While morphing between two time slices, the color values of both textures can be blended linearly by the ratio of transformation.

4.2 Mesh transformation

To move a Continental Mesh on the spherical surface, the common representation of *Euler poles* [8] can be realized by quaternion rotation. A Euler Pole depicts a point on the globe's surface, defining a rotational axis through this point an the center of the globe. Every vertex of a mesh may be moved on a spherical surface by rotating it around an axis defined by a given Euler Pole. Having the sphere originate at $(0,0,0)^T$ in space the coordinates of an Euler Pole already represent the vector of the rotational axis. Furthermore, a mesh may also rotate around its central point while simultaneously moving along a trajectory on the spherical surface. A rotational axis through a central point is used to rotate all vertices and achieves a circular rotation of the complete mesh. The set of per-vertex-transformation vectors is precomputed by storing the euclidean distance of every pair of corresponding vertices. Because a rotational technique is used for moving whole meshes over the globe's surface for longer distances, it is negligible that the transformed vertices do not move on an arc-like trajectory but on straight lines through space, since these transformations remain quite small. Given the vertex set V, the transformation vectors $\vec{t_i} \in T \subset \mathbf{R}^3$, the global rotation axis $\vec{r} \in \mathbf{R}^3$ and the corresponding angle $\phi \in \mathbf{R}$, the circular rotation axis $\vec{r_c} \in \mathbf{R}^3$ and the corresponding angle $\theta \in \mathbf{R}$, a transformation ratio $r \in \mathbf{R}$ and a rotation-function $rotate : \mathbf{R}^3 \times \mathbf{R}^3 \times \mathbf{R} \rightarrow \mathbf{R}^3$ using quaternion rotation, the following steps have to be computed consecutively:

1. Rotate every vertex around the global rotational axis $\vec{r}$ about the ratio r of the rotation-angle ϕ (see Fig.4)

$$\forall \vec{v_i} \in V \qquad \vec{v_i'} = rotate(\vec{v_i}, \vec{r}, r \cdot \phi) \tag{1}$$

2. Rotate every vertex around the central rotational axis $\vec{r_c}$ about the ratio r of the rotation-angle θ

$$\forall \vec{v_i'} \in V' \qquad \vec{v_i''} = rotate(\vec{v_i'}, \vec{r_c}, r \cdot \theta) \tag{2}$$

3. For every vertex $\vec{v_i}$ add the ratio r of the corresponding transformation-vector $\vec{t_i}$ (see Fig.5)

$$\forall \vec{v_i''} \in V'' \qquad \vec{v_i'''} = \vec{v_i''} + r \cdot \vec{t_i} \tag{3}$$

The vector $\vec{v_i'''}$ is the final rotated vertex, which can be now rendered to the screen.

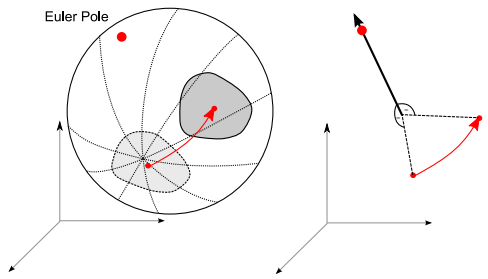


Figure 4: Rotation of mesh around a global axis. After defining an Euler pole for a continental plate, it can be rotated around the axis through this point.

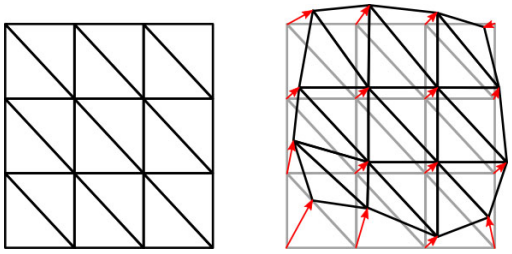


Figure 5: Every vertex has its individual 3d transformation vector, allowing smooth transformations of the mesh's shape.

The texture coordinates are constant for all times and only the ratio of blending two texture layers together changes over time.

5 MANUAL CORRESPONDENCE MATCHING

By using the Continental Mesh model as a representation of the continental drift, it is necessary to preprocess the texture data first. The texture maps of the continents are a basis for the selection and manual correspondence matching of continents. Creating a simulation of the Earth's tectonic movement consists of these three succeeding steps.

- Loading and pre-processing of input data (texture maps)
- Using texture data to create morphable continental meshes
- Morphing and display of meshes

The second step involves manual work, since for every time slice a set of meshes has to be created and adapted to the following time slice.

5.1 Input data

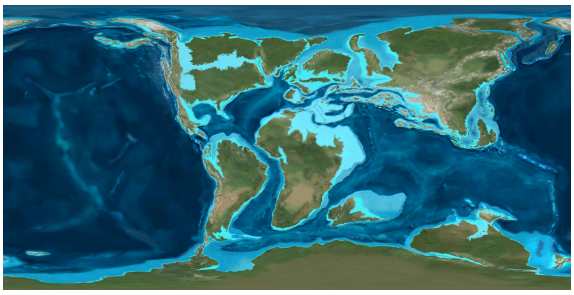


Figure 6: This is a sample map of the Earth's appearance 105 million years ago (from [4]), which is used as an input texture map for this time slice. The input texture maps include information on both vegetation (e.g., desert regions in the subtropics) and climate (e.g., no ice caps on poles).

Altogether, we used 26 texture maps of rectangular shape distributed between 600 million years ago and the present day (see Fig.6). This data set was provided by the Northern Arizona University [4]. Each texture map has a resolution of 3000 px x 1500 px. Time differences between two consecutive textures range from 15 million years to 40 million years. In addition to geographic information, differences in vegetation and climate are visible.

After loading each texture from disk, pixels belonging to oceans are tried to be identified, using their color-hue value to distinguish if a given pixel represents water or landmass. So far this is not an adaptive process and some regions might be labeled false. However, using the texture set provided by [4] oceans are identified very well using this technique. During all succeeding stages, only pixels depicting continents, islands and ice masses, such as polar caps, are considered.

5.2 Editor Workflow

The editor is used for creating smooth transitions between each pair of consecutive time slices. It is loaded with a certain configuration defining which time slices are available and which texture files to load from disk (see Fig.7). This initial data provides a textured spher-

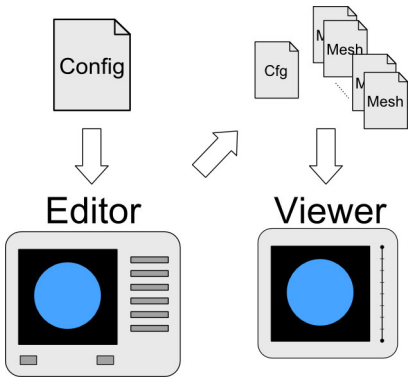


Figure 7: Processing and propagation of data throughout the program. The editor uses a config file containing information about available time slices and file paths to their texture maps. After manual correspondence matching of the input data, a configuration file for the viewer and a set of meshes is created and finally displayed by the viewer.

ical, static mesh for each time step. The user is able to move back and forth between these time steps and may also rotate the globe in any preferred way. Having a single globe for each time slice, the user selects several disjoint regions, each corresponding to a single continental plate (see Fig.9). These highlighted regions are converted into separate, disjoint Continental Meshes. Please note that these selections do not actually have to correspond to the actual continental plates. Any selection of a region that moves more or less uniformly is valid. For each Continental Mesh, a global transformation as well as several local transformations are defined by the user. The texture coordinates of the transformed vertices according to the next time slice's texture are computed automatically. This process has to be done for each continental plate in each time step. Creating and editing Continental Meshes from a textured globe follows a certain workflow (see Fig.8).

1. Selection of faces
2. Creation of a mesh from this selection
3. Adjusting the mesh to the next time slice
4. Saving the mesh to disk

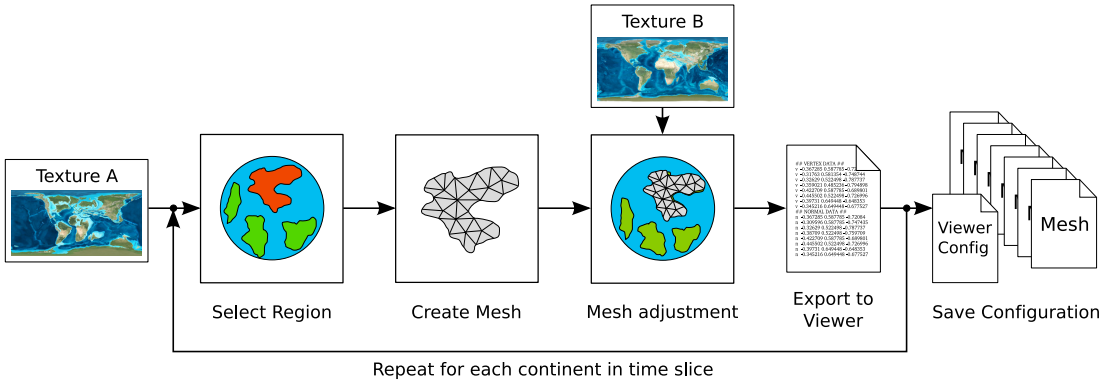


Figure 8: Editor Workflow. The user selects a continental plate from the first time slice (Texture A) and provides correspondences to the second key frame. This must be done for every continental plate. The meshes and their transformations are saved to disk.

At first the user selects faces on the globe's surface representing a single continental plate. The user may adjust and refine the shape of the current selection, which is quite useful in regions where two continental plates are very close or touch each other. Then the selected region is exported into a new Continental Mesh. This mesh can be moved on the globe's surface to match the corresponding region's position in the succeeding time slice. Also its shape may be adjusted by selecting and moving single vertices. This helps to match coastlines and important details of the moved region. To give a feedback of the current matching accuracy, the underlying globe is textured with the succeeding time slice's texture in this editing step. In common cases and depending on the size of a given continental plate, a created mesh consist of about 100 to 300 vertices. Although this may sound very labor-intensive, this part of the workflow is actually quite efficient due to two facts. First, the alignment of continental plates is often quite good after setting a global transformation during the previous step. This transformation has to be defined only once per mesh and affects all vertices at the same time. Second, the mesh resolution is quite coarse (we place a vertex at roughly each 4 degrees of longitude or latitude), so that in most cases only a few vertices along the coastal lines have to be adjusted. Still, we observe that this resolution is sufficient to achieve the desired visual accuracy. The adjustment itself is done by a simple click-and-drag action per vertex while the moved vertices always stay on the globe's surface during translation. After aligning everything properly, the user can save this particular mesh and continue to select and modify new continental meshes. Finally, everything is saved into a configuration file.

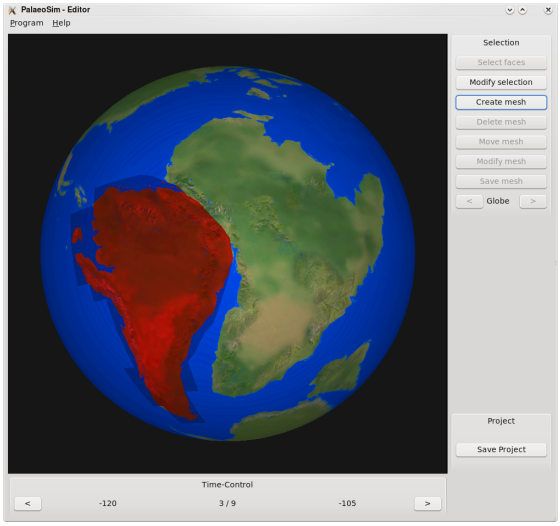


Figure 9: Editor Window. While selecting a mesh (here representing the South American continent). The user is able to select specific regions and move/adapt them to their position in the succeeding time slice. The buttons on the right allow to select and modify meshes and save this configuration. The buttons on the bottom are used to navigate through available time slices.

6 RESULTS

The task of displaying the continental drift in real-time is solved by using the Continental Mesh data structure for each continent and a datastucture to manage several time slices and their corresponding sets of Continental Meshes. We are able to display a linear interpolation of a continent's shape and texture between two succeeding time slices. Using different sets of Continental Meshes for each time step also makes it easy to create merging or even diverging continents. Since

such events frequently occurred in Earth's history, being able to display this process in a proper way is essential. After having created appropriate data from texture maps using the editor module, the viewer application (see Fig.11) can be started with the created configuration (see Fig.10). The user is able to navigate through time using a timeline slider. Manual correspondence matching between two time slices takes approximately 30 to 60 minutes, depending on the number of continents and islands present at that time. For the whole data set of 26 time slices, two working days were spent. Using a higher resolution than this does not necessarily improve the result's quality, since continental motions usually do not change very rapidly and a higher resolution would only increase the amount of manual work. So using a larger texture set for the same time span is not necessary.

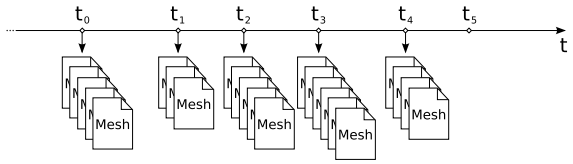


Figure 10: A basic display of the timeline used by the viewer for loading the correct set of meshes for each time slice. For every point in time the set of meshes of the preceding time slice is chosen and displayed after transformation.

The viewer loads the correct set of meshes for any given point in time and transforms these meshes by linearly interpolating their position, orientation, shape and texture. This is done in real-time using the OpenGL pipeline and taking advantage of GLSL shader programs to interpolate textures and display shaded meshes. Since every Continental Mesh is handled seperately, they are accumulated during rendering into a final framebuffer, which will be displayed after all continental plated have been rendered. Since only approx. 3200 faces are rendered and only two textures have to be loaded separately, real-time frame rates can be achieved on standard hardware. Shaders, that adapt to the graphics cards capabilities, e.g. available texture units used during accumulation, are used to speed up the rendering process. When the user drags the time slider, the surface of the globe immediately alters its appearance (see Fig.12). The angle of view can be changed by dragging and rotating the rendered globe in any direction. The user might also zoom into view parts of the globe in more detail. The motion of each continental plate drift can be viewed at any speed and from any angle.

7 CONCLUSION

Using the presented approach, it is possible to display the continental drift through time in a visually pleasing

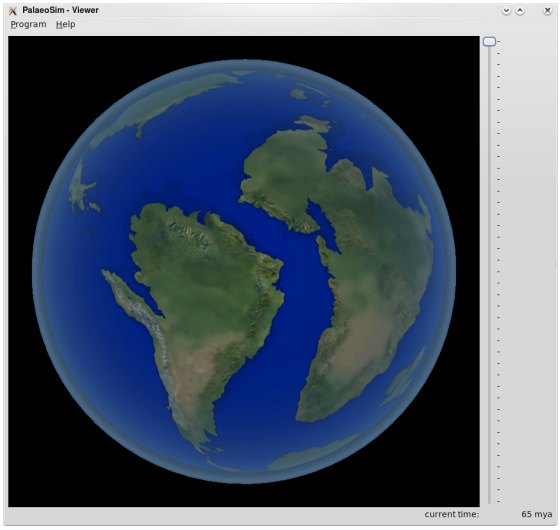


Figure 11: Viewer window displaying South America and Africa 105 mya (million years ago). By using the time slider on the right, the user is able to freely navigate through the available period of time.

Figure 12: Several interpolated views of the globe in four steps between 65 mya and 90 mya. One noteworthy event during this period is the breakup of eastern Siberia, which separated Sakhaline and the Japanese islands from the mainland. An automatic matching in this region between the two time instances can hardly be devised.

manner. The user is able to view the motion of continental plates and their transformation on a 3D globe from arbitrary perspectives. Thanks the employed morphing model, the amount of time that has to be spent on producing transitions between time slices is reasonable. The user does not need to know how to use complex modelling software such as Maya or 3dsMAX to create and texture a morphing geometry, but instead is provided a system to quickly match continental plates through consecutive time slices. The editing process is very intuitive and is easy to use even for people not familiar with commercial modelling software. The set of possible actions to deform and move geometry is pre-

defined and the user is able to quickly create the desired results. In contrast to several exitsing pre-computed animations, the user is able to move back and forth in time at arbitrary speed and may watch the continental drift from any preferred perspective. Being able to view the earth's surface from all possible points of view makes it easier to understand the transformation processes that have happened in the past, which was the aim of our system. There are still some issues with the current system. Selection and transformation of geometry is so far a purely manual task. In the future, user interaction should concentrate solely on resolving ambiguous cases, e.g., by defining the global transformation of continents. State-of-the-art shape matching techniques could then provide for accurate local motion estimation of coast-lines and other fine structures. Also the linear texture blending allows not to properly display the formation of lakes, rivers and other details placed within continental meshes. Erosion techniques could be used to display more transformations in these cases more realistic. Also, the current project does not yet display the ocean beds. These are of interest since many diverging zones like the Mid-Atlantic Ridge, where new crust is being created through uprising magma, are located in these regions. Another major improvement will be the inclusion of altitude into the model. Although changes in local shape can currently make up for changes in altitude, a more realistic model would explain the appearance and disappearance of islands and mountain ranges. Also slices through the Earth's mantle would be possible, enabling a visual exploration of the underlying geology and tectonic processes.

REFERENCES

[1] Marc Alexa. Recent advances in mesh morphing. *Computer Graphics Forum*, 21(2):173–196, 2002.

[2] B. Schunck B. K. P. Horn. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[3] Simon Baker, Stefan Roth, Daniel Scharstein, Michael J. Black, J.P. Lewis, and Richard Szeliski. A database and evaluation methodology for optical flow. *Computer Vision, IEEE International Conference on*, 0:1–8, 2007.

[4] Ronald Blakey. Global Plate Tectonics and Paleogeography. `http://jan.ucc.nau.edu/~rcb7/`. [Online; Update: May 18 2009].

[5] Rick Companje, Nico van Dijk, Hanco Hogenbirk, and DaniĆa Mast. Globe4d: time-traveling with an interactive four-dimensional globe. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 959–960, New York, NY, USA, 2006. ACM.

[6] Insitute for Geophysics. The PLATES Project. `http://www.ig.utexas.edu/research/projects/plates/`. The University of Texas at Austin.

[7] C. M. R. Fowler. *The Solid Earth: An Introduction to Global Geophysics, 2nd Edition*. Cambridge University Press, 2004.

[8] Dipl.-Ing. Michael Heinert. *Systemanalyse der seismisch bedingten Kinematik Islands*, pages 30–32. Institut für Geodäsie und Photogrammetrie der Technischen Universität Braunschweig, 2008.

[9] Richard E. Parent James R. Kent, Wayne E. Carlson. Shape transformation for polyhedral objects. *ACM SIGGRAPH Computer Graphics*, 26(2):47–54, 1992.

[10] H. Drange M. Bentsen, G. Evensen and A. D. Jenkins. Coordinate transformation on a sphere using conformal mapping. *Monthly Weather Review*, 127(12):2733–2740, 1999.

[11] National Oceanic and Atmospheric Administration (NOAA). Science on a sphere. `http://sos.noaa.gov/`. [Online; Update: June 17 2009].

[12] Frederick J. Vine Philip Kearey, Keith A. Klepeis. *Global Tectonics, 3rd Edition*. WileyBlackwell, 2008.

[13] Christopher R. Scotese. The paleomap project. `http://www.scotese.com/`. [Online; Update: June 17 2009].

[14] Alfred Wegener. *Die Entstehung der Kontinente und Ozeane*. Alfred-Wegener Institut für Polar und Meeresforschung, 1915.

[15] J. Tuzo Wilson. A possible origin of the hawaiian islands. *Canadian Journal of Physics*, 41:863–870, 1963.

Accelerated Streak Line Computation Using Adaptive Refinement

Alexander Wiebel
Max Planck Institute for Human
Cognitive and Brain Sciences
Stephanstraße 1A
04103 Leipzig, Germany
wiebel@cbs.mpg.de

Qin Wang Dominic Schneider Gerik Scheuermann
Image and Signal Processing Group
University of Leipzig
PF 100920
04009 Leipzig, Germany
schopenkitto@hotmail.com, {schneider,scheuer}@informatik.uni-leipzig.de

ABSTRACT

We introduce an improved algorithm for streak line approximation in 2D and 3D time-dependent vector fields. The algorithm is mainly based on iterative refinement of an initial coarse approximation of the streak line. The refinement process is steered by a predicate indicating the local approximation quality. We apply the algorithm to several real-world data sets to prove its applicability and robustness. An error analysis and a comparison show that the algorithm produces more accurate streak lines in shorter time compared to previous approaches. Finally we show how the new algorithm can help to improve the construction of streak surfaces.

Keywords: Flow visualization, streak lines, adaptive refinement, unsteady flow, time-dependent vector fields.

1 INTRODUCTION

Unsteady flows are ubiquitous in nature and consequently in science. To be able to deduce the laws governing the behavior of flowing fluids scientist perform experiments and simulations. In both cases the effects under investigation appear to be *invisible* quite often. Making them visible is crucial and common. In experiments, a widely used technique is the injection of visible material into the fluid. Examples are dye for hydrodynamic and smoke for aerodynamic experiments. If a material is constantly injected from a fixed location, the patterns that become visible in the flow are called *streak lines*.

The same patterns are achievable for vector fields stemming from computational fluid dynamics (CFD), i.e. from flow simulations. The patterns are again called *streak lines*. Here a certain number of virtual particles is placed in the vector field at a fixed location at consecutive times and the particles are computationally traced through time and space [KL96, SMAM99]. The whole procedure will be described in more detail in the next section.

The aim of this paper is to significantly accelerate the computation of such streak lines in flow data given as 2D or 3D time-dependent vector fields. The central idea is to reduce the number of traced particles while keeping the accuracy of the streak line approximation as high as possible, i.e. to trace only particles at those parts of the streak line that experience strong deformation by complex patterns in the vector field.

2 STATE OF THE ART

The first part of this section outlines the mathematical basis for particle tracing and the techniques that were used for streak line approximation up to now. The second part summarizes work related to streak lines and their effective use for flow visualization.

Streak Line Computation

As already adumbrated, streak lines can be described as a continuum of particles that were injected into a flow at the same location but at different times. To enable their computational approximation we will give the mathematical definition of streak lines. To ease understanding of the definition, we first review the definitions of two other line types, streamlines and path line, too.

For the following definitions let

$$\mathbf{v} : \mathbb{R}^3 \times [t_{min}, t_{max}] \rightarrow \mathbb{R}^3$$

be a continuous time-dependent vector field. Let $\mathbf{a} \in \mathbb{R}^3$ be the position of a particle in space and let $t \in [t_{min}, t_{max}]$ be a certain time.

Stream Lines Stream lines are integral curves $\mathbf{c}_{\mathbf{a},t}(u)$ of vector fields which are tangential to the vectors of a field's domain. They can be interpreted as trajectories of particles in a steady flow. For time-dependent vector fields streamlines are of little use as they stay in a single time step. Thus they do not show actual

particle motion but theoretical trajectories of particles with infinite velocity.

$$\begin{aligned} u &\mapsto \mathbf{c}_{\mathbf{a},t}(u) \\ \mathbf{c}_{\mathbf{a},t}(0) &= \mathbf{a} \\ \frac{\partial \mathbf{c}_{\mathbf{a},t}}{\partial u}(u) &= \mathbf{v}(\mathbf{c}_{\mathbf{a},t}(u),t) \end{aligned}$$

Here u is a time-independent parameter, t selects the time from which the vectors $\mathbf{v}$ are taken and $\mathbf{a}$ is the streamline seed at $u = 0$.

Path Lines In contrast to stream lines, path lines $\mathbf{p}_{\mathbf{a},s}$ in unsteady flow, indeed, are the paths of moving particles. Path lines are obtained by integration over space *and* time.

$$\begin{aligned} t &\mapsto \mathbf{p}_{\mathbf{a},s}(t) \\ \mathbf{p}_{\mathbf{a},s}(s) &= \mathbf{a} \\ \frac{\partial \mathbf{p}_{\mathbf{a},s}}{\partial t}(t) &= \mathbf{v}(\mathbf{p}_{\mathbf{a},s}(t),t) \end{aligned}$$

Here s is the seed time. Note that path lines and stream lines are identical for steady flow.

Streak Lines Streak lines $\mathbf{l}_{\mathbf{a},t}$ are imaginary lines connecting the locations of particles that were released into a flow at consecutive time steps. The lines can be observed when looking at the particles at a certain time t. As streak lines consist of particles and we need to trace these along their paths through the field, it is useful to describe streak lines in terms of path lines:

$$s \mapsto \mathbf{l}_{\mathbf{a},t}(s) = \mathbf{p}_{\mathbf{a},s}(t) \qquad (1)$$

Note that t is fixed and s varies. Like path lines, streak lines coincide with stream lines in the steady case.

It is worth mentioning that there exists another type of streak lines. In contrast to ordinary streak lines, the seed point for the particles varies in this new concept, i.e. instead of a constant position $\mathbf{a}$ they use a location $\mathbf{a}(s)$ moving along a path $\mathbf{q}$. These so called *generalized streak lines* $\mathbf{l}_{\mathbf{q},t}(s) = \mathbf{p}_{\mathbf{a}(s),s}(t)$ where introduced by Wiebel et al. [WTS+07] just recently. Their paper shows the usefulness of generalized streak lines in the context of flow separation at boundaries of object immersed in the flow.

Related Work

A general overview of line based visualization of fields can be found in the State-of-the-Art report by McLoughlin et al. [MLP+09]. In the following we will go into detail only for a number of streak line techniques especially relevant for our work.

An early implementations of streak lines for large unsteady flow fields was reported by Lane [Lan93]. In the literature we found only few papers dealing with accelerating streak lines. All of the reported approaches are complementary to what we present. One of the first ideas was trying to accelerate streak line computation by simply accelerating particle tracing. The first paper in this direction was written by Kenwright and Lane [KL96]. They increase the performance of the particle tracing by improving the cell search which is necessary for interpolation in unstructured grids. Later papers use also improved integration schemes, see [MLP+09] for details.

A paper by Sanna et al. [SMA00] might seem similar to our work because they use *adaptive streak lines* for visualizing unsteady flows. However their method is quite different and does not have the aim to accelerate streak line computation. They are interested in a large number of densely seeded streak lines that produce visualizations similar to texture-based flow visualization [LHD+04]. Their approach is only adaptive regarding the seeding of the streak lines which is steered by the local vorticity of the flow field.

Becker et al. [BLM95] extend the one dimensional streak lines to flow volumes that are seeded at several positions lying on a polygonal surface instead of only from one position. Particles are injected and traced from all positions simultaneously thus producing an evolving volume. They use an adaptive insertion of particles to keep the flow volume dense. Their criterion for insertion of new particles is the distance between particles. This is similar to what we present as distance criterion for streak lines. We will show that a distance based criterion is not the best approach to steer refinement.

Just recently flow visualization research has focused on streak surface visualization. Streak surfaces can be observed when particles are seeded from a line instead of a point. They correspond to a continuum of streak lines. The methods reach from a straight forward algorithm producing special streak surfaces called *eyelet path surfaces* [WS05], over GPU implementations for structured grids [vFWTS08, BFTW09], to a method that manages a complete triangulation of the surface and is also applicable to data given on unstructured grids [KGJ09].

Although many papers report the effectiveness of streak lines [Lan93, KL96, MLP+09] for flow analysis others emphasize that care has to be taken when interpreting the resulting visualizations [Ham62, WH80, KS86].

3 ACCELERATED STREAK LINE COMPUTATION

Before we start to explain the actual technique, we should state that it is not needed for streak lines in

steady flow. As streak lines are identical to streamlines in the steady case, the usual streamline approximation methods are sufficient. Also note, that the presented technique is applicable to generalized streak lines without any changes because both, ordinary and generalized streak lines, have the same digital representation as a polyline connecting the particles injected at different times.

Idea

The idea of the approach we present is to accelerate the computation of a streak line by reducing the number of particles traces that have to be computed. It is clear that we can only compute an approximation of the real streak line because we can trace only a finite number of particles in finite time. However, the aim still has to be as accurate as possible. Thus, we try to trace as few particles as are needed for a certain accuracy. We begin with a very coarse approximation, i.e. with only few particles started at equidistant points in time. Each traced particle results in a sample on the streak line (Fig. 2 left). The samples are joined with straight lines to form a polyline approximation of the streak line. Actually, the polyline is a linear interpolation of the samples. Please note that even though the particles all have the same distance in time, the samples do not have to have the same distance in space. In fact, most of the time this is not the case because the streak line is usually stretched and compressed.

To increase the accuracy of the approximation one decreases the time interval between the traced particles. The simplest and most common approach for this refinement is to decrease all time intervals uniformly, yielding an equidistant seeding in time again. This is most conveniently achieved by seeding one new particle in each interval between the particles of the coarser approximation. Thus is it possible to reuse the previously computed samples. The sampling density is doubled by this step.

The approach we suggest is to add new particles only in intervals in which the accuracy is low, i.e to steer the refinement by the local accuracy. The middle and right images in Figure 2 show the result of the two approaches applied to the coarse approximation shown in the left image of the same figure. Comparing the number of samples it becomes clear that the common approach is less efficient, as it produces more samples for a comparable accuracy.

Accuracy Estimation

In order to be able to seed particles only in intervals with low accuracy we need to measure the accuracy somehow. The best measure would be the distance between the approximated line and the correct streak line. As we do not have the correct streak line, we need to estimate the accuracy from the approximation alone.

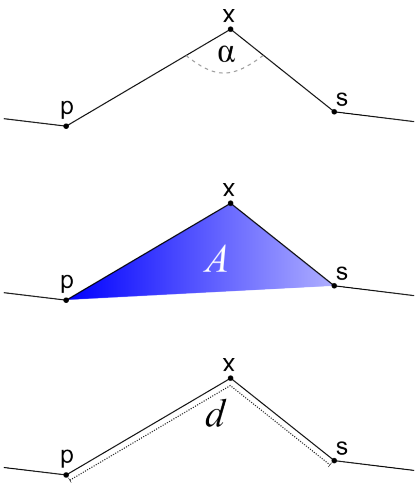


Figure 1: Different refinement criteria: angle α, area A and distance d.

Therefore we introduce three heuristic quality criteria in this section.

Figure 1 illustrates the three different quality criteria we employ for steering the refinement process. For each of the criteria we consider three consecutive points on the polyline representing the approximation: The current point x, its predecessor p and its successor s. Let $\mathbf{x}_p = p - x$ and $\mathbf{x}_s = s - x$.

Angle As the angle α between $\mathbf{x}_p$ and $\mathbf{x}_s$ comes closer to 180° the polyline becomes more similar to a straight line at x. As we use linear interpolation between the samples, a straighter polyline implies a higher accuracy. Thus we prescribe the desired quality using a minimum angle threshold k_α. The desired quality is reached if

$$k_\alpha < \arccos(\mathbf{x}_p \cdot \mathbf{x}_s)$$

with $\mathbf{x}_p$ and $\mathbf{x}_s$ being normalized vectors.

Area A small area A also indicates a relatively straight line. Thus we prescribe the desired quality using a maximum area threshold k_A. The desired quality is reached if

$$k_A > \frac{1}{2}\|\mathbf{x}_p \times \mathbf{x}_s\|.$$

Distance A small distance between samples naturally guarantees a high quality. Thus we prescribe the desired quality using a maximum distance threshold k_d. The desired quality is reached if

$$k_d > \|\mathbf{x}_p\| + \|\mathbf{x}_s\|.$$

We would like to mention, that one could make the above computations directly in homogeneous coordinates as suggested by Skala [Ska06]. It can be shown

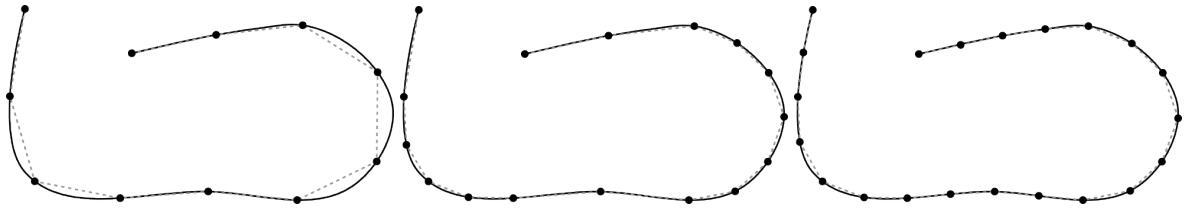


Figure 2: Comparison of refinement based on angle criterion and simple up-sampling. From left to right: Initial sampling of streak line, adaptive refinement and equidistant sampling with half of initial distance. For comparable precision adaptive refinement needs only 16 samples compared to 21 with simple sub-sampling.

that a solution of linear equations is equivalent to a generalized cross product.

Algorithm

In the following we describe the procedure for the adaptive streak line computation. As mentioned earlier, the first step of our algorithm constructs a coarse approximation of the desired streak line with the common method. This can be done quickly because the coarse approximation includes only the tracing of very few particles. Having the coarse line, we start with a first iteration along the line. We consider three consecutively computed particle positions and the intervals between them as a triplet. For each such triplet, starting with the one that has the youngest particles, we evaluate the quality criterion. If the criterion evaluates false we mark both intervals for refinement. After evaluation and marking have been performed for all triples we check all intervals. For each marked interval we trace a new particle starting at the time exactly in the middle of the times of the two particles bounding the interval. This completes the first iteration. We repeat the steps described for the first iteration until no intervals are marked during the evaluation phase or a maximum number of iterations has been reached. This completes the computation for the whole streak line. In the first case we have the streak line with desired accuracy ready. The second case is only for convenience. Particularly, it avoids infinite loops that might occur due user chosen thresholds that are beyond the sensible numerical bounds.

For the angle criterion we have implemented an additional condition that avoids endless refinement at sharp edges of the correct streak line. We omit the details of this condition here, as it will turn out later that the performance of the angle criterion is the worst. So we do not want encourage its reimplementation by the reader.

4 RESULTS

We applied our novel method to a number of data sets to prove its usefulness and robustness.

In this section we analyze two of these examples to demonstrate the efficiency of the new method and determine which of the quality criteria should be chosen. We compute streak lines using five different methods for both examples. At first we compute a reference streak line using the common method with a very high resolution. In all examples the reference streak lines have 5000 samples while the other streak lines compared against it consist of only up to several hundred particles. The reference streak line is considered to be the ground truth. To compare the efficiency of the different methods we compute a streak line with each of them. These streak lines start at the same position and cover the same time interval. For each method we vary the threshold for the used quality criterion and note the number of samples together with the achieved accuracy. The accuracy is measured by comparing the streak lines with the reference streak line and computing their distance. Our distance measure is evaluated as follows: For each point of the reference streak line we compute the shortest distance to the current streak line. Then we compute the mean of these distances. The resulting number is the average distance of the reference streak line to the current streak line and thus the error of the current streak line. We use the samples on the reference streak line to compute the distance and not those of the current streak line in order to have the same sampling density for all distance computations. This is important to make the computed distances comparable.

The four different streak lines we compare against the reference streak line are the three described above, i.e. angle-based, area-based and distance-based adaptive streak lines and a non-adaptive streak line that is equidistantly sampled with a prescribed number of samples. We carry out the comparison for three test cases, i.e. streak lines from one position in a 2D data set and streak lines from two different positions in a 3D data set. The 2D data set is the simulation of the flow around a cylinder (see Figure 3). The three-dimensional data set represents the fluid flow around a cuboid (see Figure 4). Figures 3 and 4 show the shapes and start positions of the considered streak lines.

The results of the analysis of the different streak line methods are shown as graphs in Figure 7. The graphs reveal that the angle criterion performs worst. The adaptive computation using the angle criterion is even worse than the non-adaptive method. The results of the distance-based method are not that clear. In the first two examples, where the streak line has nearly

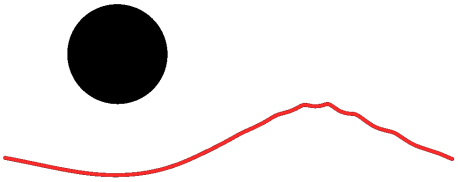


Figure 3: Streak line in flow around cylinder.

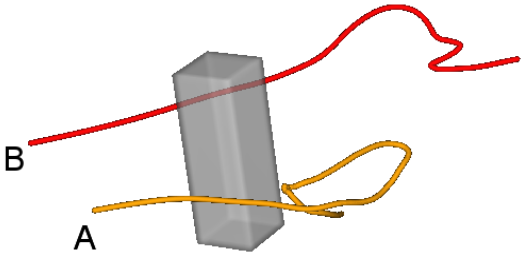


Figure 4: Streak lines in flow around cuboid. Labels A and B mark the seed points for the orange resp. red streak line.

no straight parts that could be exploited by the adaptiveness, the distance-based refinement is not as accurate as the non-adaptive streak lines (upper two images). If the real streak line, however, comprises some straight parts both, the non-adaptive and the distance-based adaptive method, perform very similar (lower image). The best method in all examples is the area-based adaptive method. It is only slightly better where the adaptiveness is not very useful (upper two images) but develops its full efficiency where the adaptiveness can sample straight parts sparsely (lower image). In the latter case the area-based method performs approximately three times better than the distance-based and the non-adaptive method (note the log scale of the graphs). In all our experiments the time used for particle tracing outweighs the time used for evaluating the quality criteria.

Finally, it should be noted that the number of computed samples increases monotonic with monotonic changes of the the quality criteria (increasing angle, decreasing area or decreasing distance).

Read-world Example

The cylinder and cuboid data sets discussed so far are of academic nature. We applied our new algorithm also on a real application data set: an aerodynamic simulation of a delta wing. The resulting streak lines can be seen in Figure 6. As the images show the methods works also on this complex data given on a large unstructured mesh consisting of 11 million cells (tetrahedra and prisms). As the acceleration method itself does not depend on the underlying mesh, it performs well as expected.

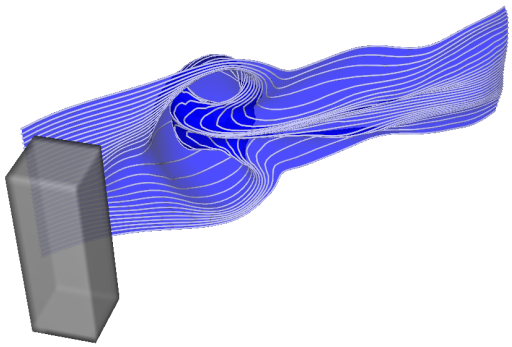


Figure 5: Thirty streak lines in flow around cuboid. Surface connecting the streak lines in blue.

Streak Surface

Our method is suitable to accelerate streak surface algorithms that build up the surface by a number of streak lines. By accelerating each streak line computation as described in this paper the whole computation time for a surface can be reduced. The streak surface shown in Figure 5 is constructed in this way, i.e. using a number of streak lines computed with the adaptive method and connected by a greedy tiling as described by Hultquist for his stream surface algorithm [Hul92].

Note, that the latest published algorithms for streak surface computation which we mentioned in the related work section chose different techniques. They do not construct streak lines explicitly but trace the particles of the streak surface independently.

5 CONCLUSION

In this paper we have presented a novel adaptive method to approximate streak lines. We have applied the method to data sets in 2D and 3D to demonstrate its usefulness. Additionally, we have carried out a thorough performance analysis that showed that only one of the three suggested quality criteria outperforms the conventional equidistant sampling. The analysis showed that the area-based method performs three times better than the common non-adaptive method.

We are aware that the initial coarse approximation might miss some of the smaller features of the correct streak line. The Nyquist-Shannon sampling theorem implies this restriction. However, smaller features are most often found in a later refinement step resulting from the evaluation of one of our quality criteria in its neighborhood.

For the future, we plan to enhance the accuracy between the sample points by replacing the linear interpolation producing polylines by higher order interpolation schemes that produce smooth lines.

6 ACKNOWLEDGMENTS

We wish to thank Markus Rütten from German Aerospace Center (DLR) in Göttingen for providing

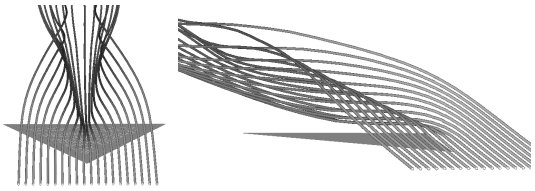


Figure 6: Streak lines in delta wing data set.

the cylinder dataset. The cuboid dataset results from direct numerical simulation carried out with the NaSt3DGP flow solver. NaSt3DGP was developed by the research group in the Division of Scientific Computing and Numerical Simulation at the University of Bonn. It is essentially based on the code described in a book by Griebel *et al.* [GDN98]. A version of the NaSt3DGP code, as well as related information and documentation is available for download at `http://wissrech.iam.uni-bonn.de/research/projects/NaSt3DGP/index.htm`. This work was partially supported by DFG grant SCHE 663/3-8.

REFERENCES

[BFTW09] Kai Bürger, Florian Ferstl, Holger Theisel, and Rüdiger Westermann. Interactive Streak Surface Visualization on the GPU. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2009)*, 15(6):to appear, November-December 2009.

[BLM95] Barry G. Becker, David A. Lane, and Nelson L. Max. Unsteady Flow Volumes. In Gregory M. Nielson and Deborah Silver, editors, *Proceedings of the 6th Conference on Visualization '95*, pages 329 – 335, Washington, DC, USA, 1995. IEEE Computer Society.

[GDN98] M. Griebel, T. Dornseifer, and T. Neunhoeffer. *Numerical Simulation in Fluid Dynamics, a Practical Introduction*. SIAM, Philadelphia, 1998.

[Ham62] Francis R. Hama. Streaklines in a perturbed shear flow. *Physics of Fluids*, 5(6):644–650, June 1962.

[Hul92] Jeffrey P. M. Hultquist. Constructing Stream Surfaces in Steady 3D Vector Fields. In Arie E. Kaufman and Gregory M. Nielson, editors, *Proceedings of the 3rd conference on Visualization '92*, pages 171 – 178, Boston, MA, 1992.

[KGJ09] Hari Krishnan, Christoph Garth, and Kenneth I. Joy. Time and streak surfaces for flow visualization in large time-varying data sets. *Proceedings of IEEE Visualization '09*, October 2009.

[KL96] David N. Kenwright and David A. Lane. Interactive time-dependent particle tracing using tetrahedral decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):120–129, 1996.

[KS86] M. Kurosaka and P. Sundaram. Illustrative examples of streaklines in unsteady vortices: Interpretational difficulties revisited. *Physics of Fluids*, 29(10):3474–3477, 1986.

[Lan93] David A. Lane. Visualization of Time-Dependent Flow Fields. In *Proceedings of the conference on Visualization '93*, pages 32 – 38, 1993.

[LHD+04] Robert S. Laramee, Helwig Hauser, Helmut Doleisch, Benjamin Vrolijk, Frits H. Post, and Daniel Weiskopf. The State of the Art in Flow Visualization: Dense and Texture-Based Techniques. *Computer Graphics Forum*, 23(2):203 – 221, 2004.

[MLP+09] Tony McLoughlin, Robert S. Laramee, Ronald Peikert, Frits H. Post, and Min Chen. Over Two Decades of Integration-Based, Geometric Flow Visualization. pages 73–92, Munich, Germany, 2009. Eurographics Association.

[Ska06] V. Skala. Length, area and volume computation in homogeneous coordinates. *International Journal of Image and Graphics*, 6(4):625–639, 2006.

[SMA00] Andrea Sanna, Bartolomeo Montrucchio, and R. Arina. Visualizing unsteady flows by adaptive streaklines. In *WSCG*, 2000.

[SMAM99] Andrea Sanna, Bartolomeo Montrucchio, R. Arina, and L. Massasso. A 3d fluid-flow visualizer for entry level computers. In *WSCG*, 1999.

[vFWTS08] W. von Funck, T. Weinkauf, H. Theisel, and H.-P. Seidel. Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization 2008)*, 14(6):1396–1403, November - December 2008.

[WH80] David R. Williams and Francis R. Hama. Streaklines in a shear layer perturbed by two waves. *Physics of Fluids*, 23(3):442–447, 1980.

[WS05] Alexander Wiebel and Gerik Scheuermann. Eyelet Particle Tracing - Steady Visualization of Unsteady Flow. In Cláudio T. Silva, Eduard Gröller, and Holly Rushmeier, editors, *IEEE Visualization 2005 - (VIS'05)*, pages 607–614. IEEE Computer Society, October 2005.

[WTS+07] Alexander Wiebel, Xavier Tricoche, Dominic Schneider, Heike Jänicke, and Gerik Scheuermann. Generalized streak lines: Analysis and visualization of boundary induced vortices. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2007)*, 13(6):1735–1742, Nov.-Dec. 2007.

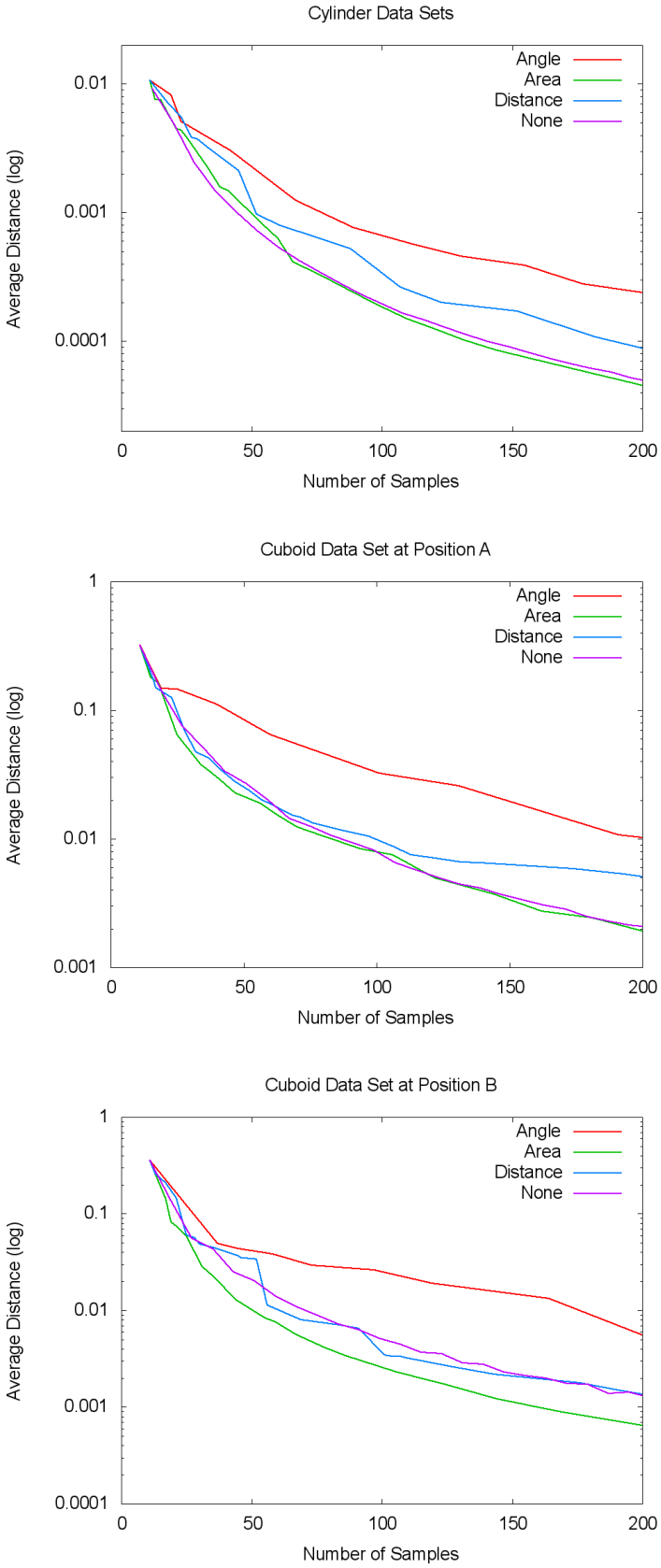


Figure 7: Error of non-adaptive streak line and adaptive streak lines using angle, area and distance criteria. Error is plotted versus number of samples. Streak line used as ground truth consists of 5000 samples.

Interactive Shadow design tool for Cartoon Animation -KAGEZOU-

Shiori Sugimoto
Waseda University
Shiori-s@
akane.waseda.jp

Hidehito Nakajima
Waseda University
nakajima@mlab.
phys.waseda.ac.jp

Yohei Shimotori
Waseda University
y-shimotori@
akagi.waseda.jp

Shigeo Morihsima
Waseda University
Shigeo@waseda.jp

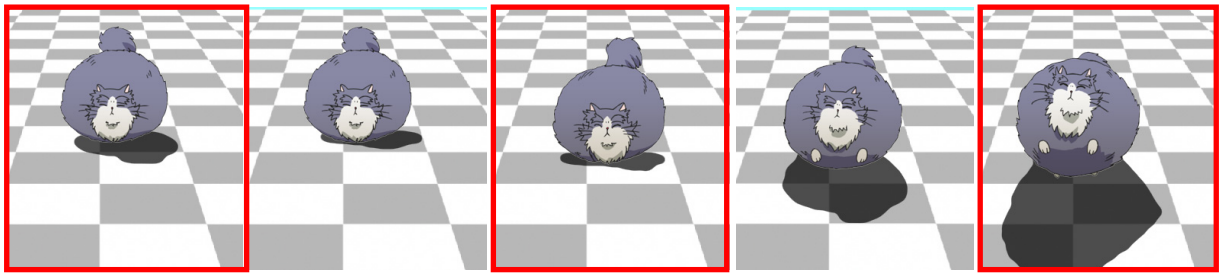


Figure 1: Anime-like shadow animation created with our system
We edit the images in the three key-frames (highlighted in red). In the left image, the characters and their shadows are depicted in their initial states. Then, when the characters crouch down, their shadows shrink at the same time (middle). Finally, when the characters jump up, their shadows stretch in a similar manner (right).

ABSTRACT

In traditional 2D Anime, shadows are drawn by hand, and play a significant role in the creation of symbolic visual effects related to the character's position and shape. However shadows are not always drawn as a result of time constraints and a lack of animators. We develop a shadow generation system that enables animators to easily create shadow animation layers based on character outlines. Our system is both simple and intuitive. The only inputs required are the character animation layers generally used in the Anime industry. Shadows are automatically generated based on these inputs, and then generated shadows are fine-tuned by simple mouse operations. First, shadows are rendered using Shadow Map Method based on the transparency information of the character animation layers. Subsequently, our system applies some filters that enable the generated shadow shape to convert into the effective shadow shape such as the elliptical shape or the wavy shape. Through these processes, our system enables animators to create simple Anime-like shadow animation easily and in a short time.

Keywords

Cartoon animation, Cel animation, interactive techniques, Non-photorealistic rendering, Shadows

1. INTRODUCTION

Shadows play a significant role in specifying a character's position ([THO81]). In other words, audience is able to perceive whether a character is standing on the ground or not from the position of shadows (see Fig.2). In cartoon animation, shadows are rendered not only to represent the relationship between objects and backgrounds but also to draw the attention of audience to particular characters and scenes. Unlike the detailed shadows of photo-realistic animation,

Figure 2: Specifying the position of character
(a) Uncertain position (b) Standing (c) Floating in

shadows in cartoon animation can be simplified because they are not required to be totally realistic. Despite importance of shadows, animators might not draw shadows at all because of a lack of animators and time constraints. Since drawing detailed shadows in each individual frame requires a substantial amount of time, animators frequently do not have enough time to draw detailed shadows according to their intentions ([PET00]).

To solve this problem, we develop a system that can produce simple shadows automatically. Our final goal is to enable animators to easily create a layer of shadow animation in a short time. Our system requires only hand-drawn layers of character animation as inputs. Figure 3 shows how the layers are composited.

First, animators input layers of the character animation. Because these inputs are transparent except for the character, shadows are rendered automatically by Shadow Map Method. In general, Shadow Map Method is used for a 3D object. On the other hand, our system applies Shadow Map Method to transparence information in layers of character animation. Subsequently, several simple shadow shapes are designed by fine-tuning these shadows. The parameters in our system are the position where characters stand on the ground, the position of the light, and parameters of Filters for deforming a shadow's shape. Our system enables animators to produce simple Anime-like shadow animation by fine-tuning these parameters in each key-frame.

2. RELATED WORK

Several user interfaces have been presented for editing shadows in computer graphics. Petrovic et al. [PET00] proposed an innovative method that creates Anime-like shadows semi-automatically. Our system inherits their key concept of shadow generation. Their method requires an amount of time to create shadow animation. On the other hand, our method enables animators to create shadow animation quickly. Pellacini et al. [PEL02] developed a user interface that enables animators to edit shadows directly on the editing screen. Like their interactive interface, we implement our user interface that allows animators intuitive and simple mouse operations. Decoro et al. [DEC07] created a user interface that can be used to design several forms of shadows for rendering non-photorealistic shadows. We are influenced by this concept and develop our system that simplifies shadows as they appear in Anime. Nakajima et al. [NAK07] developed a tool that takes advantage of both 3D and 2D techniques. Their goal was to editing shadows for Anime-like expression to 3DCG model intentionally. Conversely, our method focuses on editing shadows in 2D, and enables animators to create simple shadow animation quickly using 3DCG techniques.

Likewise, there is a lot of previous research on Anime, cartoon animation, and non-photorealistic rendering ([Gooch and Gooch 2001]). Lake et al. [LAK00] developed several real-time methods of rendering for cartoon-like animation, but they did not address shadowing techniques. Regarding highlighting techniques in Anime, Anjyo and Hiramitsu [ANJ03], Anjyo et al. [ANJ06] achieved a tool for editing highlights tweakably. Likewise shade in Anime, Todo et al. [TOD07] developed a system that enables animators to edit shade according to their preference**.** Their goal was to emulate Anime-like edit on 3DCG model.

Our system obtains the outline of the character as input data, and then uses this shape to create a layer of shadow animation. Juan and Bodenheimer [JUA06] developed a method that extracts the character from existing Anime sequences, and then generates inbetween frames using these characters. This method is useful for creating animations using archived Anime. However, when animators need to create an original animation from nothing there is no need for them to extract the character from composite animation as the complete animation consists of separated layers such as the character layer, shadow layer and background layer. Thus, to create shadow animation semi-automatically, our method focuses on the Anime work-flow that creates animations by composing several layers without extracting the character from existing Anime.

3. AUTOMATIC SHADOWING FROM ANIME SEQUENCE LAYERS

In this section, we describe our system for creating a shadow animation from 2D Anime sequences. First, animators set character animation layers sequence as input. Subsequently, our system renders shadows by applying Shadow Map Method to the input. Animators can fine-tune shadows by adjusting parameters with simple mouse operations. As a result, simple shadow animation is produced in a short time. Figure 3 shows the implementation of our system in the Anime work-flow.

3.1. Inputs

With our system, animators use layers of character animation as input. There are several layers that correspond to each component, such as the character layer, shadow layer, and background layer, in the Anime work-flow. Animators compose each layer for final rendering. Usually, everything within these layers is transparent apart from the characters themselves, so that our system is able to apply Shadow Map Method to them.

Our system generates the Silhouette plane using transparency information of these input images. This silhouette is usually similar to the character shape.

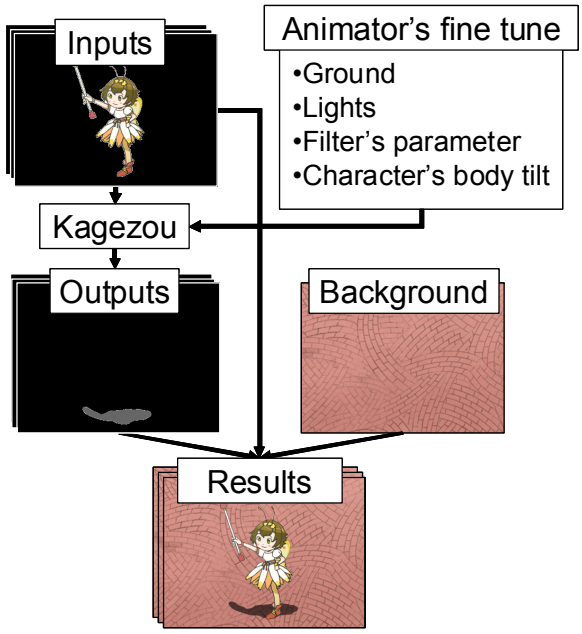


Figure 3: Implementing our system in the Anime work-flow.

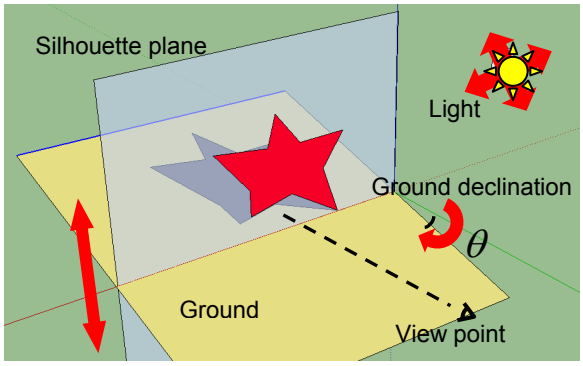


Figure 4: Location of Silhouette plane, View point, and Ground.

Silhouette plane is set vertically from the view point. The ground position is based on a declination θ between the silhouette plane and ground.

3.2. Location of Silhouette planes, View point and Grounds

Our system renders shadows onto the ground using silhouette plane with transparence information. Figure 4 shows the locational relationship between the silhouette plane, view point and ground respectively. First, in our system the silhouette plane is set vertically to the direction from the view point of the camera (see Fig.4). We then define the position of the ground based on the declination θ between the silhouette plane and the ground. As a result, our system enables animators to render a character's shadow from the input

3.3. Deforming Character's Silhouette

If the character stands slantwise, generated shadows do not come in contact with the character's feet (see Fig.5 left). For solving this problem, our system generates the deformed silhouette from input image.

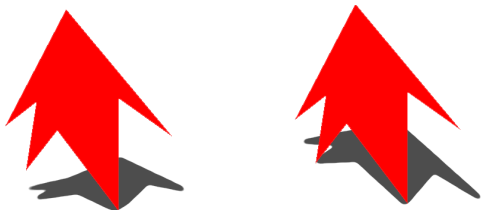


Figure 5: Deforming character's silhouette. Left : The shadow which is generated using character's silhouette do not come in contact with a character's right foot. **Right :** The shadow which is generated using deformed silhouette seem to come in contact with the feet.

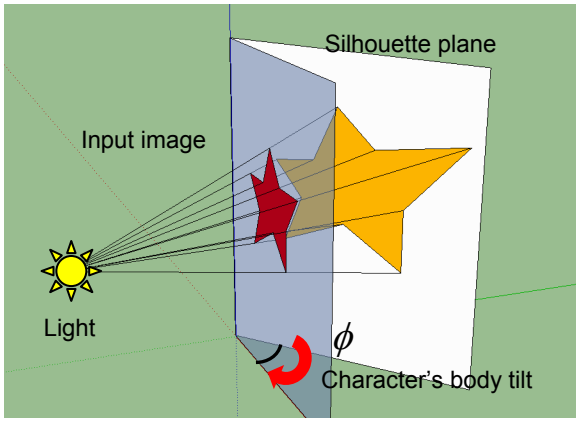


Figure 6: Location of Input, Silhouette plane. Input image is set vertically to the direction from the light. The position of the Silhouette plane is based on character's body tilt on the declination ϕ between the input image and the silhouette plane.

First, in our system the input image is set vertically to the direction from the light (see Fig.6). We subsequently define the character's body tilt on the declination ϕ between the input image and the silhouette plane. Finally, we project the character's silhouette onto the Silhouette plane using Shadow Map Method. The projected Silhouette is deformed, looks like rotated character's silhouette.

As a result, our system generates the shadows coming in contact with the character's feet (see Fig.5 right). In our system, animators can create these fine shadows easily using the parameter of Character's body tilt.

3.4. Applying Shadow Map Method to the input layer

Although there several possible shadowing techniques exist that could be incorporated into our system, we use Shadow Map Method. Shadow Map Method is commonly used for rendering shadows ([WIL78]). Several research projects have been dedicated to developing a means of producing high-quality shadows in a short amount of time (Lokovic and Veach [LOK00], Stamminger and Drettakis [STA02]). Since our algorithm is not essentially related to these several improvements of Shadow Map Method, we do not discuss this issue in our paper.

There are two advantages of using Shadow Map Method in our system. One is its high-speed performance that enables the interactive fine-tuning of shadows, though ray-tracing would not be available for interactive editing shadows. The other advantage is that Shadow Map Method is flexible use with various backgrounds. In general, in our system, shadows are drawn on flat ground. However, we would like our system to be able to handle complicated backgrounds in addition to flat ground. Though simple affine transformation, calculated by the relationship between the input, light, and the ground, is possible for flat ground, simple affine transformation is not practical for a complicated background. Therefore, we apply Shadow Map Method to our system.

In general, Shadow Map Method is applied to 3D objects. On the other hand, our system renders shadows using a hand-drawn character layer as input. First, our system stores the distance from the light to a character layer and the plane of shadow projection in the depth buffer looking from the light position. A texture visualized by this stored information is called a Shadow Map. Then, looking from an initial view point, our system finally renders shadows by referring to the Shadow Map.

3.5. Fine-Tuning Operation

When our system renders shadows, there are several factors which are used in fine-tuning: position of light, ground position, ground declination, and Character's body tilt. Red arrows in Figure 4, 6 show several fine-tuning parameters.

Light Position: In our system, animators are required to set the light first. The shape and the scale of shadows depend on this factor.

Ground Position: In order for our system to render a shadow, animators are required to set a ground position. Since audience recognize the position of objects in Anime by their shadows on the ground, it is essential to set the ground position (see Fig.2). Initially, the ground position is the lowest position of intransparent region domain. Our system then allows animators to easily fine-tune the ground position using simple mouse operations.

Ground Declination: Animators can set the ground declination in the same way: thus allowing animators to create various types of shadowing.

Character's body tilt: In the case that the character stands slantwise, animators have to set the character's grounding points for setting character's body tilt.

3.6. The Filters

Our system applies some filters that enable the generated shadow shape to convert into the effective shadow shape. These filters enable animators to create the effective shadow animation quickly. In this section, we describe two effective filters.

3.6.1. Simplification Filter

Since our research objective is shadowing in 2D Anime, it is important that the shape of shadows are simplified. The simplest shape of a shadow is considered to be an ellipse, so our system needs to gradually convert complex shadow shapes into ellipses. Gaussian filter is applied to the shadow for simplification by controlling Simple parameter denoting *s* and the Distance parameter denoting *d* interactively. Simplification filter S is represented by following equations.

$$S(u,v) = \frac{1}{2\pi\sigma} e^{-p(v)\frac{\left(u^2+v^2\right)}{2\sigma}} \qquad (1)$$

$$P(v) = \begin{cases} \dfrac{s|v-v_0|}{d} & : |v-v_0| < d \\ s & : |v-v_0| \geq d \end{cases} \qquad (2)$$

σ is the standard deviation of the Gaussian distribution, and is the lowest position of characters. (See Figure 7 as an example).

Simple parameter: *s* in our system, the kernel of Gaussian filter is controlled for making the shape of shadows ellipse gradually. As *s* becomes larger, the shape of shadows becomes simpler. Animators can adjust the parameter *s* for simplifying shadows overall.

Distance parameter: *d* Animators can adjust Distance parameter *d* to convert the shape of shadows locally. As for shadows in photorealistic situation, the shadow of a character's legs tends to be drawn accurately compared to that of the upper body. In traditional 2D Anime, the shadow of a character's legs tends to be drawn in detail as well as in photorealistic; however, the shadow of the upper body is simpler than photorealistic. Our system enables animators to control Simplification filter according to Distance parameters locally. When d equals 0, shadows are simplified overall, which means all kernels are equal in each area of shadows. On the other hand, when *d* does not equal 0, is determined by the distance from a character to the shadow: the farther shadows from the lowest position of the character are, the larger is. The shapes of shadows become simpler in proportion to the distance of the shadows (see Fig.7 right). By applying these parameters, our system achieves simplification of shadows (see Fig.7).

3.6.2. Bump Mapping Filter

For creating the shadow on the swayed ground, like water, requiring substantial time to draw, our system provides Bump Mapping Filter. This filter enables the shadow shape to convert simple shape into the waving or striped shape. Bump Mapping technique is applied to the shadow for distortion.

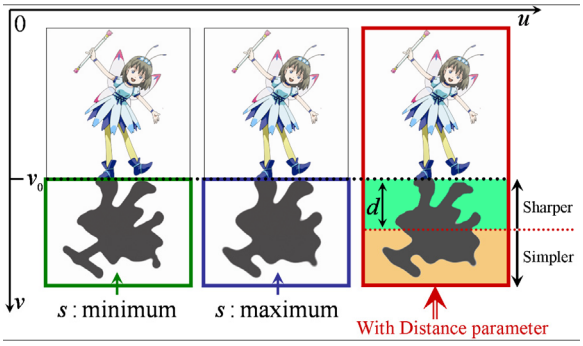


Figure 7: Applying the Simplification filter. **Left:** the initial shadow. **Middle:** rounded shadow. **Right:** applying Distance parameter *d*.

We describe the case of generating the shadow on the water (see Fig.8 right). First, the texture **D**, the basis for Bump Mapping, is created by the equation 3. **D** stores the wave direction of each pixel.

$$\begin{cases} D_u(i,j) = \cos\left[\pi(ai + bj + vel \cdot t)/\lambda\right] \\ D_v(i,j) = \sin\left[\pi(ai + bj + vel \cdot t)/\lambda\right] \end{cases} \quad (3)$$

λ is the wavelength, *a* and *b* are the parameter related to the wave direction, *vel* is the velocity of the wave, and *t* is time sequence. Subsequently, our system creates Bump Map **D'** by applying the following formula to **D**. **A** is the amplitude parameter.

$$\begin{pmatrix} D'_u \\ D'_v \end{pmatrix} = \begin{pmatrix} D_u \\ D_v \end{pmatrix} \begin{pmatrix} A_{00} & A_{10} \\ A_{01} & A_{11} \end{pmatrix} \quad (4)$$

Compositors can easily create a variety of waves by tuning above mentioned parameters. Finally, our system creates random waves by mapping two sets of Bump Map **D'** with shadow layer.

3.7. Lighting from overhead of the character

Our system enables animators to create shadows by applying Shadow Map Method to 2D characters. However, lighting from overhead of the character plane can not be rendered at all by same algorithm. To solve this problem, when a light is at overhead of characters plane, our system axisymmetrically sets four virtual lights from the character's overhead. We define this domain where the light would be set as a virtual light area, and animators can fine-tune this domain. Figure 9 shows an overview of this extension algorithm in the view from the vertical direction of a character plane.

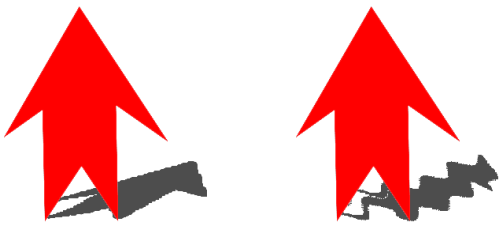


Figure 8: Applying the Bump Mapping filter. **Left :** Applying no Filter. **Right :** Applying the Bump Mapping Filter. Generated shadow looks like the shadow on the water.

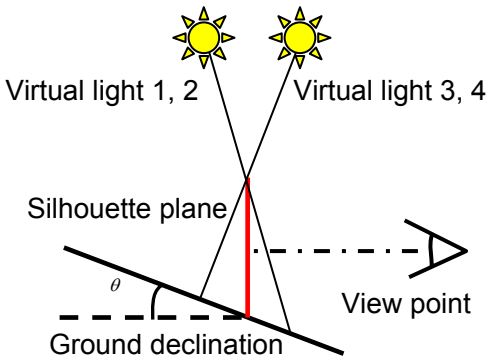


Figure 9: Overview of lighting from the overhead

4. RESULTS AND DISCUSSION

In this section, we demonstrate examples of fine-tuning shadows and animation results created by our system. Our prototype system runs on an Intel T7800 2.60-GHz platform with NVIDIA GeForce Go, and Direct X as the graphics API. The input size we used for the purposes of this paper is approximately 1000 * 1000 pixels as is commonly used in the cartoon animation industry as a previewing size. The frame rate range for fine-tuning shadows in our system is more than 30 fps for all operations.

Figure 11 shows examples of fine-tuning shadows using our system. The image on the far left shows a character used as input. The next one is the default shape of shadows as output. We then fine-tune the light position, the ground position, declination, and the Simplification filter's parameters *s*, and *d*. Since animators can fine-tune these parameters in each frame, they can create their desired shadow key-frame animation (see Fig.1).

Figure 12 shows practical animation created using our system. The top row images represent the character animation used as input. We set the input into our system, and fine-tune only in the first frame to create simple shadows. We then compose each layer; inputs, outputs, and backgrounds (see 2nd row, figure 12). As a result of the simplification of the shadows,

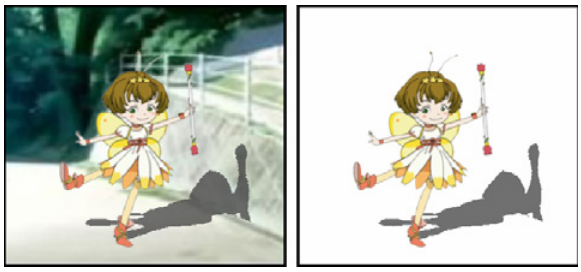


Figure 10: Extensions for 3D backgrounds.

audience is encouraged to focus more on the characters themselves. Next, we create other simple shadows (3rd row in figure 12). Finally, we create shadows on the water (bottom row in figure 12). This animation is fine-tuned only in the first frame.

Our system enables animators not only to create simple shadows edited only in the first frame, but also to create Anime-like shadows used in several key-frames. Figure 1 shows Anime-like shadow animation created with our system. We edit the shadows in these key-frames for creating Anime-like shadows. As demonstrated in Figure 1, 2, 11 and 12, our system enables animators to create different types of simple shadows, which can be rendered more easily, and in a shorter time than was previously possible using hand-drawn methods.

5. CONCLUSION AND FUTURE WORK

We have developed an innovative system that enables animators to create simple shadow animation easily and quickly. Since our procedure is simple and intuitive, animators can use this system on general-purpose computer. The operation examples in Figure 11 demonstrate that animators can fine-tune shadows interactively using Shadow Map Method. In addition, Shadow Map Method also enables animators to imitate complicated backgrounds by adding 3D shapes primitives on the ground as can be seen in Figure 10. The images represent a shadow on a wall. In Figure 1 and 12, our animation results demonstrate the practicality of our system in the creation of simple shadow animation in Anime.

As mentioned in Section 2, several research projects re-use the archived Anime by extracting characters from animation sequences. The research is certainly useful for creating customized animations from archives, however, when animators need to create original animation from nothing, these approaches might not be practical. For creating original animation, animators firstly draw or render images at each layer. They then create the final animation by composing these layers. That is, extracting characters from Anime sequences is not suitable for the actual Anime work-flow. By focusing on Anime work-flow, our system innovatively improves the efficiency of shadow rendering.

Prior to our research, Petrovic et al. [2000] developed a system that enables animators to create shadow animation from Anime sequences. Their method, although more effective than previous hand-drawn practices, is still labor intensive and time-consuming because animators must create character's 3D model frame by frame. On the other hand, by using input

images directly, our system enables animators to create Anime-like shadow animation in a shorter time than is possible using Petrovic's method, by simply setting a few factors in the key frames. For Figure 12, it took less than 10 minutes to create shadow animation for a 336-frame cycle with 2 layers. With our system, by fine-tuning the shadows only in the first frame, animators can create simple shadow animation without Anime-like direction such as key-framing as can be seen in Figure 11. Furthermore, animators can create Anime-like key-frame shadow animation, such as that shown Figure 1. Even in such case, our system enables animators to render shadows in less time than Petrovic's method.

However, several issues need to be addressed in our future work. Currently, animators can use the easy interface that enables adding several types of simple primitives to handle several types of background situations. In actual Anime, 2D character animation could be combined into 3D background. To leverage this 3D background data, we need to customize our system to process the background.

In addition, our prototype system described in this paper is a stand-alone system, which has not been integrated into an authorized pipeline such as "Adobe After Effect" (TM). In the future, with a plug-in version of our system, animators would be able to create animation more effectively.

6. ACKNOWLEDGMENTS

This research is supported by Japan Science and Technology Agency CREST project.

The character and background in Figure 1, 2, 3, 7, 10, 11, and 12 are courtesy of HUIS TEN BOSCH / yumenotomo / Shinichi Ichikawa.

REFERENCES

[ANJ03] ANJYO, K., AND HIRAMITSU, K. 2003. Stylized highlights for cartoon rendering and animation. IEEE Computer Graphics and Applications 23, 4, 54–61.

[ANJ06] ANJYO, K., WEMLER, S., AND BAXTER, W. 2006. Tweakable Light and Shade for Cartoon Animation. Proceedings of the 4th International Symposium on Non-photorealistic animation and rendering, 133-139.

[DEC07] DECORO, C., COLE, F., FINKELSTEIN, A., AND RUSINKIEWICZ, S. 2007. Stylized Shadows. Proceedings of the 5th international symposium on Non- photorealistic animation and rendering, 77-83. [4] GOOCH, B., AND GOOCH, A. 2001. Non-Photorealistic Rendering. A. K. Peters.

[JUA06] JUAN, C., AND BODENHEIMER, B. 2006. Re-using Traditional Animation: Methods for Semi-Automatic Segmentation and Inbetweening. Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation, 223-232.

[LAK00] LAKE, A., MARSHALL, C., HARRIS, M., AND BLACKSTEIN, M. 2000. Stylized rendering techniques for scalable real-time 3D animation. Proceedings of Non-Photorealistic Animation and Rendering 2000, 13–20.

[LOK00] LOKOVIC, T., AND VEACH, E. 2000. Deep shadow maps. Proceedings of the 27th annual conference on Computer graphics and interactive techniques, 385-392.

[NAK07] NAKAJIMA, H., SUGISAKI, E., AND MORISHIMA, S. 2007. Tweakable Shadows for Cartoon Animation. Proceedings of the 15th international Conference in Central Europe on Computer Graphics, 233-240.

[PEL02] PELLACINI, F., TOLE, P., AND GREENBERG, D. 2002. A user interface for interactive cinematic shadow design. ACM Transactions on Graphics, Proceedings of ACM SIGGRAPH2002 21, 3, 563–566.

[PET00] PETROVIC, L., FUJITO, B., WILLIAMS, L., AND FINKELSTEIN, A. 2000. Shadows for cel animation. In Proceedings of SIGGRAPH2000, 511–516.

[STA02] STAMMINGER, M., AND DRETTAKIS, G. 2002. Perspective shadow maps. ACM Transaction on Graphics, 21, 3, 557-562.

[THO81] THOMAS,F., AND JOHNSTON,O. Disney Animation: The Illusion of Life. Walt Disney Productions, New York, 1981.

[TOD07] TODO, H., ANJYO, K., BAXTER, W., AND IGARASHI, T. Locally Controllable Stylized Shading. 2007. ACM Transactions on Graphics Volume 26, 3, Article No. 17

[WIL78] WILLIAMS, L. 1978. Casting curved shadows on curved surfaces. ACM SIGGRAPH Computer Graphics, 12, 3, 270-274.

Figure 11: Fine-tuning shadows.
The image on the far left shows an input character. Next, the initial shadow is rendered by our system using default fine-tuning parameters. Starting from the initial shadow, we fine-tune in this order the light position, ground position, ground declination, Simplification parameter, and Distance parameter.

Figure 12: Results produced by our system.
The images on the top row show input images. After inputting these images into our system, we fine-tune several parameters only in the first frame to create different types of simple shadows: simple shadows related to character's shape (2nd row images), simple shadows (3rd row images), and shadows on the water (bottom row images).

Parametric Curves Variations with Respect to a Given Direction

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

Modeling dynamic characters, in observance of calligraphic rules can be done with the help of parametric curves. The class of Bézier curves is a powerful tool for modeling the outlines of the surface razed by a nib's calligrapher. Unfortunately, after capturing the character's hand-drawn outlines, the surface delimited by the outlines doesn't correspond to the space to shade when writing the character. Some of the curves pieces of the outlines are to be subdivided at some appropriate points. In order to identify these points, decompositions of the Bézier curves involved, with respect to particular directions, are necessary. This paper presents a simple and effective mathematical method for doing such decompositions.The proposed method helps to determine the points as extrema of certain functions. Then, the method is implemented and used to design some Arabic dynamical characters. This will help in building dynamic fonts respecting calligraphic strong rules.

Keywords

Bézier curves, decompositions with respect to a given direction, Arabic dynamical characters.

1. OVERVIEW

One of the big challenges in typesetting Arabic alphabet based texts consists on designing Arabic dynamical fonts respecting strong rules of a very well established calligraphy. The characters are context dependent. Some of them are to be stretched and inter-connected with small curves called *Kashida* [MEss 87].

Traditional software assisting to develop fonts, such as FontCreator [HiLo 05] and others don't provide enough support for developing Arabic fonts which can respect the calligraphic rules. In particular, stretching characters, with small curves, according to the context is not available. The general features and characteristics of a software to support these special needs are out of the scope of this paper. One of the preliminary problems to solve before designing applications for generating fonts can be stated as follows: after capturing hand written characters, or more precisely the nib's motion outlines, blacken the surface delimited by the Bézier curves [Bézi 77] representing the movement of the nib's vertices will not produce the original character. In order to reproduce the original characters, these Bézier curves have to be decomposed in some particular points. Then the nib's motion have to be considered according to the resulting curves separately. How to identify these points? This paper proposes a mathematical method to do so. finding out these points leads to the study of the *variations of parametric curves with respect to a given direction*. The method presented in this paper concerns parametric curves in general. Cubic Bézier curves are particular cases. Such curves are used in font development languages such as PostScript [Adob 99] and Metafont [Knut 86] .

After showing the need for a method for studying variations of parametric curves according to a given vector, the proposed mathematical method will be presented in detail. Then the way to de-

compose some Bézier curves will be presented and illustrated by an example.

2. PROBLEM

One of the ways to build a program that can help in developing Arabic fonts can be described as follows: After displaying a scanned image of the character, the font developer will seek the draw with it's mouse pointer as if it were a nib's head. Therefore, the program will generate the PostScript encoding corresponding to the character in the font. The program will have to take into account the curvilinear stretching, or *Kashida* [MEss 87], of the characters .

In the Arabic writing, the nib's head can be represented as a shaded rectangle whose width is one/a sixth of the length. The character is materialized by the surface razed by this rectangle. According to the writing style, Naskh, Roqaa, Diwany etc..., the angle between the rectangle's length and the baseline can vary a little or remain unchanged while writing. For instance, in the Naskh style this angle remains at about 70° with the baseline [MEss 87, MEss 94] (See Figure 1).

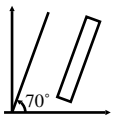


Figure 1. Nib's head in Naskh style, in 12mm size.

In the Arabic writing, many characters are dynamic, the shape and size of the character are context dependent. The stretching or change of size of the character in a given context is not a linear scaling nor a simple enlargement. Therefore, no suitable vectorial modeling can be found of a character through focusing on the outlines of the surface razed by the nib's head. The optimal solution consists on the curves modeling the nib's head *motion* instead of those representing the razed surface outlines. Then, the draw modeling the character is to be shaded taking into account the nib's head movement.

The rectangle modeling the nib's head can be considered as a rigid body (the length, width and angles between sides are invariant). The movement of one of the rectangle vertices determine entirely the movement of the other vertices through simple translations. The rectangle represented in Figure 2 has a length $M_{10}M_{20}$ and a width $M_{10}M_{40}$. It makes an angle α with the baseline.

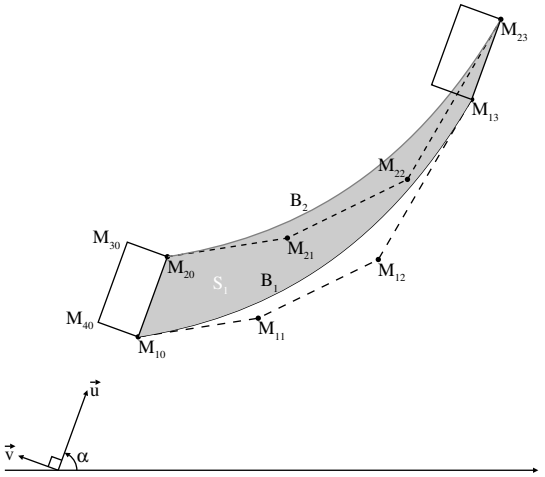


Figure 2. Surface S1 razed by the side $[M_{10}M_{20}]$

The movement of the vertex M_{10} is represented by the Bézier curve with four control points M_{10}, M_{11}, M_{12} and M_{13}. From now, we make the assumption that the angle α remains invariant while the rectangle is in motion (all movements are reduced to simple translations, no rotations are performed). Taking into account rotations will need supplementary efforts on approximation. Now, it is out of the scope of this paper. The three vertices will seek a similar movement through translations of vectors $\overrightarrow{u}$, $\overrightarrow{u}+\overrightarrow{v}$ and $\overrightarrow{v}$ respectively ($\overrightarrow{M_{10}M_{20}}$, $\overrightarrow{M_{10}M_{30}}$ and $\overrightarrow{M_{10}M_{40}}$). How to shade the surface razed by the rectangle? A way to do so consists on shading the surface razed by each of the rectangle's sides. So for the segment $[M_{10}, M_{20}]$, we will have to shade the surface delimited by the Bézier curve B_1 with control points M_{10}, M_{11}, M_{12} and M_{13}, the segment $[M_{13}, M_{23}]$, the Bézier curve B_2 with control points M_{23} , M_{22}, M_{21} and M_{20}, and finally the segment $[M_{20}, M_{10}]$. The points M_{20}, M_{21}, M_{22} and M_{23} are derived from M_{10}, M_{11}, M_{12} and M_{13} trough the translation of vector $\overrightarrow{u}$. Consider the movement of the segment $[M_{10}, M_{20}]$, then we define the Bézier outlines associated to this movement to be the *multi-curve* $(B_1, [M_{13}, M_{23}], B_2, [M_{20}, M_{10}])$. This way is meaningful. Actually, the PostScript language supports operators for shading outlines as with graphic development languages such as Java [Java 05].

This method of shading surfaces doesn't provide always the desired results. Actually, consider a very smart nib (a nib with a negligible thickness (width)) or simply a segment in motion (in general case, the surface razed by the nib's head is ob-

tained through considering the surfaces razed by all the edges). As in Figure 3 where the segment $[M_{10}, M_{20}]$ is in movement, the extremity M_{10} follows the cubic Bézier's curve B. The surface razed by the segment is presented in Figure 4 and the surface delimited with the outlines associated to the movement doesn't fit exactly with the surface razed by the segment $[M_{10}, M_{20}]$ (See Figure 5).

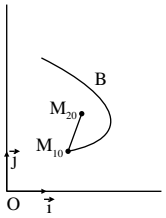


Figure 3. M_{10} **path**

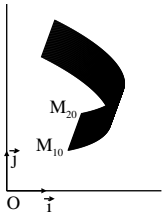


Figure 4. The Surface razed by the segment $[M_{10}, M_{20}]$

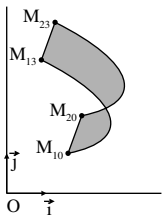


Figure 5. Surface delimited by the outlines associated to the movement

Now, let us decompose the curve B with the generalized algorithm of refinement [Bars 85, Jeff 81, Hosa 80, Gold 82] (case of none median decomposition) with respect to the coefficient $T = 0.3613981051$. We get two Bézier curves B_1 and B_2 that can represent the vertex's M_{10} movement (See Figure 6). Let S_1 and S_2 be the surfaces delimited by the outlines associated to the curves B_1 and B_2 (See Figure 7 and 8). Superposing S_1 and S_2 will give a shaded surface that fits exactly with the surface razed by the segment (See Figure 9).

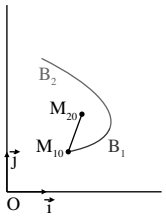


Figure 6. M_{10} **path Decomposition**

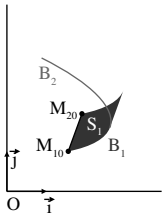


Figure 7. Area corresponding to B_1

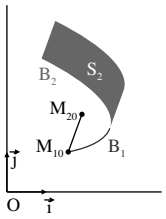


Figure 8. Area corresponding to B_2

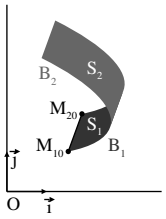


Figure 9. Superposing areas S_1 **and** S_2

The coefficient of decomposition T is an extrema of the Bézier parametric curve. Of course, there is no order relation in an affine plane. Next, we'll define comparison operators that will make the study of parametric curves as easy as the study of real functions with a single variable.

3. ORDER IN AN AFFINE PLANE IN $\mathbb{R}^3$

Consider $\mathcal{P}$ an affine plane in $\mathbb{R}^3$ containing the origin O. Let $\overrightarrow{u}$ be a given vector in $\overrightarrow{\mathcal{P}}$, the director vectorial plane associated to $\mathcal{P}$. We will

define in $\mathcal{P}$ an *equality operator* $\underset{\overrightarrow{u}}{=}$ and an *order operator* $\underset{\overrightarrow{u}}{\leq}$ with respect to the vector $\overrightarrow{u}$.

In the sequel, we consider:

- The affine space $\mathbb{R}^3$ with the rectangular Cartesian system of reference $R\left(O, \overrightarrow{\imath}, \overrightarrow{\jmath}, \overrightarrow{k}\right)$.
- An affine plane $\mathcal{P}$ in $\mathbb{R}^3$ containing the origin O. Then, $\overrightarrow{\mathcal{P}}$ stands for the vectorial space of translations in $\mathcal{P}$. A direct orthonormal basis of $\overrightarrow{\mathcal{P}}$ will be denoted by $p = \left(\overrightarrow{P_1}, \overrightarrow{P_2}\right)$.
- Let $\overrightarrow{u} \in \overrightarrow{\mathcal{P}}$.

Definition 1 (Equality with respect to a vector in an affine plane)
We define an equality relation $\underset{\overrightarrow{u}}{=}$ *, among points in* $\mathcal{P}$*, with respect to the vector* $\overrightarrow{u}$ *by:*
$\forall\, M_1, M_2 \in \mathcal{P}$,
$M_1 \underset{\overrightarrow{u}}{=} M_2 \Leftrightarrow \left(\overrightarrow{u} \wedge \overrightarrow{OM_1}\right) = \left(\overrightarrow{u} \wedge \overrightarrow{OM_2}\right)$

Definition 2 (Direct orthogonal range with respect to a vector and a basis)
Let M *in* $\mathcal{P}$*, the value* $\left(\overrightarrow{u} \wedge \overrightarrow{OM}\right) \cdot \left(\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right)$*, denoted by* $\mathcal{R}_{(\overrightarrow{u},p)}(M)$*, is called the direct orthogonal range of* M *with respect to a given vector* $\overrightarrow{u}$ *and a normed direct basis* p*.*

Remark 1
The direct orthogonal range is the component of the surface vector with respect to $\overrightarrow{P_1} \wedge \overrightarrow{P_2}$.

Property 1 *Let* $M_1, M_2 \in \mathcal{P}$*, then:*
$M_1 \underset{\overrightarrow{u}}{=} M_2 \Leftrightarrow \mathcal{R}_{(\overrightarrow{u},p)}(M_1) = \mathcal{R}_{(\overrightarrow{u},p)}(M_2)$.
This property allows to establish a relation of equivalence in the set of points $\mathcal{P}$*. Classes of equivalence are identified as parallel lines in* $\mathcal{P}$ *with the same direction* $\overrightarrow{u}$*.*

Proof (Property 1)
The implication from left to right is obvious.
Conversely,
suppose that $\mathcal{R}_{(\overrightarrow{u},p)}(M_1) = \mathcal{R}_{(\overrightarrow{u},p)}(M_2)$
it means $\left(\overrightarrow{u} \wedge \overrightarrow{OM_1}\right) \cdot \left(\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right) = \left(\overrightarrow{u} \wedge \overrightarrow{OM_2}\right) \cdot \left(\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right)$
We have $\overrightarrow{u} \in \overrightarrow{P}$, $\overrightarrow{OM_1} \in \overrightarrow{P}$ and $\overrightarrow{OM_2} \in \overrightarrow{P}$.
Therefore $\left(\overrightarrow{u} \wedge \overrightarrow{OM_1}\right) = \lambda \left(\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right)$ and $\left(\overrightarrow{u} \wedge \overrightarrow{OM_2}\right) = \beta \left(\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right)$.
It follows that
$\left(\overrightarrow{u} \wedge \overrightarrow{OM_1}\right) \cdot \left(\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right) = \lambda \cdot \left\|\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right\|^2$ and
$\left(\overrightarrow{u} \wedge \overrightarrow{OM_2}\right) \cdot \left(\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right) = \beta \cdot \left\|\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right\|^2$
But $\left\|\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right\| \neq 0$.
Consequently $\lambda = \beta$.
We then get $\left(\overrightarrow{u} \wedge \overrightarrow{OM_1}\right) = \left(\overrightarrow{u} \wedge \overrightarrow{OM_2}\right)$
and finally $M_1 \underset{\overrightarrow{u}}{=} M_2$. ■

Definition 3 (Order with respect to a vector in an affine plane)
The relation $\underset{\overrightarrow{u}}{\leq}$ *defined in the set* $\mathcal{P}$*, with respect to the direction* $\overrightarrow{u}$*, by:*
$\forall\, M_1, M_2 \in \mathcal{P}$*, we have:*
$M_1 \underset{\overrightarrow{u}}{\leq} M_2 \Leftrightarrow \mathcal{R}_{(\overrightarrow{u},p)}(M_1) \leq \mathcal{R}_{(\overrightarrow{u},p)}(M_2)$
is a total order.

Proof (Definition 3)

- Reflexivity: obvious
- Antisymmetry:
  Let $M_1, M_2 \in \mathcal{P}$ such that $M_1 \underset{\overrightarrow{u}}{\leq} M_2$ and $M_2 \underset{\overrightarrow{u}}{\leq} M_1$.
  then $\mathcal{R}_{(\overrightarrow{u},p)}(M_1) \leq \mathcal{R}_{(\overrightarrow{u},p)}(M_2)$ and $\mathcal{R}_{(\overrightarrow{u},p)}(M_2) \leq \mathcal{R}_{(\overrightarrow{u},p)}(M_1)$.
  So $\mathcal{R}_{(\overrightarrow{u},p)}(M_1) = \mathcal{R}_{(\overrightarrow{u},p)}(M_2)$
  according to the property 1, it follows
  $M_1 \underset{\overrightarrow{u}}{=} M_2$
- Transitivity:
  Let $M_1, M_2, M_3 \in \mathcal{P}$ such that $M_1 \underset{\overrightarrow{u}}{\leq} M_2$ and $M_2 \underset{\overrightarrow{u}}{\leq} M_3$.
  Then $\mathcal{R}_{(\overrightarrow{u},p)}(M_1) \leq \mathcal{R}_{(\overrightarrow{u},p)}(M_2)$ and $\mathcal{R}_{(\overrightarrow{u},p)}(M_2) \leq \mathcal{R}_{(\overrightarrow{u},p)}(M_3)$.
  It follows $\mathcal{R}_{(\overrightarrow{u},p)}(M_1) \leq \mathcal{R}_{(\overrightarrow{u},p)}(M_3)$.
  So $M_1 \underset{\overrightarrow{u}}{\leq} M_3$
- Total order:
  Let $M_1, M_2 \in \mathcal{P}$, then we have
  $\mathcal{R}_{(\overrightarrow{u},p)}(M_1) \leq \mathcal{R}_{(\overrightarrow{u},p)}(M_2)$ or $\mathcal{R}_{(\overrightarrow{u},p)}(M_2) \leq \mathcal{R}_{(\overrightarrow{u},p)}(M_1)$
  and therefore, $M_1 \underset{\overrightarrow{u}}{\leq} M_2$ or $M_2 \underset{\overrightarrow{u}}{\leq} M_1$

Thus $\underset{\overrightarrow{u}}{\leq}$ is a total order relation. ■

4. FUNCTIONS FROM $\mathbb{R}$ INTO AN AFFINE PLANE IN $\mathbb{R}^3$

The equality and order relations defined before will allow studying parametric functions defined

from an interval in $\mathbb{R}$ into an affine plane in the space $\mathbb{R}^3$ reported to a rectangular Cartesian system of reference $R\left(O,\overrightarrow{\imath},\overrightarrow{\jmath},\overrightarrow{k}\right)$ in terms of extrema and variations.

In the following, let $\mathcal{P}$ be an affine plane contained in $\mathbb{R}^3$, containing the origin O, with director vectorial plane associated $\overrightarrow{\mathcal{P}}$ and a normed direct basis $\left(\overrightarrow{P_1},\overrightarrow{P_2}\right)$.

Consider a given vector $\overrightarrow{u}\in\overrightarrow{\mathcal{P}}$ and a closed interval $[a,b]\subset\mathbb{R}$.
Let f be a parametric function (parametric curve) such that:

$$\begin{array}{rccl} f & : & [a,b] & \longrightarrow & \mathcal{P} \\ & & t & \longmapsto & (f_1(t),f_2(t)) \end{array}$$

where f_1 and f_2 are two real functions defined on $[a,b]$.

In the following, we will establish some lemmas and generalize theorems about real functions in order to take into account parametric functions with respect to a given vector. Before that, let us give an obvious property that will next be used.

Property 2 *Let f be a continuous function defined on the interval $[a,b]$. Suppose that f is differentiable on $]a,b[$. Then $\mathcal{R}_{(\overrightarrow{u},p)}\circ f$ is continuous on $[a,b]$, differentiable on $]a,b[$ and $\left(\mathcal{R}_{(\overrightarrow{u},p)}\circ f\right)' = \mathcal{R}_{(\overrightarrow{u},p)}\circ f'$.*

Theorem 1 (Rolle's theorem) *If f is a continuous function on $[a,b]$ such that:*

- *$f(a) \underset{\overrightarrow{u}}{=} f(b)$ and*
- *f is differentiable on $]a,b[$*

then $\exists c\in]a,b[$ such that $f'(c) \underset{\overrightarrow{u}}{=} O$ (O is the point with null coordinates in $\mathcal{P}$).

Proof (Theorem 1)
The function $\mathcal{R}_{(\overrightarrow{u},p)}\circ f$ is continuous on $[a,b]$, and differentiable on $]a,b[$ (from the property 2) and $\left(\mathcal{R}_{(\overrightarrow{u},p)}\circ f\right)(a) = \left(\mathcal{R}_{(\overrightarrow{u},p)}\circ f\right)(b)$.
from Rolle's theorem for real functions of one variable we get:
$\exists c\in]a,b[$ such that $\left(\mathcal{R}_{(\overrightarrow{u},p)}\circ f\right)'(c) = 0$.
Consequently
$\exists c\in]a,b[$ such that $\mathcal{R}_{(\overrightarrow{u},p)}(f'(c)) = 0$.
Thus, $\exists c\in]a,b[$ such that
$\left(\overrightarrow{u}\wedge\overrightarrow{Of'(c)}\right)\cdot\left(\overrightarrow{P_1}\wedge\overrightarrow{P_2}\right) = 0$.
It follows $\exists c\in]a,b[$ such that
$\left(\overrightarrow{u}\wedge\overrightarrow{Of'(c)}\right) = \overrightarrow{(0,0,0)}$,
because $\overrightarrow{u}\wedge\overrightarrow{Of'(c)}$ and $\overrightarrow{P_1}\wedge\overrightarrow{P_2}$ are collinear and $\overrightarrow{P_1}\wedge\overrightarrow{P_2}$ is not null.
Then, we will have
$\exists c\in]a,b[$ such that $\overrightarrow{u}\wedge\overrightarrow{Of'(c)} = \overrightarrow{u}\wedge\overrightarrow{OO}$.
finally $\exists c\in]a,b[$ such that $f'(c) \underset{\overrightarrow{u}}{=} O$. ■

Theorem 2 (Mean value theorem) *If f is a continuous function on $[a,b]$, differentiable on $]a,b[$, then $\exists c\in]a,b[$ such that $f'(c) \underset{\overrightarrow{u}}{=} \frac{f(b)-f(a)}{b-a}$.*

Proof (Theorem 2)
As in the previous proof (Theorem 1), consider the real function $\mathcal{R}_{(\overrightarrow{u},p)}\circ f$.
By the Mean value theorem for real functions of one variable, we get:
$\exists c\in]a,b[$ such that $f'(c) \underset{\overrightarrow{u}}{=} \frac{f(b)-f(a)}{b-a}$. ■

Definition 4 (Monotony) *A function f is monotone increasing (respectively decreasing) on $[a,b]$ with respect to the direction $\overrightarrow{u}$ if and only if:*
$\forall t_1,t_2\in[a,b]$, $t_1\le t_2\Rightarrow f(t_1) \underset{\overrightarrow{u}}{\le} f(t_2)$ *(respectively $\forall t_1,t_2\in[a,b]$, $t_1\le t_2\Rightarrow f(t_2) \underset{\overrightarrow{u}}{\le} f(t_1)$).*

Remark 2

The monotony is *strict* whenever the order relation $\underset{\overrightarrow{u}}{\le}$ is used.

Lemma 1 *The functions f and $\mathcal{R}_{(\overrightarrow{u},p)}\circ f$ have the same monotony.*

Theorem 3 (variations sens) *If f is continuous on $[a,b]$ and differentiable on $]a,b[$ then:*

- $\forall t\in]a,b[$, $f'(t) \underset{\overrightarrow{u}}{\ge} O \Leftrightarrow$ *f is monotone increasing on $[a,b]$ with respect to the direction $\overrightarrow{u}$.*
- $\forall t\in]a,b[$, $f'(t) \underset{\overrightarrow{u}}{\le} O \Leftrightarrow$ *f is monotone decreasing on $[a,b]$ with respect to the direction $\overrightarrow{u}$.*

Proof (Theorem 3)
That's an obvious corollary of the lemma 1. ■

Definition 5 (Extrema) *The function f admits a maximum (resp a minimum) at the point $t_0\in[a,b]$ with respect to the direction $\overrightarrow{u}$ if and only if there exist $t_1,t_2\in[a,b]$, $t_1\neq t_2$ such that:*
$t_0\in]t_1,t_2[$ *and* $\forall t\in[t_1,t_2]$ $f(t) \underset{\overrightarrow{u}}{\le} f(t_0)$ *(resp $f(t_0) \underset{\overrightarrow{u}}{\le} f(t)$).*

Lemma 2 *The functions f and $\mathcal{R}_{(\overrightarrow{u},p)} \circ f$ have the same extrema.*

Theorem 4 (Extrema and derivative) *Suppose that f is continuous on $[a,b]$ and differentiable on $]a,b[$. Let $t_0 \in]a,b[$. Then if $\overrightarrow{u} \wedge \overrightarrow{Of'(t)}$ vanishes at t_0 and changes its direction, f admits an extrema in t_0 with respect to the direction $\overrightarrow{u}$.*

Proof (Theorem 4)
As previously, consider the real function $\mathcal{R}_{(\overrightarrow{u},p)} \circ f$.
The vector $\overrightarrow{u} \wedge \overrightarrow{Of'(t)}$ is collinear with $\overrightarrow{P_1} \wedge \overrightarrow{P_2}$, and $\overrightarrow{u} \wedge \overrightarrow{Of'(t)}$ is null at t_0 and changes in direction. This will imply that $\mathcal{R}_{(\overrightarrow{u},p)} \circ f'$ is null at t_0 and changes in sign,
it follows that t_0 is an extrema of $\mathcal{R}_{(\overrightarrow{u},p)} \circ f$ on $[a,b]$
We will have from the lemma 2, that t_0 is an extrema of f on $[a,b]$. ■

Remark 3

At $t_0 \in]a,b[$ which is an extrema according to the theorem 4, the directional tangent $\overrightarrow{Of'(t_0)}$ is collinear with the vector $\overrightarrow{u}$.
In the Arabic document processing area, there is no system that typeset Arabic documents with a quality that is like the one in the calligrapher handwritten texts. The simple cause is that the development of fonts respecting the Arabic calligraphy rules isn't a simple continuation of the works done concerning the latin documents. The problem is not simple as the majority had thought in the beginning. The first step and mandatory in this area is to offer to the community a good formalization of the problem. So the goal of the paper is not to *invent a new theory but to give an adequate and direct mathematical formalism.* The theory studied here would be extended in some coming works to offer mathematical ways to develop dynamic fonts of Arabic letters in stretchable outlines when the nib's head is translated without and with rotations.

5. BÉZIER CURVES DECOMPOSITION

Up till now, a way to decompose parametric curves characterizing a segment motion has been developed. So, the surface razed by the segment can be obtained. It is the same for any rigid body. Now, we will focus on the polar Bézier curves. In the following, we'll give a method for decomposing Bézier curves of any degree. An example with a cubic Bézier curve will be given as illustration.

- Let $\mathbb{R}^3$ be the affine space with the rectangular Cartesian system of reference $R\left(O, \overrightarrow{\imath}, \overrightarrow{\jmath}, \overrightarrow{k}\right)$.
- Let $\mathcal{P}$ denote the $\mathbb{R}^2$ affine plane with direct normed basis $p = (\overrightarrow{\imath}, \overrightarrow{\jmath})$ passing through the origin O.
- Let $[M_{10}, M_{20}]$ be a segment to be translated in $\mathcal{P}$.
- $[a,b] = [0,1]$.
- The Bézier curve B describes the movement of M_{10}. Suppose that their control points M_{10}, M_{11}, $\ldots, M_{1n}$ are such that $M_{1i} = (x_i, y_i, 0)$.

The curve B can be defined in $\mathbb{R}^3$ through considering a null component with respect to the vector $\overrightarrow{k}$:

$$\begin{array}{rcccl} B & : & [0,1] & \longrightarrow & \mathcal{P} \\ & & t & \longmapsto & \displaystyle\sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} M_{1i} \end{array}$$

In order to decompose a Bézier's curve into monotone curves, with respect to the direction $\overrightarrow{M_{10}M_{20}}$, we should determine the set of extrema of the direct orthogonal range $\mathcal{R}_{(\overrightarrow{M_{10}M_{20}},p)} \circ B$ on $[0,1]$. Then, these extrema will be classified by increasing order. Let $T_0 = \{t_{01}, t_{02}, \ldots, t_{0m}\}$ be this ordered set, with $m \leq n-1$. Thus, we decompose B with respect to t_{01}, we get two Bézier curves B_0 and B_1 such that B_0 has a constant monotony with respect to $\overrightarrow{M_{10}M_{20}}$ on $[0,1]$ and B_1 is a curve to decompose. We don't need to determine the extrema of B_1 with respect to $\overrightarrow{M_{10}M_{20}}$ because the set of these extrema is $T_1 = \left\{t_{11}, t_{12}, \ldots, t_{1(m-1)}\right\}$ where $t_{1i} = \frac{t_{0(i+1)} - t_{01}}{1 - t_{01}}$ (the formula can be established by a an easy calculation). We continue the process until obtaining the last monotone curve. This method is presented as an automatic algorithm in the following.
Suppose that we have already defined the following data structures and algorithms:
`Point` a data structure modeling a point in $\mathbb{R}^3$,
`Vector` a data structure modeling a vector in $\overrightarrow{\mathbb{R}}^3$, (the Point data structure can be used instead of this one),
`Bezier` a data structure modeling a Bézier's curve,
`PtVect(M10:Point,M20:Point):Vector` a function that results in the vector defined by two points.

ExtremaNbr(B:Bezier,U:Vector):INTEGER a function that results in the number of extrema of B with respect to U,

Extremas(B:Bezier,U:Vector): array[1..NMax]of REAL a function that results in an array of extrema of B with respect to U,

LDecp(B:Bezier,t:REAL):Bezier a function that results in the left Bézier curve by decomposing B with respect to t,

RDecp(B:Bezier,t:REAL):Bezier a function that results in the right Bézier curve by decomposing B with respect to t.

Now, we will give the meaning of some variables and algorithms.

M10, M20 are the points which determine the segment,

U is the decomposition direction,

B is the curve to decompose,

Nx is the B extrema number according to U,

AExt is the array displaying all the extrema of B according to U,

DBC is the array displaying the Bézier curves obtained by decomposition of B according to U,

Ctrd and *Ctre*: are indexes for accessing DBC and AExt arrays

```
⋮
M10,M20 :  Point;
U : Vector;
B : Bezier;
Nx :  INTEGER;
AExt :  array[1..Nmax] fo REAL;
DBC : array[1..Nmax+1] of Bezier;
Ctrd, Ctre :  INTEGER;
BEGIN
 /* B, M10, M20 initialising
 ⋮
 U ← PointVector(M10,M20);
 Nx ← ExtremaNumber(B,U);
 AExt ← Extremas(B,U);
 FOR Ctrd=1, Nx, +1
 BEGIN
  DBC[Ctrd]←LDecp(B,AExt[Ctrd]);
  B←RDecp(B,AExt[Ctrd]);
  FOR Ctre=Ctrd+1, Nx, +1
  AExt[Ctre]←
   (AExt[Ctre]-AExt[Ctrd])/(1-AExt[Ctrd]);
 END
 DBC[Nx+1]←B;
END
```

We can remark that there is no need for two separate functions LDecp and RDecp since both parts of the subdivided curve can be computed by one execution of the algorithm of de Casteljau. We use two functions for better clarity and also when we use the algorithm of de Casteljau we must extract the decomposition parts separately.

The Method will be illustrated through an example modeling the movement of a segment $[M_{10}, M_{20}]$ with a cubic Bézier curve B with two extrema with respect to the vector $\overrightarrow{M_{10}M_{20}}$ (See Figure 10).

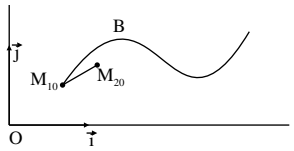


Figure 10. A cubic Bézier curve of a movement

The surface razed by the segment $[M_{10}, M_{20}]$ (See Figure 11) is different from the surface delimited by the outlines associated to the movement (See Figure 12). The reason behind this is that B is not monotone with respect to $\overrightarrow{M_{10}M_{20}}$.

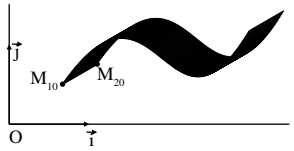


Figure 11. Surface razed by the segment $[M_{10}, M_{20}]$

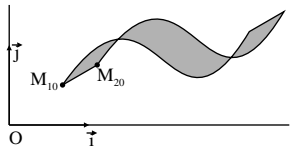


Figure 12. Surface delimited by the associated outlines

We will have to decompose the curve B into monotone sub-curves with respect to the direction $\overrightarrow{M_{10}M_{20}}$. The B control points are $M_{10} = (95, 50)$, $M_{11} = (130, 100)$, $M_{12} = (135, 20)$ and $M_{13} = (165, 70)$. We determine the direct orthogonal range of B. In order to find the extrema on $]0, 1[$, we derive this range. Then, we find that the direct orthogonal range has two extrema $T_{01} = 0.1571942250$ and $T_{02} = 0.8301512030$. We decompose B with respect to T_{01} to obtain two curves; B_1 monotone, and a curve B_2 to decompose again (See Figure 13).

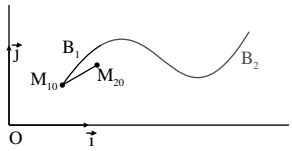


Figure 13. First decomposition

Then, the curve B_2 is decomposed with respect to $T_{11} = \frac{T_{02}-T_{01}}{1-T_{01}}$, say for instance, $T_{11} = 0.7984721960$. The decomposition gives two curves B_{21} and B_{22}. Both of them are monotone with respect to $\overrightarrow{M_{10}M_{20}}$ (See Figure 14).

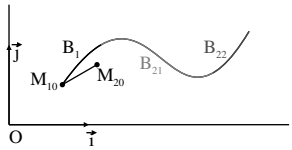


Figure 14. Second decomposition

If we superpose all the surfaces delimited by the outlines associated to the three curves B_1, B_{21} and B_{22} (See Figure 15), we get exactly the surface in Figure 11.

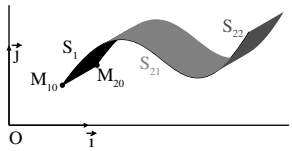


Figure 15. Superposition of the surfaces of decomposition

6. CONCLUSIONS

A method for the determination of the extrema of a parametric curve is now ready for use. This method has been used to find the decomposition's coefficients of cubic parametric curves. This allows generating the PostScript code which supports the movement of an Arabic nib that remains with a constant angle of inclination with the baseline. The case where this angle is changing will be presented in a next work. The study of both the Bézier curves variations and angle's variations will lead to use also approximations with polar curves.

7. REFERENCES

[Adob 99] Adobe. PostScript Language reference. Third edition, Adobe Systems Incorporated, Library of Congress Cataloging-in-Publication Data, 1999.

[Bars 85] 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.

[Bézi 77] Pierre E. Bézier, Essai de Définition Numérique des Courbes et des Surfaces Expérimentales, PhD dissertation, Université Pierre et Marie Curie, Paris, 1977.

[HiLo 05] High-Logic, Font Creator Manual, http://www.high-logic.com, 2005 .

[Java 05] Sun, Java 2D Programmer's Guide, http://java.sun.com/products/jdk/1.2/docs/guide/2d/spec/j2d-title.fm.HTML,2005.

[Knut 86] D.E. Knuth, The Metafont Book, Computers and Typesetting, Vol. C. Reading MA: Addison-Wesley. 1986.

[MEss 87] Mahdi Essayed Mahmoud, Learning Arabic Calligraphy: Naskh, Roqaa, Farsi, Tholoth, Diwany, Ibn Sina, Publisher, Cairo, Egypt, 1987.

[MEss 94] Mahdi Essayed Mahmoud, Learn your self Arabic Calligraphy: Naskh, Roqaa, Farsi, Tholoth, Diwany, Ibn Sinaa publisher Cairo, Egypt, 1994.

[Jeff 81] Jeffrey M. Lane, Richard F. Riesenfelf, Bounds on Polynomial, BIT, Vol 21, No 1, pp. 112-117, 1981.

[Hosa 80] Mamoru Hosaka, Fumihiko Kimura, A Theory and Methods for Free Form Shape Construction, Journal of Information Processing, Vol 3, No 3, pp. 140-151, 1980.

[Gold 82] Ronald N. Goldman, Using Degenerate Bézier Triangles and Tetrahedra to Subdivide Bézier Curves, Computer-Aided Design, Vol 14, No 6, pp. 307-311, November 1982.

Partial 3D Shape Matching Using Large Fat Tetrahedrons

J.S.M. Vergeest,
A. Kooijman, Y. Song

Delft University of Technology
Landbergstraat 15, NL-2628 CE
Delft, The Netherlands

j.s.m.vergeest@tudelft.nl

ABSTRACT

We present a method of automatic alignment of two partially matching 3D shapes. The algorithm selects large fat tetrahedrons (LFT) composed of 4 vertices in one shape and exhaustively searches in the other shape for sets of 4 vertices being compatible with the tetrahedron. By selecting such salient tetrahedrons that are relatively wide and fat, although also being not too unlikely to be contained in the overlap region, the cost of search can be reduced. The method is relatively insensitive to noise and not depending on the existence of local shape features nor on feature correspondences. When implemented on a GPU in Cuda, two point sets of 40,000 each can be aligned within seconds. The method is intended to support interactive 3D scan registration applications.

Keywords

Scan view registration, partial shape matching, fat tetrahedron, GPU, Cuda

1. INTRODUCTION

Partial shape matching is an essential step in the reconstruction of geometric objects. One important application is 3D scanning of physical objects. To construct a geometric model from a physical object, multiple scan views are taken, each consisting of range data, *i.e.* 3D points representing the object's surface. Since the orientation of the object relative to the scanning device is different for different scan views, the collection of points from all scan views do not as such represent the object's surface. First the points need to be aligned to each other, that is be transformed to a common coordinate system. The process of aligning the scan views is called scan view registration. From the aligned point sets the surface of the object can be reconstructed, either fully or partially, depending on the coverage of the scan views. If the surface can be fully reconstructed, it can be assumed to represent the boundary of a volume, or solid model. Then a solid model can be derived, which can serve as input to a CAD (Computer-Aided Design) system for further modeling and processing. To base a design on an existing object or on reuse of precedent models is an important paradigm in some industries, such as industrial design engineering. Such a method cannot be successful unless occasional users of scanning devices can easily operate the system. However, the registration task is, even nowadays, still an impeding factor. In practice the user could be a stylist who has manually created a clay model of a future household device. Whereas taking the scan views of the clay model is a commonsense task to him/her, the registration of view pairs is not. The scanning system's manufacturer normally offers an interactive software package, allowing the user to designate correspondences he/she observes among the scan views, as to provide a starting position for a shape matching algorithm, typically based on the ICP (Iterative Closest Point) method. Each scan view has to be aligned, by the user, with scan view(s) aligned previously. Generally this way of operating the scanner is perceived as slow and tedious, both by incidental users and by trained operators.

To improve the operation speed and ease of 3D scanning, the equipment can be enriched with mechanical or magnetic location/orientation trackers attached to the scanner or to the object being sensed or to both of them. Another, commercially available,

solution is based on the detection of optical markers attached to the object being scanned.

If no additional devices or markers are applied and if one does not want to rely on human intervention, the registration problem depend on automatic partial shape matching. Several approaches have been reported to the problem of partial shape matching. From here on we assume that the input data consists of unordered point sets only. That is, we will not rely on preprocesses that generate surface meshes, nor on additional information such as color, texture or material properties of the scanned object. We focus on the kernel problem of matching two point sets. Most methods make use of geometric descriptors and/or feature points. A geometric descriptor is rule to assign a characterization vector to each point in a data set, where the vector is typically computed from points in the neighborhood around the point. The geometric descriptor can take many forms, including moments, FFT coefficients, spin images etc [Johnson 1999]. Since geometric descriptors are invariant under rigid body transformations, they can be used to detect correspondences between two shapes. However, the number of points in a scan view can be large, possibly leading to too many descriptors, causing too long processing time. Then one can attempt to define feature points of shape. Feature points are invariant under rigid body transformations as well, and they are typically small in number. If correct feature points can be generated within the overlap region of two scan views, then the approximate transformation can be easily derived and be applied to the point sets to bring them into a position from which ICP can be successfully used. However, feature points are commonly derived from point differences (for example local curvature) and hence sensitive to noise, which may cause wrong correspondence assignments. Defining a feature using integrated quantities rather than using derivatives reduces the influence of noise [Gelfland 2005]. Another approach to diminish sensitivity to noise and data outliers is taken by [Aiger 2008]. He collects sets of 4 planar points in both point clouds. If a particular set from one point cloud is approximately congruent with one from the other point cloud, a candidate corresponding pair of 4-points sets is found. If the 4-points set is relatively wide, then the method is less sensitive to noise. In this paper we will not further review existing work on partial shape matching. We refer to [Gelfland 2005] and references therein for a more extended description of registration methods.

Our approach is inspired by the 4-points congruent sets as in [Aiger 2008]. We look for 4-points sets which define a large fat tetrahedron (LFT). The assumption is that the geometry of a large tetrahedron is relatively rare and therefore can serve to detect correspondences in the two point clouds. However, since true correspondence exist in the overlap region only, a bound must be set to the maximum size of the tetrahedrons. Secondly, since the number of fat tetrahedrons in point sets can be very large, a straightforward comparison of two sets of tetrahedrons (each derived from one point cloud) would not be efficient. Our algorithm derives a limited number of fat tetrahedrons from one point cloud. Then each tetrahedron is tested for being approximately congruent to any point neighborhood of the other point set. Although the latter step of the algorithm is expensive, it can be easily parallelized and the correct approximate transformation can be found in about 10 seconds in a Cuda implementation.

In the next section the LFT algorithm will be described. In section 3 we present the achievements of the method. Conclusions and recommendations are given in section 4.

2. THE LFT ALGORITHM

Let two point sets A and B be given, originating from sampling of a portion of the surface of a three-dimensional object. There may exist subsets $A' \subseteq A$ and $B' \subseteq B$ such that A' and B' are samples of the same subsurface of the object. A' and B' are then said to represent an overlap region of the samples.

Let a set of sets B_i be a partitioning of B be defined as follows. A three-dimensional grid is constructed, aligned with a bounding box of B. The grid has the size of the bounding box of B. The block-shaped grid elements all have the same size and have index i, $i = 1, ..., n_G$, where n_G is the number of grid elements of B. Each grid element encloses zero or more points of B. Each point of B is enclosed by exactly one grid element. B_i is the set of points enclosed by grid element indexed i.

Let A' and B' be the largest overlap of A and B, informally defined as follows. Assuming that A and B are range images of a physical object, let S_A and S_B informally be defined as the portions of the object's surfaces represented by A and B, respectively. Then, both A' and B' represent a superset of the surface $S_A \cap S_B$. Depending on the extent of $S_A \cap S_B$, A' and B' each may contain zero up to as many points as the cardinality of A and B, respectively.

Let $B'_i = B_i \cap B'$. Our search strategy is based on the assumption that the overlap region is connected and has the extent of at least the size of a grid element. In such cases there might exist sets B_i containing multiple points of B_i'. A property of any point of B' is that its Euclidian distance to A is relatively small, provided that A and B are defined in the same coordinate system. However, since A and B originate

from independent sampling processes, they will in general be defined in different coordinate systems. The difference between the two coordinate systems can be described by a rigid body transformation M, such that MB and A are defined in the same coordinate system, where MB is the set of points of B to which transformation M has been applied. Let b_{ij} be the jth point in B_i, with $j = 1, ..., n_i$, where n_i is the number of points in B_i.

We could naively define a search algorithm as follows:

Exhaustive search algorithm

```
for r = 1, ..., n_r //  loop over possible transformations
  for i = 1, ..., n_G  // loop over grid elements
    n_ri = 0           // number of nearby points so far
    for j = 1, ..., n_i // for points in one grid element
      for s = 1, ..., n_A // loop over points in A
        if | M_r b_ij – a_s| <  δ then n_ri++ // evaluate dist.
      endfor
    endfor
  endfor
endfor,
```

where r defines the parameters of a rigid body transformation M_r. One must then assume that the problem can be solved by considering a finite number n_r of transformations. Typically, r is a 6-dimensional vector defining translation and rotation in 3D space. n_r is the total number of configurations achieved by stepping over finite intervals of translation and rotation. The points contained in A are denoted by a_s with $s = 1, ..., n_a$. The value n_{ri} is the number of those points in grid element i that are close to A after applying transformation M_r to B. When $n_{ri} > 3$ (or any other threshold) we could define r as a candidate parameter set for an approximate alignment transformation. The value of δ (in the algorithm) is typically set to a small multiple of the sampling spacing of A and B.

It is well-known that exhaustive sampling is generally too slow for obtaining scan view registration within seconds, as would be needed for the purpose of interactive 3D scanning. The critical factors here are the step sizes in the six dimensions of configuration space, that define the number n_r of point cloud distance computations. However, using the GPU and adaptive step sizes might prove exhaustive search be feasible in the future [Kooijman 2009].

To reduce the number of distance calculations, we need to extend the algorithm in order to select "promising" transformations. If, for a particular transformation Mr, an (even small) number of points in a grid element is, after transformation, very close to A then these points could be taken as candidates for being points in the overlap region. However, in regions where the object's surface has low curvature, points in the overlap region poorly define the proper transformation M_r. Therefore we need also to consider the degree of planarity of the points close to A. When a small set of points (4 or more) of B_i is close to A *and* if these points are sufficiently non-planar, then the transformation to match this set with A is a relatively good candidate of the M_r we are looking for. Relying on this principle we base the algorithm on matching 4 points to A, where the 4 points are contained in the same grid element. The 4 points, denoted l_1, l_2, l_3, l_4, are selected from B_i such that they form L, a large least-planar set, or large fat tetrahedron (LFT) as follows: l_1 and l_2 are the points in B_i which are furthest apart. l_3 is the point in B_i furthest from the line through l_1 and l_2, that is it maximizes $|(l_3 - l_1) \times (l_2 - l_1)|$. l_4 is the point in B_i furthest from the plane defined by l_1, l_2 and l_3, that is it maximizes $| ((l_3 - l_1) \times (l_2 - l_1)) \cdot (l_4 - l_1) |$.

If L is contained in B' then the directed Hausdorff distance of M_rL to A will be small for the proper transformation M_r. To test whether this is the case and to determine M_r we proceed as follows. At first, we apply a translation to L such that point l_1 coincides with a given point $a_s \in A$ (the following procedure will be repeated for all points in A). We will leave point l_1 unchanged but otherwise rotate L around this pivot point to find out whether the remaining three points can be positioned close to A, which will be the case when there exist points m_2, m_3, $m_4 \in A$, such that $| M_r l_j - m_j |$ is small for $j = 2, 3, 4$. Since (l_1, l_2) is the longer edge of tetrahedron L, all candidate points m_2, m_3 and m_4 must be on or be contained in the sphere of radius $|l_2 - l_1|$ centered at a_s. Let us define $A_s \subseteq A$ as the set of points of A either on or inside this sphere. Candidate points m_2 will be on the surface of sphere up to some accuracy depending on sampling spacing and the precision of the sampling process. Let $m_2' \in A_s$ be such a candidate point. Then L is rotated about point l_1 such that point l_2 comes closest to m_2', which implies that a_s, l_2 and m_2' are collinear. Now there is one degree of freedom left for L, namely its rotation about the line through l_1 and l_2. If point l_3 would (approximately) hit any point m_3 in A_s then L has attained a configuration which should be further evaluated. If, then, point l_4 is close to any point m_4 in A_s, L is called being matching to A. The implied transformation M_r can be determined, as described later. All points in B_i are then subjected to transformation M_r and their distance to A is determined. If a sufficient fraction of the points are close to A then M_r is saved as a candidate alignment transformation.

Algorithm based on LFT (Large Fat Tetrahedron)

for $i = 1, \ldots, n_G$ // loop over grid elements
  select l in B_i // l is large fat tetrahedron
    for $s = 1, \ldots, n_A$ // loop over points in A
      move L such that $l_1 = a_s$ // anchoring 1st point of *LFT* to A
      determine A_s // points of A in sphere at l_1, of radius $|l_1 - l_2|$
      for moves of L such that l_2 reaches A_s // move 2nd point of *LFT* to A
        for moves of L such that l_3 reaches A_s // move 3d point of *LFT* to A
          if l_4 near A_s then evaluate goodness of matching, using points in M_rB_i
          endif
        endfor
      endfor
    endfor
  endfor

The number of candidate transformations M_r depends on the various distance criteria that should be set in the algorithm. As a final step, the candidate transformations are used to compute the set M_rB, involving all points of B, and the degree of matching of the set to A is determined. The transformation producing the best alignment is the outcome of the method. The goodness of an alignment is defined as the number of points in M_rB closer to A than a predefined threshold distance.

The transformation M_r can be written in explicit form as follows. We define transformation M_1 to bring point l_1 to a_s as translation $T(a_s - l_1)$. Let us apply M_1 to L. Subsequently L is transformed by M_2 such that the vector $l_2 - a_s$ is rotated about direction vector $d_1 = (l_2 - a_s) \times (m_2' - a_z)$ to reach direction $(m'_2 - a_s)$, which requires the rotation angle

$$\alpha_1 = \text{acos}((l_2 - a_s) \cdot (m_2' - a_z)) / (|l_2 - a_s| \, |m_2' - a_z|).$$

The axis of rotation goes through point a_s, that is the rotation is indeed around point l_1 of L. Therefore $M_2 = T(a_s)\ R(d_1,\ \alpha_1)\ T(-a_s)$, where $R(d_1,\ \alpha_1)$ is the rotation of angle α_1 about the line through the origin, with direction d_1. An implementation note: the program gets a bit simpler and slightly more efficient if we initially transform all points a_i with $T(-a_s)$, that is point cloud A is moved such that a_s reaches the origin and from then on $T(a_s)$ would become identity. The next transformation, M_3, is a rotation about the current direction vector $l_2 - a_s$ by angle α_2. This angle is determined by point m_3 and computed as the angle between the vectors n_1 and n_2, the vectors normal to the triangles $(a_s,\ l_2,\ l_3)$ and $(a_s,\ l_2,\ m_3)$, respectively, that is $n_1 = (l_2 - a_s) \times (l_3 - a_s)$, $n_2 = (l_2 - a_s) \times (m_3 - a_s)$ and $\alpha_2 = \text{acos}(n_1 \cdot n_2) / (|n_1|\ |n_2|)$. The rotation form n_1 toward n_2 is in positive sense about vector $d_2 = n_1 \times n_2$. Thus $M_3 = T(a_s)\ R(d_2\ ,\ \alpha_2)\ T(-a_s)$. The total transformation gets $M_r = M_3\ M_2\ M_1$, and accounting for some canceling translations we obtain:

$$M_r = T(a_s)\ R(d_2,\ \alpha_2)\ R(d_1,\ \alpha_1)\ T(-l_1). \qquad (1)$$

M_r can be interpreted as a transformation of corner l_1 of the LFT to the origin, then two rotations about lines through origin and finally translating l_1 to point a_s. The explicit matrix form of transformation $R(v, \alpha)$, $v = (v_0, v_1, v_2)$ is

$$R(v,\alpha) = U \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} U^T, \qquad (2)$$

with

$$U = \begin{pmatrix} u_{00} & u_{01} & u_{02} \\ u_{10} & u_{11} & u_{12} \\ u_{20} & u_{21} & u_{22} \end{pmatrix}, \text{ and}$$

$$\begin{pmatrix} u_{02} \\ u_{12} \\ u_{22} \end{pmatrix} = \frac{1}{\sqrt{v_0^2 + v_1^2 + v_2^2}} \begin{pmatrix} v_0 \\ v_1 \\ v_2 \end{pmatrix},$$

$$\begin{pmatrix} u_{01} \\ u_{11} \\ u_{21} \end{pmatrix} = \frac{1}{\sqrt{v_0^2 + v_1^2}} \begin{pmatrix} -v_1 \\ v_0 \\ 0 \end{pmatrix} \text{ and}$$

$$\begin{pmatrix} u_{00} \\ u_{10} \\ u_{20} \end{pmatrix} = \begin{pmatrix} u_{01} \\ u_{11} \\ u_{21} \end{pmatrix} \times \begin{pmatrix} u_{02} \\ u_{12} \\ u_{22} \end{pmatrix}.$$

The transformation of equation (1) can be implemented efficiently using this form to transform larger number of points in B_i as to evaluate the closeness of M_rB_i and/or M_rB to A.

3. IMPLEMENTATION AND RESULTS

First, to illustrate the steps of the LFT algorithm we apply it to two scan views (A and B) from the Stanford Bunny data set [Stanford 2009]. The scan views consist of about 40,000 points each. The relative position and orientation of A and B are completely arbitrary, as a result of the particular scanning process. The scan views were decimated to 3185 and 2406 points, respectively (Figure 1) to increase the processing speed. In Figure 2 scan view B is shown as well as one of the tetrahedrons. This tetrahedron happened to be contained in the overlap region. The tetrahedron was selected by the algorithm as follows in this particular example run. The subdivision grid for points B was chosen to consist of 5×5×5 elements. Out of these 125 elements, 8 contained more than 96 points, which was a lower threshold (discussed later), as set for this particular run. In each of these 8 grid elements the LFT was constructed as described in Section 2. Each of the 8 tetrahedrons was sufficiently fat (criterion discussed later) and was subjected to the exhaustive test of congruence with any subset of A. Depending on the criterion for points of a tetrahedron to coincide with a point of A, the number of sets in A congruent to a tetrahedron was typically between 20,000 and 200,000, which is the gross number of candidate transformations per each grid element. For each transformation, the fraction of points M_rB_i (that is the points inside the current grid element) closer than some preset distance to A was computed. If this fraction exceeded a predefined threshold, the number of points in M_rB close to A was determined. The latter computation, involving the full point set B, occurred relatively seldom, between 0 and 300 times only, which is less than 1% of the number of candidate transformations.

In Figure 3 the number of points in M_rB close to A is shown for the candidate transformations M_r, where the x-axis represents the amount of rotation implied by matrix M_r. A couple of almost similar transformations, having rotation angle near 90 degrees, produce the highest score. From the ground truth information that we have about the scan views, we can conclude that the correct alignment was obtained. The points M_rB, which are matching to shape A are shown, together with points A in Figure 4.

The computation of this partial shape matching process took 450s on an ordinary PC, which could be reduced to about 10s when implemented on a GPU, not including a couple of seconds time to import and decimate the point clouds.

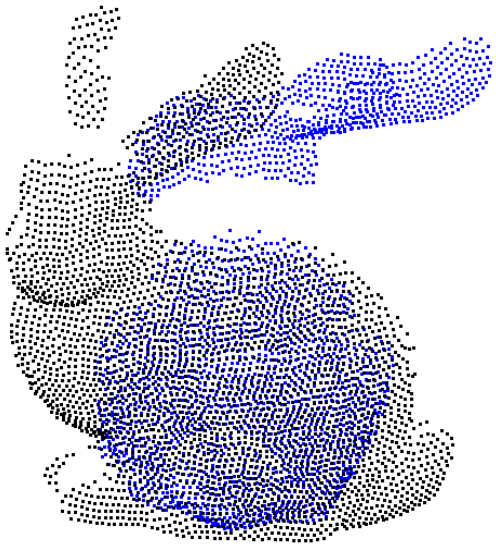


Figure 1. Two scan views (*A* and *B*) of Bunny before alignment.

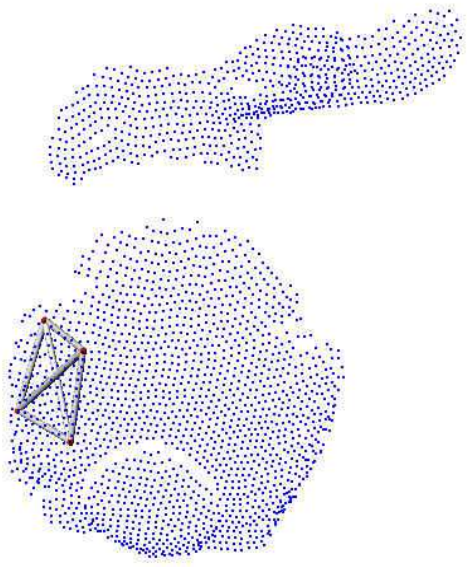


Figure 2. Points in scan view *B* and the best performing LFT depicted by its 4 edges.

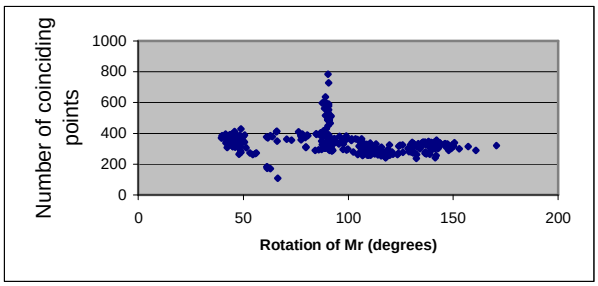


Figure 3. Number of points of M_rB close to A versus rotation angle of candidate transformations M_r.

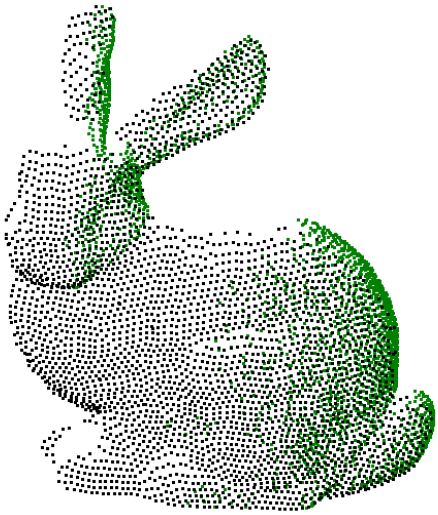


Figure 4. Result of the alignment; point sets *A* and *Mr B*.

As a next example two scan views of a human face, consisting of about 18,000 points each, were taken using our Minolta Vivid700 laser scanner, see Figure 5. Since the overlap region of the two views was relatively large, a good fit could be obtained with sets down-sampled to 9% of the original cardinality. The rotation angle plot is shown in Fig. 6. The computation time for this particular on CPU (not GPU) was about 40s, not including file read/write and cloud decimation, which took together 10s. The best transformation matrix obtained by the algorithm from the down-sampled data differed only little compared to the one from the original data. The amounts of rotations, for example, of the two transformations differed by 0.6 degrees. It is an indication that the result of the algorithm is suited for further processing by common ICP-based registration software.

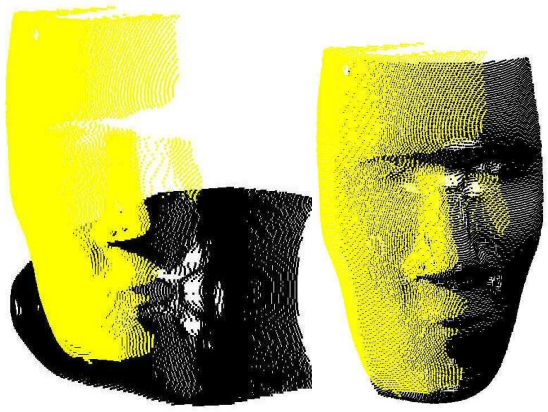


Figure 5. Two scan views before (left) and after partial shape matching (right).

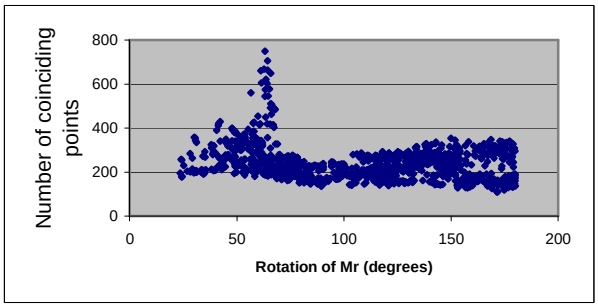


Figure 6. Number of points of M_rB close to A versus rotation angle of candidate transformations M_r for the scan views shown in Figure 5.

In Figure 7 two scan views of an automobile shape are shown. The physical, scaled, model was about 25cm long and very much simplified, containing no detailed shape features. The top view of the car model also contains data from the left side of the car, as can be seen in Figure 7, however that part of the surface is recorded at a small incident angle at relatively poor accuracy. Yet, the tetrahedron selected by the algorithm in scan view *B* (see Figure 8) could be matched to the correct location in scan view *A*, thus providing a good initial match, shown in Figure 9. The tetrahedron is quite large because the size of each grid element is relatively large. We did not remove "outliers" such as data due to the supporting platform and therefore the bounding box got large and so did the grid elements. It can be noted that the background data of scan view *A* has not affected the outcome of the algorithm but has slowed down the search process, since each point in *A* is tested by the LFT algorithm as a potential corner point of the tetrahedron. Using the full data, without decimation of the point clouds (containing about 10,000 points each), the computation time is 15s on GPU. When the point clouds are reduced to about 2000 each, the computation is accelerated

dramatically to less than 1s on GPU or 19s on a CPU. The resulting transformation was still approximately correct, *i.e.* sufficiently accurate to reach fine registration using a commercial tool based on ICP or a general purpose minimizer based on steepest descent.

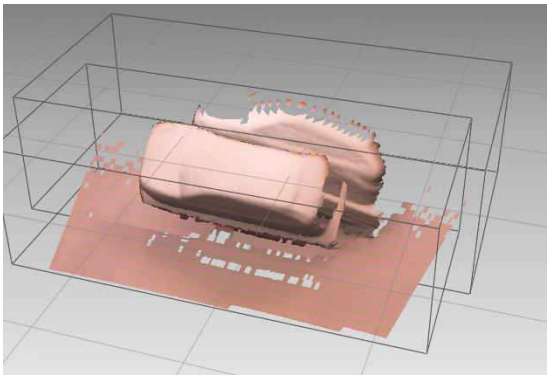


Figure 7. Top view and left side view of a simple car model.

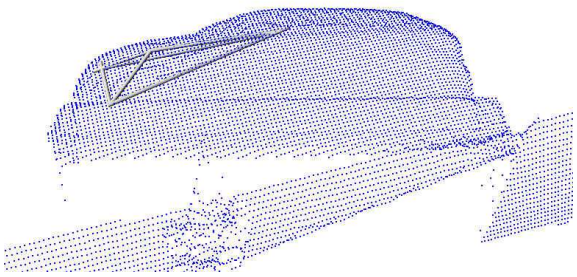


Figure 8. Point cloud *B* representing the left side view of the car and an LFT leading to the correct transformation M_r. The LFT is depicted by its four edges.

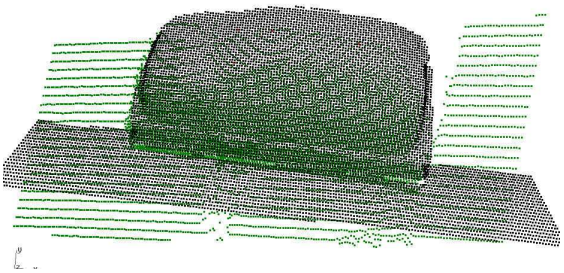


Figure 9. Correct matching of the two scan views from the car model.

4. DISCUSSION AND CONCLUSIONS

The partial shape matching algorithm based on large fat tetrahedrons (LFT) has been evaluated using a, as yet still small, number of data sets.

Both the goodness of the transformation M_r and the amount of computation time spent by the algorithm depend on the settings of various tolerance criteria. For example, if the thickness of the spherical surface of A_s is too small, the correct position of l_2 could be missed; however if the thickness is taken too large then the number of candidate tetrahedrons can become excessive. Similar reasoning can be applied to the number of grid elements subdividing the bounding box, and other parameters mentioned. In our implementation, the chief parameter is the "assumed scanning distance" *d*, which is the typical closest distance between the measured points. All other geometric thresholds in the algorithm are currently defined proportional to *d*.

The algorithm has provided the correct transformation for the partial match in all cases we investigated so far. More examples are being studied.

The distance parameter *d* is critical with respect to computation time. For example, whereas the matching example of Figure 3 took 10s with *d* set to 1.5 units, the computation took 70s with $d = 2.0$. The outcome of the algorithm was the same, but there were relatively many false candidate tetrahedron placements (4 million compared to 0.5 million at $d = 1.5$). The increase of the number of false hits can be seen by comparing Figure 10 to Figure 3.

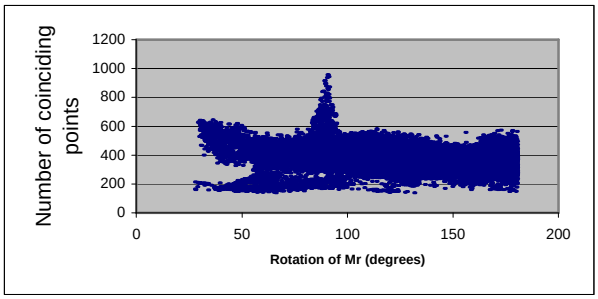


Figure 10. Number of points of M_rB close to *A* versus rotation angle of candidate transformations M_r for the same scan data samples as in Figure 3, with larger parameter *d*.

If the grid element size gets smaller than the overlap region of the two scan views then the algorithm would no longer be able to find a correct match. There should also be a lower limit on the number of points in a grid element to decide whether an LFT will be considered. We have set the default threshold to $n_B / n_G^{2/3}$, which is appropriate for point sets *B* representing a smooth surface.

The numerical tests indicate that the method is not very sensitive to down-sampling of the point clouds and hence could be scalable. The down sampling algorithm that we applied simulates lower precision scanning, and was intentionally not designed to

preserve sharp curvatures. Indeed, the LFT method is not dependent on high curvature features, but on "medium-size" aspects, which are preserved under down-sampling.

In addition the method discards flat regions, including the planar "outlier" points as in the car model, since flat tetrahedrons are by definition non-fat. However, if a grid element contains a mixture of data from the model and from a supporting table, wrong tetrahedrons could emerge.

As mentioned, the computation time depends strongly on the degree of decimation and the values to which the threshold parameters are set. From the numerical experiments we learned how to tweak these parameters to settings which make the computation very fast; however, for a given pair of point clouds it is not *a priory* known how to chose the optimal parameters. The algorithm could be extended with pre-scans of the data in order to estimate an optimal setting.

Since the computation time to match two scan views remains below 10s using the GPU or even the CPU, the LFT method seems very suitable to practically support interactive and mobile scanning processes.

As mentioned, the data samples we used to evaluate the algorithm originate from repositories, such as in [Stanford 2009], or from measurements we conducted ourselves. To our knowledge there is not yet a suitable database containing data samples that could be used for benchmarking purposes for partial shape matching. Indeed, there are databases containing data samples to evaluate shape retrieval algorithms. However, these data samples represent shapes, not partial shapes. The authors look forward to suggestions to achieve at a repository of partial shapes to benchmark scan view registration methods.

REFERENCES

[Aiger 2008] Aiger, D., Niloy, M., Cohen–Or, D. 2008. 4–Points Congruent Sets for Robust Pairwise Surface Registration. ACM Trans. Graph. 27, 3.

[Johnson 1999] Johnson, A.E. and M. Hebert, "Using spin-images for efficient multiple model recognition in cluttered 3-D scenes," IEEE Trans. Pattern Anal. Mach. Intell., vol. 21, no. 5, pp. 433–449, 1999.

[Gelfland 2005] Gelfland, N., Mitra, N. J., Guibas, L. J., Pottmann, Robust global registration. In Proc. Symp. Geometry Processing, Eurographics, pp 197–206.

[Kooijman 2009] Kooijman, A., Vergeest, J.S.M., A GPU-Supported Approach to Registration of 3D Scan Data. Proceedings of the Gravisma 2009, V. Scala (Ed), in press.

[Stanford 2009] Stanford Scan data Repository, http://graphics.stanford.edu/data/3Dscanrep

PhotoPath: Single Image Path Depictions from Multiple Photographs

Christopher Schwartz, Ruwen Schnabel, Patrick Degener, Reinhard Klein
Institute for Computer Science II, University of Bonn

ABSTRACT

In this paper we present PhotoPath, an image synthesis technique that allows the visualization of paths through real world environments in a single intuitive picture. Given a series of input photographs taken with a hand-held camera along the path, our method generates an image that creates the impression of looking along the path all the way to the destination regardless of any curves or corners in the route. Rendered from a pedestrian perspective, this visualization supports intuitive wayfinding and orientation while avoiding occlusion of path segments in the image.
Our approach intentionally avoids an involved and error-prone scene reconstruction. Instead we advocate the use of planar geometry proxies estimated from the sparse point-cloud obtained with Structure-from-Motion. The proxy geometry is spatially deformed in order to align the curved path with the straight viewing direction of the synthesized view. Finally, we propose a novel image composition algorithm accommodating for parallax and occlusion artifacts due to the approximation errors of the actual scene geometry.

Keywords: image-based-rendering, navigation, non-photorealistic-rendering, space-deformation

1 INTRODUCTION

To ease wayfinding in unknown environments, Degener et al. [7] recently proposed an efficient visualization for short paths as they are typically encountered public places like hotels, airports or museums. Their method intuitively conveys directions in a single 'warped' image of the path that gives the impression of looking along the path all the way to the destination. Unlike traditional floor plans, warped images show the path from the visitor's perspective and thus contain realistic portrayals of landmarks which enables intuitive orientation and wayfinding. In contrast to conventional photographs in which segments of the path that lie around a corner usually are occluded, the visibility of the entire path is ensured by a space deformation that aligns the route with the viewing direction. Degener et al. demonstrate the effectiveness of their visualization in a comparative user-study. An example of a warped image is shown in Figure 1. Such images are valuable and inexpensive navigation aids that can be handed out to visitors. However, the approach of Degener et al. suffers from a severe practical limitation: It supposes that a fully modeled, high quality textured 3D model of the scene is already given which, naturally, seldom is the case. To overcome this hurdle we want to use nothing

Figure 1: Given a series of input photographs and a sparse 3D point-cloud created by a Structure-from-Motion process our method generates a single image that summarizes the views along a user specified path (The path points are depicted in blue in the middle). In the synthesized image curves and corners in the path are straightened such that the impression is created of looking along the path all the way to the destination. Beside a use as navigational aid such images also possess a certain artistic appeal.

Plzen, Czech Republic.

but a series of photographs taken along the path with an ordinary hand-held camera.

Unfortunately, the extraction of accurate 3D geometry from photographs is a difficult task. Even though a set of photographs could in principle suffice for a full-scale multiview reconstruction [13, 20] this is relatively unreliable in general. In particular dense stereo matching is error prone if applied to images with a large baseline and many specular surfaces. However, recently approaches such as Photo Tourism [17] and [16] have demonstrated that Structure-from-Motion (SfM) can be used to robustly extract a sparse 3D point-cloud from a set of unordered photographs with relatively large baselines. The question now is if this sparse 3D information already suffices to generate warped images.

In this work we present PhotoPath that, rather than relying on a full-scale reconstruction, has at its heart a novel image-based rendering technique that is based on a sparse set of 3D-points reconstructed from image features in a SfM step.

Our approach is based on a coarse, not necessarily globally consistent, proxy geometry for the scene. As typical paths in urban or indoor environments are strongly dominated by planar geometry (facades, floors, walls or ceilings), our method simply fits a set of planar proxies to the sparse 3D point cloud. To avoid unmanageable occlusion issues, we introduce an innovative new technique for finding suitable boundaries to the potentially infinite planar proxies, based on a energy minimization in the space of the input images. Obviously however, an inaccurate proxy geometry as we advocate poses challenges for the subsequent image-based rendering due to frequently occurring incorrect occlusion and parallax effects. We demonstrate that these issues can be addressed effectively with our novel image stitching method that minimizes parallax and occlusion artifacts and at the same time is powerful enough to allow for the space deformation necessary for straightening the path. While our algorithm produces good results fully automatically – except for a few parameter settings – we also allow for interactive user intervention in all steps of the method to let the user further improve the final result.

2 RELATED WORK

Image Based Rendering (IBR) methods that create novel views from a set of input images have a long tradition in computer graphics. Individual approaches differ mostly in the richness of the underlying geometry representation. In general, to achieve a given rendering quality IBR methods have to make a trade-off between the required number of input images and the geometric detail of the employed proxy object. While the paper of Levoy and Hanrahan [11], proposing Light Field Rendering, was purely image based using no notion of geometry at all, this comes at the expense of a high number of required input views. The Lumigraph system [10] demonstrated that rendering quality drastically improves if a simple 3D proxy for the scene geometry is used. At the other extreme is view dependent texture mapping as used by Debevec et al. [6] in the Facade system: Starting from simple architectural models, Debevec et al. combine projections of images taken from different viewpoints into textures to increase realism. A generalization of these approaches was given by Buehler et al. [4], who presented an algorithm for unstructured lumigraph rendering that gives compelling renderings even for a sparse set of input images if a detailed proxy surface mesh approximating the scene is given. If depth maps are available for each image, it is also possible to apply classical image warping techniques [5, 12] to obtain novel views. If the density of ray sampling or the accuracy of the geometric proxy is insufficient, as in our case, all of the above IBR methods are prone to ghosting artifacts. As shown by Eisemann et al. [9] these ghosting artifacts can be reduced by warping projected images on the proxy. However, they assume that visibility is only affected by small geometry errors, whereas in our case, no reliable visibility function can be derived at all.

Our method can also be seen as an image-based rendering with a non-standard multi-perspective camera. In this area, the work of Agarwala et al. [1] [2] is related to our approach in two respects. In [2] they introduced user controllable Markov Random Field (MRF) optimization in the context of image stitching and in [1] they used a single proxy plane for composing a series of photographs into a multi-viewpoint panorama that can capture e.g. the side of a street in a single image. Our method is based on the same optimization principle and allows for user intervention in a similar manner but uses an objective function specifically designed for our scenario.

For an in-depth discussion of related work concerning Route Visualization for Navigation, we refer the interested reader to the paper of Degener et al. [7]. Let us here only shortly mention that for successful navigation a correspondence between map and environment has to be established by the observer. Humans perform this task by identifying and matching landmarks in a scene (e.g. walls, corners, doors) with their map representation [14]. This is greatly facilitated by our composite images since they contain real-world depictions of all these features and are therefore easily understood and memorized (the slight distortion in the image is usually not an issue for a human observer).

3 OVERVIEW

Our method requires an input set of photographs $\mathscr{I} = \{I_1, \ldots, I_N\}$ which have been registered by a SfM process such that every picture I_i is associated to a camera

C_i with respective orientation matrix R_i and projection center t_i. Moreover the SfM also supplies a sparse 3D point-cloud $\mathscr{P} = \{p_1, \ldots, p_M\}$ constructed from corresponding image features in the input images.

Given the user supplied path $\mathscr{X}$ defined by a sequence of points $\mathscr{X} = \{x_1, \ldots, x_Q\}$ it is the goal of our algorithm to synthesize a single image, which we call path-image, from the input photographs in which the entire path, i.e. from x_1 to x_Q, is visible and the surroundings remain easily recognizable.

In order to create the path-image, we use $\mathscr{P}$ to estimate a set of planes $\mathscr{A} = \{a_1, \ldots, a_S\}$ that serves as a rough approximation to the real-world geometry. The planes in $\mathscr{A}$ are in general unbounded, i.e. of infinite extent. If such unbounded planes were used as geometric proxies this would result in an unmanageable amount of occlusion errors during image based rendering since planes from very distant parts of the scene easily extent into the current view. It is therefore crucial that bounded proxies are generated in order to obtain approximate visibility information. To this end each input image I_i is partitioned into disjoint regions $I_i = \bigcup_j T_j^i$, where a region T_j^i is assigned as projective texture to a plane $a_j \in \mathscr{A}$ respectively. Thus, each such region results in a bounded, projectively textured 3D-plane $a_j^i \in \mathscr{B}$ which will be used for composing the final path depiction (see Fig. 2). This way occlusion errors are dramatically reduced and it is now feasible to resolve the remaining inaccuracies with our image stitching approach.

Before the path-image is composed, a non-linear space deformation ϕ is applied to the set of bounded proxy planes $\mathscr{B}$. Given the user specified path $\mathscr{X}$ defined by a sequence of points $\mathscr{X} = \{x_1, \ldots, x_Q\}$, ϕ is designed to align the points in $\mathscr{X}$ with the viewing direction, i.e. the Z-axis. The final image I is then composed from the images of the deformed proxies $P(\phi(\mathscr{B})) = \{P(\phi(a_j^i))\}$ under a standard perspective projection P. The composition is based on MRF optimization and selects for each pixel in I the most suitable proxy according to our novel objective function.

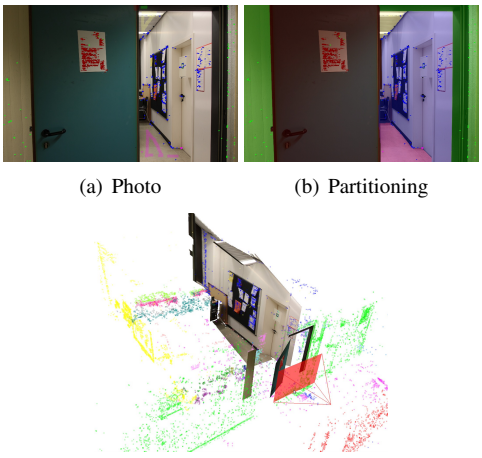
3.1 Preprocessing

We use the publicly available 'Bundler' SfM system [18] for computation of the sparse point-cloud $\mathscr{P}$ and the camera parameters and orientations.

3.2 Proxy Geometry

Given the sparse point-cloud $\mathscr{P}$ our aim is to find a set of planes $\mathscr{A}$ that roughly approximates the underlying geometry. Our algorithm is based on a RANSAC procedure that greedily extracts best fitting planes from $\mathscr{P}$ (see e.g. [15]). To this end the user has to define two parameters: (1) A tolerance value ε that defines the maximal point-to-plane distance allowed for a point to be considered fitted by the plane and (2) a minimum number of fitted points for a plane to be considered valid. Since the user has some knowledge about the scene, e.g. relative object dimensions, it is usually possible to find suitable parameter settings very quickly. The plane detection is fast enough (about a second) so that parameter settings can be evaluated and adjusted interactively. Due to the error tolerance of the following steps the planar approximation need not be very exact, in particular not every point in $\mathscr{P}$ has to be approximated by a planar proxy. Indeed, ignoring clusters of outliers or misaligned parts in $\mathscr{P}$ may often improve the final result.

Thus, after the plane detection $\mathscr{P}$ is partitioned into disjoint point sets $\mathscr{P}_{a_j} \subset \mathscr{P}$ corresponding to the plane $a_j \in \mathscr{A}$ respectively and a set $\mathscr{R}$ of remaining points, i.e. $\mathscr{P} = (\bigcup_{a_j \in \mathscr{A}} \mathscr{P}_{a_j}) \bigcup \mathscr{R}$.



(a) Photo (b) Partitioning

(c) Textured proxies

Figure 2: Semi-automatic partitioning of an input photograph. (a) The input photograph with seed pixels depicted as small colored pixels. Additionally the user has marked some regions to aide the partitioning process (shown as transparently colored pixels). (b) The resulting partitioning. (c) The partitioned imaged parts are projected onto the respective bounded proxies which are visible from the input image. Points in $\mathscr{P}$ corresponding to different planar proxies are rendered in different colors respectively. Please note, that user guidance is usually not required and the partitioning can run fully automatically.

4 INPUT IMAGE PARTITIONING

The proxy geometry $\mathscr{A}$ consists of a set of planes of which each has principally infinite extent. However, in order to generate the final image, the planes need to be bounded. In this work, rather than constructing a globally consistent set of bounded planes, we propose to perform a partitioning of each input image into (necessarily finite) parts corresponding to planes in $\mathscr{A}$. While our approach does neither guarantee a correct partitioning nor consistency between images, such errors are handled later in the image composition optimization. In fact, this simplicity is an important ingredient to the

practicability of our approach as usually quite involved algorithms (which in general nonetheless cannot guarantee a correct solution) are required in order to derive a globally consistent reconstruction.

Since, by principle, every $p \in \mathscr{P}$ originates from featurepoints in at least two different images and every image supports at least five points in $\mathscr{P}$, we can identify subsets $\mathscr{P}^i \subset \mathscr{P}$ of points which originate from featurepoints in image I_i respectively. To find a partitioning $I_i = \bigcup_j T_j^i$ for image I_i the points in $\mathscr{P}^i$ are projected into I_i. The projected locations of points in $\mathscr{P}_{a_j}^i = \mathscr{P}^i \bigcap \mathscr{P}_{a_j}$ serve as seeds for regions T_j^i in I_i corresponding to plane a_j. In order to find the regions we make use of a simple yet surprisingly effective heuristic: Since image edges often correlate with discontinuities in the scene geometry, we try to align the region borders with image edges and assert at the same time that the region T_j^i contains all the projected points from $P_i(\mathscr{P}_{a_j}^i)$. These conditions are easily captured in a MRF-like energy of the following form:

$$E(\mathscr{L}) = \sum_{x \in I} D_x(\mathscr{L}(x)) + \lambda \sum_{(x,y) \in \mathscr{N}} S_{xy}(\mathscr{L}(x), \mathscr{L}(y)) \tag{1}$$

where $\mathscr{L} : I \to \mathscr{A}$ is a labeling assigning planes to pixels, x and y are pixels in I and $\mathscr{N} \subset I \otimes I$ is the set of neighboring pixels in I. The term $D_x(\mathscr{L}(x))$ gives the cost of assigning plane $\mathscr{L}(x)$ to pixel x while the term $S_{xy}(\mathscr{L}(x), \mathscr{L}(y))$ describes the relation between neighboring pixels and gives the cost of simultaneously assigning $\mathscr{L}(x)$ to x and $\mathscr{L}(y)$ to y. The parameter λ introduces a weighting of the two energy-terms. Finding the regions T_j^i then amounts to finding a labeling $\mathscr{L}^*$ that minimizes (1).

For the partitioning D_x is used to introduce the seed locations for the regions, i.e. it encourages assignment of label a_j at projected locations of $\mathscr{P}_{a_j}^i$. Thus we define

$$D_x(a_j) = \begin{cases} 0 & x \notin \bigcup_{a_k \in \mathscr{A}} P(\mathscr{P}_{a_k}^i) \vee x \in P(\mathscr{P}_{a_j}^i) \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

In order to align region boundaries with image edges we set S_{xy} as follows:

$$S_{xy}(a_j, a_k) = \begin{cases} 0 & a_j = a_k \\ g(\|\nabla I\|) & a_j \neq a_k \end{cases} \tag{3}$$

where ∇I is the image gradient and $g(x)$ is a function approaching zero for larger values. In our experiments we use a four-connected neighborhood and a weighting factor of $\lambda = 50$. As in our case $\|\nabla I\|$ only takes values in $[0,1]$, the simple choice of $g(x) = 1.0 - x$ is sufficient.

To find $\mathscr{L}^*$ we employ standard MRF optimization techniques (see e.g. [19] and [3]). Since every image is treated independently the image partitioning can be performed in parallel straightforwardly such that we are able to process even a large number of images in a matter of a few minutes.

The effectiveness of our approach is best understood in conjunction with the nature of the SfM process. Since $\mathscr{P}$ is derived from image feature correspondences it naturally contains more points in textured areas or near image discontinuities. Thus, while our simple heuristic would generally fail in textured areas due to the abundance of image edges, the additional seed points constrain the solution in a sensible manner. Conversely, in uniform image regions the number of seed points is low, but so is the number of image edges and these edges are much more likely to coincide with actual discontinuities in the geometry. Additionally we give the user the opportunity to influence the partitioning by marking regions of images that should belong to a specified proxy plane a_j, see Fig. 2. The marked regions are simply interpreted as additional seeds by our system.

After the partitioning we create the final geometric proxies on a per-image basis. Each region T_j^i is triangulated in the image domain and projected onto a_j to give a bounded planar approximation $a_j^i \in \mathscr{B}$ which is projectively textured from I_i.

5 SPACE DEFORMATION

In this work we employ the space deformation introduced by Degener et al. [7] which is responsible for the straightening of the path while also ensuring visibility of the surrounding geometry.

The space deformation ϕ is derived from the user specified path-points $\mathscr{X} = \{x_1, \ldots, x_Q\}$ such that $\phi(\mathscr{X}) = \{\phi(x_i), \ldots, \phi(x_Q)\}$ lie along the negative z-axis, i.e. the viewing direction of the final image.

These conditions alone however do not yet suffice to specify all the important attributes of ϕ for creation of a single image path depiction. Therefore every path-point is also equipped with four auxiliary points which are used for specifying up- and right- direction as well as a scaling factor.

Finally a thin-plate spline [8] is used to define the globally smooth space deformation ϕ that adheres to the user specified constraints. In our implementation we derive an initial set of appropriate constraints automatically and give the user the opportunity to interactively adjust these in two ways:
First, the user has the possibility to change the spacing between transformed path points, to adjust the amount of image space a certain segment of the path occupies in the warped image. This can be used to reduce the space needed for long uninterrupted sections of the path in favor of complicated junctions. Furthermore, in many indoor situations, it might also be reasonable to adjust the scaling defined by the auxiliary points, e.g. to widen a narrow doorway along the path, providing a better view on the space behind.

6 IMAGE COMPOSITION

The final image I is composed from the deformed, bounded proxies $\phi(\mathscr{B}) = \{\phi(a_j^i) \mid a_j^i \in \mathscr{B}\}$. First, for each such proxy an image $I_j'^i = P(\phi(a_j^i))$ is rendered under the same projection P which is also used for the composed image I. It is these images which will be stitched in order to form the final path depiction.

The stitching is formulated as MRF energy minimization as in (1) [2][1]. In order to achieve visually appealing results we identify the following properties that the final image should possess [4]:

Visibility in the composed image should be consistent within in the image itself, i.e. a chair in front of a wall should not be partially occluded by the wall. Moreover visibility in the composed image should mimic visibility in the real world, i.e. if there is a chair in front of a wall in the real scene it should also appear in the composed image.

Angular deviation between the viewing ray of a pixel in the final image and the line of sight under which the photograph used for filling that pixel was taken should be minimal. A small angle reduces parallax artifacts and therefore increases visual quality of the stitched image.

Continuity between neighboring pixels in the composed image. That is neighboring pixels in the composed image should have similar colors as neighboring pixels in the original input images. This ensures recognizability of image features in the composed result.

Resolution sensitivity means that the image should be composed from those input photographs that possess the above properties and at the same time give the highest resolution when projected into the final composite.

All these aspects are captured in our MRF energy formulation that follows the pattern of (1). For the stitching, the labeling $\mathscr{L} : I \to P(\phi(\mathscr{B}))$ now assigns each pixel one of the projected proxies as color source. The terms D_x and S_{xy} have to be redefined to reflect the above desired properties. D_x will be responsible for visibility, angular deviation and resolution sensitivity while S_{xy} is used to encourage continuity. Thus we set D_x as

$$D_x(I_j'^i) = \alpha V_x(I_j'^i) + \beta A_x(a_j^i) + \gamma R_x(a_j^i) \qquad (4)$$

where $V_x(I_j'^i)$ is the depth value of $I_j'^i$ at pixel x and therefore rewards correct visibility as proxies closer to the eye receive lower cost. A_x is the term for controlling angular deviation of viewing rays from camera line of sight and R_x penalizes low resolution.

Compared to previous approaches such as [4] our setting differs in that the proxy geometry is subject to a space deformation ϕ. As a result, camera line of sights are effectively bent after application of the deformation. Therefore, in order to compare the direction under which a proxy is viewed after the deformation to the direction of the original camera line of sight we also need to apply ϕ to the direction of the line of sight (see Figure 3. Let $\vec{v}_x$, $\|\vec{v}_x\| = 1$ denote the direction of the viewing ray of pixel x which intersects $\phi(a_j^i)$ in p_x and let $s(t) = c_i + t\vec{r}$ be the line of sight under which $p_x' = \phi^{-1}(p_x)$ is observed by camera i. Then $\frac{d\phi}{dt}(s(\cdot)) = d\phi(\cdot)\vec{r}$ is the direction with which the bent line of sight intersects $\phi(a_j^i)$. Thus we let

$$A_x(a_j^i) = 1 - \alpha \max(0, \vec{v}_x' d\phi(p_x')\vec{r}) \qquad (5)$$

with normalizing factor $\alpha = 1/\|d\phi(p_x')\vec{r}\|$.

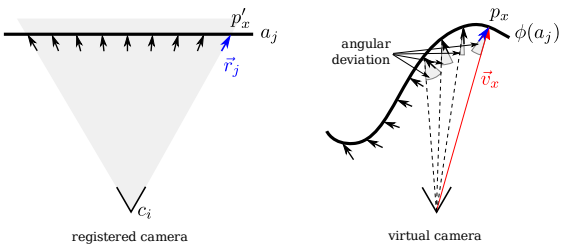


Figure 3: The angular deviation criterion under a space deformation ϕ. On the left hand side, the unbounded proxy plane a_j and the camera of image I_i are shown, as well as one example viewing ray $s(t) = c_i + t\vec{r}$ highlighted in blue. On the right hand side, the deformed proxy geometry $\phi(a_j)$ is depicted. One viewing ray of the virtual camera with view direction $\vec{v}_x$, used to project the deformed proxy, is highlighted in red. At the intersection point p_x between $\phi(a_j)$ and the viewing ray, the angular deviation is indicated as the angle between the viewing ray direction $\vec{v}_x$ and the deformed viewing direction $d\phi(p_x')\vec{r}$.

The last term R_x prefers proxies with a texture sampled in a resolution similar to that of the target image. Thus the sampling rate has to be determined in the input photograph as well as in the target image. Let x' be the pixel of I_i in which camera i observed $p_x' = \phi^{-1}(p_x)$ and let p_y' be the point of a_j^i which was observed in the neighboring pixel $y' \in I_i$. Then we define

$$R_x(a_j^i) = (\|p_x' - p_y'\| - \|p_x - p_y\|)^2 \qquad (6)$$

Finally the term S_{xy} is given as

$$S_{xy}(I_j'^i, I_l'^k) = \sum_{u \in \{\mathscr{N}_x \cap \mathscr{N}_y\}} \|I_j'^i(u) - I_l'^k(u)\|^2 \qquad (7)$$

where $\mathscr{N}_x$ is the neighborhood of pixel x, including x itself. S_{xy} computes the sum of squared distances between the corresponding neighbor pixel values in $I_j'^i$ and $I_l'^k$. This penalizes compositions with visible seams between different image patches.

User interaction

Using the D_x term it is possible to easily integrate user specified constraints into the optimization. In our system the user is able to encourage or discourage the use

of a given image in certain parts of the target image. This is realized by interactively marking the respective regions of the target image similarly to the partitioning interface.

6.1 Optimization

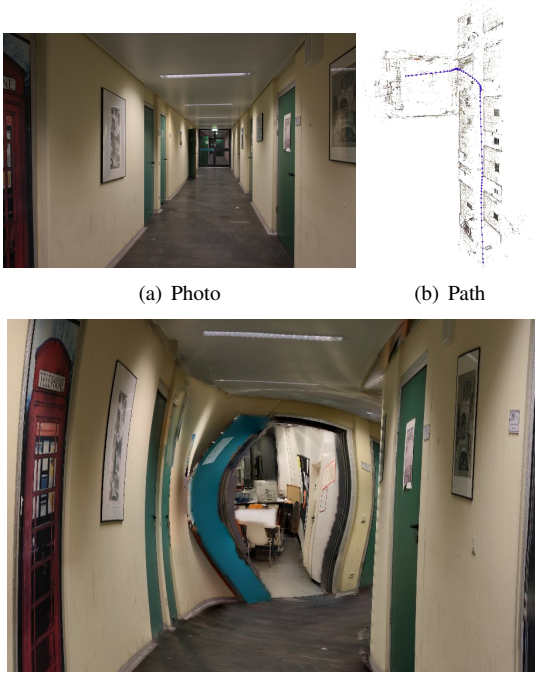
Two Step Stitching

To find a labeling $\mathscr{L}^*$ for the composition we need to minimize (1) where D_x and S_{xy} are defined as stated above. The algorithms we employed to this end in Sec. 4 are efficient if the number of labels is relatively small, i.e. a few hundred. However, in case of the image composition the number of labels can be very large, i.e. $|\mathscr{B}| = O(|\mathscr{I}||\mathscr{A}|)$. Therefore we propose the following two-step procedure to find the final image composite:
(1) We compose a single image I'_j for each proxy plane $a_j \in \mathscr{A}$ from all the I'^i_j using the energy described above.
(2) Then we compose these I'_j into the final result I, using the same energy formulation. The data-cost-terms for single bounded-planar-proxies a^i_j, however, are now replaced by a look-up using the composition computed in step 1.
This dramatically reduces the number of labels in each step. In the first stage there are only $O(|\mathscr{I}|)$ labels while there are exactly $|\mathscr{A}|$ labels in the second phase. Of course the final result is usually not the same as obtained when directly minimizing the full problem, but we found the results to be visually equivalent in our experiments. The reason for this is that the set of images I'^i_j usually is highly redundant due to the overlap between the input photographs I_i and therefore a large subset of the possible labelings in the full optimization actually produce very similar results. In our approach on the other hand this redundancy is eliminated in the first step so that the second step only has to resolve ambiguities between the different proxies.

Image Shifting

Since the scene geometry is only approximated by a coarse proxy geometry, parallax artifacts are to be expected. In order to deal with these artifacts we, first of all, identified the need to formulate an angular deviation constraint to minimize these artifacts, but we also extend the stitching algorithm to shift the input images $I \in \mathscr{I}$ by some pixels to further eliminate parallax effects. This extension can easily be implement by simply adding shifted versions $P(\phi(a^i_j))'_{x,y} = P(\phi(a^i_j))_{x+s_x,y+s_y}$ of the projected proxy geometry to the input of the stitching algorithm. The allowed shift $(s_x, s_y) \in \{(x,y) \in \mathbb{N}^2 : |x| \leq \sigma \wedge |y| \leq \sigma\}$ is specified by the parameter σ. Higher values of σ allow better compensation for bigger parallax artifacts. This, however, entails a significant increase in optimization times, since the number of labels in the optimization grows quadratically with σ.

Because only small values of σ can be used in practice, the angular deviation costs, which in theory vary for different shifted versions of a projection, do not need to be recomputed, as the changes in the view direction are negligibly small for differences of only a few pixels.



(a) Photo (b) Path

(c) Image

Figure 4: A path through a corridor into a kitchen. (a) A photograph taken from the starting location. (b) The user specifies the path in the sparse 3D point-cloud (path points depicted in blue). (c) The final image composition after the space deformation.

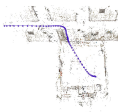
Gradient Domain Fusion

In order to further reduce visible discontinuities between different patches, which might occur for example due to differences in material appearance under varying viewing angles, we deploy a Poisson image editing technique on the final image, which in the context of image stitching is also referred to as *gradient domain fusion* by Agarwala et al.. For details on this optimization, please see [2].

7 RESULTS

We have applied our method to two different scenarios. In Fig. 1 we show a path through a supermarket. The input data consists of 378 images taken with a handheld camera in about 25 minutes. Care has been taken to ensure sufficient overlap between images as required for SfM. Apart from that no further acquisition planning was necessary. The reconstructed 3D points are

(a) Photo



(b) Path

(c) Automatic result

(d) User corrected image

Figure 5: A path from a kitchen into a corridor. (a) A photograph taken from the starting location. (b) The user specified path. (c) The result generated by our approach fully automatically. (d) The final result after a few user corrections.

(a) Photo

(b) Photo

(c) Image

Figure 6: An overview visualization of a museum's foyer. (a) The foyer is too large to fit into a single photograph. (b) Our rendering on the other hand shows all the exhibits and adjoining corridors in the room.

depicted in Fig. 1 middle together with the visualized path shown in blue. A total of 69 planar proxies were automatically extracted from these points. In the partitioning stage it was occasionally necessary to manually mark floor and ceiling due to lack of feature points in these low texture regions (roughly 20% of the input images). The final stitching result was obtained without further user interaction. As shown in the right of Fig. 1 the resulting image gives a good overview of the path clearly depicting features along the way which serve as landmarks during navigation. Please note that warped path-images are not intended to convey spatial survey knowledge but rather unfold their true power if the sequence of visible features matches the observer's immediate surroundings. A few inconsistencies due to geometric approximation errors become apparent under close examination. However, these minor artifacts are perceptually insignificant and do not compromise the visualization's purpose.

The corridor dataset depicted in Figure 4 and 5 was taken in an office environment which exhibits far less natural salient image features. Since features are fundamental for SfM, a slightly increased number of images was necessary. In total 400 images were taken with a hand held camera in approximately 30 minutes. From the sparse point cloud a total of 29 planar proxies were extracted. For the path emanating from the kitchen shown in Fig. 5 we applied a manual correction in the composition step to resolve occlusion problems: In this way the door was removed to give a better view into the corridor. Moreover a false occlusion due to the lack of seed points on the homogenous table surface in the front was corrected (see Figure 4 (c) and (d)).

Finally, the museum dataset (see Fig. 6) consists of 91 pictures and 13 proxy planes. This image's primary purpose is to give an overview of the museums foyer with special emphasis on the adjacent corridors. Again it was necessary to occasionally give some manual hints in the partitioning phase on the featureless white walls, but the respective planes were automatically extracted. Note that the smudgy brown area on the lower left in the image is due to unobserved areas in the input dataset.

In our approach most of the processing time is spent in the preprocessing step: The SfM step took roughly 5 hours for 400 images. Apart from preprocessing, the runtime of our method is dominated by the first step in the optimization described in Section 6.1: For all data sets it took about 15 minutes to compose textures on all proxies. Most time was spent on the floor and ceiling planes which are visible in many images (4.4 minutes on average). The final composition step took only 1 in the corridor and museum scenarios and 1.8 minutes for the supermarket dataset at an output resolution of 900×600 pixels. The times for input image partitioning (4 seconds on average) and warping are negligible.

8 CONCLUSION

This paper presents a novel approach to creating high-quality single image visualizations of short, possibly curved, routes from a series of input photographs. To accomplish this goal we introduce a novel image-based rendering technique able to deal with space deformations. It can be seen as a hybrid between unstructured lightfield rendering [4], panorama and image stitching [1, 2] as well as exploration of image collections [17]. We propose the use of simple proxy geometry estimated from a sparse set of 3D points obtained by SfM. Our proxies give only a rough approximation and are explicitly allowed to deviate from the actual scene geometry. A global energy minimization in the image domain stitches the deformed proxies' images and effectively handles parallax and occlusion artifacts and thus allows for very loose fitting proxies. The synthesized images are valuable and inexpensive navigation aids that can be handed out to visitors in public places. Moreover, we believe that beyond functional aspects, warped images also possess a high aesthetic value.

Our method is currently restricted to scenes predominated by piecewise planar geometry. While this is met by most indoor and urban environments, our method is not well suited if the scene geometry is more complex and cannot be reasonably approximated by planes, e.g. in a forest.

ACKNOWLEDGEMENTS

The authors would like to express their gratitude towards the supermarket *Comet Verbrauchermarkt* and the museum *Rheinisches Landesmuseum* in Bonn, Germany.

REFERENCES

[1] A. Agarwala, M. Agrawala, M. Cohen, D. Salesin, and R. Szeliski. Photographing long scenes with multi-viewpoint panoramas. In *SIGGRAPH*, pages 853–861, 2006.

[2] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen. Interactive digital photomontage. In *SIGGRAPH*, pages 294–302, 2004.

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001.

[4] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *SIGGRAPH*, pages 425–432, 2001.

[5] S. E. Chen and L. Williams. View interpolation for image synthesis. In *SIGGRAPH*, pages 279–288, 1993.

[6] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *SIGGRAPH*, pages 11–20, 1996.

[7] P. Degener, R. Schnabel, C. Schwartz, and R. Klein. Effective visualization of short routes. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1452–1458, 2008.

[8] J. Duchon. Spline minimizing rotation-invariant semi-norms in sobolev spaces. In *Constructive Theory of Functions of Several Variables*, volume 571 of *Lecture Notes in Mathematics*, pages 85–100, 1977.

[9] M. Eisemann, B. D. Decker, M. Magnor, P. Bekaert, E. de Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. Floating Textures. *Proc. Eurographics*, 27(2):409–418, 4 2008.

[10] S. Gortler and R. Grzeszczuk. The lumigraph. *SIGGRAPH*, pages 43–54, 1996.

[11] M. Levoy and P. Hanrahan. Light Field Rendering. In *SIGGRAPH*, pages 31–42, 1996.

[12] L. McMillan and G. Bishop. Plenoptic modeling: an image-based rendering system. In *SIGGRAPH*, pages 39–46, 1995.

[13] M. Pollefeys, L. J. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, 2004.

[14] C. C. Presson. The development of map-reading skills. *Child Development*, 53(1):196–199, 1982.

[15] R. Schnabel, R. Wahl, and R. Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, June 2007.

[16] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski. Finding paths through the world's photos. In *SIGGRAPH*, pages 1–11, 2008.

[17] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *SIGGRAPH*, pages 835–846, 2006.

[18] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *Int. J. Comput. Vision*, 80(2):189–210, 2008.

[19] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(6):1068–1080, 2008.

[20] M. Vergauwen and L. Gool. Web-based 3D reconstruction service. *Machine Vision and Applications*, 17(6):411–426, 2006.

Divergence Analysis of Discrete 2-d Shapes

David Brunner and Guido Brunnett
Chemnitz University of Technology
Germany
{brunner, brunnett}@informatik.tu-chemnitz.de

ABSTRACT

In this paper we introduce a second order differential operator in order to partition discrete objects into meaningful parts. One of these parts contains all points of the object's medial axis which is a widely used shape descriptor.

Since aliasing is a fundamental problem for methods based on the boundary of discrete objects we use a bilateral filter to reduce the artefacts significantly.

As a result of the partitioning and filtering we obtain a lattice point set (a superset of the medial axis) which is robust against noise and aliasing and which is rotationally invariant. The lattice point set can be used for instance to compute the medial axis or similar shape descriptors.

Keywords: 2-d shape descriptor, bilateral filter, skeletonization, curve skeletons

1 INTRODUCTION

One typical goal for processing a discrete 2-d shape $\mathscr{S}$ is finding a good shape descriptor. One possible descriptor of $\mathscr{S}$ is the medial axis which is defined as the set of centres (and radii) of maximal disks in $\mathscr{S}$, whereas a disk is said to be maximal in $\mathscr{S}$, if it is not completely covered by any other disk in $\mathscr{S}$ (cf. [1, 12]). Such descriptors are widely used in many applications such as shape analysis, shape representation, skeletonization, segmentation, and shape matching (cf. [10, 5, 9]).

In this paper we use the divergence of a scalar distance field in order to describe the shape's local geometry. We show that only points located at the medial axis have a high negative divergence. Of course these points could be used in the computation of the medial axis as the contributions of Siddiqi et al. show (cf. [15, 2, 4]). On the other hand, points at concave boundary regions have a positive divergence and may be used in a segmentation approach.

But this property is only helpful for objects with a smooth boundary and therefore aliasing is a fundamental problem. Boundary based approaches are notoriously instable to small boundary perturbations. Hence, the results of corresponding algorithms such as for the medial axis are not rotationally invariant. Many methods follow the concept of reducing the influence of aliasing using some sort of heuristic. In the context of skeleton extraction pruning approaches are used for such a reducing effect [16, 7, 14].

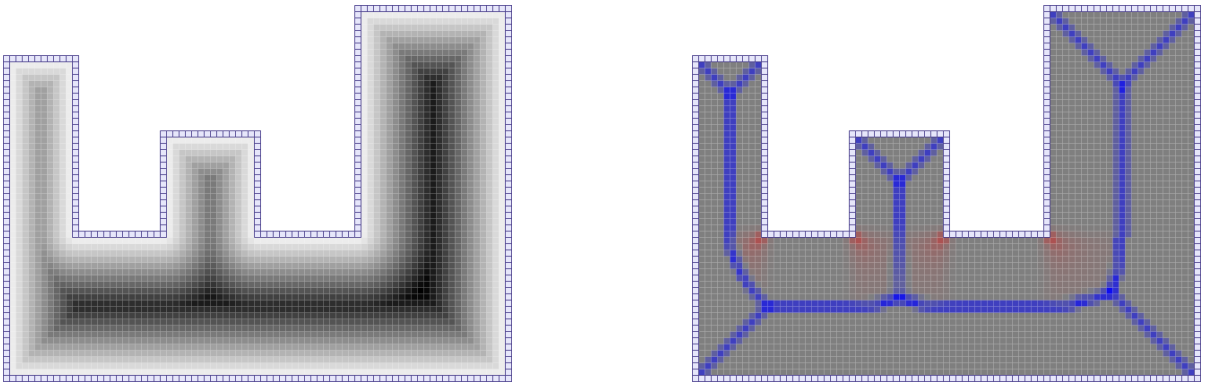
In contrast to such heuristics, we suggest a significant reduction on the basis of a bilateral filter (whose theoretical foundations have been widely developed, recently). The bilateral filter has its origin in [17] and is typically used in applications such as denoising of images [19] or meshes [6]. Practically, the filter can be applied at interactive speed thanks to efficient numerical schemes [18]. A good survey of this field can be found in [11].

Since the points with a high negative divergence are of interest for a shape descriptor we utilize the bilateral filter to emphasize the respective regions. We show that the filtered point set is suitable to construct a shape descriptor which is rotationally invariant and very robust against boundary perturbations.

This paper is structured as follows: After introducing some mathematical basics and notations in section 2 we explain an adapted Laplace operator and the partitioning in section 3. In section 4 we analyse artefacts which are caused by the aliasing and describe the bilateral filter which is used to reduce these artefacts significantly. To demonstrate the qualities and effectiveness of the introduced operator and filter we present in section 5 as an exemplary application a skeletonization approach which uses the filtered point set to construct a shape descriptor. The paper is concluded with our results in section 6.

2 BASIC DEFINITIONS AND NOTATIONS

We use the Cartesian point lattice $\mathbb{Z}^2$ as the mathematical model for describing a tessellation of the 2-d Euclidean space $\mathbb{E}^2$ into cell elements. The coordinates of a lattice point $p \in \mathbb{Z}^2$ are denoted by p_x, p_y. For the



(a) (b)

Figure 1: In (a) the distance field and (b) the divergence are shown. Gray cells in (b) have zero, blue cells have a negative and red cells a positive divergence.

Cartesian lattice we define the local neighborhood of size k for lattice points by

$$N_k(p) = \left\{ q \in \mathbb{Z}^2 : \max\left(|p_x - q_x|, |p_y - q_y|\right) = k \right\} \quad (1)$$

which is based on the chessboard metric.

A discretized object O is defined as a connected finite set of lattice points $\mathbb{Z}^2$. Using the notion of a binary image it is common to say that each point of O belongs to the foreground while the points of $\overline{O} = \mathbb{Z}^2 \setminus O$ form the background.

The border of a discretized object O is defined by the set O_B of all points $p \in O$ whose local neighborhood $N_1(p)$ contains at least one point q with $q \in \overline{O}$.

3 OBJECT'S PARTITIONING

For further analysis of a discrete object O we want to partition it into three disjoint parts. The first part O_I is some kind of scaffold of the object and is similar to the medial axis. The second part O_{II} contains points which are influenced by a concave boundary. The remaining third part O_{III} is not of interest because it contains no applicable information.

Due to partition objects we start with a distance mapping. Therefore, we use a distance transform which is an efficient standard procedure to compute the minimal Euclidean distance of each object point to the background (cf. [3, 13]). Figure 1(a) shows an example.

We change this definition a little by defining the minimal Euclidean distance $d_{\min}$ of $p \in O \cup \overline{O}$ to the boundary O_B instead to the background:

$$d_{\min}(p, O_B) = \min\left(\|p,q\| : q \in O_B\right) \quad (2)$$

where $\|p,q\|$ denotes the Euclidean distance between two lattice points. We define the final distance mapping

$$d(p) = \begin{cases} -d_{\min}(p, O_B): & p \in \overline{O} \\ 0: & p \in O_B \\ d_{\min}(p, O_B): & p \in O \end{cases}. \quad (3)$$

We need to define negative distances for background points in order to make it possible to have a suitable second derivative for boundary points, too. Otherwise, all boundary points would have positive divergence and thus concave boundary would not be recognizable (since all boundary points would belong to O_{II}).

After the distance transform, we use an adapted Laplace operator to compute the divergence of distance map. The Laplace operator is a second order differential operator in $\mathbb{E}^2$ and in our case defined as the divergence of the gradient of the distance field.

The gradient of the distance field is defined to be the vector field whose components are the partial derivatives of the distance field. The gradient of a point $p = (p_x, p_y)^T \in O$ is defined by:

$$\nabla d(p) = \left(\frac{\partial d(p)}{\partial x}, \frac{\partial d(p)}{\partial y} \right)^T. \quad (4)$$

which we will compute by a finite difference since we use a discrete scalar field:

$$\nabla d(p) = \frac{1}{2} \begin{pmatrix} d((p_x+1, p_y)^T) - d((p_x-1, p_y)^T) \\ d((p_x, p_y+1)^T) - d((p_x, p_y-1)^T) \end{pmatrix}. \quad (5)$$

Finally, we compute the divergence of the vector field to measure the magnitude of the field's sink at a given point (see figure 1(b)). In general, the divergence of a differentiable vector field is defined as the sum of the partial derivatives of the i-th component:

$$\Delta d(p) = \nabla^2 d(p) = \frac{\partial^2 d(p)}{\partial x^2} + \frac{\partial^2 d(p)}{\partial y^2}. \quad (6)$$

Siddiqi et al. state in [15] that this definition cannot be used since the gradient of the distance field is not free of singularities and hence is not differentiable. Therefore, they suggest a numerically stabble solution, which we will adapt, such that $\Delta d(p) \mapsto [-1, 1]$:

$$\Delta d(p) = \frac{1}{|N_1(p)|} \sum_{q \in N_1(p)} \angle(q - p, \nabla d(q)) \quad (7)$$

where $|N_1(p)|$ is the number of local neighbor points, 8 in the 2-d case, and where $\angle \mapsto [-1,1]$ is a function which computes the cosine of angle α between two vectors $\vec{u},\vec{v}$ taking account of zero vectors:

$$\cos(\alpha) = \angle(\vec{u},\vec{v}) = \begin{cases} 0 & : \vec{u} \equiv \vec{0} \vee \vec{v} \equiv \vec{0} \\ \dfrac{\langle \vec{u},\vec{v} \rangle}{\|\vec{u}\| \, \|\vec{v}\|} & : \text{else} \end{cases} . \quad (8)$$

Now we analyse the divergence values for different point classes. Let $D(p) \subset O$ be the set of points of the maximal disc in O with p as disc center. That is, $D(p)$ describes the disc with the largest radius of all possible discs at p. Further, let

$$B(p) = D(p) \cap O_B \quad (9)$$

be the set of all border points in $D(p)$.

We are not only interested in the border points of $B(p)$. Since the divergence of a point p (defined in eq. (7)) depends on its local neighbors $N_1(p)$, the closest boundary points of those neighors are of interest, too. In order to analyse the divergence value at p we need to consider the gradients of the point set

$$\mathscr{B}(p) = \bigcup_{q \in N_1(p)} B(q). \quad (10)$$

Note that

$$B(p) \subset \bigcup_{q \in N_1(p)} B(q). \quad (11)$$

We observe for points p located at the medial axis the set $\mathscr{B}(p)$ contains points belonging to different boundary segments. For all other points p outside the medial axis $\mathscr{B}(p)$ contains only border points located at exactly one boundary segment. In figure 2 the points p_1, p_2 are and p_3, p_4 are not part of the medial axis.

Obviously, $\Delta(p)$ correlates with the gradients of the points of $\mathscr{B}(p)$. For points in $\mathscr{B}(p)$ belonging to different boundary segments, the gradients of $q \in N_1(p)$ are oriented towards p and vary strongly. Thus, $\Delta(p)$ is negative (see points p_1, p_2 in figure 2). In contrast, for points p having in $\mathscr{B}(p)$ only border points of one boundary segment, $\Delta(p)$ may be either zero for points $b \in \mathscr{B}(p)$ having the same gradient (see point p_3 in figure 2) or may be positive if the points $b \in \mathscr{B}(p)$ describe a concave boundary segment (see point p_4 in figure 2) and some of the gradients of the points $q \in N_1(p_4)$ are oriented away from p_4.

This observation is now used to classify the points into three sets so that a given object can be partioned according to the divergence values. We define (using $\dot\cup$ (instead of $\cup$) to indicate a union of disjoint sets)

$$O = O_I \,\dot\cup\, O_{II} \,\dot\cup\, O_{III}, \quad (12)$$

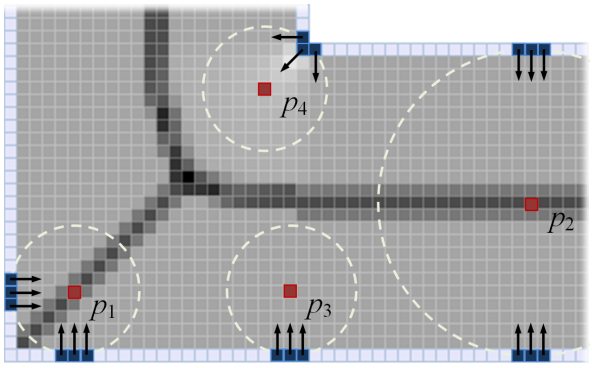


Figure 2: The divergence at different points (red) is influenced by the respective closest border points (dark blue). Dark cells representing a negative, light cells a positive divergence. The dotted circles indicate the discs $D(p_1) - D(p_4)$.

with

$$O_I = \{p : \Delta(p) < 0\}, \quad (13)$$
$$O_{II} = \{p : \Delta(p) > 0\}, \quad (14)$$
$$O_{III} = \{p : \Delta(p) = 0\}. \quad (15)$$

Fig. 1(b) illustrates the partitioning for an object with a smooth boundary (i.e. without aliasing). The blue points belong to O_I, the red points to O_{II}, and the light gray points to O_{III}.

The possible uses of these subsets of an object O are various: The set O_I contains points, which are used for example to compute the medial axis of an object. The points contained in O_{II} may be used to support a decomposition approach, since objects are often decomposed at concave regions. The points of O_{III} can be ignored depending on the particular object analysis approach, resulting in reduced computational costs.

4 FILTERING ALIASING BASED ARTEFACTS

The object shown in figure 1 contains no aliasing based artefacts, because all border segments are axis aligned. Aliasing effects occur when the object in figure 1 is rotated or noise is added and surely when an object contains non-axis aligned edges (see figure 4). The partition of O based on eq. (12) is unclear in such cases.

Obviously, the noise or aliasing causes the appearance of stripe-shaped regions of positive or negative divergence (see figure 3(a) and 3(b)). In a first filtering step the gradient field is smoothed to reduce the effects partly. Afterwards, a second filtering step is applied to the divergence field. For both filtering steps the bilateral filter is the appropriate method for removing the mentioned artefacts. This filter technique has the advantage to smooth image data while preserving its discontinuities.

In our case we intend to smooth small gradient and divergence variances while preserving the greater variances. For this purpose we adapt the well-known bilateral filter. In 2-d the Gaussian kernel is given by

$$g_\sigma(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right). \tag{16}$$

Using this kernel we define the bilateral filter f_{grad} for lattice points p and two parameters $\sigma_d \in \mathbb{N}$, $\sigma_\nabla \in \mathbb{R}$ by:

$$f_{\text{grad}}(p) = \frac{1}{w(p)} \sum_{q \in N_{2\sigma_d}(p)} g_{\sigma_d}(\|p-q\|) g_{\sigma_\nabla}(a(p,q)) \nabla(q) \tag{17}$$

where $N_{2\sigma_d}(p)$ is the local neighborhood at p of size $2\sigma_d$ (see eq. (1)) and $w(p)$ is a normalization factor with

$$w(p) = \sum_{q \in N_{2\sigma_d}(p)} g_{\sigma_d}(\|p-q\|) g_{\sigma_\nabla}(a(p,q)). \tag{18}$$

The function $a(p,q) \mapsto [0,1]$ together with eq. (8) is defined by

$$a(p,q) = \left(\frac{1 - \angle(\nabla(p), \nabla(q))}{2}\right) \tag{19}$$

and expresses the difference between the gradients of p and q.

The amount of filtering is influenced by the parameters σ_d and σ_∇. On the one hand σ_d influences the size of the local neighborhood at p. For instance let $\sigma_d = 1$ then $N_{2\sigma_d}(p)$ defines a 5×5-mask. On the other hand the weight g_{σ_d} has the intention to decrease the influence of distant points. The second parameter g_{σ_∇} intends to decrease the influence of p's neighboring points q with a differing gradient in comparison to $\nabla(p)$.

We denote the filtered gradient at point p with $\dot{\nabla}(p)$ and the divergence based on the filtered gradients with $\dot{\Delta}(p)$:

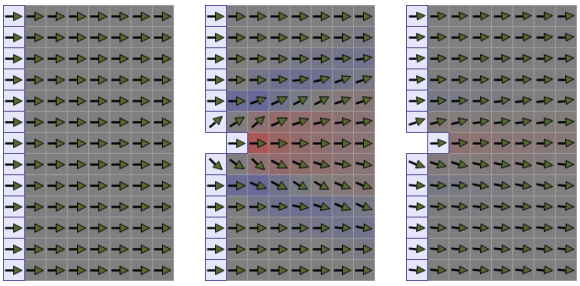
$$\dot{\Delta}d(p) = \frac{1}{|N_1(p)|} \sum_{q \in N_1(p)} \angle(q-p, \dot{\nabla}d(q)). \tag{20}$$

See figure 3(b) and 3(c) to comprehend the filtering effect.

After applying the bilateral filter to the gradients, the divergence variation is small for neighboring points $p,q \in O_I$ in comparison to the variance between neighboring points $p \in O_I$ and $q \in O_{III}$. With this being the case, again a bilateral filtering is suitable in order to further reduce the artefacts.

Therefore, in eq. (17) the weight g_{σ_∇} is replaced by the weight g_{σ_Δ} and we get:

$$f_{\text{div}}(p) = \frac{1}{w(p)} \sum_{q \in N_{2\sigma_d}(p)} g_{\sigma_d}(\|p-q\|) g_{\sigma_\Delta}(\dot{\Delta}(p) - \dot{\Delta}(q)) \dot{\Delta}(q) \tag{21}$$



(a) (b) (c)

Figure 3: Gradient and divergences for (a) a smooth boundary, (b) with added noise, and (c) after gradient filtering.

with $w(p)$ again as the sum of all weigths. The weight g_{σ_Δ} intends to decrease the influence of p's neighboring points q with a divergence value different from $\dot{\Delta}(p)$.

We denote the filtered divergence value at point p with $\ddot{\Delta}(p)$ and redefine the decomposition of O according to the filtered divergence field:

$$O = \ddot{O}_I \cup \ddot{O}_{II} \cup \ddot{O}_{III}, \tag{22}$$

with

$$\ddot{O}_I = \left\{p : \ddot{\Delta}(p) < 0\right\}, \tag{23}$$

$$\ddot{O}_{II} = \left\{p : \ddot{\Delta}(p) > 0\right\}, \tag{24}$$

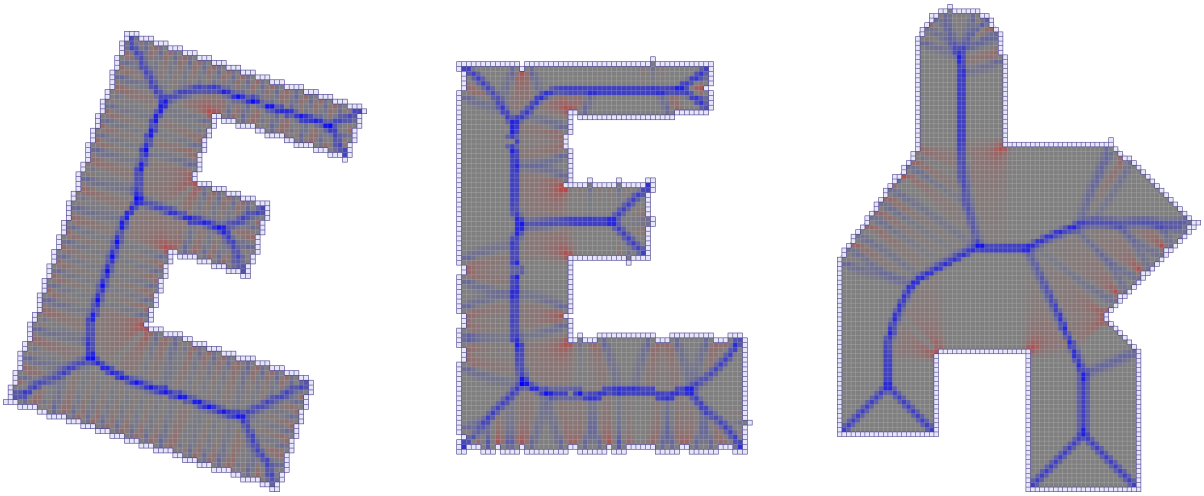
$$\ddot{O}_{III} = \left\{p : \ddot{\Delta}(p) = 0\right\}. \tag{25}$$

Note, that filtering the distance map itself wouldn not be suitable, since the variances of the distance values are only marginal and significant features would not preserved.

5 SAMPLE APPLICATION

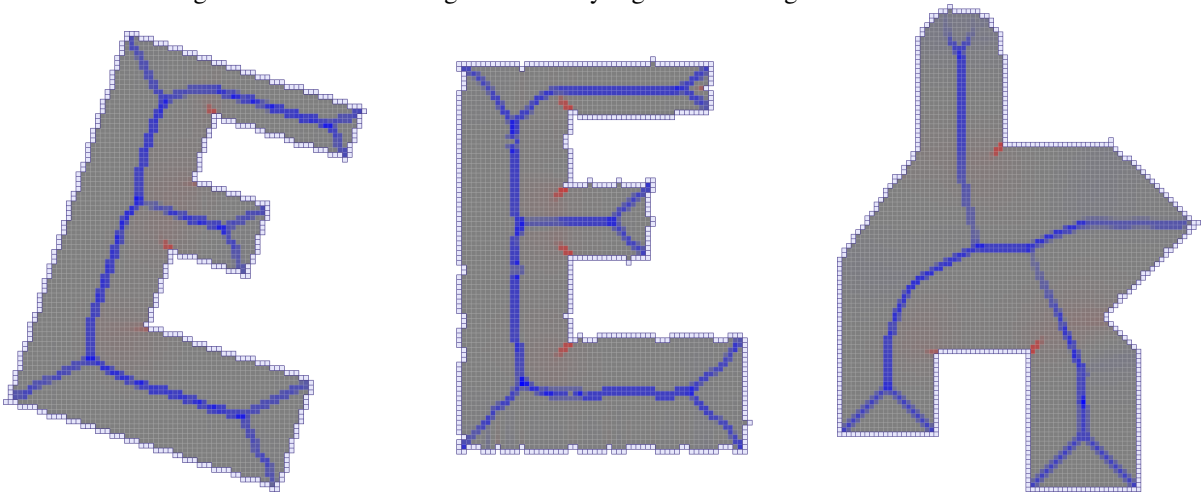
In this chapter we present a skeletonization approach on the basis of the findings of the previous chapters in order to evaluate our shape descriptor. Therefore we use a thinning method which offers two important advantages. It is fast and furthermore the only known method where the topological equivalence of the skeleton with the initial discretized object can be guaranteed [8] (the definition of topological equivalence is based on the same number of connected components, tunnels, and cavities). Thinning algorithms iteratively remove border points (i.e. points with at least one neighbor point belonging to the background) from a discretized object. It is common to call a removable point *simple* if its removal does not change the topology of the discretized object ([8]).

We extend the standard thinning algorithm by using the filtered divergence values. This is done firstly, by sorting the removable points according to the filtered divergence values, such that points with high positive divergence are prioritized and points with high negative



(a) (b) (c)

Figure 4: For non-axis aligned boundary segments aliasing based artefacts occur.



(a) (b) (c)

Figure 5: Filtering results for $\sigma_d = 2$, $\sigma_\nabla = 0.05$, and $\sigma_\Delta = 0.01$ are shown.

divergence are considered at last (as suggested in [4]). Secondly, a divergence-based threshold t is used to prevent the removal of points with a high negative divergence. The resulting algorithm is structured as follows:

```
do {
   something changed = false;
   bp = determine all border points;
   sort bp; // desc. divergence values

   for each p in bp {
      if (p is simple) and (Δ̈(p) ≥ t) {
         remove p
         something changed = true;
      }
   }
} while something changed
```

After applying the sketched algorithm the resulting skeletons are not necessarily thin. The thinning algorithm is once again applied to the remaining point set with changed conditions. The threshold-based condition is replaced by the curve-end-point-condition, which states that a point with only one foreground-neighbor must not be removed:

```
do {
   ⋮
   for each p in bp {
      if (p is simple) and (|N1(p)| > 1) {
         remove p
         something changed = true;
      }
   }
} while something changed
```

Sorting the border points by divergence changes the order of point removal. By removing a point p the removability of its neighbors may be changed. That is, a point q may not be removed only because its neighbor (which has a higher divergence) p is removed before considering q.

The resulting skeletons provide very positive properties: they are efficient to compute, topological equivalent to the original object, centered, rotationally invariant, and robust against noise (see figure 6). Furthermore, the divergence-based threshold t allows the generation of hierarchical skeletons – which is a very rare possibility with regard to the wide range of available skeletonization methods. See figure 7 for a sample.

6 RESULTS AND CONCLUSION

It is clearly recognizable that bilateral filtering of the divergence values reduces the aliasing based artefacts substantially. As a beneficial consequence, the point set $\ddot{O}_I$ is rotationally invariant and robust against noise. For this reason, operators or algorithms (such as skeletonization) which base on this point set, yield results with these properties. In the case of the medial axis based on the filtered point set $\ddot{O}_I$ we get a robust skeleton for noisy and rotated 2-d objects.

Another observation could be interesting for shape decomposition: Repeated filtering of the divergence values diverges quickly. The resulting connected sets of points $\in \ddot{O}_I$ are representatives for all major parts of the whole shape (see figure 8).

The presented approach may be easily adopted to discrete 3-d images.

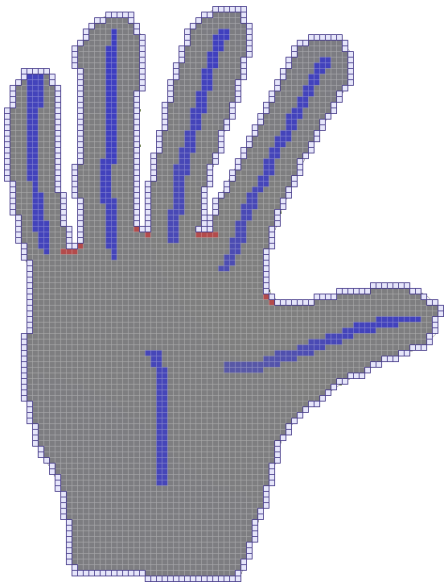


Figure 8: The set $\ddot{O}_I$ diverges if the divergence filter is repetitive applied. The resulting set may be used for a shape segmentation.

REFERENCES

[1] Harry Blum. A Transformation for Extracting New Descriptors of Shape. In Weiant Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, 1967.

[2] Sylvain Bouix and Kaleem Siddiqi. Divergence-based medial surfaces. In *ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part I*, pages 603–618, London, UK, 2000. Springer-Verlag.

[3] Jr. Calvin R. Maurer, Rensheng Qi, and Vijay Raghavan. A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(2):265–270, 2003.

[4] Pavel Dimitrov, Carlos Phillips, and Kaleem Siddiqi. Robust and efficient skeletal graphs. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:1417, 2000.

[5] Peter J. Giblin and Benjamin B. Kimia. On the local form and transitions of symmetry sets, medial axes, and shocks. *Int. J. Comput. Vision*, 54(1-3):143–156, 2003.

[6] Thouis R. Jones, Frédo Durand, and Mathieu Desbrun. Non-iterative, feature-preserving mesh smoothing. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 943–949, New York, NY, USA, 2003. ACM.

[7] Tao Ju, Matthew L. Baker, and Wah Chiu. Computing a family of skeletons of volumetric models for shape description. *Comput. Aided Des.*, 39(5):352–360, 2007.

[8] T. Yung Kong and A. Rosenfeld. Digital topology: introduction and survey. *Comput. Vision Graph. Image Process.*, 48(3):357–393, 1989.

[9] Arjan Kuijper, Ole Fogh Olsen, Philip Bille, and Peter Giblin. Matching 2d shapes using their symmetry sets. In *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, pages 179–182, Washington, DC, USA, 2006. IEEE Computer Society.

[10] Frederic F. Leymarie and Benjamin B. Kimia. The shock scaffold for representing 3d shape. In *In Proc. of 4th International Workshop on Visual Form (IWVF4*, pages 216–229. Springer-Verlag, 2001.

[11] Sylvain Paris, Pierre Kornprobst, Jack Tumblin, and Frédo Durand. A gentle introduction to bilateral filtering and its applications. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, page 3, New York, NY, USA, 2007. ACM.

[12] John L. Pfaltz and Azriel Rosenfeld. Computer representation of planar regions by their skeletons. *Commun. ACM*, 10(2):119–122, 1967.

[13] E. Remy and E. Thiel. Exact medial axis with euclidean distance. *Image and Vision Computing*, 23(2):167–175, February 2005.

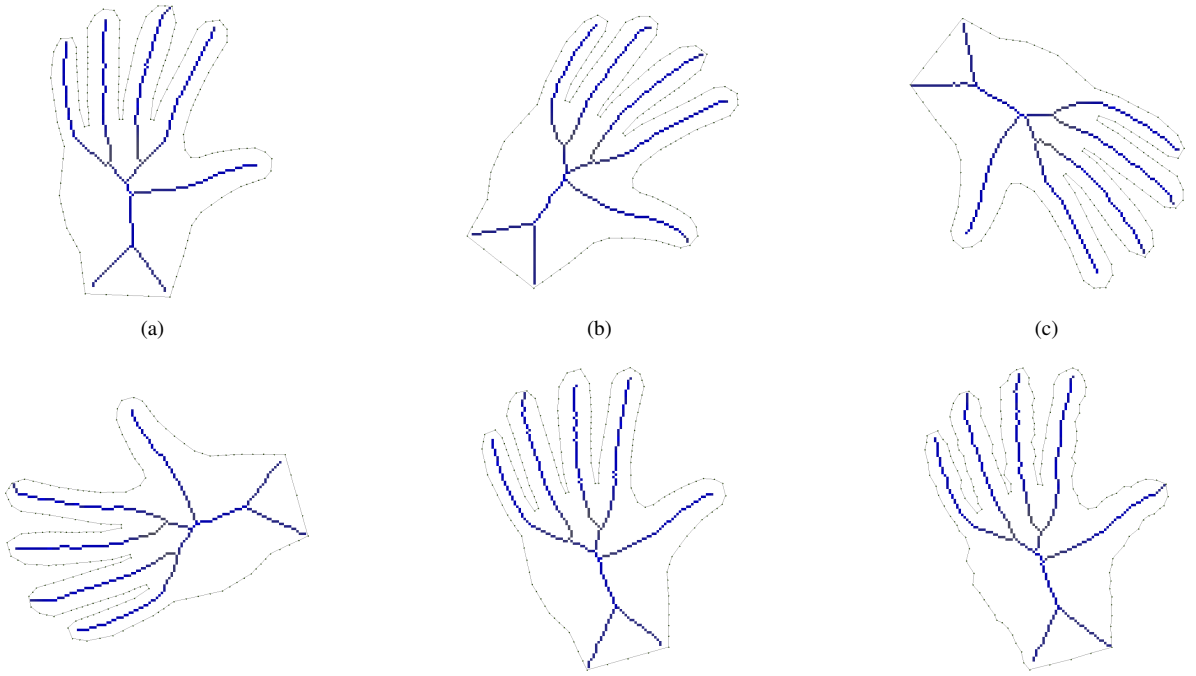
[14] Dennie Reniers and Alexandru Telea. Segmenting simplified surface skeletons. *Discrete Geometry for Computer Imagery*, 4992/2008(1-3):262–274, 2008.

[15] Kaleem Siddiqi, Sylvain Bouix, Allen Tannenbaum, and Steven W. Zucker. The hamilton-jacobi skeleton. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 828, Washington, DC, USA, 1999. IEEE Computer Society.

[16] Stina Svensson and Gabriella Sanniti di Baja. Simplifying curve skeletons in volume images. *Comput. Vis. Image Underst.*, 90(3):242–257, 2003.

[17] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 839, Washington, DC, USA, 1998. IEEE Computer Society.

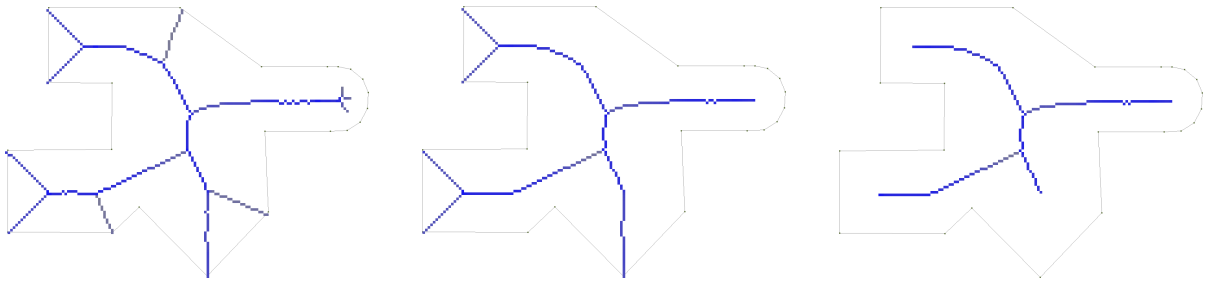
[18] Ben Weiss. Fast median and bilateral filtering. *ACM Trans.*



(a) (b) (c)

(d) (e) (f)

Figure 6: The contour is rotated in (a)-(e) and noise is added in (f). The resulting skeletons are nearly identically except for almost unnoticeable deviations.



(a) (b) (c)

Figure 7: A hierarchical skeleton may be constructed using different thresholds. For the skeletons in (a) all points p with $\ddot{\Delta}(p) \leq -0.1$, in (b) with $\ddot{\Delta}(p) \leq -0.3$, and in (c) with $\ddot{\Delta}(p) \leq -0.5$ are preserved.

Graph., 25(3):519–526, 2006.

[19] Buyue Zhang and Jan P. Allebach. Adaptive bilateral filter for sharpness enhancement and noise removal. *IEEE Transactions on Image Processing*, 17(5):664–678, 2008.

Simulating Real-Time Cloth with Adaptive Edge-based Meshes

T. J. R. Simnett R. G. Laycock A. M. Day
School of Computing Sciences, University of East Anglia
Norwich, NR4 7TJ, UK
{t.simnett|robert.laycock|amd}@uea.ac.uk

ABSTRACT

We present an approach to simulating detailed cloth in real-time using an adaptive edge-based mesh, by enhancing its usability and performance. We show how simple seaming can be used to combine multiple adaptive meshes into garments, accomplished with the use of discontinuous material co-ordinates. Performance is improved by decoupling the mesh adaption and simulation steps, allowing efficient data structures to be exploited for the simulation and collision detection. Greater memory efficiency is achieved by pre-allocating a pool of memory to be used by any mesh at any hierarchical level. The collision detection process is integrated into the edge-based adaption technique, enabling a garment to be coarsened and refined repeatedly, such that no new vertices are created inside of the object in collision. Our technique is illustrated for a 67k triangle character wearing a T-shirt adaptively refined to 6k triangles in real-time.

Keywords: Adaptive mesh, cloth simulation, garment

1 INTRODUCTION

The real-time simulation of cloth and particularly clothing is a challenging task, both the deformation calculations and collision detection are expensive thus limiting what is achievable with finite resources. Of course very detailed and realistic cloth animations can be simulated, albeit offline with very fine meshes. However, they are limited by how long the user is prepared to wait for the result. We would like to be able to simulate detailed clothing in real-time, since many applications require it such as virtual reality, games and garment prototyping. Mass-spring networks are an approach made popular for simulating real-time cloth by Provot, where an inverse dynamic procedure to correct the lengths of super-elongated springs particularly enhanced its useabilty [Pro95]. Mass-spring networks are favoured for their speed and real-time applications compared to more complex physical models [DB99]; however, the cost of the simulation in either case is proportional to the size of the mesh. Taking into account hardware advances over the last ten years, more is achievable in real-time now but the quality of offline simulations will always exceed that of real-time simulations. An approach to narrow the gap between offline and real-time systems concerns the use of adaptive meshes. Instead of using a fine regular mesh, a much coarser mesh is used, which is subsequently adaptively refined only in regions considered most important. Important regions are those that require a greater density of polygons needed to model important aspects of visually realistic cloth; most commonly wrinkles in areas of high curvature, and to accurately resolve collisions with objects. Coarsening, being the opposite of refinement takes place in regions that no longer require a high density of polygons.

We continue to build on Simnett et al's [SLD09] work, improving the edge-based adaptive refinement approach, which works on triangular meshes specially enhanced with connectivity information. The subdivision is the result of splitting edges in the mesh, this is performed using application specific edge-based criteria. This approach allows very fast incremental updates to the mesh, whereby only two adjacent triangles require updating on each edge split to ensure a conforming mesh. We use a mass-spring network for our simulation, with Verlet numerical integration and, importantly, make use of Provot's inverse dynamic procedure to correct the lengths of super-elongated springs [Pro95]. The mass-spring network shares the same topology with the adaptive mesh, where the mass of each vertex is calculated as one third of the adjacent triangle's mass, which is updated as the mesh refines.

The main contributions of this paper are:

- Incorporating multiple configurations into the transition states of edge-based adaptive meshes for improved real-time performance.
- Multiple edge-based adaptive meshes are combined to form garments. A T-shirt, constructed from two meshes attached at their seams, is illustrated in real-time.
- Collision aware mesh adaption technique enables a garment to be coarsened and refined resulting in no vertices of the cloth penetrating the character's body.

Firstly we will survey existing work in related areas and the rest of the paper is organised into the following sections detailing our work: Edge-Based Adaptive Mesh and Configurations, Seaming, Adaptive mesh Memory, Adaptive Cloth and Collision Detection.

2 PREVIOUS WORK

Adaptive meshes are a good way of reducing the cost of simulating cloth without sacrificing heavily on the detail and have been used a number of times previously, with the adaption criterion most commonly based on curvature [VL03][LV05]. Villard and Boroucharki found their adaptive mesh improved simulation times as much as six times [VB05]. Memory usage and access can be particularly problematic in the case of large meshes where disk storage is required [VL03]. Seaming, that is the joining of meshes together, is of particular interest to cloth simulations, allowing actual garments to be constructed in a similar way to reality by way of physically stitching pieces of flat material together to form garments. Ma et al [MHB06] presented a method that automatically constructed a seam surface along an arbitrary path on the surface of an irregular mesh. They simulated seam puckering on 3D garments with a mass-spring network, which produced realistic wrinkles but it unfortunately requires the use of very fine meshes beyond the capability of real-time simulation. Pabst et al [PKST08] looked at the influence of seams on the bending of fabric, together with an accurate bending model. If using a relatively coarse mesh, they found that it must be locally refined around the seams to smooth out the abrupt changes that would otherwise be present in the bending stiffness between adjacent elements, thus increasing the costs. Their method produced excellent realism, comparing a real garment with that of a simulated one with each step taking 65ms to calculate the bending forces. Durapinar et al's [DG07] virtual garment design and simulation system featured automatic pattern generation by cutting a regular mesh along lines between defined corner vertices. A mass-spring system was used with seaming taking 4.313 seconds for a skirt with 1400 vertices, finalized by combining vertices into one by adding each vertex's spring forces to the other. Seaming is not always mentioned for character garments and no specific details were included for the garment using the adaptive mesh in [LV05], suggesting that a 3D mesh in the shape of the garment is used instead. We believe a simple approach is prudent for real-time simulation. We detail our approach in Section 4 that will allow seaming of an edge-based adaptive mesh inorder to allow garments to be created. Collision detection research is a large field, many algorithms and data structures have been designed for this. Without suitable collision detection and response, cloth simulations are severely limited and complex algorithms are used for high fidelity offline animations. Bridson et al.'s [BFA05] work generates high quality wrinkles with complex collisions. By robustly processing collisions, contacts and friction, with a collision aware post processing step a surface is subdivided and iteratively smoothed for rendering. Their approach took approximately two minutes per frame for a piece of cloth with 150 x 150 nodes. Self collisions for cloth is particularly expensive to compute. Govindaraju et al. [GKJ+05] presented a method that accurately detected all self-collisions for a 23K triangle cloth dress in 400-550 ms. Their chromatic decomposition partitions a fixed topology mesh into independent sets, and uses a linear-time culling algorithm performing 1D overlap tests on the CPU and a 2.5D on the GPU. Achieving real-time performance puts tight limits on the complexity of the cloth and collision testing. Fuhrmann et al. [FGL03] achieved interactive animation of cloth with self-collision by approximation; it considered only pairs of particles and held them apart using a bounding hierarchy of particles.

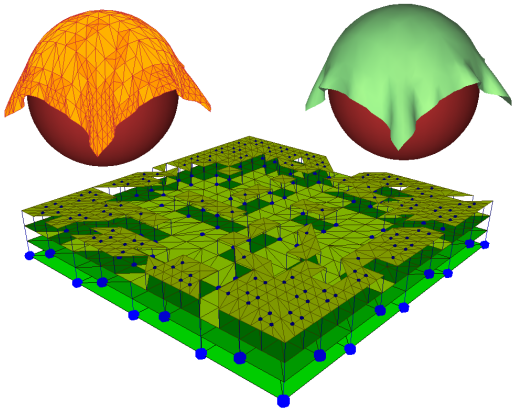


Figure 1: Cloth draped over a sphere, rendered with smooth lighting and coloured levels together with an illustration of the adaptive mesh's hierarchy.

3 EDGE-BASED ADAPTIVE MESH

The Edge-Based approach to adaptive meshes allows fast incremental changes to the mesh, refining or coarsening by one level at a time. Edges in the mesh are

split in order to subdivide the mesh, and curvature, edge length and edge collisions have proved suitable criteria for controlling this for cloth modelling. Coarsening of the mesh is achieved by rejoining previously split edges. When an edge is split, a new vertex is generated at the centre and two child edges are created. Triangles are dealt with using a state based retriangulation scheme, where the triangles adjacent to the split (or rejoined) edges are internally retriangulated. This is undertaken to build a conforming mesh with new additional internal edges and internal triangles. Upon reaching a full subdivided triangle (1-to-4 split), further refinement of child edges and triangles are allowed. The adaptive mesh forms a complex hierarchical tree structure of levels, with edges containing two child edges and a central vertex, triangles contain upto four internal triangles and three edges. The highest level in use of any part of the mesh constitutes the whole mesh used for the cloth simulation. Figure 1 shows a simple simulation of a piece of cloth draped over a sphere, for which the hierarchy is illustrated.

3.1 Configurations

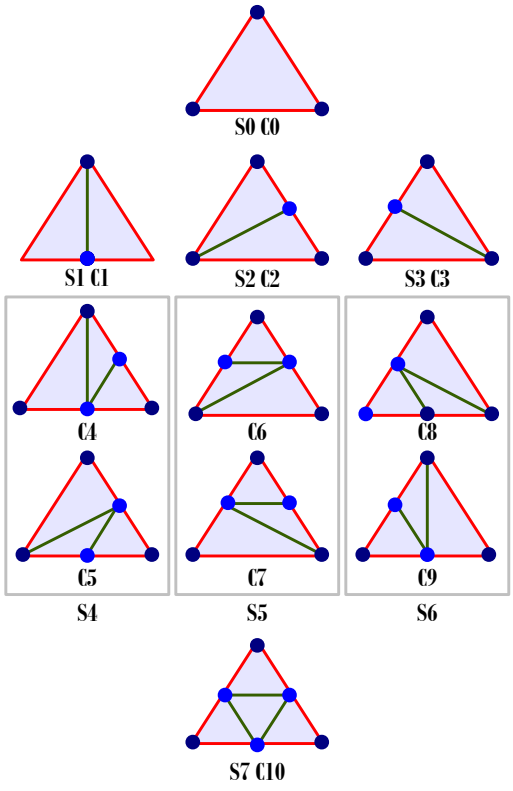


Figure 2: Triangle states and configurations showing internal triangulations of a parent triangle.

The state based triangulation approach enabled the fast determination of the new triangulation by checking the change in status of external edges [SLD09]. We have since expanded this approach to include the other three triangulations missing from the previously defined eight states. Each of the three new triangulations share the same state as an existing one, that is the status of the edges are the same. We therefore require an additional identifier other than state for the triangulations; we call this a triangle's "configuration". There are a total of eleven configurations starting with an empty parent triangle to a completely sub-divided one (1-to-4 split) and all the possibilities in-between. We map every state to a configuration, except in three cases where now each of those states maps to two possible configurations. See Figure 2 for the states and configurations.

Deciding on the configuration to use is straight forward in the case of the 1-1 mappings. For the other 3 states, we can consider a transition from the previous configuration to one that provides the minimal change, this helps to keep the forces more consistent between steps as the cloth adapts. For example: consider that when transitioning from C1 to C4/C5, the algorithm would choose C4 as the configuration that provides minimal change. Therefore, if C5 was chosen instead the change would be visually detectable in the rendered cloth (unless the vertices were co-planar).

It can be seen that many transitions between two configurations have common features. This is because it is not ideal to have to configure the whole parent triangle when it is re-triangulated. Instead, a lower level approach to re-triangulation that exploits the similarities is possible, although it increases code complexity considerably. The previous configuration is reused as much as possible, changing only as necessary. There are a total of 100 possible transitions that may be used by the adaptive mesh. Table 1 shows a comparison of the speed up that is achieved with this approach for a selected number of cases.

Transition	Time (μs)	Improved Time (μs)
C0 to C1	0.218079	0.199257
C1 to C4	0.316416	0.177083
C1 to C5	0.312889	0.236971
C2 to C5	0.312540	0.173625
C4 to C1	0.238927	0.070819
C4 to C10	0.435041	0.257016

Table 1: Time comparison of the new faster re-triangulation approach compared to the work in [SLD09].

Some transitions such as C0 to C4/C5, are both equally costly and we need to either use a default selection or select it based on some criteria. We have investigated the use of three simple approaches based on edge curvature, edge length and edge rest length. The two length criteria choose the configuration that gives the shorter new edge length, based on either

the deformed length or rest length. The curvature criterion looks to see which way the parent triangle is bending and selects the best configuration to conform to this. Figure 3 shows very small differences between different selection criteria. The reason for this is that in most cases there is a faster transition available that will be used in precedence to the selection criteria, such that it will have little effect overall. Therefore we consider the use of a default criteria, since it requires no additional processing while not negatively impacting on the refinement.

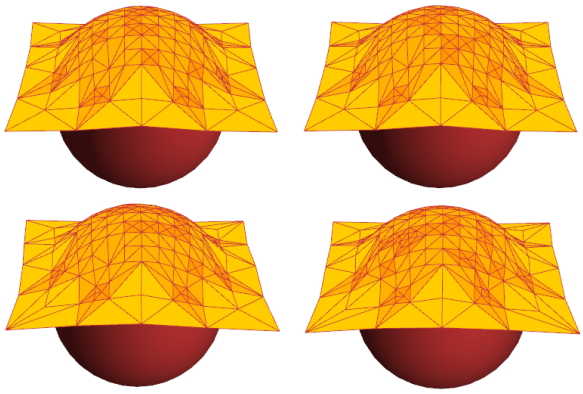


Figure 3: Four spheres are illustrated using four different transition selection methods. (Top Left: Default Choice, Top Right: Edge Length, Bottom Left: Edge Rest Length, Bottom Right: Edge Curvature) A selection method is used for transitions that are equally costly, otherwise the fastest minimal change transition is used.

4 SEAMING

Complicated models for seaming do not currently appear feasible for real-time simulations; therefore we take a simple approach and perform seaming on the base level of our adaptive mesh. We accomplish this by merging the vertices and connecting the edges of seams together. Firstly the user can interactively select links between groups of vertices and pairs of boundary edges using the mouse or alternatively use previously stored links. The mesh can be simulated using the coarse mesh if desired before actual seaming occurs to bring the meshes together, (linked seam vertices are constrained together). Next we perform the actual merging; where initially the vertices and edges are removed from the meshes. A new vertex is created for each group of vertices that are linked and is stored as part of the garment, each new vertex has a total mass of all the combined vertices together with a merged adjacent triangle list. The connectivity of the boundary edges and triangles in the meshes must be updated as to reference the new merged vertex rather than each of the group. Consequently, we can use the merged vertices instead of the old ones in the simulation. Finally boundary edges are connected into pairs, setting their pointers to point to each other, which are now stored in a seam edge as part of the garment. Seaming for a T-Shirt is illustrated in Figure 4, using a mesh for the front and one for the back with seams above the shoulders, under the arms and down the sides.

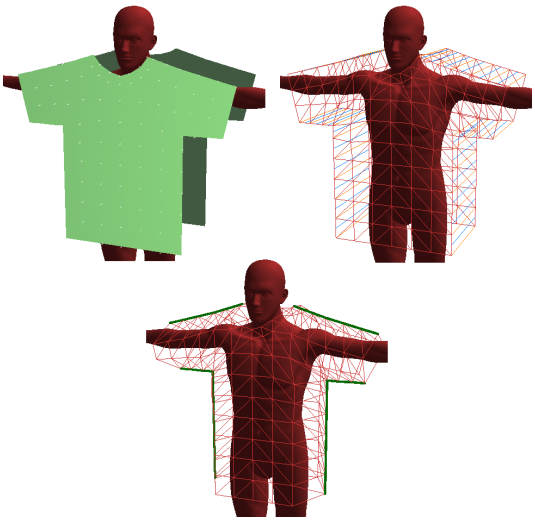


Figure 4: Interactive Seaming, Top Left: initial base meshes (rendered with normals), Top Right: seaming links defined (vertex links in orange, edge links in blue), Bottom Centre: after seaming, seams in green.

4.1 Discontinuous Material Co-ordinates

There is a challenge with managing material co-ordinates for the meshes; 2D material co-ordinates allow the quick determination of un-deformed lengths needed for the physics calculations between any two points of the mesh and they are readily acquired from flat textile patterns. Ultimately there exists discontinuous co-ordinates along the seam lines between multiple meshes and this is also the case for a single mesh (e.g. a single mesh seamed to form a cylinder). Storing the material co-ordinates in the vertices, alone cannot support this. We present a method that supports discontinuous co-ordinates between any base triangle in the mesh, and continuous co-ordinates are used within triangles as they are subdivided. To support this, material co-ordinates were needed to be moved from the vertices into the edges. Since an edge between adjacent triangles consists of two oppositely directed pieces, each side can have its own material co-ordinates. Where the material co-ordinates are continuous, both sides of the edge will hold references to the same material co-ordinates. In the case of discontinuous co-ordinates, each side will hold reference to a different set of co-ordinates. Figure 5 illustrates

this, three meshes are to be seamed together with three sets of continuous material co-ordinates. Vertices are merged but the material co-ordinates are not, then edges are connected together. It should be noted that the boundary edge lengths should match between pairs for best results, as one edge is designated as the control edge whose length will be used for the spring forces in the simulation.

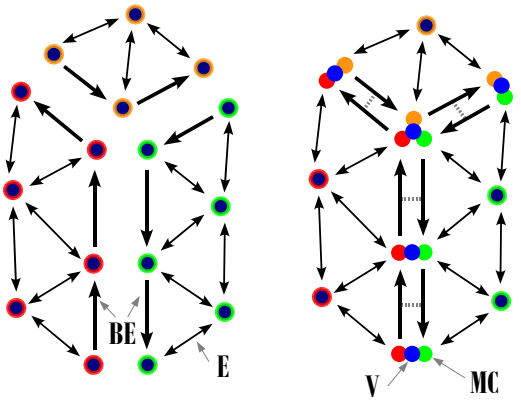


Figure 5: Three meshes each with their own set of material co-ordinates (red,orange,green), seamed together with discontinuous material co-ordinates (MC) along their boundary edges (BE). Vertices are shown in blue (V).

5 ADAPTIVE MESH MEMORY

Memory considerations are often overlooked in real-time simulations where we favour pre-computation wherever possible, and consider algorithm efficiency of most importance. However, adaptive meshes require a large amount of memory to store subsequent levels even for small base meshes. Real-time constraints often force the use of pre-allocation for all levels that may be needed, this is because the operating system's managed dynamic memory imposes a relatively large overhead for creation and deletion. In regards to real-time cloth, the purpose of the adaptive mesh is to allow refinement only in areas that require more triangles (typically high curvature). There is a limit on the overall number of triangles in a piece of cloth that can be simulated and rendered in real-time, and the adaption criteria should be chosen accordingly. We can make use of this to achieve large memory savings, by only pre-allocating the maximum memory needed by the mesh over the course of a real-time simulation. The memory is divided into pools, one for each of the principle building blocks of the mesh: triangles, edges, vertices and material co-ordinates. As the mesh is refined and coarsened the pools are used dynamically, in a similar way but with almost no overhead compared to system managed memory. We use a linked list approach with two lists, which allow constant time access to the pools. One list stores unused objects to be taken from the pool, and the other holds list nodes ready for when an object is put back onto the pool. There is an additional advantage to this approach, the pools may be shared simultaneously by multiple meshes; each mesh does not need significant resources allocated to it which may not be fully utilised. It is difficult to predict the exact requirements for a complex simulation especially with collisions, however, experimentation can yield good results. Taking the simulation in Figure 10 as an example, the base mesh requires 0.3 MB and an additional 7.2 MB in the memory pools was needed with allowances to support upto approximately 7,000 triangles. The full refinement to level 3 would require 15.6 MB in total, this means in this example, our approach has saved 8.3 MB or around 53% of the memory needed over that of pre-allocation for the adaptive mesh.

6 ADAPTIVE CLOTH

6.1 Separation of Adaption and Simulation

The original approach in [SLD09] was to perform both adaption and simulation in the same algorithm and do this at each update. We have found this not to be efficient in cases of very small time steps, often these very small steps are needed to ensure numerical stability of the simulation when using finer meshes. All the edges in the mesh will be checked against the criteria for each time step for adaption. When very few changes have taken place, the cost of checking will surpass the actual cost of updating the mesh structure per step. Therefore, we have since decoupled the adaption and simulation allowing more resources for the simulation to run at a rate required for stability. The adaptive mesh will update a constant number of times each second (typically at 30Hz), we refer to this as a major step. Minor steps refer to the simulation steps, which are performed after each major step, the number of these depend on the time step required to give stable results. Collision detection and response is performed once at the end of each major step rather than every minor step to save time when vertices are only moving very small distances.

To further increase performance, it is not necessary to recalculate the triangle normals (using the cross product) and vertex normals (average of adjacent triangles) during the adaption step since the vertices are not moved. Also, new vertices are initially co-linear with their parent edge, and therefore fast interpolation is sufficient to determine their normal.

6.2 Data Structure Traversal

Our adaptive mesh's hierarchy enables an efficient algorithm for refinement but imposes an overhead for the

simulation where the hierarchical data structure must be traversed many times each step. We have alleviated this cost by constructing a temporary list of pointers to triangles, edges and vertices each time the adaptive mesh is updated, they are particularly effective now that the adaption is decoupled and the lists may be valid and used for a number of simulation steps. Table. 2 shows the time in milliseconds for each simulation step, the cost of creating the list, and the improved simulation timing that results from using the list, hence the cost of the data structure traversal to the simulation can be seen.

Level	Simulation (ms)	List Creation (ms)	Sim. with List (ms)
Base	0.049832	0.001453	0.033908
1	0.214280	0.006593	0.136197
2	0.880447	0.033203	0.573544
3	2.964748	0.136163	2.407764

Table 2: Simulation times for a single step using standard traversal and temporary lists updated each adaptive step. A base mesh of 128 triangles is used and the cost of list creation can be seen.

7 COLLISION DETECTION

Although geometric shapes provide straight forward collision checks by use of their parametric equations, building complex objects out of them is not easy. The decision was made to implement a general method that can be used with any 3D triangle model. The only requirement is that the triangles of the mesh must be outward facing (determined by the vertex winding) forming a surface where the cloth will not be allowed to penetrate through into the object. A problem with the highly flexible nature of cloth is that even if the vertices are not penetrating the object, edges and triangles may intersect the object's surface causing very noticeable visual artefacts. The use of the adaptive mesh, greatly increases the cloths ability to approximate the underlying surface it is in contact with, but this alone is not sufficient. A full triangle-triangle collision approach provides a potential solution although at a cost to processing compared to only vertex-triangle collision detection. We seek a compromise for use with real-time simulations; a suitable approach is to leave a region above the surface to hide these intersections. If too great a region is used, the gap will be very noticeable and unrealistic; relatively too small and intersections will still be visible. To enable this, we use a separate mesh for collision and rendering. The original mesh is loaded, and then sent to the graphics card to be stored in a vertex buffer object. To construct the collision mesh, we first calculate smooth normals for the vertices (which may be the same as used for rendering). Using these normals we expand the mesh by moving the vertices in the direction of their normals with an adjustable offset. The result is a shell around the original object, giving us the region to hide intersections within. The originally loaded mesh can be deleted from main memory if desired, leaving it only on the graphics card for rendering so that memory requirements are not doubled by using the collision mesh. We allow the mesh to be arbitrarily rotated and translated, by calculating and storing the transformations using a 3x3 matrix for the rotation and a 3D vector for the translation. The cloth is simulated in world co-ordinates, and vertices are first transformed into the object's local coordinate space to undergo the collision checks. Considering a cloth vertex (mass) has a current position and a previous position (which was outside of the object), the collision with a static object can be determined by checking that the vertex's path has not intersected the object.

The object is potentially a large triangle soup, and therefore we initially must partition it into a more efficient structure for collision testing inorder for real-time computation. A 3D grid of cells is used, where each cell is an axis aligned bounding box that contains a list of all the triangles that are inside or partially overlapping.

The main collision algorithm between a cloth vertex and a object proceeds as follows. The previous and current position are transformed into the object's coordinate space, the transformed points are hence referred to as A and B. Each grid Cell that the path AB intersects is tested for collisions by testing the vertex with each triangle within the cell. Figure 6 illustrates the vertex triangle collision test, firstly the segment AB is intersected with the triangle's plane to find an intersection point (IP). If IP exists, the barycentric co-ordinates are calculated to determine if this lies within the triangle face; if this is so, the surface point (SP) is calculated by projecting B in the direction of the triangle's normal. The current position is moved to the SP after transforming it back into world co-ordinates.

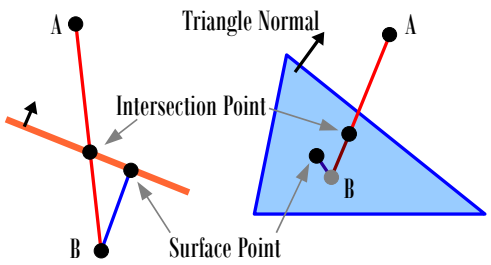


Figure 6: Vertex-triangle collision: The vertex's current position (B) is moved to the surface point, if the intersection point is within the triangle.

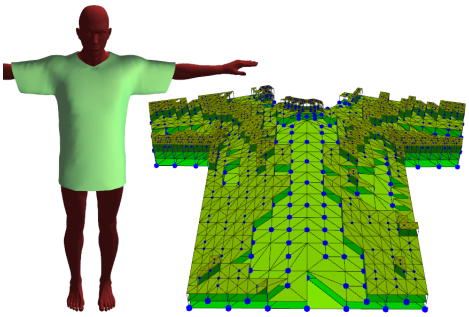


Figure 7: A T-Shirt draped on a static character, with the adaptive hierarchy of the front mesh illustrated.

7.1 Collision within Mesh Adaption

Collision routines are called from within the adaption step, as necessary, in a number of places. When an edge is split, the newly created vertex in the centre of the edge can be created inside other objects, a situation that is prevented under normal circumstances. We must be able to determine when this has occurred and find a suitable location to relocate the vertex to. We tackle both of these problems in one step, by constructing a ray pointing outwards from the object and testing it for collision with the collision mesh. The direction we use is the average of the edges two end normals. If there is a collision, then the vertex is moved to the intersection point adjusting the previous position to preserve the velocity, see Figure 8. Figure 9 shows three snap shots of the cloth simulation on a character, and demonstrates how the coarse base mesh may be used to speed up the initial placement of the garment.

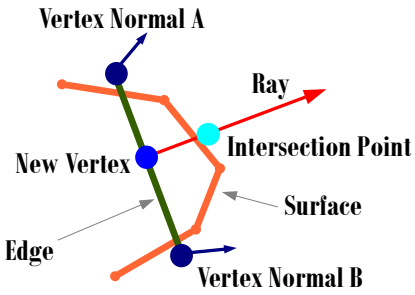


Figure 8: Resolving a new vertex that has been generated inside an object. It is moved to the intersection point between the directed ray and the surface.

7.2 Conflicting Criteria: Edge Collision and Curvature.

When using multiple criteria to control the refinement an coarsening of the mesh, problems can arise from these conflicting. An edge may be in collision and therefore should be split, but the curvature criteria may determine that the edge should not be split and would

Figure 9: Left: coarse mesh, Centre: immediately after two adaption steps (incremental approach implies a change of one level for each step and level two is required in some areas to resolve the collisions), Right: after 1 second (30 adaption and 120 simulation steps), the cloth has relaxed.

therefore be rejoined in the next step. Previously Simnett et al. [SLD09] used a rejoin wait count, with the aim to prevent rapid flipping between coarsening and refining an edge by specifying a number of steps before an edge was allowed to rejoin after having been split. This was although visually effective, caused the flipping to still occur each time the rejoin wait was completed. We have changed how it is used in order to improve upon this and the performance. If edge collision splitting is enabled for the cloth; before an edge is rejoined, it is checked for collisions. The purpose being is that if there is an edge collision and it is rejoined, it will almost certainly be split again at the next step. The edge in collision will be prevented from rejoining, saving the costs of the edge join and retriangulation. The additional collision check has a cost associated with it too, so we make use of the rejoin wait count again but now it saves the cost of collision and curvature checks. However, there is a trade off as rapid coarsening can bring force, integration and collision savings for the simulation. A balance between a too long rejoin wait and not long enough is important; we find a rejoin wait count of ten steps works well, which will basically spreads out the cost of the rejoin checks over the ten steps as they are not synchronized.

8 RESULTS

We implemented our cloth simulation using C++ and used Opengl for rendering, the results presented in this paper were performed using a PC with a Intel 2.66 GHz Core i7 920 using a single thread with 6GB of RAM. The limit of our approach is approximately seven thousand triangles, with the simulation times being the limiting factor where mesh adaption takes less than 10% of the total time for each major step. Figure. 10 shows a mesh of this size, worn as a T-shirt by a static character that is simulated in real-time. Each major step takes 22ms running at 30Hz, with 4 minor or simulation steps running at 120 Hz. Figure 7 shows the corresponding adaptive hierarchy for the front mesh of the T-Shirt. The accompanying video features the character, where two meshes are initially seamed together and the cloth is simulated in real-time. To demonstrate the adaptive mesh and collision detection more robustly, a

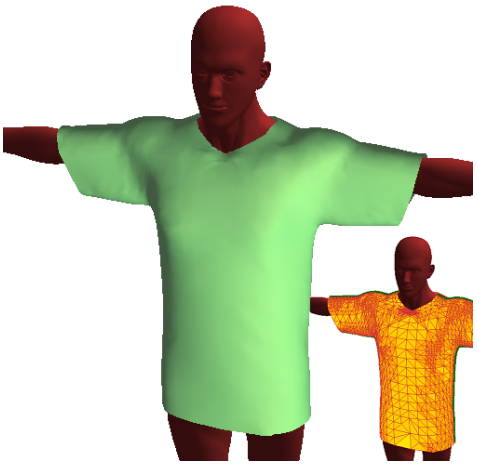


Figure 10: A T-shirt is draped on a static character with 67k triangles. The cloth is constructed from two base meshes seamed together, which totals 316 triangles, and are adaptively refined to 6199 out of a possible 20224 (Level 3) triangles. It takes approximately 22ms for each adaption and 4 simulation steps, running at 30Hz in real-time.

force on the cloth is introduced from a movable virtual fan. The video can be found at the following url: http://www.crowdsimulationgroup.co.uk/Cloth.wmv

9 CONCLUSION

We have presented a number of improvements to the edge-based adaptive mesh approach presented in [SLD09], such as increased performance by decoupling mesh adaption from simulation, using a more efficient data structure for traversal and introducing new configurations to speed up re-triangulation. Furthermore, we have implemented a robust collision detection scheme for the cloth with arbitrary triangle meshes and have integrated this into the adaption process, which together with seaming allows the real-time simulation of a detailed T-shirt on a virtual character. The improved handling of memory has provided a saving of 53% for this simulation, compared to the pre-allocation of the complete hierarchy. In the future we would like to simulate garments on moving characters, and also on multiple characters simultaneously. We anticipate changes will be needed to our collision handling in order to allow non-static objects, and the grid based approach will need to be modified to support jointed characters. Our work shows that the adaptive mesh would work well with additional level of detail approaches, since an off-screen character can be simulated using purely their base mesh. Figure 9 shows that our refinement method could quickly return to detailed garments if the character were to come back into view. The fact that the base meshes are small in terms of memory requirements compared to the refined mesh, allows the memory pool to be used by multiple meshes effectively.

REFERENCES

[BFA05] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *International Conference on Computer Graphics and Interactive Techniques*. ACM Press New York, NY, USA, 2005.

[DB99] M. Desbrun and A. Barr. Interactive animation of structured deformable objects. In *In Graphics Interface*, 1999.

[DG07] F. Durupinar and U. Gudukbay. A virtual garment design and simulation system. In *IV'07. 11th International Conference*, pages 862–870, 2007.

[FGL03] A. Fuhrmann, C. Gross, and V. Luckas. Interactive animation of cloth including self collision detection. *Journal of WSCG*, 11(1):141–148, 2003.

[GKJ+05] N.K. Govindaraju, D. Knott, N. Jain, I. Kabul, R. Tamstorf, R. Gayle, M.C. Lin, and D. Manocha. Interactive collision detection between deformable models using chromatic decomposition. *Proceedings of ACM SIGGRAPH 2005*, 24(3):991–999, 2005.

[LV05] L. Li and V. Volkov. Cloth animation with adaptively refined meshes. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*, pages 107–113. Australian Computer Society, Inc. Darlinghurst, Australia, 2005.

[MHB06] Liang Ma, Jinlian Hu, and George Baciu. Generating seams and wrinkles for virtual clothing. In *VRCIA '06: Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, pages 205–211, NY, USA, 2006. ACM.

[PKST08] Simon Pabst, Sybille Krzywinski, Andrea Schenk, and Bernhard Thomaszewski. Seams and bending in cloth simulation. *VRIPHYS*, 382(1):24–41, 2008.

[Pro95] X. Provot. Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behaviour. In *Graphics Interface*, pages 147–147. Canadian Information Processing Society, 1995.

[SLD09] T.J.R Simnett, S.D. Laycock, and A.M. Day. An edge-based approach to adaptively refining a mesh for cloth deformation. In *EG UK Theory and Practice of Computer Graphics 2009*, pages 77–84, 2009.

[VB05] J. Villard and H. Borouchaki. Adaptive meshing for cloth animation. *Engineering with Computers*, 20(4):333–341, 2005.

[VL03] V. Volkov and L. Li. Real-time refinement and simplification of adaptive triangular meshes. *VIS 2003. IEEE*, pages 155–162, 2003.

A new CBIR approach based on relevance feedback and optimum-path forest classification

André Tavares da Silva
Unicamp, Brazil
atavares@dca.fee.unicamp.br

Alexandre Xavier Falcão
Unicamp, Brazil
afalcao@ic.unicamp.br

Léo Pini Magalhães
Unicamp, Brazil
leopini@dca.fee.unicamp.br

ABSTRACT

Recently some CBIR approaches have shown the use of relevance feedback to train a pattern classifier to select relevant images for retrieval. This paper revisits this strategy by using an optimum-path forest (OPF) classifier. During relevance feedback iterations, the proposed method uses the OPF classifier to decide which database images are relevant or not. Images classified as relevant are sorted and presented to the user for a new iteration. Such images are ordered according to the normalized distance using relevant and irrelevant representative images, computed previously by the OPF classifier. Our experiments show that the proposed approach requires fewer iterations, being faster and more effective than methods based on SVM.

Keywords: CBIR, Relevance Feedback, OPF.

1 INTRODUCTION

Image collections have been created and used in several applications, such as digital libraries, medicine, and biodiversity information systems. Given the size of these collections, it is essential to provide efficient and effective means to retrieve images. Such a problem has raised the interest in putting together image processing, information retrieval, and database management to design content-based image retrieval (CBIR) systems for large image collections [2].

The visual content of an image in a CBIR system is often represented by a *feature vector*, which may encode color, texture, and/or shape measures. The image is then interpreted as a *point* in the feature space. A query in a CBIR system is usually done by range (returning all images whose distance to the query image is less than a given radius) or by similarity. We will focus on the second approach where a number of the closest images to the query point are retrieved from the database. Given that the meaning of those images may differ for distinct users since they are not completely represented by the feature vector, a *semantic gap* occurs between the user's expectation and the result of the query. Thus, *relevance feedback* techniques have been investigated to reduce the semantic gap by requiring more user interaction than simply the specification of a query image. These techniques usually involve three steps: (i) a small number of retrieved images is presented to the user; (ii) the user indicates which images are relevant; (iii) the system learns the user's opinion from this feedback in order to return more relevant images in the next iteration. This process may be repeated until the user is satisfied, but it is highly desirable to finish it in a few iterations.

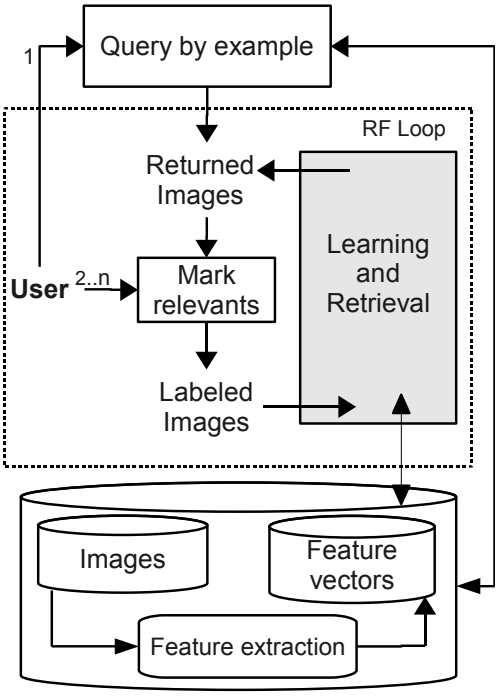


Figure 1: CBIR with Relevance Feedback.

Figure 1 shows an overview of the relevance feedback process. There are several studies on each stage of this pipeline, such as creating more robust local descriptors[15, 19] (i.e. feature vectors and distance functions to compare them) or providing scalability to

huge image databases[9, 20]. Our work focuses on the learning and retrieval process (gray box in Figure 1), especially in query classification and ranking.

Relevance feedback techniques were initially proposed for document retrieval, but have been successfully applied to CBIR systems[2, 13, 14, 17, 23]. Figure 2 illustrates three examples of simple relevance feedback techniques [7, 10]. In Figure 2a, the positive examples (relevant images) from a first iteration are used to move the next query point to their geometric center in the feature space. This idea stemmed from Rocchio's formula [13] in document retrieval and it has been successfully exploited in CBIR systems, such as MARS [14] and MindReader [6]. Two other methods use the relevant images as next query points and, depending on the distance to this multi-point query set, different isosurfaces are formed in the feature space (Figures 2b and 2c). The method we present here is also a multi-point query but, differently from those approaches, we exploit relevant and irrelevant images as query points.

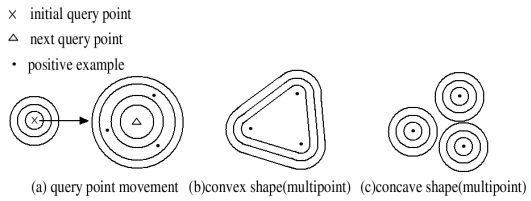


Figure 2: Simple relevance feedback techniques that change query shapes (i.e., isosurfaces with respect to the query points).

Approaches based on relevant and irrelevant images usually exploit active learning techniques to design a classifier that selects from the database the candidate relevant images for sorting by distance to the query point [1, 3, 16]. The method proposed by Tong and Chang. [16] uses Support Vector Machines (SVM) for image classification [22]. During the relevance feedback iterations, the method finds the optimum hyperplane that separates relevant and irrelevant images, presenting to the user the images closer to the hyperplane. This hyperplane is adjusted along the iterations and, after a last iteration, the method presents the images farther to the hyperplane, on its relevant side.

The method we propose here follows a similar strategy, using a faster and more effective classifier aiming to present the most relevant images in the database at each iteration, unlike SVM. For a given set of relevant and irrelevant images, the method designs an OPF (Optimum-Path Forest) classifier [12]. Only database images classified as relevant are sorted by distance and presented to the user in the next iteration. This distance is computed based on relevant and irrelevant prototypes (representative images), computed during the training of OPF classifier. We show that this strategy is actually very effective reducing considerably the number of required iterations.

In addition to that, the *distance function* used to compare images has also influence on the retrieval process. Some methods use multiple pairs of feature vectors and distance functions, called *descriptors*, and compare two images by combining their distance based on each descriptor [11, 18]. In this case, the learning from relevance feedback may also change the way to combine descriptors [18]. Our method can exploit the same framework, but we will consider in this study only a single descriptor per image.

This paper is organized as follows. Section 2 presents the proposed algorithm based on OPF classifier and an example illustrates our relevance feedback process. The experiments and results using three heterogeneous image databases are described in Section 3. As baselines for comparison, we use the method od Tong and Chang [16] and the one illustrated in Figure 2c, which uses only relevant images for multi-point query. Section 4 states the conclusions and discusses our future work.

2 CBIR USING OPF CLASSIFIER

OPF is a classification method which represents each class of objects by one or more optimum-path trees rooted at given samples, called prototypes [12]. The training samples are nodes of a complete graph, whose arcs are weighted by the distance between the feature vectors of their nodes. In relevance feedback, we have two classes: relevant images chosen by the user and irrelevant ones. The prototypes computed by the OPF classifier are then used to sort the images according to the user's selection.

Let $\mathscr{Z}$ be an image database. For every image $t \in \mathscr{Z}$, we have a feature vector $\vec{v}(t) \in \Re^n$. That is, every image may be interpreted as a point in the feature space $\Re^n$. The distance $d(s,t)$ between two images s and t is the distance between their corresponding feature vectors. For an initial query point s, the proposed method returns the N closest images in $\mathscr{Z}$ to s (query by similarity). Due to the semantic gap, the closest images to s may not be the most relevant for a given user. By marking the relevant images among the returned ones, the user creates two sets: a set $\mathscr{I} \subset \mathscr{Z}$ of irrelevant images and a set $\mathscr{R} \subset \mathscr{Z}$ of relevant images. The method then uses sets $\mathscr{R}$ and $\mathscr{I}$ to compute two optimum-path forests (OPF), one for each class. Each database image $t \in \mathscr{Z} \setminus \mathscr{I} \cup \mathscr{R}$ is then classified according to the root's label of the forest (relevant/irrelevant) which offers to t the optimum path in the graph. Only the N closest images labeled as relevant will be returned in a set $\mathscr{C}$ to the user in the next iteration. Relevant prototypes ($\mathscr{A}$) and irrelevant ones ($\mathscr{B}$), computed in the previous step, are then used to sort the images in $\mathscr{C}$ for the next iteration.

The method computes the average distance $\bar{d}_{\mathscr{A}}(t,\mathscr{A})$ between each image $t \in \mathscr{C}$ and images in the set of relevant prototypes $\mathscr{A}$. It also computes the average distance $\bar{d}_{\mathscr{B}}(t,\mathscr{B})$ between t and images in the set of irrelevant prototypes $\mathscr{B}$. Finally, a distance $\bar{d}(t,\mathscr{A},\mathscr{B})$ is computed as a normalized mean between relevant and irrelevant prototypes: $\bar{d}(t,\mathscr{A},\mathscr{B}) = \frac{\bar{d}_{\mathscr{A}}(t,\mathscr{A})}{\bar{d}_{\mathscr{A}}(t,\mathscr{A})+\bar{d}_{\mathscr{B}}(t,\mathscr{B})}$.

Algorithm 1: Relevance Feedback Algorithm

Input: A query image s, a feature extraction function v, a distance function d, a desirable number N of relevant images, an image database $\mathscr{Z}$ and a number T of iterations.
Output: An ordered list L of the N most relevant images in $\mathscr{Z}$.
Auxiliary: Sets $\mathscr{R} \subset \mathscr{Z}$ and $\mathscr{I} \subset \mathscr{Z}$ of relevant and irrelevant images, $\mathscr{A} \subset \mathscr{Z}$ and $\mathscr{B} \subset \mathscr{Z}$ of relevant and irrelevant prototypes, set $\mathscr{C} \subset \mathscr{Z}$ of images classified as relevant for the next iteraction.

```
Compute the distance d(s,t) for every image t ∈ Z.
Create an ordered list L of the N closest images t to s based on d(s,t).
Set I ← ∅ and R ← ∅.
for each learning iteration i = 1,2,...,T do
    Set C ← ∅.
    The user marks the relevant images in L, which are inserted into R and the irrelevant ones are inserted into I.
    if |R| < N then
        Compute OPF using sets I and R, resulting also A and B.
        for each image t ∈ Z\I ∪ R do
            if t is labeled as relevant by OPF then
                insert t into the set C of images classified as relevant.
            end
        end
    end
    else
        Return the final ordered list L with the N most relevant images in R, as defined by the user's selection.
    end
    Create an ordered list L with the N most relevant images in C, in increasing order of d̄(t,A,B).
end
Return the final ordered list L with the N most relevant images in R, completing it with the N − |R| relevant images in C in the increasing order of d̄(t,A,B).
```

After classifying each image in $\mathscr{Z}\setminus\mathscr{I} \cup \mathscr{R}$, the method returns to the user a new set of N relevant images, which contains the lowest values of $\bar{d}(t,\mathscr{A},\mathscr{B})$. This process is then repeated for a few iterations T and, finally, the system returns all relevant images obtained so far.

In order to illustrate the advantages of our relevance feedback approach as compared to a simple retrieval of the N closest images to s, we present an example of query image in Figure 3 from the image database Corel [21]. We use a color descriptor, called BIC, proposed by Stehling et al. [15]. The $N = 30$ closest images in that database are shown in Figure 16, where the relevant images are presented with a blue border. After $T = 3$ iterations (a reasonable number of iterations for practical situations), the system presents the $N = 30$ most relevant images found so far, as shown in Figure 17. It is important to note that the quality of this result may vary depending on the image descriptor.

Figure 3: A query image s.

3 EXPERIMENTS AND RESULTS

In order to evaluate our method, we use the BIC descriptor with the dLog distance function [15], and compare its effectiveness using precision-recall curves and two other approaches as baselines: the SVM-based method proposed by Tong and Chang [16] and the multi-point query with relevant images only, as illustrated in Figure 2c. The first, named here as SAL (SVM Active Learning), is also named SVM_{ACTIVE} or SVM_{AL} in the literature. It was chosen because it is based on a state-of-the-art technique for image classification. The second, named as QPM (Query Point Movement) [16], was selected to illustrate the importance of irrelevant images in the multi-point query set. Our approach is named here OPF_{AL} or simply OPF, because it is based on the OPF classifier.

As mentioned before, our work focuses in query classification and ranking. Indexing schemes to accelerate the search can be exploited in our method, as well as techiques for descriptor combination. But we consider in this study only a single descriptor in order to compare the proposed method against others.

The curves of precision-recall use the entire image database $\mathscr{Z}$. Thus, lines 18 and 20 of our algorithm are replaced by: Create a list L with all relevant images in $\mathscr{C} \cup \mathscr{R}$, in their increasing order of $\bar{d}(t, \mathscr{A}, \mathscr{B})$, and compute the precision-recall curve for images in L.

The experiments used three heterogeneous image databases, representing different challenges for CBIR.

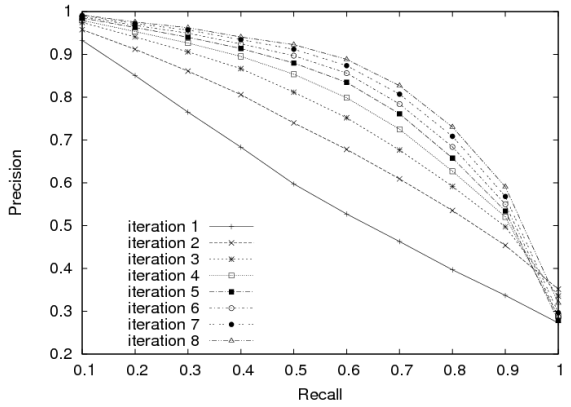


Figure 4: Mean precision-recall curves of OPF in Corel database, iterations 1 to 8.

- PASCAL [4, 5].

  This database consists of 3,448 natural images, each one containing multiple regions of interest (subimages). Each region contains one object from a class of visual objects (bikes, boats, birds). The regions are labeled by their class performing a total of 23 classes with different number of images, varying from 72 to 446 subimages each.

- Corel [21].

  This database is a collection with 200,000 images from the Corel GALLERY Magic–Stock Photo Library 2. We use a subset of 3,906 natural images, pre-classified into 85 classes. These classes have different number of images varying from 7 to 98 images each.

- ETH-80 [8].

  This database is available in the project COGVIS, serving for both psychophysical and computational studies concerning object recognition and categorization. The project includes images of objects from 8 basic-level categories performing a total of 2,384 images, distributed uniformly among the classes.

For each image database, we simulate the user behavior by using each image as initial query point and marking the relevant points (images from the same class of the query) from 30 returned images at each iteration.

First, we present in Figures 4, 5 and 6 the mean precision-recall curves of OPF for the databases Corel, ETH-80, and PASCAL, respectively, by varying the number of iterations from 1 to 8. These curves show that OPF improves its performance (the higher the precision-recall curve, the better is the method) with the number of iterations, as expected, but they also indicate the challenge degree of each database: PASCAL imposes more challenges than Corel which is more difficult than ETH-80.

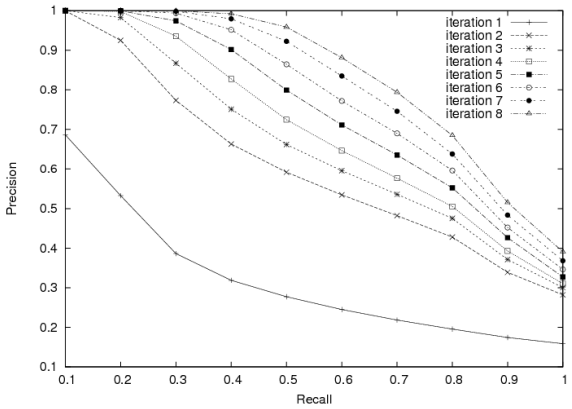


Figure 5: Mean precision-recall curves of OPF in ETH-80 database, iterations 1 to 8.

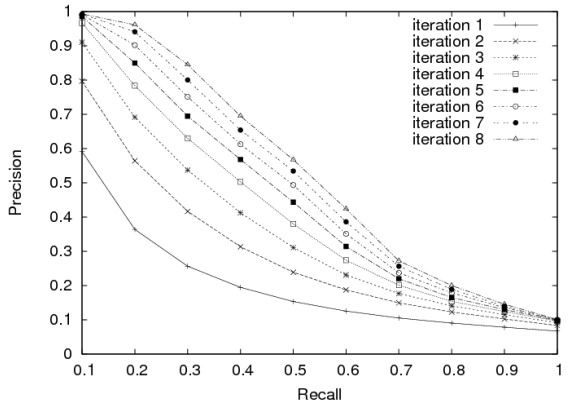


Figure 6: Mean precision-recall curves of OPF in PASCAL database, iterations 1 to 8.

Following, Figures 7 to 15 show the mean precision-recall curves of each method (OPF, QPM, and SAL) in each database (Corel, ETH-80, and PASCAL) for 3, 5 and 8 relevance feedback iterations. One may observe that OPF outperformed SAL and QPM in the most difficult databases, Corel and Pascal, and for all number of iterations. In the easiest case, ETH-80, the curves cross each other in some recall rates, but OPF is still better than the others up to 40% of recall for 3 and 5 iterations, and 50% of recall for 8 iterations. In addition to that, OPF is much faster than SAL and it learns quicker the simulated user's wish, providing effective results in fewer iterations. We consider 3 iterations as the ideal

number for practical situations. OPF has also outperformed QPM in all cases and this indicates the importance of using relevant and irrelevant points in multi-point query systems rather than only relevant points.

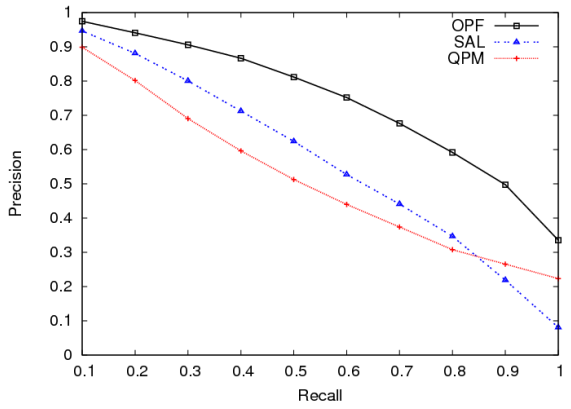


Figure 7: Mean precision-recall curves in Corel database, third iteration.

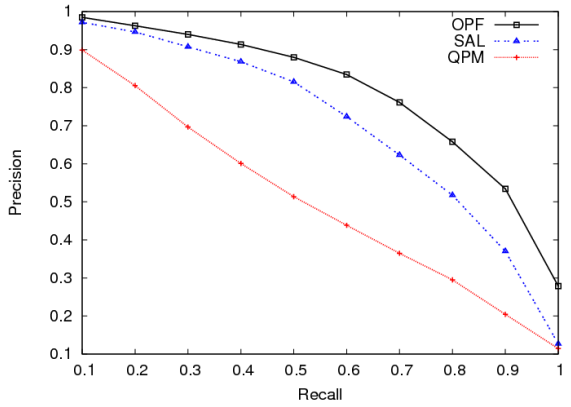


Figure 8: Mean precision-recall curves in Corel database, fifth iteration.

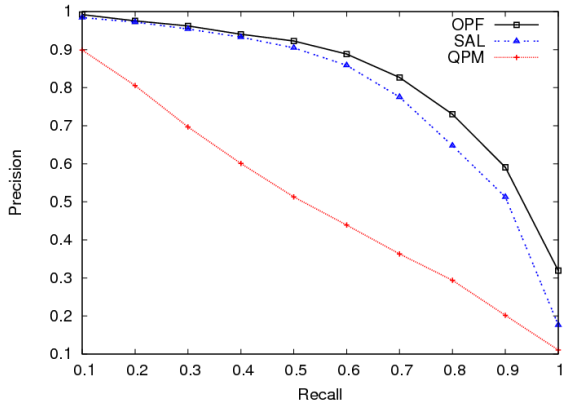


Figure 9: Mean precision-recall curves in Corel database, eighth iteration.

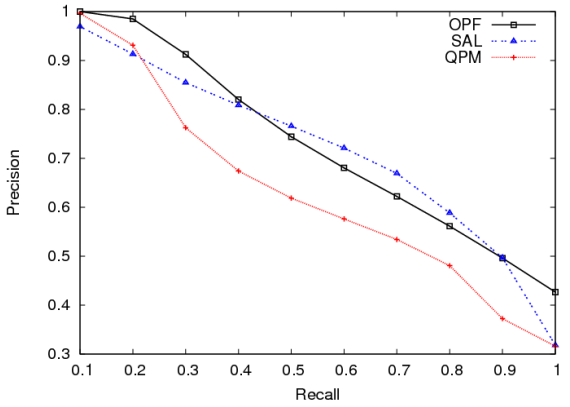


Figure 10: Mean precision-recall curves in ETH-80 database, third iteration.

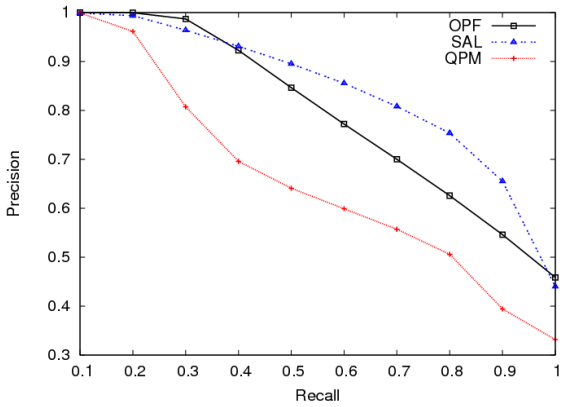


Figure 11: Mean precision-recall curves in ETH-80 database, fifth iteration.

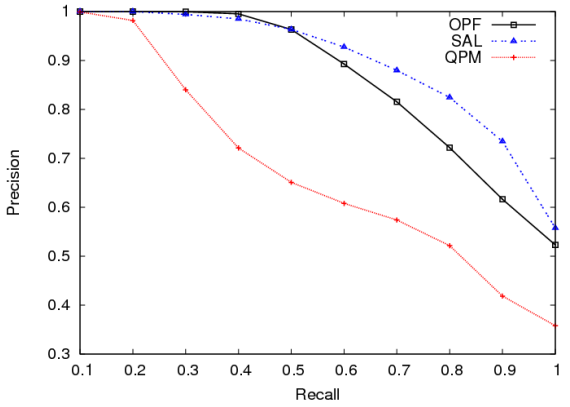


Figure 12: Mean precision-recall curves in ETH-80 database, eighth iteration.

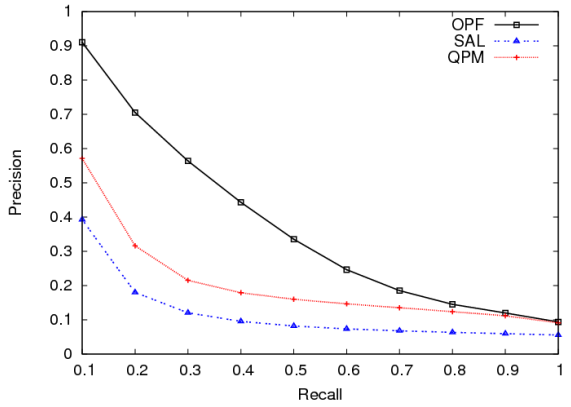


Figure 13: Mean precision-recall curves in PASCAL database, third iteration.

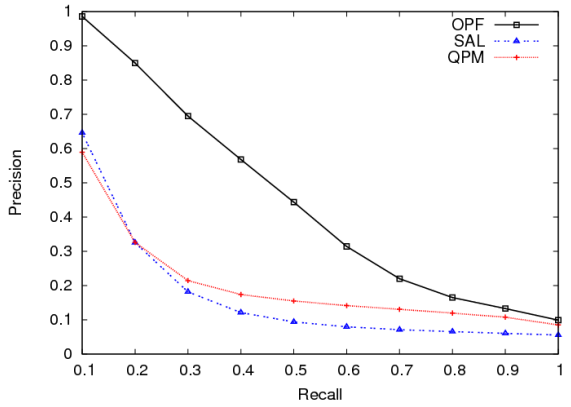


Figure 14: Mean precision-recall curves in PASCAL database, fifth iteration.

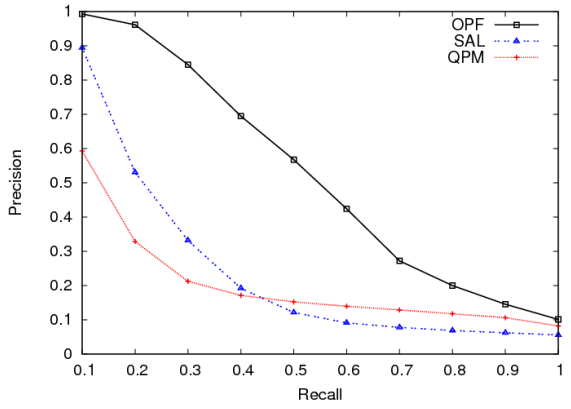


Figure 15: Mean precision-recall curves in PASCAL database, eighth iteration.

Tables 1 and 2 show the execution times of SAL and OPF, respectively. We present time values for all images of the databases for iterations 3, 5 and 8, used to compute the precision-recall curves of Figures 7 to 15.

Table 3 shows the average execution time for one iteration of SAL and OPF. Our approach could also be used with indexing schemes to further accelerate the search process and reduce the execution time. However, we present in this paper execution times without indexing structures. As the number of iterations grows, the runtime increases. In the Corel database for instance, our method takes 0.4 seconds to present images at the eighth iteration while SAL takes 10.2 seconds in the average. The tests were performed in a machine with Intel Pentium D processor at 3.4GHz and 1 GB RAM running the Linux operational system.

Table 1: Total execution time of SAL (minutes).

Database	Corel	ETH-80	PASCAL
3 iterations	1,116	420	903
5 iterations	2,170	702	1,743
8 iterations	5,330	1,120	4,483

Table 2: Total execution time of OPF (minutes).

Database	Corel	ETH-80	PASCAL
3 iterations	42.8	24.2	33.4
5 iterations	102.9	38.9	81.2
8 iterations	224.0	59.1	188.4

Table 3: Average execution time per query (seconds).

Database	Corel	ETH-80	PASCAL
SAL	5.71	3.53	1.96
OPF	0.22	0.20	0.19

Methods such as OPF and SAL classify candidate relevant images in the image database and sort them to select the N closest to the query point(s). One may ask about the relevant images misclassified as irrelevant. These images are lost by the system. Table 4 presents the percentage of images that were erroneously discarded by OPF for each database and for iterations 3, 5 and 8; the percentages of missed relevant images are insignificant. This result is even more important when we consider that the performance of CBIR systems, as mentioned before, usually increases with the number of descriptors and strategies to combine them [18], which is not being exploited in the present study.

Table 4: Percentages of relevant images missed by OPF due to classification in each database.

Database	Corel	ETH-80	PASCAL
3 iterations	0.36%	1.23%	3.37%
5 iterations	0.32%	1.21%	3.22%
8 iterations	0.27%	1.02%	3.06%

4 CONCLUSION AND FUTURE WORK

We presented a new relevance feedback technique for CBIR. This is the first time that the OPF classifier is being used and evaluated for small training sets, as required in learning by relevance feedback. Differently from the original method, we have separated the optimum-path forests of each class for classification. This constitutes a simple but very effective variant of the original method. We have also proposed a new order relation among the relevant images, which is based on the mean distances to the prototypes of the OPF classifier.

We have evaluated the method using a color descriptor, three heterogeneous image databases, two reference approaches, a few iterations, and query by similarity. The results indicated that the proposed method, named OPF, requires fewer iterations of relevance feedback. It outperformed the reference approaches in all databases and the number of missed relevant images due to classification was insignificant.

The new CBIR approach based on relevance feedback and optimum-path forest classification presented in this paper provides a solution in interactive time for practical applications. On average our method was twenty times faster than SAL.

Our future work involves the use of multiple descriptors and techniques to combine them. We intend to use other descriptors based on shape, texture and color and combine them by using techniques such as Bayesian framework, Genetic Programming [18] or other similar approaches. We also intend to investigate other image classifiers and to evaluate the methods for multiple users.

ACKNOWLEDGEMENTS

The first author thanks CNPq for financial support (140968/2007-5). The second author thanks CNPq (project ARPIs, 481556/2009-5) and FAPESP (07/52015-0).

REFERENCES

[1] M. Cord, J. Fournier, and S. Philipp-Foliguet. Exploration and search-by-similarity in cbir. In *SIBGRAPI '03: Proceedings of the Brazilian Symposium on Computer Graphics and Image Processing*, pages 175–182. IEEE Computer Society, 2003.

[2] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 40(2):1–60, 2008.

[3] L. Duan, W. Gao, W. Zeng, and D. Zhao. Adaptive relevance feedback based on bayesian inference for image retrieval. *Signal Process.*, 85(2):395–399, 2005.

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[5] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html.

[6] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Upper Saddle River, NJ: Prentice Hall, 2002.

[7] D.-H. Kim and C.-W. Chung. Qcluster: Relevance feedback using adaptive clustering for content-based image retrieval. In *In Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 599–610, 2003.

[8] B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. In *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*, pages 409–415, 2003.

[9] H. Lejsek, F. Ásmundsson, B. Jónsson, and L. Amsaleg. An efficient disk-based index for approximative search in very large high-dimensional collections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):869–883, 2008.

[10] D. Liu, K. A. Hua, K. Vu, and N. Yu. Fast query point movement techniques for large cbir systems. *IEEE Trans. on Knowl. and Data Eng.*, 21(5):729–743, 2009.

[11] R. Ohbuchi and Yushin Hata. Combining multiresolution shape descriptors for 3d model retrieval. In *Proc. WSCG 2006*, Plzen, Czech Republic, 2006.

[12] J. P. Papa, A. X. Falcão, and C. T. N. Suzuki. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, 19(2):120–131, 2009.

[13] J. J. Rocchio. *Relevance feedback in information retrieval*, pages 313–323. Prentice-Hall, Englewood Cliffs, NJ, USA, 1971.

[14] Y. Rui, T. S. Huang, and S. Mehrotra. Content-based image retrieval with relevance feedback in mars. In *In Proc. IEEE Int. Conf. on Image Proc*, pages 815–818, 1997.

[15] R. O. Stehling, M. A. Nascimento, and A. X. Falcão. A compact and efficient image retrieval approach based on border/interior pixel classification. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 102–109, New York, NY, USA, 2002. ACM.

[16] S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*, pages 107–118, New York, NY, USA, 2001. ACM.

[17] R. S. Torres and A. X. Falcão. Content-based image retrieval: Theory and applications. *Revista de Informática Teórica e Aplicada*, 13(2):161–185, 2006.

[18] R.S. Torres, A.X. Falcão, M.A. Gonçalves, J.P. Papa, B. Zhang, W. Fan, and E.A. Fox. A genetic programming framework for content-based image retrieval. *Pattern Recognition*, 42(2):217–312, Feb 2009.

[19] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: a survey. *Found. Trends. Comput. Graph. Vis.*, 3(3):177–280, 2008.

[20] E. Valle, M. Cord, and S. Philipp-Foliguet. High-dimensional descriptor indexing for large multimedia databases. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 739–748, New York, NY, USA, 2008. ACM.

[21] J. Z. Wang, J. Li, and G. Wiederhold. Simplicity: Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:947–963, 2001.

[22] L. Zheng and X. He. Classification techniques in pattern recognition. In *Proc. WSCG 2005*, London, United Kingdom, 2005.

[23] X. S. Zhou and T. S. Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia Systems*, 8(6):536–544, April 2003.

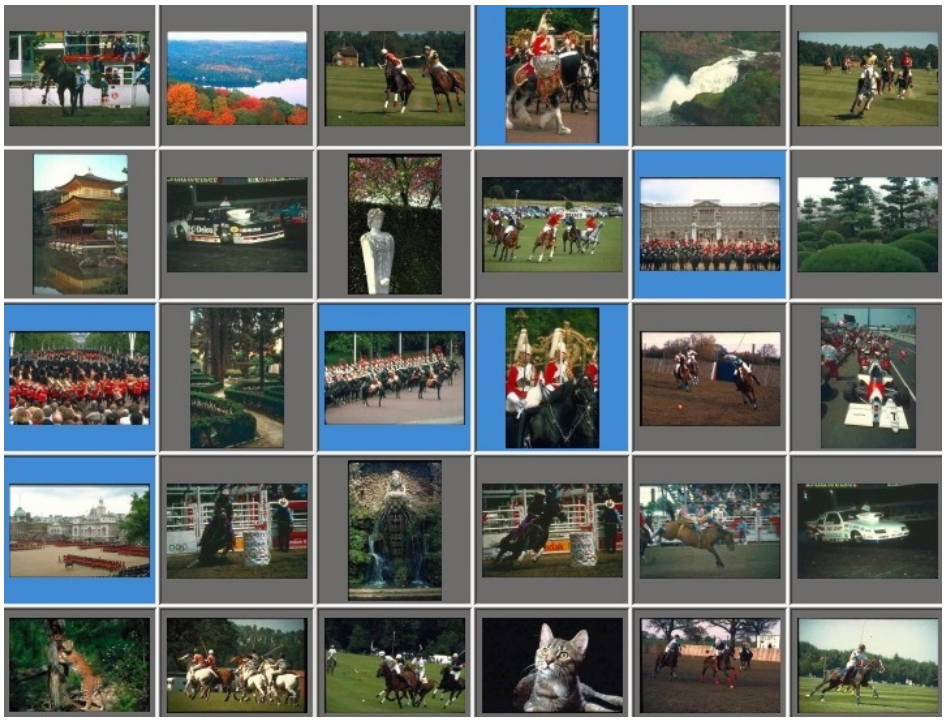


Figure 16: Closest images to s based on $d(s,t)$.

Figure 17: Result of OPF after three iterations.

Stable Explicit Integration of Deformable Objects by Filtering High Modal Frequencies

Basil Fierz Jonas Spillmann Matthias Harders
(bfierz | jonas.spillmann | mharders)@vision.ee.ethz.ch
Computer Vision Laboratory
ETH Zurich, Switzerland

ABSTRACT

In a traditional finite element method for the simulation of deformable objects, the stiffness matrix depends on the shape of the tetrahedral elements. Ill-shaped elements containing large or small dihedral angles lead to an arbitrary large condition number of the stiffness matrix, thus slowing down the simulation. In addition, high modal frequencies cannot be simulated stably if an explicit numerical time-integration scheme is considered. We propose an approach consisting of two components to address this problem: First, we isolate the ill-shaped tetrahedra by performing an eigenvalue decomposition, and then remove their high modal frequencies by directly altering their element stiffness matrices. This makes the elements softer in some directions. To prevent element inversion, we define constraints that rigidify the object along those directions. The fast projection method, implemented as a velocity filter, is employed to enforce the constraints after the temporal evolution. With our approach, a significantly larger time step can be chosen in explicit integration methods, resulting in a faster simulation.

Keywords: physically based-modeling, deformable objects, modal analysis, explicit integration, constraints

1 INTRODUCTION

The modeling and dynamic simulation of deformable objects is an active field of research in computer animation. To accomplish this, the domain of the object is commonly discretized into tetrahedral elements. Then, we assume a linear stress-strain relationship and use the finite element method (FEM) to compute the restitution forces which work against the deformation. By assuming that the nodes of the mesh constitute mass points, the equations of motion can be solved numerically to evolve the object in time.

In most standard FEM codes, the internal forces are related to the deformed geometry of an element, and work in the direction of its undeformed geometry. Consequently, the condition of the element stiffness matrix relies heavily on the shape of the element. More precisely, ill-shaped elements with particularly small or large dihedral angles result in a large condition number of the corresponding stiffness matrix, as pointed out by, *e. g.*, Shewchuk [She02a].

In turn, poorly conditioned stiffness matrices cause severe problems for the underlying numerical solver: If implicit integration methods are considered, then the convergence of an iterative solver is slowed down, resulting in a significant computational overhead. In the context of explicit integration methods, small time steps are necessary to reproduce the corresponding high frequencies and thus to stabilize the simulation. This is particularly problematic in the context of real-time simulations that require the maximum time spent to compute one *simulated second* to be less than one *real second*. The time required for the force computation and numerical integration is not dependent on the time step, therefore smaller time steps increase the time to compute one simulated second.

There exist two ways to attack this problem, notably by considering the *geometry*, or by considering the *simulation*. The easiest way to influence the geometry is by employing an appropriate meshing approach that avoids ill-shaped tetrahedra such as [LS07]. However, especially in the context of frequent topology changes in cutting simulations, this can be hard to achieve in real-time. In contrast, changing the geometry of particularly ill-shaped elements locally often results in a cascade of topological operations that propagate through the mesh, which are difficult to control [KS07]. In contrast, approaches that consider the simulation usually limit the strain and thus inhibit degenerate elements [BFA02, TPS09]. However, in the context of an explicit time-integration scheme, these methods cannot prevent instabilities completely.

In this paper, we propose a novel approach to stabilize a simulation of deformable objects in combination with an explicit time-integration method, thereby especially addressing the case of frequent topological changes such as cutting and fracturing. Our approach is a combination of two components: First, we assume that the time step of the simulation is given, based, *e. g.*, on real-time considerations. In cutting simulations, we instead demand that the time step stays constant throughout the simulation. We now identify those elements which cannot be simulated stably with that

time step. We determine the directions of the problematic vibration modes by performing an eigenvalue decomposition of the element stiffness matrix. Then, we assemble a new stiffness matrix by filtering the corresponding modal frequencies.

In order to account for the changed material properties, and to prevent element inversion of the softened elements, the second component rigidifies those elements along the determined directions. To accomplish this, we define a set of constraints and employ the fast projection method [GHF*07], implemented as a velocity filter, to enforce the constraints. We show that by employing our stabilization filter, an explicit solver can take significantly larger time steps than traditional approaches, thus economizing the additional cost of the constraint enforcement phase many times over. In addition, our method avoids the headaches that come with manually identifying and editing ill-shaped elements. Various experiments underline the benefits of our method.

2 RELATED WORK

Since most schemes for the simulation of deformable objects relate the restitution forces to the shapes of the elements, there exist a wealth of approaches handling ill-shaped elements. The approaches can be grouped into those that avoid the creation of ill-shaped elements, and those that reduce their influence on the simulation stability.

The easiest way to handle ill-shaped elements is to prevent them completely, by only generating well-shaped elements at the meshing stage. While Delaunay-based meshing approaches [She98, ACSYD05] and advancing front approaches [Sch97] obtain a very good *average* tet quality, they cannot guarantee not to create any ill-shaped tets. Unfortunately, *one* ill-shaped tet is enough to destabilize the simulation. This problem is alleviated by the approach of Labelle *et al.* [LS07]. They compute a distance field and then fill the domain with tets selected from a list of stencils. Their main benefit is that they can give guarantees on the tet qualities. However, employing their approach to remesh domains on the fly (as in [KFCO06, WT08]) is currently not possible in real-time.

As an alternative to global remeshing, it is also possible to identify ill-shaped elements and use node-snapping and edge-swapping techniques to improve their quality [ISS04, CDRR04]. Recently, Klingner *et al.*have proposed to optimize the mesh using a sequence of different optimization steps [KS07]. However, the resulting topological changes tend to be computationally intensive thus prohibiting their use in real-time simulations.

Instead of the geometry, the underlying simulation can be considered. In the field of dynamic linear systems there has been some work in trying to stabilize explicit integration schemes by determining a suitable time step [GK07, Shi04]. Both look at the properties of the dynamic system and the integration scheme. They determine a time step based on the spectral radius of the matrix of a linear solver. However, this requires calculating the eigenvalues of the system matrix, which is not feasible for interactively changing topologies. Schmedding *et al.* [RS09] define a dynamic damping term such that the simulation is stable. While their method is suitable for mass-spring systems, the extension to FEM-based deformations is not straightforward.

In order to determine the dynamic response of a deformable object to a prescribed deformation, the model frequencies of the system matrix can be computed. This has been done before by Pentland and Williams [PW89] and Hauser *et al.* [HSO03] who identified high modal frequencies as source of small amplitude vibrations causing unpleasant visible artifacts. Moreover, the high modal frequencies require small time steps. As a result they removed the high modal frequencies from the simulation in order to improve the simulation speed. However, their methods require pre-computing the vibration modes for the stiffness matrix. Real-time cutting and fracturing is not feasible using these methods, because calculating the modes online would be too expensive. In contrast to the cited works, we perform the modal analysis on a per-element basis, which results in a smaller system to be solved.

To constrain the deformation of the relaxed elements we filter the velocities, such that deformations in these elements are limited. In doing so, we build on an approach recently proposed by Thomaszewski *et al.* [TPS09]. They extend this concept to continuum based constraints and as such define a neo-hookean material law. This new material is linear within the constraints, and inextensible otherwise. While Thomaszewski *et al.* limit the strain of *all* elements, we identify the ill-shaped elements beforehand and only consider their deformation. Therefore, we obtain a significantly smaller system of constraints whose solution is feasible in real-time.

3 OVERVIEW

Our approach is a combination of two components, notably a method which determines the ill-shaped tets and reduces their high modal frequencies, and a velocity filter which rigidifies these tets afterwards. In this section, we give a short overview of our approach.

As input, we assume an object whose domain is discretized into tetrahedral elements. In addition, we assume that the material properties (Young's modulus, Poisson ratio and density) of the object are given. Further, we assume that an explicit integration method such as the explicit Euler or the Verlet scheme is employed to numerically evolve the object in time.

As a first step, we choose a minimum time step Δt_{min} for our simulation. This time step can be chosen heuristically, *e. g.*, based on the expected time t_{exp} to compute one simulation step of this object on a given hardware. Based on this time, the minimum allowable time step to simulate the object in real-time can be derived. Alternatively, in case of a cutting simulation, we just demand that a time step Δt_{min} chosen at the start of the simulation does not change during the simulation, even though

the cuts might induce ill-shaped tets requiring a smaller time step.

We then identify the elements which cannot be simulated with Δt_{min}, *i. e.*, which have one vibration mode exceeding the maximum allowable modal frequency. This process requires the solution of an Eigenvalue problem of size 12×12 for each element, as detailed in Section 4.3. We then filter the modal frequencies of the found elements along the affected directions by directly altering the stiffness matrices. This process is done *before* the computation of the restitution forces and time evolution.

The second component acts as a velocity filter, which is executed *after* the time evolution. Since we have filtered the modal frequencies of some ill-shaped tets, these elements are particularly sensitive to element inversion. In order to prevent this, we rigidify the elements along the relaxed directions. That is, we define geometric constraints on the positions, and employ a projection method to enforce these constraints directly. The result of the velocity filter are new positions and velocities satisfying the constraints, as discussed in Section 5.

4 FILTERING MODAL FREQUENCIES

In this section, we describe the first part of our approach, which determines the ill-shaped tets and relaxes the material along the bad directions. To accomplish this, we first derive an analytical relation between the element stiffness matrix and the time step of the underlying numerical integration. Equipped with this knowledge, we can determine the bad directions, and the required amount of material relaxation.

4.1 Finite element method

We assume that the restitution forces are computed by employing a linear explicit FEM with warped stiffness [MG04]. We decompose the simulation domain into tetrahedral elements. Within each tetrahedral element, linear shape functions $\mathbf{N}_i, i = 1 \dots 4$ are employed to interpolate the displacement field $\mathbf{u}$ with $\mathbf{u}(\mathbf{x}) = \mathbf{N}_i(\mathbf{x})\mathbf{u}_i$, where $\mathbf{u}_i = \mathbf{x}_i - \mathbf{x}_{i,0}$ is the displacement of the node i with undeformed position $\mathbf{x}_{i,0}$. The elastostatic equation states that the divergence of the stress σ and the external forces $\mathbf{f}^e$ form an equilibrium,

$$\nabla \cdot \sigma + \mathbf{f}^e = 0. \tag{1}$$

Applying the Galerkin discretization to the variational formulation of (1), we obtain the description of the stress σ_e for each element e:

$$\sigma_e = \mathbf{E}\varepsilon = \mathbf{E}\mathbf{B}_e\mathbf{u}, \tag{2}$$

where $\mathbf{E}$ is the material-dependent elasticity matrix, ε is the Cauchy strain [Hug87], and $\mathbf{B}_e$ is a constant matrix depending on the gradient of the shape functions $\mathbf{N}_i$. The restitution forces of each element are then linearly related to its nodal displacements using the stiffness matrix $\mathbf{K}_e = V_e\mathbf{B}_e^T\mathbf{E}\mathbf{B}_e$, where V_e is the volume of the element e. The global stiffness matrix $\mathbf{K}$ is the superposition of the element stiffness matrices $\mathbf{K}_e$, $\mathbf{K} = \sum_e \mathbf{K}_e$.

Shewchuk [She02a] showed that the condition of $\mathbf{K}_e$ depends on the cotangent of the dihedral angles of the tetrahedron e. If the angles approach 0° or 180°, the cotangent goes to infinity, which results in an arbitrary large condition number of $\mathbf{K}_e$. As a consequence, the condition number of the global stiffness matrix $\mathbf{K}$ depends on the condition numbers of the element stiffness matrices $\mathbf{K}_e$.

In a static FE analysis, we are interested in solving for the unknown deformed nodal positions $\mathbf{x}$, given prescribed boundary conditions. In this case, the problem amounts to solving

$$\mathbf{K}\mathbf{x} = \mathbf{f} \tag{3}$$

for the unknown vector $\mathbf{x}$. As Shewchuk pointed out, the convergence rate of an iterative solver for the linear system decreases with increasing condition number of $\mathbf{K}$, and thus, for ill-shaped tets, the solution of (3) becomes arbitrary expensive.

4.2 Dynamic finite element analysis

In the following, we are focusing on the dynamic simulation of a deformable object. In this case, the equations of motion of the object are

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{C}\dot{\mathbf{x}} + \mathbf{K}(\mathbf{x} - \mathbf{x}_0) = \mathbf{f}^e, \tag{4}$$

where $\mathbf{M}$ is the mass and $\mathbf{C}$ the damping matrix. For simplicity and as usual for most problems in computer animation, we assume that the mass is lumped in the nodes, resulting in a diagonal mass matrix. Furthermore, we limit the analysis to the case of an undamped system, *i. e.*, $\mathbf{C} = 0$. We will consider damping as future work. Nevertheless, we can employ viscous point damping without sacrificing our theoretical bounds.

To solve (4), the second order ordinary differential equation (ODE) is transformed into a system of first order ODEs, which are then numerically integrated in order to obtain the trajectories of the mass points. In our case, we assume an explicit time integration scheme [QSS00].

In order to characterize the stability of (4), we employ the Courant-Friedrichs-Lewy (CFL) condition from [She02b], which relates the temporal resolution, *i. e.* the time step of the numerical integration, to the spatial resolution, *i. e.*, the discretization of the mesh. The CFL condition stems from the observation that if a wave (*e. g.*, a pressure wave in our case) is crossing a discrete grid, then the time step must be less than the time for the wave to propagate through an element. More precisely, the CFL condition relates the time step of the numerical integration to the vibration mode of the underlying discretization, which is in turn inversely proportional to the propagation time of a pressure wave:

$$\Delta t < \frac{c}{\sqrt{\omega_{max}^2}}, \tag{5}$$

where Δt is the time step of the numerical integration and ω_{max} is the largest modal frequency of the body. The constant c corresponds to the stability radius of the state transition matrix [QSS00] of the underlying numerical integration scheme. In addition, c may also depend on the damping coefficient. Detailed derivations for different solvers can be found throughout the literature. Gurfil and Klein [GK07] studied the explicit Euler integration and derived $c = 2\xi$, where ξ is the damping coefficient. The explicit Euler is never stable if no damping is present. Müller *et al.* [MHTG05] present a detailed derivation for the leap frog integration scheme without damping, which results in $c = 2$. For the Verlet scheme, Klein [Kle07] derives $c = 2$ for the undamped and $c = 2(\sqrt{\xi^2+1} - \xi)$ for the damped integration scheme.

In order to determine the vibration modes of the mesh, (4) has to be transformed to a diagonal form, which is accomplished with the *whitening transform* [PW89]. This results in the generalized Eigenvalue problem

$$\Lambda\Phi = (\mathbf{M}^{-1}\mathbf{K})\Phi. \tag{6}$$

where the diagonal matrix Λ contains the eigenvalues of $\mathbf{M}^{-1}\mathbf{K}$, and the columns of the matrix Φ correspond to the eigenvectors of $\mathbf{M}^{-1}\mathbf{K}$. The eigenvalues $\lambda_i = [\Lambda]_{ii}$ correspond to the squared modal frequencies ω_i^2, and the eigenvectors $[\Phi]_i$ correspond to the directions along which the vibrations work. By plugging $\omega_{max}^2 = \lambda_{max} := \max_i \lambda_i$ into (5), we can determine the maximum time step allowed for the stable simulation of the object.

However, we are not exactly where we want to be. Firstly, we can now estimate a maximum time step for the whole object, without knowing which elements are particularly ill-shaped. However, as a single tetrahedron may make a simulation unstable, we must determin such tets. Secondly, the numerical solution of (6) is expensive and cannot be done in real-time. Thus, in the following, we show how to approximate (6) by looking at each element in isolation.

4.3 Element vibration modes

Fried [Fri72] showed that it is possible to define a reasonable upper bound for the eigenvalues of $\mathbf{M}$ and $\mathbf{K}$ by considering each element e in isolation. These upper bounds depend on the maximal eigenvalues λ_{max} of the element matrices $\mathbf{M}_e, \mathbf{K}_e$ and the maximum vertex degree n of the mesh,

$$\begin{aligned} \max_e \lambda_{max}(\mathbf{K}_e) \le \lambda_{max}(\mathbf{K}) \le n \max_e \lambda_{max}(\mathbf{K}_e) \\ \max_e \lambda_{max}(\mathbf{M}_e) \le \lambda_{max}(\mathbf{M}) \le n \max_e \lambda_{max}(\mathbf{M}_e) \end{aligned} \tag{7}$$

where $\lambda_{max}(\cdot)$ corresponds to the maximum eigenvalue of the matrix passed as argument. By analogy of [Fri72], Shewchuk derived a similar estimation,

$$\max_e \lambda_{max}(\mathbf{J}_e) \le \lambda_{max}(\mathbf{M}^{-1}\mathbf{K}) \le n \max_e \lambda_{max}(\mathbf{J}_e), \tag{8}$$

where $\mathbf{J}_e = \tilde{\mathbf{M}}_e^{-1/2}\mathbf{K}_e\tilde{\mathbf{M}}_e^{-1/2}$, $\tilde{\mathbf{M}}_e = \mathrm{diag}(m_i, m_j, m_k, m_l)$ and i, j, k, l are the nodes of e. Details are found in [She02b]. As a consequence, the maximum eigenvalue of $\mathbf{M}^{-1}\mathbf{K}$ is roughly proportional to the maximum eigenvalue of $\mathbf{J}_e$ of the worst element e. We employ a slightly different heuristic formulation which is derived from (8).

$$\lambda_{max}(\mathbf{M}^{-1}\mathbf{K}) \le \max_e \frac{1}{m_e}\lambda_{max}(\mathbf{K}_e) \tag{9}$$

where $m_e = \frac{1}{4}V_e\rho$ is the average mass of a corner of the element e with volume V_e and density ρ. In (8) the $\tilde{\mathbf{M}}_e$ and n are related. The masses of the nodes are the sum of the mass contributions of each tet. We use the observation that the ill-shaped elements typically have smaller masses than average mass $\tilde{\mathbf{M}}_e/n$. This gives an estimation of the eigenvalues and thus of the quality of an element. More precisely, if $\omega_{max}^2 = \frac{1}{m_e}\lambda_{max}(\mathbf{K}_e) \ge \left(\frac{c}{\Delta t}\right)^2$, it follows that the element e cannot be stably simulated with the time step Δt.

We note that this bound is very conservative; in practice, meshes can be found whose maximum frequency is up to a magnitude smaller than the frequency derived from (9).

4.4 Stiffness matrix assembly

We now employ the eigenvalues of $\mathbf{K}_e$ to compute the amount of material relaxation required for the stable simulation, and its eigenvectors to compute the direction of material relaxation. We first compute the matrix Λ_e by solving the corresponding 12×12 eigenvalue problem with a QR decomposition. The diagonal matrix $\Lambda_e = [\lambda_i]_{ii}, i = 1 \dots 12$ contains the squared oscillation frequencies of the element. If the frequencies exceed the maximum oscillation ω_{max} from (9) we filter the corresponding frequencies λ_i from Λ_e and form a new matrix $\Lambda_e' = [\lambda_i']_{ii}, i = 1 \dots 12$ with

$$\lambda_i' = \begin{cases} \lambda_i & \text{if } \lambda_i \le \omega_{max}^2 \\ \omega_{max}^2 & \text{if } \lambda_i > \omega_{max}^2 \end{cases} \tag{10}$$

From Λ_e', we assemble the *filtered* stiffness matrix $\mathbf{K}_e' = \Phi^T\Lambda'\Phi$. The modified material will react less to forces in the direction of the eigenvector corresponding to the modified modes. In the limit the material is infinitely soft along these directions. Thus, these modes have to be rigidified with constraints, as discussed subsequently.

The material modification can be done as a precomputation step for simulations without topological changes. In cutting or fracturing simulations, the filtering is done only for those elements whose topology has changed. This is possible due to the derived per-element approximation of the modal frequencies.

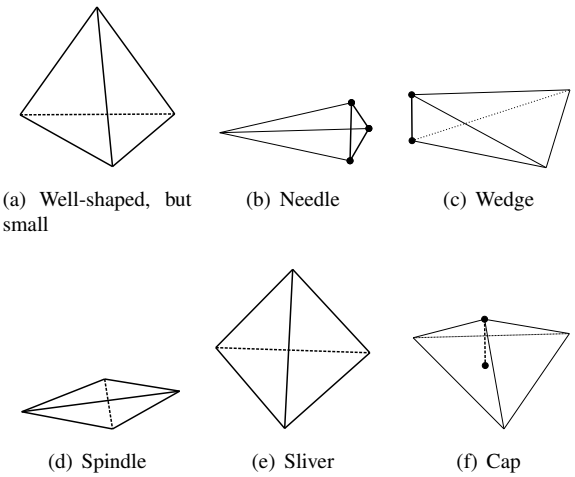
5 CONSTRAINING ELEMENT DEFORMATION

After having filtered the high-frequency vibration modes from ill-shaped elements, we obtain a stable

time integration. However, because the modified elements react less to forces in the direction of the removed modes, they tend to invert or distort. To prevent this, we replace these modes with constraints. These constraints try to keep the filtered elements as close as possible to the ground truth. The ground truth, in this case, is the element without material relaxation, and simulated with a sufficiently small time step.

To constrain the deformations of the filtered elements, we use the fast projection method [GHF*07], which is a manifold projection technique [HWL06]. The conceptual advantage is that fast projection can be implemented as a velocity filter, which directly modifies the future positions and velocities of the points after the numerical time-integration.

An intuitive way to rigidify the elements along the removed directions would be to directly employ the eigenvectors corresponding to the removed vibrations as constraint directions. However, since the eigenvectors and their derivatives cannot be computed symbolically, we instead classify the ill-shaped elements according to [BCER95]. For each type of degeneracy (see Figure 1), we use tailored constraints to rigidify the element along the filtered directions. This is because filtered directions align with the shortest distances in the tetrahedron, which can either be an edge or a height. If more than one vibration mode is filtered, or if multiple directions within the same element have been filtered, we rigidify the whole element.



(a) Well-shaped, but small (b) Needle (c) Wedge

(d) Spindle (e) Sliver (f) Cap

Figure 1: We distinguish 6 different element types [BCER95]. Highlighted line segments mark edge (b), (c) or height (f) constraints. The other elements (a), (d), (e) use volume constraints.

- Well-shaped elements as in Figure 1(a) can cause instability if they are small, compared to the average tet size in the mesh. Because all four heights are relaxed, we have to rigidify the element completely, which is done by constraining the lengths of all six edges of the tetrahedron. The length constraint $\mathbf{C}_e(\mathbf{x},\mathbf{y})$ for an edge $(\mathbf{x},\mathbf{y})$ is

$$C_e(\mathbf{x},\mathbf{y}) = \|\mathbf{y}-\mathbf{x}\|^2 - l_0^2 = 0 \qquad (11)$$

where $\mathbf{l}_0$ is the undeformed length of the edge.

- For needles, Figure 1(b), we use three edge constraints to approximate the relaxed heights. Edge constraints are shared between tetrahedra and thus reduce the overall number of constraints.

- For wedges, Figure 1(c), we use one edge constraints along the shortest edge.

- Spindles and Slivers, Figures 1(d) and 1(e), have relaxed directions perpendicular to the plane spanned by the crossing edges. As such all four heights have to be constrained. Again we use edge constraints to rigidify the whole element.

- Finally, for caps like in Figure 1(f) we set a height constraint along the shortest height. The function $\mathbf{C}_h(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{w})$ maintaining the distance h_0 between a tetrahedron corner $\mathbf{w}$ and its opposite face $(\mathbf{x},\mathbf{y},\mathbf{z})$ is

$$\begin{aligned}\mathbf{C}_h(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{w}) &= \frac{(((\mathbf{y}-\mathbf{x})\times(\mathbf{z}-\mathbf{x}))\cdot(\mathbf{w}-\mathbf{x}))^2}{\|(\mathbf{y}-\mathbf{x})\times(\mathbf{z}-\mathbf{x})\|^2} - h_0^2 \\ &= 0 \qquad (12)\end{aligned}$$

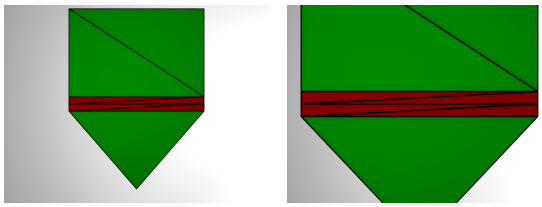
These constraints are symbolically differentiated and stacked in the constraint Jacobian matrix. Then, the fast projection method is employed to compute the feasible positions. The corresponding linear system is solved with an sparse Cholesky decomposition whose run-time is linear in the number of non-zero blocks. Therefore the method is linear in the number of constraints. Experiments indicate that very few iterations provide sufficient accuracy. Notice that in contrast to [TPS09], we do not impose constraints on the well-shaped tetrahedra, leading to significantly fewer constraints, which can be maintained efficiently.

6 RESULTS

In this section, we demonstrate our approach on a variety of dynamic simulations of deformable objects. The tetrahedral meshes have been generated with the approach described in [MT03]. To cut the meshes, we use the approach in [SHGS06] which tries to keep the number of newly created elements small. All experiments have been performed on an Intel Core2 Duo PC running at 2.5GHz.

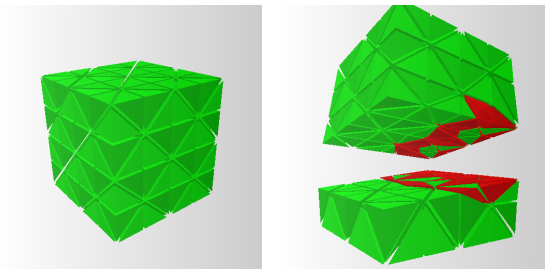
6.1 Stack of Wedges

We illustrate the working of our approach on a particularly simple object consisting of six ill-shaped, wedge-type tets, encompassed by well-shaped tets (see Figure 2). The edge length of the well-shaped tets is 1 m, the Young's modulus is $E = 10$KPa, the Poisson ratio is $\nu = 0.3$, and the density is $\rho = 0.05$kg m^{-3}. The wedges in the middle have one short edge and 5 long edges of approximately the same length. We obtain an edge ratio value of ~ 19, meaning that the longest edge is 19 times longer than the vertical edge. The computation of one simulation step takes 0.06ms. Due to the



(a) (b)

Figure 2: We illustrate the working of our approach on a particularly simple object consisting of six ill-shaped tets (red) encompassed by well-shaped tets (green). For this object, a speed-up of a factor of 3.5 is obtained by employing our approach. Right: Close-up of the six wedges.



(a) (b)

Figure 3: Our approach can also be employed in interactive cutting simulations. Left: Object before being cut. Right: After having cut the object into two parts. The cutting algorithm generates ill-shaped tets (red). By employing our method, we can keep a constant time step throughout the simulation.

ill-shaped tets, a small time step of $\Delta t_{max} = 0.03$ms is necessary to stably simulate the object, as we have determined experimentally. Notice that Δt_{max} could also have been derived from (9), but this would result in a much smaller time step because of the conservative estimation. Consequently, the computation of one physical second takes $1000/0.03 \times 0.06 = 2000ms= 2$ seconds, which is not real-time.

Now we assume that we have to simulate the object with a time step of $\Delta t = 0.7$ ms, for example because we aim at a real-time simulation. Thus, we want to relax all elements which *cannot* be simulated with Δt_{min}. By performing the eigenvalue decomposition element-wise, we isolate the six wedges and relax them along the vertical direction. Afterwards, we constrain the small edges of the wedges, resulting in 21 edge constraints that have to be enforced. The resulting relaxed object can now be simulated with a time step of $\Delta t = 0.7$ ms, as required. Notice that the new time step is more than 20 times larger than the initial one. The computation of one time step, including the constraint enforcement, takes 0.43 ms. Consequently, the simulation of one physical second takes $1000/0.7 \times 0.43 = 614$ms $= 0.61$ seconds, which is real-time. Table 1 summarizes the timings.

6.2 Cutting a Cube

Our method is particularly attractive in the context of cutting simulations. In Figure 3 we cut a cube of edge length 4 m consisting of 320 well-shaped elements with no dihedral angle smaller than 45°. The Young's modulus of the object is $E = 10$KPa, the Poisson ratio is $\nu = 0.3$, the density is $\rho = 0.06$kg m^{-3}. The cube is simulated with a time step of $\Delta t = 1$ms, and the time to compute one simulation step is 2.6ms. Thus, the simulation of one physical second takes $1000 \times 2.3 =$ 2300ms $= 2.3$ seconds. If we cut the cube, then the cutting method [SHGS06] refines the mesh in the region around the cut plane, thus generating 12 new tets and changing the geometry of 125 tets. Some of these tets will be ill-shaped, since we do not perform any further geometric optimizations. Consequently, the maximum possible time step after having cut the mesh would drop to $\Delta t_{max} = 0.42$ms, which is again experimentally determined. Thus, without our method, the computation of one physical second takes $1000/0.42 \times 2.3 = 5476$ms $= 5.4$ seconds.

We now demand that the time step stays constant throughout the simulation, *i. e.*, we relax all tets which cannot be simulated at $\Delta t = 1$ms. By performing the eigenvalue decomposition element-wise, we isolate 14 ill-shaped tets and introduce 30 constraints. By simulating the relaxed object, we can keep the time step $\Delta t_{rel} = \Delta t = 1$ms constant, as desired. The time to compute one simulation step has slightly increased to 2.4ms, which is due to the constraint handling. The computation of one physical second now takes $1000 \times 2.4 = 2400$ms $= 2.4$ seconds (see Table 1).

6.3 Liver Mesh

Generating tetrahedral meshes from surface meshes is a difficult problem, especially if conforming or surface-interpolating tetrahedral meshes are generated. Our method can be employed to simulate such meshes by employing larger time steps. To illustrate this, we consider a liver model composed of 407 tets (see Figure 4). The liver has a Young's modulus $E = 10$KPa, a Poisson ratio of $\nu = 0.3$, and a density $\rho = 1000$kg m^{-3}. 52 tets (colored red in Figure 4) have modal frequencies which are more than twice as high as the average value. These few elements require that the time step has to be halved. Looking at the classification of these tetrahedra, we can see that only six of them are considered ill-shaped. The rest are well-shaped but are very small and thus have a high modal frequency. If we dynamically simulate the liver without using our approach, we need a time step of $\Delta t_{max} = 1.2$ms. The time to perform one simulation step is 2.0ms, and therefore the time to compute one physical second is $1000/1.2 \times 2.0 = 1666$ms $= 1.6$ seconds.

Assuming that we want to simulate the liver with a time step of $\Delta t_{min} = 2$ms, the method relaxes the 52 elements, resulting in 120 constraints. The time to compute one simulation step increases to 2.6ms. However, the overall time to compute one second is now

Object	Δt	$t_{timestep}$	t_{second}
Wedges	0.03ms	0.06ms	2.0 sec.
	0.7ms	0.43ms	0.6 sec.
Cube (uncut)	1ms	2.3ms	2.3 sec.
	1ms	2.3ms	2.3 sec.
Cube (cut)	0.42ms	2.3ms	5.4 sec.
	1ms	2.4ms	2.4 sec.
Liver	1.2ms	2.0ms	1.6 sec.
	2ms	2.6ms	1.3 sec.

Table 1: This table summarizes the performance gains of our approach. Δt is the (maximum) time step of the numerical integration, $t_{timestep}$ is the time to compute one simulation step, and t_{second} is the time to compute one physical second. We simulate each object twice, once without employing our approach, and once by employing our approach. The last column indicates that in all scenarios we obtain a significant speed-up by employing our approach.

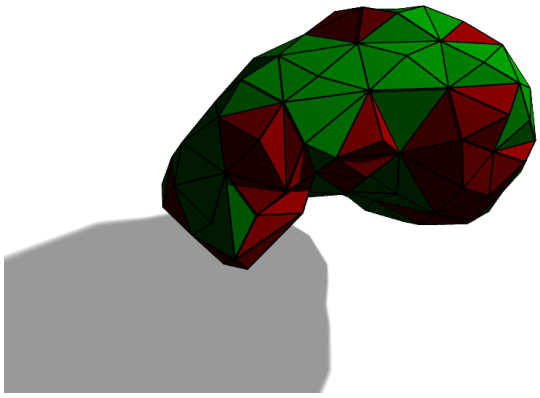


Figure 4: Our approach can also be employed in medical simulations. Here, a liver object is discretized into 407 elements. The ill-shaped tets (red) impose severe time step restrictions which can be relaxed by employing our approach.

$1000/2 \times 2.6 = 1300\text{ms} = 1.3$ seconds, which is a performance gain of 23% (see Table 1).

7 CONCLUSIONS

We have presented a method which allows us to determine ill-shaped elements based on a time step constraint. We relax the material of the ill-shaped elements in the direction of their largest vibration mode. To approximate the unmodified simulation behavior as closely as possible, we apply a velocity filter based on fast projection using constraints along the filtered directions.

Our method does not handle element inversion. But it can be combined with any inversion handling technique available, *e. g.*[ITF04] or [ST08]. Another limitation of our method is that the estimation of the elemental vibration modes is very conservative, which results in filtering more elements than actually necessary. This is because we consider each element in isolation, without looking at its neighborhood. By investigating this issue, the number of constraints could be reduced.

In addition, we want to include damping into the system. The material relaxation should stay unchanged, but updating the velocities in the fast projection step will change slightly. Additionally, we want to look at how compatible our system is with implicit integration schemes.

8 ACKNOWLEDGEMENTS

This research was supported by the EU project PASSPORT FP7 ICT-2007-223894.

REFERENCES

[ACSYD05] Alliez P., Cohen-Steiner D., Yvinec M., Desbrun M.: Variational tetrahedral meshing. ACM Transactions on Graphics (Proc. SIGGRAPH) 24, 3 (2005), 617–625.

[BCER95] Bern M., Chew P., Eppstein D., Ruppert J.: Dihedral bounds for mesh generation in high dimensions. in Proc. Symposium on Discrete Algorithms (1995), Society for Industrial and Applied Mathematics, pp. 189–196.

[BFA02] Bridson R., Fedkiw R., Anderson J.: Robust treatment of collisions, contact and friction for cloth animation. ACM Transactions on Graphics (Proc. SIGGRAPH) 21, 3 (2002), 594–603.

[CDRR04] Cheng S.-W., Dey T. K., Ramos E. A., Ray T.: Quality meshing for polyhedra with small angles. in Proc. Symposium on Computational Geometry (2004), pp. 290–299.

[Fri72] Fried I.: Condition of finite element matrices generated from nonuniform meshes. AIAA Journal 10 (1972), 219–221.

[GHF*07] Goldenthal R., Harmon D., Fattal R., Bercovier M., Grinspun E.: Efficient simulation of inextensible cloth. ACM Transactions on Graphics (Proc. SIGGRAPH) 26, 3 (2007), 49.

[GK07] Gurfil P., Klein I.: Stabilizing the explicit euler integration of stiff and undamped linear systems. Journal of Guidance, Control, and Dynamics 30, 6 (2007), 1659 – 1667.

[HSO03] Hauser K. K., Shen C., O'Brien J. F.: Interactive deformation using modal analysis with constraints. in Proc. Graphics Interface (2003), Canadian Human-Computer Commnication Society, pp. 247–256.

[Hug87] Hughes T.: The Finite Element Method. Prentice-Hall, 1987.

[HWL06] Hairer E., Wanner G., Lubich C.: Geometric Numerical Integration - Structure-Preserving Algorithms for Ordinary Differential Equations. Springer Berlin Heidelberg, 2006.

[ISS04] Ito Y., Shih A. M., Soni B. K.: Reliable isotropic tetrahedral mesh generation based on an advancing front method. in Proc. 13th International Meshing Roundtable (2004), pp. 95–106.

[ITF04] Irving G., Teran J., Fedkiw R.: Invertible finite elements for robust simulation of large deformation. in Proc. Symposium on Computer Animation (2004), Eurographics Association, pp. 131–140.

[KFCO06] Klingner B. M., Feldman B. E., Chentanez N., O'Brien J. F.: Fluid animation with dynamic meshes. ACM Transactions on Graphics (Proc. SIGGRAPH) 25, 3 (2006), 820–825.

[Kle07] Klein B.: FEM - Grundlagen der Anwendungen der Finite-Elemente-Methode im Maschinen- und Flugzeugbau. Vieweg, 2007.

[KS07] Klingner B. M., Shewchuk J. R.: Aggressive tetrahedral mesh improvement. in Proc. IMR (2007), Springer, pp. 3–23.

[LS07] Labelle F., Shewchuk J. R.: Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. ACM Transactions on Graphics (Proc. SIGGRAPH) 26, 3 (2007), 57.

[MG04] Müller M., Gross M.: Interactive virtual materials. in Proc. Graphics Interface (2004), Canadian Human-Computer Communications Society, pp. 239–246.

[MHTG05] Müller M., Heidelberger B., Teschner M., Gross M.: Meshless deformations based on shape matching. ACM Transactions on Graphics (Proc. SIGGRAPH) 24, 3 (2005), 471–478.

[MT03] Müller M., Teschner M.: Volumetric meshes for real-time medical simulations. in Proc. BVM (2003), pp. 279–283.

[PW89] Pentland A., Williams J.: Good vibrations: modal dynamics for graphics and animation. in Proc. SIGGRAPH (1989), ACM, pp. 215–222.

[QSS00] Quarteroni A., Sacco R., Saleri F.: Numerical Mathematics. Springer, 2000.

[RS09] R. Schmedding M. Gissler M. T.: Optimized damping for dynamic simulations. Proc. Spring Conference on Computer Graphics 25 (2009), 205–212.

[Sch97] Schoberl J.: Netgen - an advancing front 2d/3d-mesh generator based on abstract rules. Computing and Visualization in Science 1 (1997), 41–52.

[She98] Shewchuk J. R.: Tetrahedral mesh generation by delaunay refinement. in Proc. 14th Annual Symposium on Computational Geometry (1998), pp. 86–95.

[She02a] Shewchuk J. R.: What is a good linear element? interpolation, conditioning, and quality measures. in Proc. 11th International Meshing Roundtable (September 2002), pp. 115–126.

[She02b] Shewchuk J. R.: What is a Good Linear Element? Interpolation, Conditioning, and Quality Measures. Tech. rep., Carnegie Mellon University, 2002. Online version.

[SHGS06] Steinemann D., Harders M., Gross M., Szekely G.: Hybrid cutting of deformable solids. in Proc. IEEE conference on Virtual Reality (2006), IEEE Computer Society, pp. 35–42.

[Shi04] Shinya M.: Stabilizing explicit methods in spring-mass simulation. in Proc. Computer Graphics International (2004), IEEE Computer Society, pp. 528–531.

[ST08] Schmedding R., Teschner M.: Inversion handling for stable deformable modeling. The Visual Computer 24, 7 (2008), 625–633.

[TPS09] Thomaszewski B., Pabst S., Straßer W.: Continuum-based strain limiting. Computer Graphics Forum (Proc. Eurographics) 28 (4 2009), 569–576.

[WT08] Wojtan C., Turk G.: Fast viscoelastic behavior with thin features. ACM Transactions on Graphics (Proc. SIGGRAPH) 27, 3 (2008), 1–8.

A Threefold Representation for the Adaptive Simulation of Embedded Deformable Objects in Contact

Martin Seiler
ETH Zürich, Switzerland
seiler@vision.ee.ethz.ch

Jonas Spillmann
ETH Zürich, Switzerland
spillmann@vision.ee.ethz.ch

Matthias Harders
ETH Zürich, Switzerland
mharders@vision.ee.ethz.ch

ABSTRACT

We propose an approach for the interactive simulation of deformable bodies. The key ingredient is a threefold representation of the body. The deformation and dynamic evolution of the body is governed by a cubic background mesh. The mesh is hierarchically stored in an octree-structure, allowing for a fully local adaptive refinement during the simulation. To handle collisions, we employ a tetrahedral mesh, allowing for an efficient collision detection and response. We then show a physically-plausible way to transfer the contact displacements onto the simulation mesh. A high-resolution surface is embedded into the tetrahedral mesh and only employed for the visualization. We show that by employing the adaptive threefold representation, we can significantly improve the fidelity and efficiency of the simulation. Further, we underline the wide applicability of our method by showing both interactive and off-line animations.

Keywords: Physically-Based Animation, Deformable Models, Collision Handling, Finite Elements, Adaptivity

1 INTRODUCTION

Interactive simulation of soft tissue calls for both efficient and physically-plausible methods to model the non-linear deformations. To accomplish this, the finite element method (FEM) is commonly employed. However, the real-time simulation of larger systems with many degrees-of-freedom (DOF) quickly exceeds the capabilities of today's computer hardware. Still, a large number of DOF are necessary to faithfully reproduce the deformations. In this paper, we discuss two strategies to make the simulation of geometrically-complex deformable bodies compatible with the real-time constraints, notably the *adaptive* simulation, and the *embedding* approximation.

In an adaptive simulation of deformable bodies, as *e. g.* discussed by Debunne *et al.* [6], the DOF are dynamically arranged in *regions of interest*, and removed from the undeformed parts. This is particularly attractive in the context of surgery simulations where the interaction between the soft tissue and a surgery tool is considered. Here, the regions of interest are commonly spatially narrow regions around the tip of the tool.

The second strategy, called embedding, is orthogonal to the adaptive simulation. In order to animate a geometrically-complex surface, the surface is embedded into a coarse simulation mesh. The deformation is then governed by the simulation mesh, and the surface vertices are interpolated. Although the embedding technology is known since long [33, 23], it gained increasing attention with the recent advances of Nesme *et al.* that consider the varying material properties of the embedded body in the derivation of the stiffness matrices of the simulation mesh [24]. Still, the unification of an adaptive simulation with the embedding strategy in the context of an interactive simulator is not well investigated.

Contribution In this paper, we propose a method for the adaptive real-time simulation of complex deformable bodies. The key ingredient is a threefold body representation of the body, consisting of a coarse simulation mesh, a mid-resolution tetrahedral collision mesh, and a high-resolution surface for the visualization.

The coarse *simulation mesh* is a non-conforming rectangular grid composed of cubic elements, and governs the deformation and dynamic evolution of the body. The cubic elements allow for an octree representation, and enable the adaptive refinement. The surface-interpolating *tetrahedral mesh* is exclusively employed to detect and resolve collisions. This is because tetrahedra are well-suited for many collision detection schemes. Moreover, the tetrahedra approximate the surface of the body much better than the cubic elements of the simulation mesh, and therefore allow for a precise and physically-plausible handling of boundary conditions. We then present a formulation for transferring the contact displacements back on the simulation mesh such that the momentum is preserved. This contrasts previous approaches that commonly performed the collision handling directly on the surface and therefore degrade the performance if high-resolution surfaces are employed. By employing our threefold body representation, we can

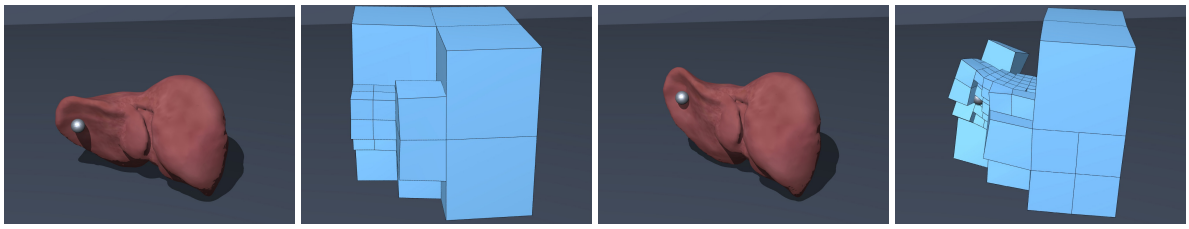


Figure 1: Adaptive resoultion of simulation mesh. A liver model is deformed by a human guided tool. Interactive simulation of a deformable liver. The adaptive resolution allows for an efficient simulation, since the DOF are arranged in the region of interest around the tool. The high-resolution surface is exclusively employed for the visualization.

adaptively simulate complex, highly deformable bodies at interactive rates, as shown in Fig. 1.

2 RELATED WORK

The physically-based simulation of soft bodies is an active area of research. For an overview, we refer to the excellent survey of Nealen *et al.* [25]. In this paper, we focus on the interactive real-time simulation of soft bodies. Due to their simplicity and efficiency, mass-spring systems [2, 33, 11] are frequently used. However, it is not easy to preserve the global deformation behavior of an object under adaptive refinement. The finite element method (FEM) builds directly on the laws of continuum mechanics [13] and is therefore better suited for the adaptive simulation. We employ a linear FE method with a co-rotational stiffness tensor, allowing to reproduce large deformations [17, 18].

Embedding To improve the efficiency, we consider an embedding strategy where a detailed surface mesh is embedded in a coarse simulation mesh [5, 33]. The surface is commonly embedded by linear interpolation, although other techniques such as harmonic coordinates (suitable for arbitrary polygons) [14, 20] or moving least squares [15] have been proposed. Recently, Nesme *et al.* have proposed to compute a kinematic relation between the coarse and the surface nodes by performing a static FE analysis, and then use this relationship for the embedding [24]. We employ linear embeddings since this is favorable in the context of an adaptive simulation.

The usage of cubic elements in the simulation mesh is favorable in the context of adaptive refinement. However, one disadvantage of a hexahedral simulation mesh is that its boundary is not conforming. Therefore, boundary conditions arising from, *e. g.*, contact handling cannot be easily imposed directly on the simulation mesh. Some authors [10, 29, 8] impose the boundary conditions directly on the surface mesh, but this degrades the performance in case of highly detailed surface meshes. Therefore, we propose a novel threefold representation where a low-resolution non-conforming hexahedral mesh governs the deformation, a mid-resolution tetrahedron mesh is employed to detect collisions and impose boundary conditions, and a high-resolution triangle mesh is used for visualization.

Adaptive deformation A widely used strategy to further accelerate the simulation of soft bodies is to employ an adaptive framework. This is usually accomplished by considering a hierarchy of meshes at different resolutions. Then, based on some level-of-detail criterion, each portion of an object can be simulated at a different resolution, and the physical properties are interpolated in between the different resolutions [6]. In case of tetrahedral meshes, the hierarchy is usually precomputed [35, 26] since dynamic splitting and merging of tetrahedra is a difficult task. This is where the use of a hexahedral simulation mesh becomes most evident. This is because the octree-refinement of hexahedrons is particularly simple [7, 23]. In turn, the different resolutions do not need to be pre-computed; instead, the adaptation can be done on the fly. This is particularly impressive demonstrated in [23] where the refinement level is governed by the position and orientation of the camera.

One problem inherent to multi-resolution is the lack of compatibility between elements at different resolutions, resulting in *incompatible nodes* or *T-junctions*. Apart from the strategy of avoiding incompatible elements by, *e. g.*, employing a red-green refinement strategy [16] or extending the model to arbitrary polygons [34, 20], there exist a wealth of approaches to enforce the compatibility explicitly. For example, this can be done by imposing constraints and solving them accordingly, *e. g.*, with the method of Lagrange multipliers [4]. The discontinuous Galerkin FE method (DG-FEM) treats each element in isolation, and employs penalty forces to weakly enforce compatibility [15]. However, it comes with the drawback of simulating more DOF than are actually desired in the simulation. A similar strategy that neglects the deformation of the elements is presented in [3]. As an alternative to these works, hierarchical multi-resolution approaches [9, 23] consider the interaction of an incompatible node with *all* its direct and indirect parents, thereby avoiding a special treatment of the constraints. However, the resulting system tends to be denser, which influences the performance of an implicit solver. We follow the approach of Sifakis *et al.* which propagates the force gradients between incompatible nodes [30]. This approach has the key advantage that

the incompatible nodes do not need to be simulated, and no special constraint handling is necessary.

3 OVERVIEW

We employ a threefold representation to simulate the deformable body. The starting point is a high-resolution surface obtained from, *e. g.*, a geometric modeling process or an MRI scan. From this surface, we compute a tetrahedral mesh whose boundary nodes interpolate the surface. The tetrahedral mesh defines the material distribution of the body. We underline that since the tetrahedral mesh is not employed for the deformation computation, the tetrahedra do not need to be particularly well-shaped.

To dynamically simulate the deformable body, the tetrahedral mesh is embedded into a coarse rectangular simulation mesh whose nodes correspond to Lagrangian mass points with associated dynamic properties. The simulation, including the handling of collisions, is done by employing a standard manifold projection method, which first assembles the global stiffness matrix $\mathbf{K}$, then performs an unconstrained time-evolution to obtain unconstrained positions $\tilde{\mathbf{x}}(t+h)$, and then projects the solution back on the geometric manifold imposed by the contact constraints to obtain non-colliding positions $\mathbf{x}(t+h)$. Since the contact handling is done with the tetrahedral mesh, the positions $\tilde{\mathbf{x}}_c$ of the tetrahedral mesh nodes must be interpolated from the simulated points, and the resulting feasible positions $\mathbf{x}_c$ must be transferred back on the simulation mesh afterwards. As a post-processing step, an oracle, *i. e.* a heuristic decision maker, performs the adaptive refinement in order to arrange the DOF according to the regions of interest.

repeat
  $\mathbf{K} \leftarrow \text{AssembleStiffnessMatrix}(\mathbf{x}(t))$
  $\tilde{\mathbf{x}}(t+h) \leftarrow \text{TimeEvolution}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{K})$
  $\tilde{\mathbf{x}}_c \leftarrow \text{UpdateCollisionMesh}(\tilde{\mathbf{x}}(t+h))$
  $\mathbf{x}_c \leftarrow \text{ComputeFeasiblePositions}(\tilde{\mathbf{x}}_c)$
  $\mathbf{x}(t+h) \leftarrow \text{TransferFeasiblePositions}(\mathbf{x}_c)$
  $\text{AdaptiveRefinement}(\mathbf{x}(t+h))$
until stop;

4 SIMULATION MESH

The simulation mesh is a partition of the simulation domain into cubic elements of different sizes. An octree is employed to store the hierarchy of elements. Since we opt for a pure local refinement of elements, we need to handle the resulting mesh incompatibility, *i. e.*, we need to detect and handle the T-junctions. Having in mind that we want to adaptively refine and un-refine the mesh on-the-fly, we first describe a compact representation of the intrinsic coordinates of the simulation elements and nodes. We then describe an approach to detect the T-junctions and discuss the deformation model as well as the numerical integration method.

4.1 Location coordinates

A path to an element of the octree is determined by the information whether we continue on the left or right side of three split planes. This corresponds to a sequence of binary decisions per dimension, with a 0 for left, and a 1 for right. We now store this decisions for each dimension separately and obtain a 3-tuple of bit sequences

$$S_i = \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix} = \begin{bmatrix} x_1 x_2 \ldots \\ y_1 y_2 \ldots \\ z_1 z_2 \ldots \end{bmatrix}$$

where, *e. g.*, $x_1 \in [0,1]$ encodes the information whether the element lies on the left or right side of the top-level plane splitting the X-axis. We denote these 3-tuples as *location coordinate*, similar to the term *location code* [21, 27, 1]. Each component of the location coordinate is now represented in fixed point arithmetic. The main benefit of this representation is that point coordinates can be exactly compared, thus avoiding the headaches that come with floating point comparisons. The navigation within the octree can now be efficiently done by adding appropriate offset vectors.

4.2 Adaptive refinement

By having assigned a unique location coordinate to each element within the octree, we have now the tools in hand to efficiently refining refine elements. We refine an element if the deformation energy of the element is above a certain threshold. This simple heuristic is based on the observation that contacts induce deformations. Then, the DOF must be arranged in order to reproduce the deformation, as we will show later. If, in contrast, the deformation energy of a refined element is below a second threshold, then we merge its sub-elements. It is important not to merge strongly deformed elements, since this results in popping artifacts [26].

To refine, we recursively split an element into eight sub-elements of same size. Further, we underline that the refinement is purely local, without restriction on the level difference between adjacent elements, which is in contrast to[7]. This avoids that refinements are propagated throughout the entire mesh, which would degrade the performance.

However, at this point we have to consider that the original simulation mesh does not correspond to the simulated body, but to the cubic bounding-box of the *embedded* body. Therefore, after a refinement, some elements might not contain any material (see Fig. **??**). These void elements do not need to be simulated. Thus, after having refined an element, we test each sub-element for material intersection, *i. e.*, we determine the intersection of the sub-elements with the tetrahedral mesh and remove the empty elements from the simulation.

Following [6], we interpolate the positions and velocities linearly. In order to adjust the mass contribu-

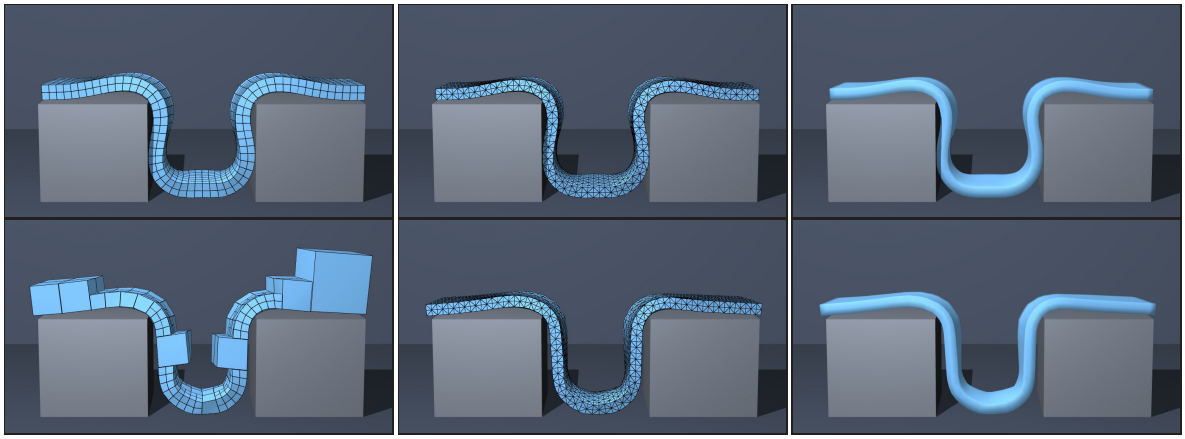


Figure 2: We employ a threefold representation of the body, notably a deformation mesh, a collision mesh, and an embedded surface. The *top row* shows the result of a full simulation while *bottom row* shows the same scene but with an adaptive simulation, where elements are refined in regions of large deformation.

tions of the sub-elements, we iterate over the embedded tetrahedral corner points and distribute their masses barycentrically on the simulation nodes. In addition, we scale the stiffness matrix in order to account for elements that are only partially filled with material. As Nesme *et al.* point out, this improves the stability, because lighter elements are now also softer [23].

4.3 Detecting T-junctions

T-junctions are inevitable if elements in an octree are refined in isolation, *i. e.*, without considering the neighborhood of the element. While refining elements in isolation is favorable with respect to efficiency and implementation ease, the T-junctions require a special treatment in order to avoid cracks during the simulation. To check whether a given node is a T-junction, we traverse its set of at most eight adjacent elements, and check whether the element is not empty *and* the node is not a corner of that element. Since T-junctions are not known a priori, this has to be done for each node. Further, since the height of the unbalanced octree is in $\mathcal{O}(N)$, the T-junction detection is in $\mathcal{O}(N^2)$ in theory. Still, since numerical issues forbid an unlimited refinement, the height of the tree can be considered as constant in practice. By imposing restrictions on the level differences, linear-time T-junction detection algorithms can be designed [7, 1], but this comes at the price of a non-local refinement.

4.4 Dynamic deformation model

The adaptive refinement of the simulation mesh results in a complex of cubic elements and mass points. To compute the elastic response of the deformable body, we rely on the linear co-rotational FE method as discussed in, *e. g.*, [18]. This approach has the advantage that for each element, the stiffness tensor $\mathbf{K}_e$ depends solely on the un-deformed point coordinates and an element rotation $\mathbf{R}_e$. Since in the element frames, the elements are axis-aligned cubes, we can pre-compute the elemental stiffness tensors in an analytical way, see Appendix. To estimate the rotation, we follow the approach described in [19]. We then assemble a global stiffness matrix $\mathbf{K}$ from the elemental stiffness matrices $\mathbf{K}_e$.

To compute the dynamic evolution of the deformable body, we assume that the mass points obey the Newton equations of motion. The time-dependent positions and velocities of the points then result from numerically integrating the equations of motion with a semi-implicit solver. We use the pre-conditioned conjugate gradient from the Taucs library[1] to solve the linear system.

4.5 T-junction handling

At this point, we have considered all nodes of the simulation mesh as DOF. However, this is not true for the T-junction nodes which are barycentrically constrained to edges or faces of adjacent elements. To enforce these constraints and avoid cracks in the simulation, we follow an approach proposed by Sifakis *et al.* in [30] and later generalized by Nesme *et al.* [24]. The key observation is that there exists a linear relation between the coordinates $\mathbf{x}_{full}$ of the full system, and the coordinates $\mathbf{x}_{redu}$ of the reduced system containing only real DOF,

$$\mathbf{x}_{full} = \mathbf{W}\mathbf{x}_{redu} \tag{1}$$

where the matrix $\mathbf{W}$ contains the barycentric coordinates of the T-junctions with respect to their parents [30]. Nesme *et al.* have shown that a similar relation holds between the nodal forces,

$$\mathbf{f}_{redu} = \mathbf{W}^T\mathbf{f}_{full} \tag{2}$$

where the vector $\mathbf{f}_{full}$ contains the forces on *all* points [24]. The operator $\mathbf{W}^T$ distributes these onto the reduced points, excluding the hard bound points. Since

[1] http://www.tau.ac.il/~stoledo/taucs/

we employ an implicit solver, we assemble the global stiffness matrix likewise as

$$\mathbf{K}_{redu} = \mathbf{W}^T \mathbf{K}_{full} \mathbf{W}. \quad (3)$$

We now employ the reduced stiffness matrix $\mathbf{K}_{redu}$ to solve for the future positions. The advantage of this approach is that the dimension of the system is reduced by the number of T-junctions, and that no special constraint treatment is required. The downside, however, is that assembling the reduced system is more expensive, and that the reduced system is denser.

5 TETRAHEDRAL MESH

While the simulation mesh governs the deformation and dynamic evolution of the body, boundary conditions such as contact constraints are imposed on the tetrahedral mesh. In this section, we describe how to update the tetrahedral mesh, handle collisions, and transfer the displacements back onto the simulation mesh.

5.1 Mesh update

We assume that each node of the tetrahedral mesh lies in exactly one element of the simulation mesh. Thus, we can derive a linear relation between the simulation mesh nodes $\mathbf{x}$ and the tetrahedral mesh nodes $\mathbf{x}_c$,

$$\mathbf{x}_c = \mathbf{H}\mathbf{x} \quad (4)$$

where the matrix $\mathbf{H}$ contains the barycentric coordinates of the tetrahedral mesh nodes with respect to the simulation mesh nodes. These correspond to the linear shape functions evaluated at $\mathbf{x}_c$. The matrix $\mathbf{H}$ needs to be recomputed only after an adaptive refinement.

Consequently, having computed the (colliding) future simulation mesh positions $\tilde{\mathbf{x}}(t+h)$, we update the tetrahedral mesh nodes with $\tilde{\mathbf{x}}_c(t+h) = \mathbf{H}\tilde{\mathbf{x}}(t+h)$. Then, the collision handling is performed on the (colliding) positions $\tilde{\mathbf{x}}_c$, as described in the following.

5.2 Collision handling

To detect inter-object collisions, we employ a spatial hashing approach [32]. Then, the penetration depths are computed with the approach described in [12]. Since both approaches rely on a tetrahedral discretization, they go hand in hand with our threefold representation.

To resolve collisions, we employ a position-based scheme similar in spirit to [28] and [31]. That is, we resolve each point-triangle collision locally while conserving the momentum. This process is repeated iteratively until all penetrations are resolved. Experiments indicate that 5-10 iterations provide sufficient accuracy. The result of the computation are collision-free positions $\mathbf{x}_c$ and corresponding displacements $\mathbf{d}_c = \mathbf{x}_c - \tilde{\mathbf{x}}_c$.

5.3 Transferring the displacements

Having computed the collision displacements, we have to transfer the solution back onto the simulation mesh in order to realize the collision response. Still, this has to be accomplished such that the momentum of the simulation mesh is as well preserved. Moreover, one has to consider that the resolutions of the simulation mesh and the tetrahedral mesh might differ significantly.

To accomplish this, we again resort to [24] which relates the forces $\mathbf{f}$ on the coarse nodes to the forces $\mathbf{f}_c$ on the embedded points as

$$\mathbf{f} = \mathbf{H}^T \mathbf{f}_c. \quad (5)$$

To apply this relation, we write $\mathbf{f}_c = \frac{1}{2}h^{-2}\mathbf{M}_c\mathbf{d}_c$, where $\mathbf{M}_c$ is the diagonal mass matrix of the tetrahedral mesh. Then, $\mathbf{f} = \frac{1}{2}h^{-2}\mathbf{H}^T\mathbf{M}_c\mathbf{d}_c$. Still, it is favorable to handle collisions on the velocity level. Thus, we convert the resulting contact forces $\mathbf{f}$ back to contact displacements $\mathbf{d}$, and obtain the final displacement transfer formulation as

$$\mathbf{d} = 2h^2\mathbf{M}^{-1}\left(\frac{1}{2}h^{-2}\mathbf{H}^T\mathbf{M}_c\mathbf{d}_c\right) = \mathbf{M}^{-1}\mathbf{H}^T\mathbf{M}_c\mathbf{d}_c \quad (6)$$

where $\mathbf{M}$ is the diagonal mass matrix of the simulation mesh nodes. We then update the positions with $\mathbf{x}(t+h) \leftarrow \tilde{\mathbf{x}}(t+h) + \mathbf{d}$, and the velocities with $\dot{\mathbf{x}}(t+h) \leftarrow h^{-1}(\mathbf{x}(t+h) - \mathbf{x}(t))$. The key benefit of this formulation is that the momentum is preserved, which is crucial for the physical plausibility of the simulation.

To understand the consequence of this transfer formulation, one has to consider the case where many tetrahedral points are embedded in a single element. In this case, the masses of the simulation nodes are larger than the masses of the collision nodes. Consequently, the resulting simulation node displacements $\mathbf{d}$ are smaller than the computed displacements $\mathbf{d}_c$, and the collisions cannot be resolved. This problem can only be overcome by dynamically refining the coarse elements in the region of the interaction. In turn, this results in smaller and lighter elements that can properly react to the collision. Therefore, a good adaptation oracle is crucial.

6 SURFACE MESH

In order to improve the fidelity, we employ a high-resolution triangle surface for the visualization. This surface is linearly embedded in the tetrahedral mesh, having the advantage that the barycentric coordinates do not need to be re-computed after refinement of the simulation mesh. Another benefit is that since we compute the non-colliding positions of the tetrahedral nodes and update the vertices of the surface mesh thereafter, visible collisions are avoided even if the simulation mesh fails to reproduce the deformation accurately enough, as discussed before.

Scene		High-res	Adaptive
Bar	element count	1024	265
	ms / time step	120	17
	speed-up factor	–	7x
Plane	element count	896	75
	ms / time step	102	8
	speed-up factor	–	13x

Table 1: Performance measurements of our method. The first row shows the timings of the adaptive bar simulation illustrated in Fig. 2. For this scenarios, we obtain a speed-up a factor of 7 by employing the adaptive model. The second row illustrates that for the interactive plane scene (see Fig. 4), we obtain a speed-up of a factor of 13 since we only refine the mesh in the region of interest around the tool.

7 RESULTS

In this section, we evaluate the performance of our method, and demonstrate our method in various challenging interactive and off-line scenarios. We underline that the pre-computation time of our method is negligible. All experiments have been performed on an Intel quad-core CPU @ 2.6 GHz with a GeForce GTX 280.

7.1 Performance

To show that the application of our method results in a significant performance gain without compromising the accuracy, we perform an experiment where an elastic bar is laid between two rigid obstacles (see Fig. 2). The length of the bar is 0.2m, its density is 2×10^5 kg m^{-3}, the Young modulus is 5×10^5 Pa, and the Poisson ratio is 0.4. We perform the experiment with a non-adaptive high resolution simulation mesh and compare it with an adaptive simulation mesh. The measurements (see Tab. 1) indicate a speed-up factor of seven with only minor loss in visual quality.

7.2 Examples

First we verify that our method yields comparable results as a reference simulation: we use tetrahedral FEM to simulate the same body at a higher resolution. Fig. 3 shows for one example that this is indeed the case.

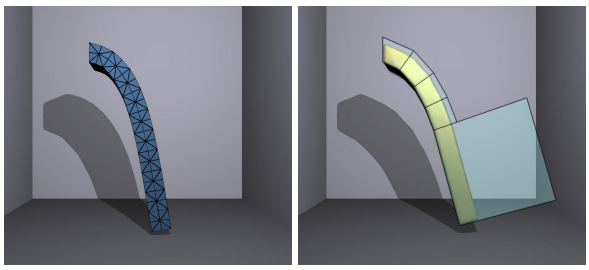


Figure 3: Qualitative comparison of a hanging bar. Left: The deformation mesh of the tetrahedral FEM consists out of 800 elements. Right: Our method embeds the body into 9 cubic elements.

Our method is particularly attractive in scenarios where a user interacts in a spatially coherent area of a large object. To illustrate this, we perform an experiment where the user interacts with an elastic plane lying on the ground (see Fig. 4). The size of the plane is 0.2m $\times$ 0.2m, its density is 400kg m^{-3}, the Young modulus is 3×10^4 Pa, and the Poisson ratio is 0.4. Since we employ a co-rotational stiffness tensor, large deformations can be reproduced plausibly, as illustrated in Fig. 4. There are on average 75 elements simulated, and the tetrahedral collision mesh contains 8k elements. The average computation of one simulation step takes 8ms, where 10^{-3}ms are spent for the adaptive (un-)refinement. The simulation runs at 19 frames per second on average. A similar performance cannot be achieved by employing a non-adaptive model (see Tab. 1). In this case, the frame rate drops to 3 fps, and an interactive manipulation is not possible.

To show that our method can also be employed for a complex off-line simulation, we drop 81 elastic toy bikes discretized into 384 tetrahedra on the ground (see Fig. 5). The computation of the dynamics of up to 3k elements takes 70ms per time step on average. The detection of 220 collisions takes 20ms and the computation of the non-colliding positions, including transferring the displacements, takes 7ms. The surface meshes of the bikes contain 13k triangles each, performing the collision handling on the surfaces directly would therefore result in a significant computational overhead. In total, the simulation takes about 7s per frame, including the rendering of 2.5 million surface triangles.

8 CONCLUSION

In this paper, we have presented an approach for robust and efficient simulation of deformable bodies. The key ingredient has been a novel threefold representation, where a coarse simulation mesh governs the deformation and dynamic evolution, a tetrahedral mesh is employed to handle collisions, and a high-resolution surface is visualized. The simulation mesh is a partition of the domain into axis-aligned cubes, and therefore enables a straight-forward adaptive refinement. We have then discussed an efficient approach to detect the T-junctions. A linear co-rotational FE method, in tandem with an implicit numerical integration method, is employed to compute the dynamic evolution.

We have further discussed a physically-plausible way to update the embedded tetrahedral mesh, and afterwards transfer the collision displacements back onto the deformation mesh. Our formulation preserves the momentum and is therefore physically plausible. Experiments indicate that we can simulate interacting deformable bodies at interactive rates.

As future work, we plan to adapt the stiffness tensor in order to reflect the embedded material, similar in spirit to Nesme *et al.* [24]. In addition, we will address topological changes such as cutting and fracture.

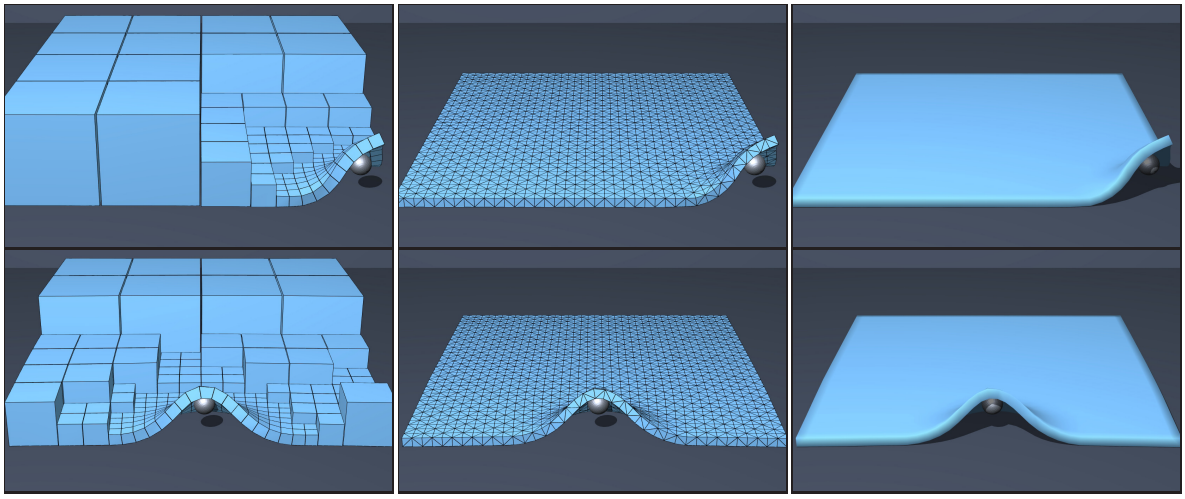


Figure 4: By employing an adaptive scheme, we can reproduce rich deformations at interactive rates. This is because a large object is refined locally, thus arranging the DOF in the regions of interest. To illustrate this, we perform a simulation of a deformable plane being manipulated by a user. The simulation runs at 19 frames per second on average. Left: The hexahedral simulation grid. Middle: The tetrahedral collision mesh. Right: The high-resolution surface mesh.

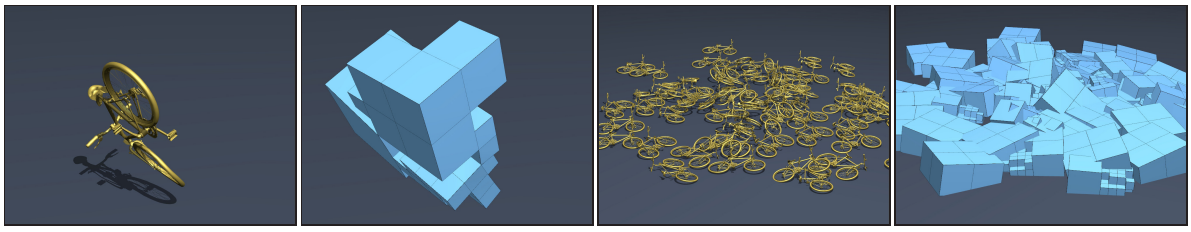


Figure 5: Massive scenario of bikes falling on the ground. The collisions induce deformations, which in turn make that the objects are being refined. The surface meshes of the bikes contain 13k triangles each. Since we perform the collision handling on the tetrahedral mesh, the simulation of the full scenario depicted on the right takes only 7s per frame.

This work has been supported by the Swiss CTI project ArthroS.

REFERENCES

[1] Aizawa K., Tanaka S.: A constant-time algorithm for finding neighbors in quadtrees. vol. 31, IEEE Computer Society, pp. 1178–1183.

[2] Bridson R., Fedkiw R., Anderson J.: Robust treatment of collisions, contact and friction for cloth animation. ACM Transactions on Graphics (2002), 594–603.

[3] Botsch M., Pauly M., Wicke M., Gross M.: Adaptive space deformations based on rigid cells. Computer Graphics Forum 26, 3 (2007), 339–347.

[4] Carey G. F.: Computational Grids: Generation, Adaptation and Solution Strategies. Taylor & Francis, 1997.

[5] Capell S., Green S., Curless B., Duchamp T., Popović Z.: Interactive skeleton-driven dynamic deformations. ACM Transaction on Graphics (Proc. SIGGRAPH) 21, 3 (2002), 586–593.

[6] Debunne G., Desbrun M., Cani M.-P., Barr A. H.: Dynamic real-time deformations using space and time adaptive sampling. in Proc. SIGGRAPH (2001), pp. 31–36.

[7] Dequidt J., Marchal D., Grisoni L.: Time-critical animation of deformable solids: Collision detection and deformable objects. Computer Animation and Virtual Worlds 16, 3-4 (2005), 177–187.

[8] Faure F., Barbier S., Allard J., Falipou F.: Image-based collision detection and response between arbitrary volumetric objects. in ACM Siggraph/Eurographics Symposium on Computer Animation, SCA 2008, July, 2008 (2008).

[9] Grinspun E., Krysl P., Schröder P.: CHARMS: a simple framework for adaptive simulation. ACM Transactions on Graphics (Proc. SIGGRAPH) 21, 3 (2002), 281–290.

[10] Galoppo N., Otaduy M. A., Tekin S., Gross M., Lin M. C.: Soft articulated characters with fast contact handling. Computer Graphics Forum 26, 3 (2007), 243–253.

[11] Georgii J., Westermann R.: Mass-spring systems on the gpu. Elsevier Science (July 2005).

[12] Heidelberger B., Teschner M., Keiser R., Mueller M., Gross M.: Consistent penetration depth estimation for deformable collision response. in Proc. Vision, Modeling, Visualization (2004), pp. 339–346.

[13] Hughes T.: The Finite Element Method. Prentice-

Hall, NJ, 1987.

[14] Joshi P., Meyer M., DeRose T., Green B., Sanocki T.: Harmonic coordinates for character articulation. ACM Trans. Graph. 26, 3 (2007), 71.

[15] Kaufmann P., Martin S., Botsch M., Gross M. H.: Flexible simulation of deformable models using discontinuous galerkin fem. Graphical Models 71, 4 (2009), 153–167.

[16] Molino N., Bridson R., Teran J., Fedkiw R.: A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. in Proc. 12th International Meshing Roundtable (2003), pp. 103–114.

[17] Müller M., Dorsey J., McMillan L., Jagnow R., Cutler B.: Stable real-time deformations. in Proc. ACM SIGGRAPH/Eurographics symposium on Computer animation (2002), pp. 49–54.

[18] Müller M., Gross M.: Interactive virtual materials. in Proc. Graphics Interface (2004), pp. 239–246.

[19] Müller M., Heidelberger B., Teschner M., Gross M.: Meshless deformations based on shape matching. ACM Transaction on Graphics (Proc. SIGGRAPH) 24, 3 (2005), 471–478.

[20] Martin S., Kaufmann P., Botsch M., Wicke M., Gross M. H.: Polyhedral finite elements using harmonic basis functions. Comput. Graph. Forum 27, 5 (2008), 1521–1529.

[21] Morton G.: A computer oriented geodetic data base and a new technique in file sequencing, 1966.

[22] Muller M., Teschner M., Gross M.: Physically-based simulation of objects represented by surface meshes. in CGI '04: Proceedings of the Computer Graphics International (2004), pp. 26–33.

[23] Nesme M., Faure F., Payan Y.: Hierarchical multi-resolution finite element model for soft body simulation. in 2nd Workshop on Computer Assisted Diagnosis and Surgery, March, 2006 (2006).

[24] Nesme M., Kry P. G., Jerábková L., Faure F.: Preserving topology and elasticity for embedded deformable models. in SIGGRAPH '09: ACM SIGGRAPH 2009 papers (2009).

[25] Nealen A., Müller M., Keiser R., Boxermann E., Carlson M.: Physically Based Deformable Models in Computer Graphics. in Eurographics-STAR (2005), pp. 71–94.

[26] Otaduy M., Germann D., Redon S., Gross M.: Adaptive deformations with fast tight bounds. in Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation (2007), pp. 181–190.

[27] Samet H.: The Design and Analysis of Spatial Data Structures (Addison-Wesley series in computer science). Addison-Wesley Pub (Sd), 1989.

[28] Spillmann J., Becker M., Teschner M.: Non-iterative computation of contact forces for deformable objects. Journal of WSCG 15, 1-3 (2007), 33–40.

[29] Steinemann D., Otaduy M. A., Gross M.: Efficient bounds for point-based animations. in Proc. IEEE/Eurographics Symposium on Point-Based Graphics (2007), pp. 57–64.

[30] Sifakis E., Shinar T., Irving G., Fedkiw R.: Hybrid simulation of deformable solids. in Proc. ACM SIGGRAPH/Eurographics symposium on Computer animation (2007), pp. 81–90.

[31] Spillmann J., Teschner M.: An Adaptive Contact Model for the Robust Simulation of Knots. Computer Graphics Forum (Proc. Eurographics) 27, 2 (2008), 497–506.

[32] Teschner M., Heidelberger B., Müller M., Pomerantes D., Gross M. H.: Optimized spatial hashing for collision detection of deformable objects. in Proc. Vision, Modeling, Visualization (2003), pp. 47–54.

[33] Teschner M., Heidelberger B., Muller M., Gross M.: A versatile and robust model for geometrically complex deformable solids. in Proc. Computer Graphics International (2004), pp. 312–319.

[34] Wicke M., Botsch M., Gross M.: A finite element method on convex polyhedra. Computer Graphics Forum (Proc. Eurographics) 26, 3 (2007), 355–364.

[35] Wu X., Downes M., Goktekin T., Tendick F.: Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. Computer Graphics Forum 20, 10 (2001), 349–358.

A CUBIC ELEMENT STIFFNESS

In this section, we discuss a way to analytically compute the element stiffness tensor $\mathbf{K}_e$, similar in spirit to [22]. For general hexahedra the usual approach is to use the isoparametric approach and perform a numerical integration of the resulting integral. For axis-aligned cubes this is not necessary and a simple parametrization of the integration domain is straight-forward. From continuum mechanics we know that a body's deformation energy is given by

$$E_{def} = \int_\Omega \frac{1}{2}\sigma^T \varepsilon dV = \int_{z_1}^{z_2}\int_{y_1}^{y_2}\int_{x_1}^{x_2} \frac{1}{2}\varepsilon^T \mathbf{D}\varepsilon \, dxdydz \qquad (7)$$

with the strain ε and the linear elasticity matrix $\mathbf{D}$. To interpolate the deformation within the element, we employ *natural coordinates*

$$N_i(\xi,\eta,\chi) = \frac{1}{8}(1+\xi_i\xi)(1+\eta_i\eta)(1+\chi_i\chi) \qquad (8)$$

where ξ_i, η_i, and χ_i denote the positions of the corners in natural coordinates. Then the deformation field $\mathbf{u}$ within the element is interpolated by $\mathbf{u}(\xi,\eta,\chi) = \sum_{i=1}^{8} N_i(\xi,\eta,\chi)\mathbf{u}_i$. The Cauchy strain $\varepsilon_c(\mathbf{u}) := \frac{1}{2}(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)$ depends on the spatial derivative

$$\nabla\mathbf{u} = [\frac{\partial\mathbf{u}}{\partial x}, \frac{\partial\mathbf{u}}{\partial y}, \frac{\partial\mathbf{u}}{\partial z}]^T = \mathbf{J}^{-1}[\frac{\partial\mathbf{u}}{\partial\xi}, \frac{\partial\mathbf{u}}{\partial\eta}, \frac{\partial\mathbf{u}}{\partial\chi}]^T \qquad (9)$$

where

$$\mathbf{J} = \begin{pmatrix} \frac{\partial x}{\partial\xi} & \frac{\partial y}{\partial\xi} & \frac{\partial z}{\partial\xi} \\ \frac{\partial x}{\partial\eta} & \frac{\partial y}{\partial\eta} & \frac{\partial z}{\partial\eta} \\ \frac{\partial x}{\partial\chi} & \frac{\partial y}{\partial\chi} & \frac{\partial z}{\partial\chi} \end{pmatrix} \qquad (10)$$

is the Jacobian matrix. For a general hexahedron, x, y and z are functions of ξ, η and χ, and therefore the partial derivatives are involved functions in the integration coordinates, requiring a numerical evaluation of the integral in (7). However, since we have cubes as integration elements, we obtain much simpler functions

$$x(\xi) = 1/2(x_2 - x_1)\xi \qquad (11)$$

$$y(\eta) = 1/2(y_2 - y_1)\eta \qquad (12)$$

$$z(\chi) = 1/2(z_2 - z_1)\chi \qquad (13)$$

with a corresponding constant and diagonal Jacobian $\mathbf{J}$. Thus the integral (7) can be evaluated analytically. The corresponding stiffness tensor results from basic FE theory [13].

Component-Based Isosurface Extraction for Multiple Dataset Visualization

Shiyuan Gu
Department of Computer Science,
Department of Mathematics,
Louisiana State University,
USA (70803), Baton Rouge, LA
gshy@math.lsu.edu

Bijaya B. Karki
Department of Computer Science,
Louisiana State University,
USA (70803), Baton Rouge, LA
karki@csc.lsu.edu

ABSTRACT

In the situations where isosurfaces are comprised of many disjoint components, two or more datasets can be visualized simultaneously by processing only a subset of isosurface components. The components of interest can be selected by exploiting interdataset coherency at the level of individual voxels and components. Thus, only those components (identified as voxel coverages or voxel sets) which differ significantly among the datasets under consideration are extracted as needed while the similar components were extracted only once from a reference dataset. Since the polygons are extracted/rendered as a whole component, the rendered isosurfaces are crack-free. We use three user-defined thresholds to control multiple dataset visualization (MDV) so that important relationships (differences and similarities) among the datasets can be explored with an improvement in the overall performance. If the data-coherency can not be defined easily, MDV can still benefit from the on-the-fly processing of the individual components.

Keywords: Multiple dataset visualization, data coherency, isosurface extraction, component, crack-free

1 INTRODUCTION

Multiple dataset visualization (MDV) is desirable for developing a more complete understanding of a given system or problem for which two or more datasets are available (e.g., [aoy07, chi97, kha06]). Unlike normal visualization in which each dataset is processed independent of other dataset(s), MDV processes multiple datasets of interest together so that the cross-correlations (such as differences and similarities) among them can be better explored. A straightforward approach is to completely process each dataset using an existing visualization technique (such as isosurface extraction [lor87]) and then simply display the resulting images together. There are two issues with this approach. Firstly, it is not always possible to process fast and completely all the datasets under consideration due to the limited computing and storing capabilities. For instance, a modern dataset can be arbitrarily large so minimizing the processing/storage requirements is crucial in MDV. Secondly, even if it is possible to handle all datasets simultaneously, it may not be effective for comparison purpose. For instance, there are situations where the underlying structures are highly complex and widely spread in a three-dimensional space. The users often face tremendous challenge in visually identifying the similar and dissimilar regions. It is, therefore, desirable that MDV can automatically detect and highlight such regions.

Several approaches have been explored in the context of visualization of multiple datasets. In [chi97], the authors have proposed visualization spreadsheets for comparing multiple datasets. By displaying the outputs for several datasets in a tabular form, the users can apply various operations for the cells of the spreadsheets (e.g., subtracting two cells and rendering the difference in another cell). A recent MDV work involves overlaying multiple visualizations within a single view or rendering them in multiple views [aoy07]. While these approaches primarily deal with the interfaces (i.e., how the display outputs are organized), the scalable adaptive MDV presented in [kha06, kha07] deals with visualization at the processing level using isosurface extraction and texture mapping.

To improve MDV performance, one can exploit the coherency between the datasets to be visualized ([kha08]). Data coherency is simply a measure of similarity between the datasets. In this approach, a given multiple set of data are first divided into two groups: reference datasets and non-reference datasets. The reference datasets are those which are completely processed and whose polygon data are used subsequently to also represent the parts of the isosurfaces of the non-reference datasets. The required data-coherency test is performed by comparing a non-reference dataset with a reference dataset block

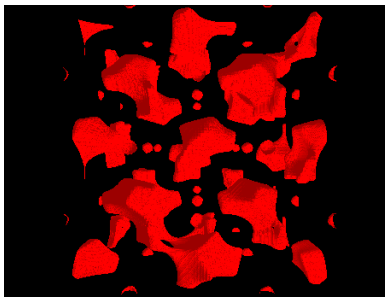


Figure 1: Multicomponent isosurface

by block. Only the polygons (triangles) within the blocks (i.e., octree nodes) which significantly differ from the corresponding blocks of the reference dataset are extracted; the polygons in other blocks are simply retrieved from the reference dataset.

In this paper, we perform component-based isosurface extraction in combination with interdataset coherency to support multiple dataset visualization. Our MDV is shown to be useful in the situations where the underlying isosurface structure is comprised of many small spatially disjoint components. An example is shown in Fig. 1 for the electron density isosurfaces. Here, different isosurface components correspond to electron distributions around different atomic sites. Similarly, in the case of medical data such components may correspond to different suborgans or tissue structures. A typical scenario can be that two or more datasets represent some changes which are localized. For instance, one vacancy site (an missing atom) in a crystal is likely to affect the electron distribution only in its neighbourhood. A component can thus change its size/shape freely without affecting the other components. Therefore, identifying and subsequently processing such individual components can be useful in comparing datasets and hence in MDV. As stated above, our method assumes the data coherency, i.e., it assumes that the shape differences between isosurfaces are reflected in a subset of voxels. It differs from the data coherency method of [kha08] in that we analyze both the component similarity and voxel similarity while only the voxel similarity was considered previously. There are two advantages of considering both types of differences. First, the isosurfaces are guaranteed to be crack-free while the previous study ([kha08]) suffers from cracks due to the inconsistency between extracted and approximated parts of isosurfaces (Fig. 2). Secondly, the proposed approach is more effective in identifying the interesting structural differences and suppressing the noises. In many cases, a small change in a big isosurface component may not be of interest because it can be due to some noise or can be so small that it is visually undetectable. We can avoid an unnecessary processing of the corresponding polygons by specifying appropriate thresholds.

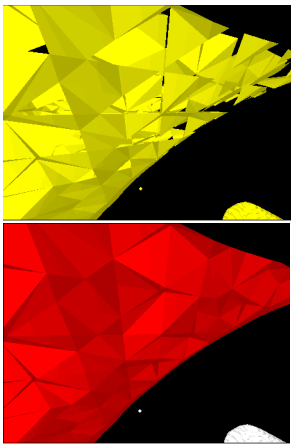


Figure 2: Cracks (top) and no cracks (bottom)

2 MULTI-COMPONENT ISOSURFACE EXTRACTION

The first important step in the proposed multi-component isosurface extraction method is to identify the components themselves. An exact identification of a component can be computationally expensive. However, we can approximate an isosurface component by its voxel coverage - a set of voxels which contribute to the component, which is determined as described below. It is easy to realize that two components must be separated spatially from each other by, at least, one voxel gap. The second important step is to perform the interdataset coherency test. For this purpose, we divide given datasets into two categories: reference datasets (*RDS*) and non-reference datasets (*NRDS*). Consider the simplest case of two datasets for which we have one *RDS* and one *NRDS*. *RDS* is completely processed and the polygon data are stored for later retrieval. On the other hand, *NRDS* is processed for partial isosurface extraction. Only those components which differ from or do not exist in *RDS* are directly extracted from *NRDS*; all similar components are rendered using the corresponding *RDS* polygon data. Thus, the two types of datasets are processed differently.

The *RDS* isosurface extraction is performed using Algorithm 1. This algorithm processes a reference dataset to generate the polygon data representing isosurface component and a lookup table $RDS.map$ representing the relationship between voxels and the components. An entry $RDS.map[v_i]$ indicates the component the voxel v_i belongs to. We define $RDS.map[v_i]$ to be zero if v_i does not contribute to the isosurface. $RDS.map$ is used later during processing of *NRDS* (Algorithm 2). $RDS.map$ allows us to easily find out whether a newly extracted component in *NRDS* overlaps with any component in *RDS*. In Algorithm 1, $v_i.iss$ and $v_i.p$ are true/false flags indicating whether the voxel v_i contributes to the isosurface and whether

the polygons within it have been extracted, respectively. These two flags allow us to skip many voxels. The variable *comp* counts the number of components. To find a component, we start from a voxel in that component and expand the component by checking its neighbors and then their neighbors successively. This expanding process can be implemented by a queue as follows. We traverse each voxel of *RDS*. If a voxel contributes to an unexplored component, we extract the polygons within the voxel and check for its neighbors. We put all its neighbors that also contribute to the isosurface into a queue (denoted as Q). We repeat this process for each member in the queue until all members in the queue are processed. We record the component a voxel belongs to in *RDS.map* every time polygons are extracted from the voxel. We also group the polygons in a component together for an efficient retrieval. Note that since the polygons that belong to the same component are extracted successively, the grouping can be done simply by recording the first extracted polygon in the component if the polygons are saved in a list.

We use the Marching Cubes algorithm to extract the polygons in a voxel. In order to determine whether a voxel contributes to the isosurface or not, we need to compare its scalar values to a given isovalue. It is not a cheap process. One can find the maximum and minimum of the scalar values of each voxel and use the octrees to speed up the extraction. Here, our major goal is to illustrate the component-based isosurface extraction approach. It can be extended to octrees ([she96, wil92]) and other fast isosurface schemes ([yar98]).

For each non-reference dataset, we first compute the per-voxel differences from a reference dataset. The voxel differences between two datasets can be measured in many ways. Different measures vary in their sensitivity to noise and their ability to capture the change. Generally speaking, the measures more capable to capture the change are also more sensitive to noise. Some choices include the differences in one of the eight values or the maximum or the average value between two voxels. Computing the voxel difference can be time-consuming. However, if we choose a difference measure independent of isovalue such as those just mentioned, we need not recompute the differences when we change the isovalue. It is important to note that isosurfaces need to be explored over a wide range of isovalues. We can also divide the whole volume into blocks and assign the same voxel difference to all voxels in the same block as in the previous study ([kha08]).

Once the voxel differences are computed, the polygon generation for a non-reference dataset can be performed with Algorithm 2 only for the components which contain sufficiently large number of significant

Input: a reference dataset *RDS*
Output: polygons in *RDS*, *RDS.map*

Initialize all variables to zeros/false.
FOR each voxel v_i in *RDS*
  IF $v_i.iss$ AND NOT $v_i.p$ **THEN**
    extract polygons in v_i
    $v_i.p \leftarrow true$
    $comp \leftarrow comp + 1$
    $RDS.map[v_i] \leftarrow comp$
    $Q[0] \leftarrow v_i$
    $head \leftarrow 0$, $tail \leftarrow 1$
    WHILE $head < tail$
      FOR each neighbor nb_j of Q[head]
        IF $nb_j.iss$ AND NOT $nb_j.p$ **THEN**
          extract polygons in nb_j
          $nb_j.p \leftarrow true$
          $RDS.map[nb_j] \leftarrow comp$
          $Q[tail] \leftarrow nb_j$
          $tail \leftarrow tail + 1$
        ENDIF
      ENDFOR
      $head \leftarrow head + 1$
    ENDWHILE
  ENDIF
ENDFOR

Algorithm 1: Pseudo-code for isosurface extraction for a reference dataset

voxels. These are the voxels whose difference are greater than the user-defined $threshold1$. In Algorithm 2, $v_i.p$ is a true/false flag which indicates whether the voxel has ever entered into the queue Q or not. We use the flag $v_i.p$ to avoid putting the same voxel into Q multiple times. $voxel.diff$ denotes the voxel difference, $threshold1$, $threshold2$, $threshold3$ are user-defined parameters. $threshold2$ and $threshold3$ are for measuring the differences between components while $threshold1$ is for measuring the similarity between voxels. nV is a counter counting the number of voxels in the component. nSV is a counter counting the number of significant voxels in the component. nV and nSV together with $threshold2$ is for determining whether a component is significantly different and hence should be extracted. $NE[.]$ is an array of true/false flags and $NE[c_i]$ indicates whether there are newly extracted polygons in the corresponding region of $RDS.c_i$ (the c_i component in *RDS*). $NE[.]$ is used when we decide the components to be retrieved from *RDS*.

Algorithm 2 scans through all significant voxels and finds their components. The polygons in a component are extracted only if the component contains sufficiently large number (determined by $threshold2$) of significant voxels. The polygons of the reference dataset are retrieved if no polygons are

newly extracted in the corresponding region in the non-reference dataset and the overlap of the *NRDS* isosurface and *RDS* isosurface are sufficiently large (determined by $threshold3$). The effects of these three thresholds are presented in the next section.

Input: a non-reference dataset *NRDS*, polygons in the reference dataset *RDS*, *RDS.map*, $threshold1$, $threshold2$, $threshold3$.
Output: polygons in *NRDS*

Initialize all variables to zeros/false
FOR each voxel v_i in *NRDS*
  IF $v_i.iss$ AND NOT $v_i.p$ AND $v_i.diff >$ $threshold1$ **THEN**
    $v_i.p \leftarrow true$
    $nV \leftarrow 1$; $nSV \leftarrow 1$
    $Q[0] \leftarrow v_i$
    $head \leftarrow 0$; $tail \leftarrow 1$
    WHILE $head < tail$
      FOR each neighbor nb_j of Q[head]
        IF $nb_j.iss$ AND NOT $nb_j.p$ **THEN**
          $nV \leftarrow nV + 1$
          IF $nb_j.diff \geq threshold1$ **THEN**
            $nSV \leftarrow nSV + 1$
          ENDIF
          $nb_j.p \leftarrow true$
          $Q[tail] \leftarrow nb_j$
          $tail \leftarrow tail + 1$
        ENDIF
      ENDFOR
      $head \leftarrow head + 1$
    ENDWHILE
    IF $nSV > threshold2 \times nV$ **THEN**
      FOR $k = 0, \ldots tail - 1$
        extract polygons in $Q[k]$
        $NE[RDS.map[Q[k]]] \leftarrow true$
      ENDFOR
    ENDIF
  ENDIF
ENDFOR
FOR each component $c_i (i \geq 1)$ in *RDS*
  IF NOT $NE[c_i]$ **THEN**
    $n1 \leftarrow$ number of voxels in c_i
    $n2 \leftarrow$ number of voxels in c_i whose counterparts in *NRDS* also contain polygons.
    IF $n2 > threshold3 \times n1$ **THEN**
      retrieve the polygons in c_i
    ENDIF
  ENDIF
ENDFOR

Algorithm 2: Pseudo-code for isosurface extraction for a non-reference dataset

3 THRESHOLDS AND THEIR SIGNIFICANCE

In this section, we analyze how thresholds control the outputs. We consider a two-dimensional case but our discussion and all conclusions apply to a three-dimensional case. Imagine that the components from the reference dataset and the non-reference dataset are laid over each other in space (Fig. 3). We color the *RDS* isosurface (isoline) blue and the *NRDS* isosurface (isoline) red. There are three cases for each component in the output for a non-reference dataset:

- Case I: One component from the non-reference dataset does not touch any component from the reference dataset (Fig. 3 (a)). If the voxel covering of the component contains more than $threshold2$ significant voxels, the polygons from the non-reference dataset are extracted. Otherwise, the polygons are missing in the output.

- Case II: One component from the reference dataset does not touch any component from the non-reference dataset (Fig. 3 (b)). This polygons are not retrieved and hence is missing in the output.

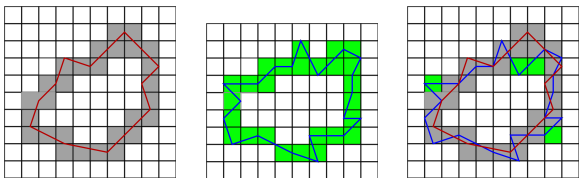
- Case III: One component from the reference dataset overlaps another component from the non-reference dataset (Fig. 3 (c)).

  - Case III (i): The component from the non-reference dataset contains more than $threshold2$ portion of significant voxels. The polygons contained in that component from the non-reference dataset (i.e. the red one) will be extracted. The polygons from the reference dataset (i.e., the blue one) are not retrieved and are missing in the output.

  - Case III (ii): The component from the reference dataset contains no more than $threshold2$ portion of significant voxels and the overlapping of these two components is more than $threshold3$ portion of the total number of the voxels in the component from the reference dataset (In other words, the green voxels in Fig. 3 (c) count less than $1 - threshold3$ portion of the number of the voxels in the component from the reference dataset). The polygons from the reference dataset (the blue one) are retrieved and the polygons from the non-reference dataset are missing in the output. In other words, the exact isosurface of the non-reference dataset (the red one) in that region is approximated by the isosurface of the reference dataset (the blue one).

– Case III (iii): Otherwise, no polygons are extracted or retrieved for the non-reference dataset.

Here, *threshold*1 is for capturing the difference locally while the *threshold*2 and *threshold*3 are for capturing the global difference (between components). By choosing different combinations of three thresholds, users can define the similarity and suppress noise in a flexible way. If all three thresholds are set zero, the output are the exact isosurfaces from the non-reference dataset. From the above analysis, we can also see that we do not introduce any cracks or any self-intersection to the output isosurfaces since the polygons are extracted or retrieved as a whole component.



(a) Case I (b) Case II (c) Case III

Figure 3: Effect of thresholds. The blue and red curves represent an exact component of the isosurface from the reference dataset and the non-reference dataset, respectively

4 PERFORMANCE ANALYSIS

We visualize the electron density data for the perfect and defective (one atom missing) crystals of MgO with the multicomponent isosurface extraction method. We use the perfect crystal dataset as the reference dataset and the defective crystal dataset as the non-reference dataset. The size of each dataset is $356 \times 380 \times 252$. Performance measurements were carried out in Windows XP with Pentium (R)4 3.40 GHz and 2.0 GB RAM.

We use the difference between the scalar values at the upper left corners of the two voxels as the voxel difference. We calculate the number of triangles and polygon generation time by varying *threshold*1 for *threshold*2 and *threshold*3 fixed at 0.25 and compare isosurfaces for four isovalues of 0.01, 0.018, 0.02 and 0.023. As shown in Fig. 4, the number of triangles for the exact *NRDS* isosurface (which are directly processed) varies strongly with isovalue. Correspondingly, the isosurface extraction time increases with isovalue. Note that the time (350 *ms*) for the one-time computation of voxel differences is not included. By choosing *threhold*1, *threshold*2 and *threshold*3 to be 0.0015, 0.25 and 0.25, respectively, we successfully capture the dissimilar components by much fewer triangles in about half of the time. The exact and approximated isosurfaces for *NRDS* shown in Fig. 5 are visually indistinguishable. The different components are highlighted so we can see easily that some disjoint components in the perfect crystal merge together in the defective crystal. As *threshold*1 decreases, more and more voxels in the non-reference dataset turn into significant voxels. However, in our approach more significant voxels do not necessarily lead to more extracted triangles (which is the case in the previous method in [kha08]) because we also consider the similarity between components. If we increase *threshold*2, some components may be discarded so less overhead in polygon extraction and the performance improves. *threshold*3 does not involve polygon extraction but only retrieval for rendering and so it will not affect performance dramatically. Domain-specific knowledge is needed to tune these values. However, there are some visual indications in the case when the threshold values are not set correctly. For example, large blank space of the resulting image may indicate that *threshold*2 or *threshold*3 is set too high. We may need to do further investigation for that particular portion of data.

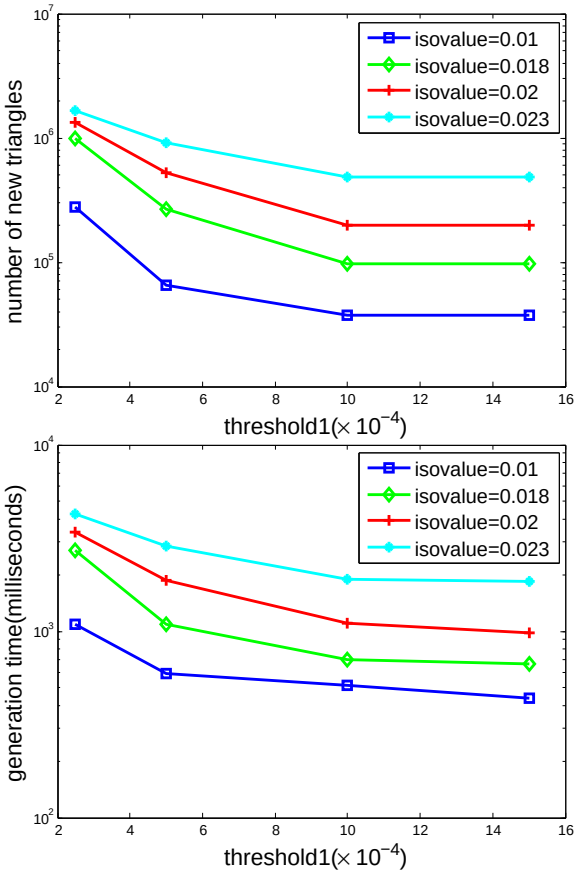


Figure 4: Number of new triangles (top) and generation time (bottom) for the non-reference dataset as a function of *threshold*1

Component-based isosurface extraction is not useful if the isosurfaces under consideration contain only

one or two large components that extend over almost the entire scene. If this is the case, then these big components are either extracted or ignored based on the thresholds. If the components are not extracted, then large portions of the isosurfaces are retrieved or missing. If they are extracted, it takes more time than the normal processing time.

5 ON-THE-FLY ISOSURFACE EXTRACTION

Our method assumes the data coherency. In some situations, the shape similarity/difference may not be reflected by voxel values. Consider the case where two isosurfaces differ only by a shift of one voxel size. The shapes of the isosurfaces are the same but the voxels differ a lot. Here, we provide a way to handle such situations to take the advantage of multi-component isosurface extraction. Instead of extracting the polygons at once, we use a simple shape to represent each component after we find them using Algorithm 1. If the user recognizes and selects a region of interest, then the actual isosurfaces within that regions are exacted. We show an example in Fig. 6 where we use a bounding box to represent a component. Since only simple shapes are stored and the exact isosurfaces are extracted on the fly, we not only improve the time and space but also the user interaction.

6 CONCLUSIONS

In this paper, we have proposed a combination of multi-component isosurface extraction and data coherency methods to support multiple dataset visualization. Our approach can detect and highlight interesting structural differences. By exploring the voxel similarity and component similarity between datasets, we can improve time and space requirements in the situations where only a subset of isosurface components differ significantly between the datasets. We also provide on-the-fly isosurface extraction based on component comparison.

ACKNOWLEDGEMENTS

This work is supported by the NSF Career (EAR 0347204) and NSF/EPSCoR (EPS-0701491) grants.

REFERENCES

[aoy07] Aoyama, D.A., Hsiao, J.-T. T., Rdenas, A.F.C., and Pon, R.K. TimeLine and visualization of multiple-data sets and visualization querying challenge. Journal of Visual Languages and Computing & Computing, vol. 18, 2007

[chi97] Chi, E.H., Riedl, J., Barry, P., and. Konstan, J. Principles for information visualization spreadsheets. Proceedings of the 1997 Information Visualization Symposium.

[han05] Hansen, C.D., and Johnson, C.R. The Visualization Handbook. Elsevier Academic Press, 2005.

[kha06] Khanduja, G., and Karki, B.B. Multiple datasets visualization with isosurface extraction. Proceeding (541) Visualization, Imaging, and Image Processing, 2006

[kha07] Khanduja, G., and Karki, B.B. Using graphics hardware for multiple datasets visualization. WCSG07 Proceedings (Ed. 2007, V. Skala), pp. 161-168, ISBN 978-80-86943-02-2

[kha08] Khanduja, G., and Karki, B.B. Exploiting data coherency in multiple dataset visualization. Proc. of the 10th Int' l. Conf. on Computer Graphics and Imaging (CGIM' 08), 20

[lew03] Lewiner,T., Lopes, H., Vieira, A.W. and Vieira, G. Efficient implementation of Marching Cubes' cases with topological guarantees. Journal of Graphics Tools, 2003.

[lor87] Lorensen, W.E., and Cline, H.E. Marching cubes: a high resolution 3D surface construction algorithm. ACM Computer Graphics, vol. 21, 1987

[nub03] Nubera, E., LaMarb, E. C., Hamanna, B., and Joya, K.I. Approximation of time-varing multiresolution data using temporal-spatial reuse. Visualization and Data Analysis, 2003

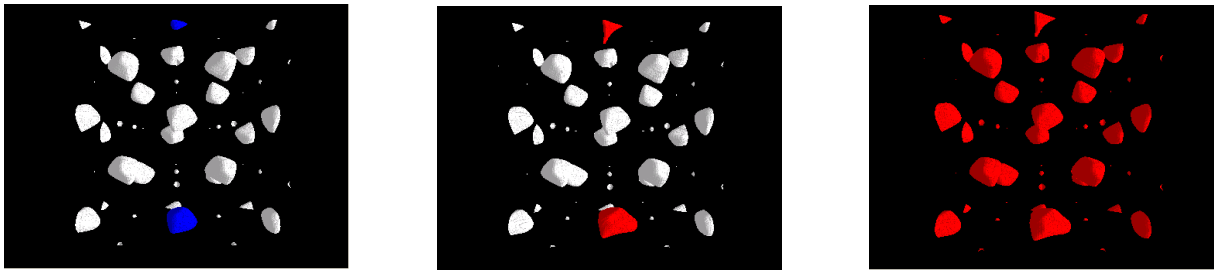
[she96] Shekhar, R., Fayyad, E., Yagel, R., and Cornhill, J.F. Octree-based decimation of marching cubes surfaces, IEEE Visualization, Proceedings of the 7th conference on Visualization 1996.

[wil92] Wilhems, J. and Gelder, A.V. Octrees for faster isosurface generation. ICM Trans. Graphics, Vol. 11, No. 3, pp. 201-227, July 1992.

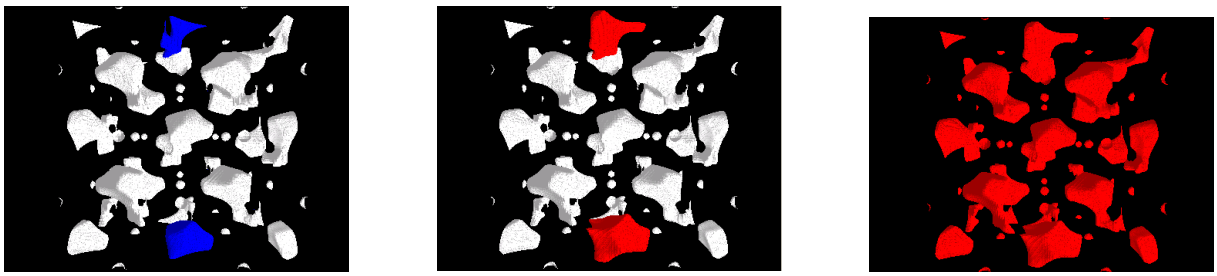
[yar98] Livnat, Y., and Hansen, C. View Dependent Isosurface Extraction. IEEE Visualization, 1998.

[web01] Weber, G.H., Kreylos, O., Ligocki, T.J., Shalf, J.M., Hagen, H., Hamann, B., and Joy, K.I. Extraction of crack-free isosurfaces from adaptive mesh refinement data. Approximation and Geometrical Methods for Scientific Visualization, pp.19-40, 2001

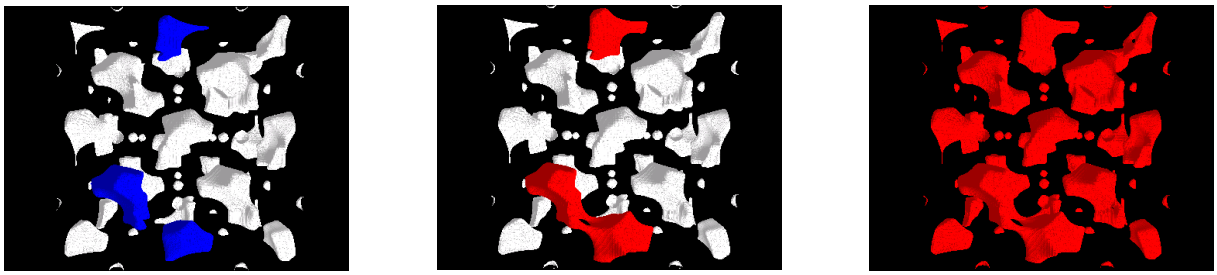
[wes99] Westermann, R., Kobbelt, L., and Ertl, T. Real-time exploration of regular volume data by adaptive reconstruction of iso-surfaces. The Visual Computer, vol. 15, pp. 100-111, 1999.



0.42M triangles | 0.037M new triangles | 0.43M new triangles



1.03M triangles | 0.098M triangles | 1.0M triangles



1.34M triangles | 0.20M triangles | 1.3M triangles

1.65M triangles | 0.49M triangles | 1.6M triangles

Figure 5: The first column are the isosurfaces for the reference dataset. The second column are the approximated isosurfaces for the non-reference dataset. The third column are the exact isosurfaces for the non-reference dataset. The blue components are the components not retrieved and hence are unique to the reference dataset. The red components are extracted exactly from the non-reference dataset. The white components are recognized as common components by our approach. The rows are for the isovalue $0.01, 0.018, 0.02, 0.023$ respectively. We can easily see that some disjoint components from the reference dataset merge together to form a new component in the structure from the non-reference dataset. Those components are successfully identified by our approach.

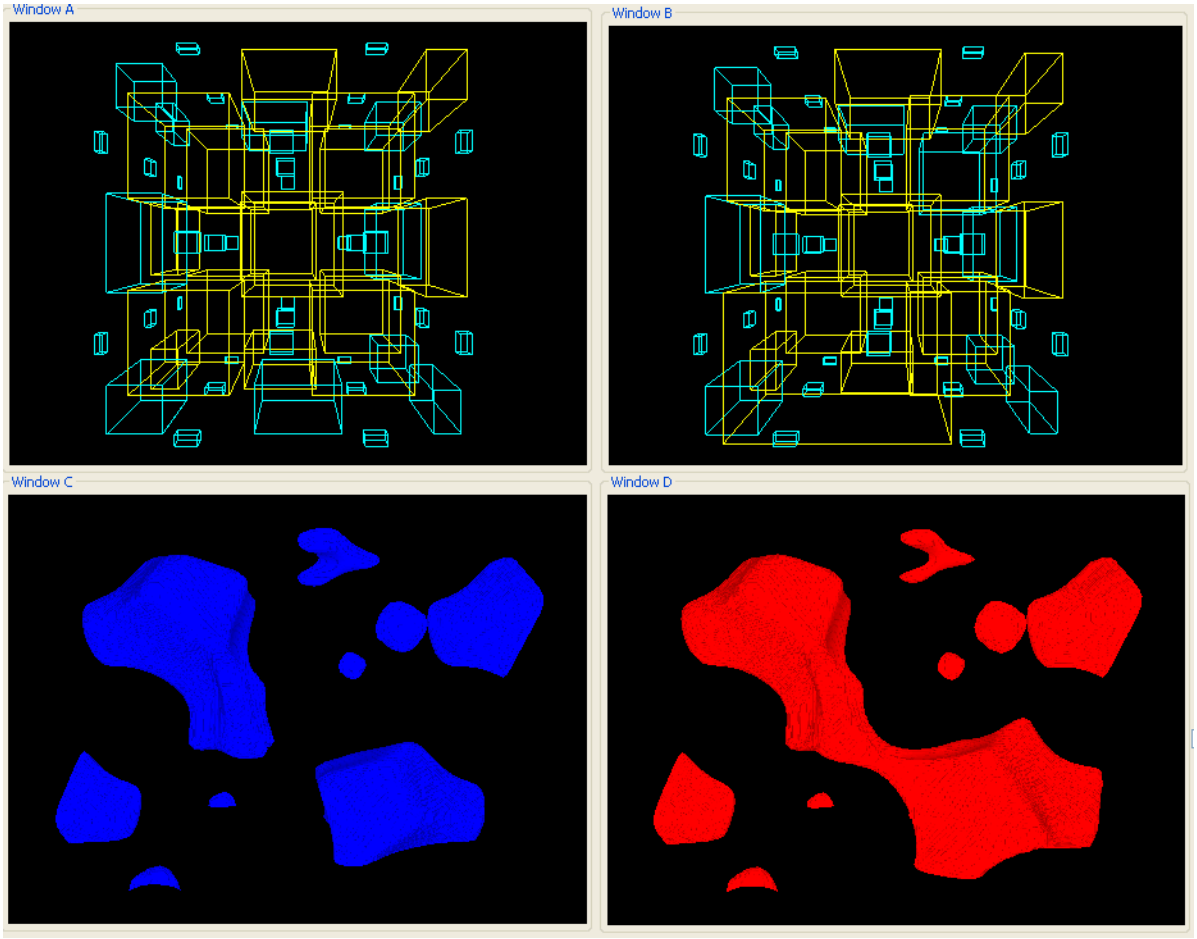


Figure 6: On-the-fly isosurface extraction. Each component is represented as its bounding box. The boxes are colored according to their size. A user recognizes a region of interest by noticing a big yellow box unique to one dataset (i.e., the big yellow box in the bottom left of the top right window) and selects the region so that the isosurfaces within that region are extracted from both datasets.

Automated Motion LoD with Rigid Constraints

Junghyun Ahn
VRLab, EPFL
IC ISIM, Station 14
1015, Lausanne
Switzerland
junghyun.ahn@epfl.ch

Byung-Cheol Kim
EECS, KAIST
335 Gwahangno, Yusung
305-701, Daejeon
South Korea
ciel@vr.kaist.ac.kr

Daniel Thalmann
VRLab, EPFL
IC ISIM, Station 14
1015, Lausanne
Switzerland
daniel.thalmann@epfl.ch

Kwangyun Wohn
GSCT, KAIST
335 Gwahangno, Yusung
305-701, Daejeon
South Korea
wohn@kaist.ac.kr

ABSTRACT

Motion LoD (Level of Detail) is a preprocessing technique that generates multiple details of captured motion by eliminating joints. This LoD technique is applied to movie, game or VR environments for the purpose of improving speed of crowd animation. So far, replacement techniques such as 'impostor' and 'rigid body motion' are widely used on real-time crowd, since they dramatically improve speed of animation. However, our experiment shows that the number of joints has a greater effect on the animation speed than anticipated. To exploit this, we propose a new motion LoD technique that not only improves the speed but also preserves the quality of motion. Our approach lies in between impostor and skeletal animation, offering seamless motion details at run time. Joint-elimination priority of each captured motion is derived from joint importance, which is generated by the proposed posture error equation. Considering hierarchical depth and rotational variation of joint, our error equation measures posture difference successfully and allows finding key posture of the entire motion. This 'motion analysis' process contributes error reduction to the next 'motion simplification' stage, where multiple details of motion are regenerated by the proposed motion optimization. In order to reduce the burden of optimization, all the terms of the objective function - distance, string, and angle error - are defined by joint-position vectors. In this aspect, a constrained optimization problem is formulated in a quadratic form. Thus, a sequential quadratic programming (SQP), a nonlinear optimization method, is suitable for resolving this problem. As the result of our experiment, the proposed motion LoD technique improves the animation speed and visual quality of simplified motion. Moreover, our approach reduces the preprocessing time and automates the whole process of LoD generation.

Keywords

Level of detail, Crowd animation, Real-time animation, Motion analysis, Motion optimization

1. INTRODUCTION

In recent animated products, the number of joints and polygons of articulated bodies has been increased substantially. Furthermore, the number of bodies that can be rendered in crowd scenes is also increasing. According to these trends, many studies on crowd animation have been presented to date. These works can be classified into two major categories: (1) realism enhancement - behavior manipulation [TT94] [MT01][ST05][YMP+09], collision detection [Rey87] [TCP06][PAB07], and (2) speed improvement of crowd animation. In the past, enhancement of realism was the major area of research with regard to crowd animation. At present, however, due to GPU evolution, the demands of real-time crowd animation have increased and, for this reason, speed of animation is becoming another important research topic. Skinning vertices are transformed in the GPU rather than in CPU. Therefore, at each frame, the CPU sends to GPU only the initial pose of mesh and the transformation matrix of joints [Dom01]. This animation mechanism reduces the calculation time of the dynamic mesh and relatively increases the burden of joint transformation. In order to demonstrate this, we conducted an experiment on the effect of joints. Joints and polygons are both simplified into 8 levels and finally 64 levels of detail are created. For each detail, 10,000 articulated bodies are cloned and animated. Our experiment shows that the number of joints remarkably affects the overall speed of crowd animation. In comparison with polygon reduction (see Fig. 1), joint elimination appears to be even more effective. Although this skeletal simplification approach cannot surpass the animation speed of impostor technique [ABT00][TLC02][DHO+05], our

experiment shows that the speed of the lowest detail (8% detail) is about four times faster than the original animation (100% detail), without any other physical simulation or interpolation, *i.e.*, just for playback.

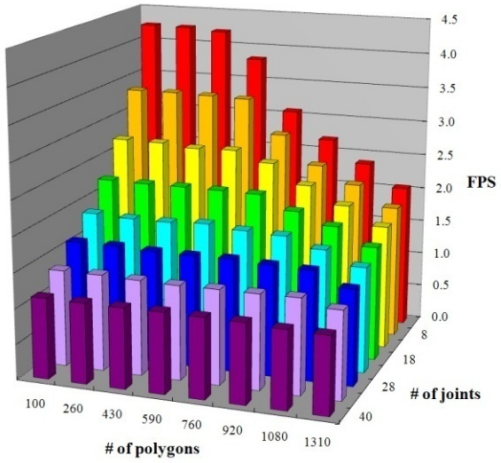


Figure 1. Experiment on 10,000 walking virtual humans (VHs). A VH consists of 40 joints and 1,310 polygons. Joints and polygons are simplified at a rate of 100, 82, 70, 58, 45, 33, 20, and 8%. Each bar represents animation speed of 10,000 VHs by frame per second (FPS) at GeForce 7800 GTX 512 MB and Pentium D 3.0 Ghz.

Our major contributions are as follows:

Automatic level generation: Each joint has different hierarchical depth and rotational variance per motion. If a joint has fast moves, it can be considered that it contains more information than a slow joint. The priority of elimination of joint will be decided by this importance measure. In this paper we present an equation that automatically generates this importance.

Fast preprocessing: From the motion capture system, a number of motions are accumulated. Animators who have access to these captured motions spend lots of time to eliminate joints with low importance and to re-create simplified motion. Our simplification method optimizes simplified motion by excluding mesh parameters in the objective function and by using a fast nonlinear optimization solver.

Preserved motion quality: If the bone length is not consistent during motion, the quality of motion will decrease. Previous works on skeletal simplification [JT05][AOW06] mentioned about this bone length problem. In this approach, we succeed to avoid bone length variation by defining constraints of rigid bone.

From these contributions, our approach not only reduces preprocessing time of optimization, but also considers the quality of simplified motion. Moreover, automatic generation of joint importance shows the practicality and flexibility of this method.

2. RELATED WORKS

There are two approaches on crowd animation for improving the speed: (1) the 'impostor' [ABT00] [TLC02][DHO+05], wherein an image sequence is applied instead of articulated motion, and (2) the 'skeletal simplification' for simulation LoD [CH97] [CIF99][PW99][BK04][RGL05][KRK08] or motion LoD [GJG+03][JT05][AOW06]. Among these works the impostor is more efficient in terms of increasing the speed of crowd. However, this image technique is compromised by the following disadvantages:

Reality: If the camera approaches an animated virtual human (VH) or moves rapidly around the crowd, the realism of the motion is deteriorated.

Memory: To animate a long take or multiple motions, a great deal of memory space is necessary for saving entire image sequences.

Interactivity: In a real-time environment, interaction between the user and VH would not be easy.

Skeletal simplification, meanwhile, can gradually decrease detail of motion and at run time it can adjust motion details by controlling the number of joints. Moreover, the skeletal simplification can overcome these reality, memory, and interactivity problems.

Over the last decade, many researches on skeletal simplification have been presented. [CH97] and [PW99] simplified manually the hierarchy of an articulated figure in order to improve the speed of physical simulation and facilitate convergence. However, these skeletal simplifications are unable to apply in the crowd motion using motion capture data. [GJG+03] proposed a manually constructed level of articulation to improve speed of the real-time networked environment. [BK04] described a method for simulating motion of complex plants. They used a preprocessing method to generate different plant structure, along with a set of simulation LoD. [AW04] proposed a joint posture clustering (JPC) method in order to reduce the number of transformation and improve the speed of animation. However, reducing transformation is not as fast as eliminating joint of motion. [RGL05] presented an adaptive algorithm for computing forward dynamics of articulated bodies using motion error metrics. Their approach simplifies the dynamics of a multi-body system, based on the desired number of degrees of freedom and forces. [JT05] generated different levels of motion based on given animating mesh sequences. The joint structure is reconstructed by clustering rotation of triangles. [AOW06] proposed an optimized motion LoD for real-time crowd animation. In order to generate a simplified motion, they minimized error between the original and the simplified motion by designing a linear system that optimizes skinning matrices by least square approximation (LSA).

In the early period of skeletal simplification, joints were eliminated for the purpose of improving speed of physical simulation (simulation LoD). Recently, however, eliminating joint from a captured motion (motion LoD) became another important issue.

3. MOTION LOD FRAMEWORK

The overall animating pipeline of our approach is depicted in Fig. 2. First, at the preprocessing stage, motion and mesh are divided into a number of details through our motion and geometric LoD [Hop96]. The number of details is given by user and a discrete level of mesh is generated by edge collapsing. Simplified motion is connected to the corresponding mesh level by 'motion mapping'. The background scene is also analyzed to populate simplified bodies into the scene. A depth map is generated from the top orthographic view of the VH's movable region. This map is later used for calculating height and limiting boundary of VH's movement. At the run-time stage, preprocessed results are gathered into the 'simulation' module. During simulation, VH's root position, view frustum culling, and projected size of VH from the camera coordinates are generated and sent to the 'scene generation' via 'LoD control' or directly. Finally, the whole scene is rendered for each frame and camera attributes are modified for the next simulation loop.

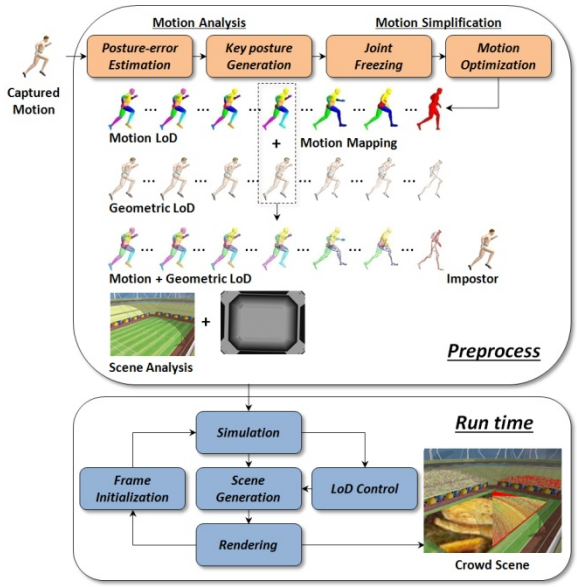


Figure 2. The overall pipeline of motion LoD.

The proposed motion LoD consists of two sub-parts - motion analysis and simplification. Every motion has different properties, thus the way of simplifying motion must be distinguished among each motion. The proposed 'motion analysis' enables us to distinguish motion by generating key posture of motion and priority list of joint elimination. In the 'motion simplification' stage, each motion level is generated. Joints that are selected from the priority list become frozen joints (see Section 5). At the 'joint freezing' stage, frozen joints are applied to the simplified structure of motion. Finally, at the 'motion optimization', our nonlinear optimization algorithm calculates a new simplified motion by minimizing the error between the original and the simplified motion.

4. MOTION ANALYSIS

Before simplifying a skeletal structure of motion, we need to know the elimination priority of joints. This priority list is created by sorting joint importance, which are measured by the sum of posture error. In this section, we propose a basic equation on posture error and show how we extract key posture and derive joint importance from the posture error.

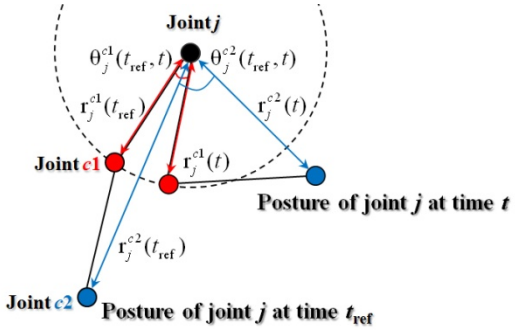


Figure 3. The evaluation terms of posture error $E_j(t_{ref}, t)$. The length $r^c_j(t)$ is the distance between joint j and its child c at frame t. The radian $\theta^c_j(t_{ref}, t)$ is the angle of $p_c(t_{ref})p_jp_c(t)$, where $p_c(t)$ is the position of joint c at frame t.

4.1. Posture Error of Joint

Our posture error equation considers two important factors - hierarchical depth and rotational variance of joint. For example, if a joint lies near the root of hierarchy, its rotation will propagate to descendents. Therefore, a higher level joint has a higher posture error than a lower level joint. The rotational variance is a more intuitive factor. If a joint rotates in a wide range during motion then the difference will increase. As described in Eq. (1), the posture error $E_j(t_{ref}, t)$ is a posture difference of joint j from frame t_{ref} to t.

$$E_j(t_{ref}, t) = \frac{\sum_{c=1}^{m} \left\| \left[r^c_j(t_{ref}) + r^c_j(t) \right] \theta^c_j(t_{ref}, t) \right\|}{2\pi r_{max}} \qquad (1)$$

Basically, the posture error equation is a normalized summation of joint-trajectory distance between t_{ref} and t. Each specific term of equation is depicted in Fig. 3. The constant m is the total number of children of joint j and $2\pi r_{max}$ is the normalization value of

posture error equation. In our experiment, we set the r_{max} value as the summation of r^c_{root} (c=1, …, m). In this equation, the hierarchical factor is covered by the number of descendants m and the average length of $r^c_j(t_{ref})$ and $r^c_j(t)$. Meanwhile, the rotational factor is covered by $\theta^c_j(t_{ref}, t)$. Some previous works proposed a different way to evaluate posture error [LCR+02] [KPS03]. Our posture error equation automatically generates a normalized and weighted error.

4.2. Key Posture of Motion

From the Eq. (1), we create a 2D array of posture errors for each joint, where the row parameter is t_{ref}, column is t and array value is $E_j(t_{ref}, t)$. The key posture is generated from the minimum sum of the row *i.e.* for each joint j, we select a reference frame t_{ref} and calculate the sum of posture errors on entire frame t ($1 \le t \le n$). The constant n is the number of motion frames. This error summation proceeds for all t_{ref} ($1 \le t_{ref} \le n$). As defined by Eq. (2), the reference frame t_{ref} with the minimum row sum is set to t^{key}_j, which is the key frame of joint j.

$$t_j^{\text{key}} = \arg\min_{t_{\text{ref}}(1 \le t_{\text{ref}} \le n)} \left[\sum_{t=1}^{n} E_j(t_{\text{ref}}, t)\right] \quad (2)$$

As the result, each joint contains different key frame t^{key}_j. Key posture of motion is generated by applying matrix or quaternion that is defined at frame t^{key}_j. Since the key frame of each joint is extracted from the existing motion frames, the key posture doesn't violate human constraint. As depicted in Fig. 4, we applied our key posture algorithm to several motions.

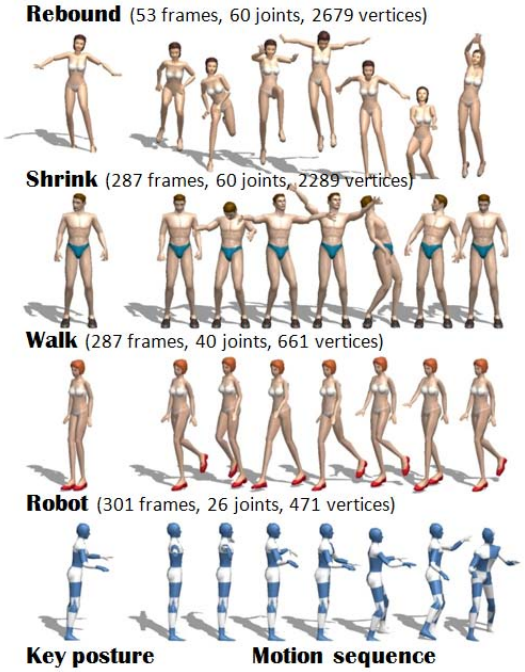


Figure 4. The key posture generation results.

4.3. Joint Importance

Joint importance is a measure of average variance of a joint on the entire motion. From this value, we can generate the priority list for joint elimination. The joint importance ε_j can be obtained from Eq. (2). As defined by Eq. (3), the sum of the row t^{key}_j is normalized by the number of frames. A joint with low importance has a higher elimination priority. The value ε_j is used on the motion simplification stage as for creating the priority list for joint elimination.

$$\varepsilon_j = (1/n) \sum_{t=1}^{n} E_j(t_j^{\text{key}}, t) \quad (3)$$

5. MOTION SIMPLIFICATION

In this section, we describe how to minimize error between original and simplified motion. The goal is achieved by two stages - joint freezing and motion optimization. In order to construct a priority list of joint elimination and to generate frozen joints, the joint importance ε_j is applied. The basic terms of the proposed objective function are acquired from joint position vectors and frozen joints.

5.1. Frozen Joint

Before removing joint from a motion, we freeze joint (make it rigid) in order to keep useful parameters such as rigid bone length and angle. Frozen joints are selected by ε_j and by the number of joints that will be eliminated during motion simplification. The local transformation of frame t^{key}_j which has the minimum error sum over all frames is applied as a frozen joint. For each frozen joint, the bone length and cosine angle constraints can be defined as in Eq. (4).

$$b_{c,p}(t) = \left\|\mathbf{x}_c(t) - \mathbf{x}_p(t)\right\|^2 \quad (4)$$
$$\theta_j(t) = \frac{\left[\mathbf{x}_c(t) - \mathbf{x}_j(t)\right] \cdot \left[\mathbf{x}_p(t) - \mathbf{x}_j(t)\right]}{b_{c,j}(t) b_{j,p}(t)}$$

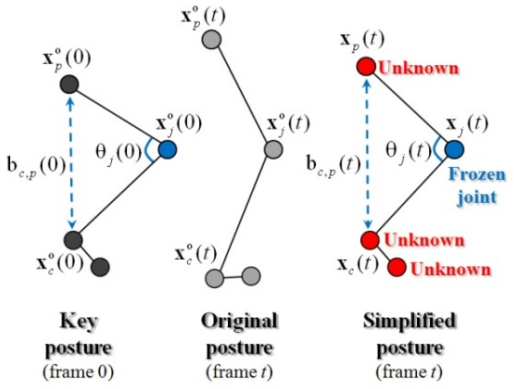


Figure 5. The frozen joint and motion attributes for optimization. A simplified posture is a posture of which we want to minimize the error.

As depicted in Fig. 5, the unknown motion consists of position vectors of joint $\mathbf{x}_j(t)$, functions of bone length $b_{c,p}(t)$, and cosine angle $\theta_j(t)$, where p is parent and c is child of joint j. Here the cosine angle $\theta_j(t)$ is defined from 0 to π. For every frozen joint j, known values $b_{c,p}(0)$ and $\theta_j(0)$ of the key posture (keys are saved into frame zero) are pre-calculated. The rest pose of a VH's mesh is modified by the key posture, including skinning weights re-arrangement on the frozen area. If a vertex is related to a frozen joint, the skinning weights move to its parent joint.

5.2. Motion Optimization

The basic idea of motion optimization is to minimize the sum of difference between the original motion $\mathbf{x}^o_j(t)$ and simplified motion $\mathbf{x}_j(t)$. Due to the frozen joints, the objective function must consider additional hard constraints such as the bone length and joint angle. As was defined by Eq. (4), these functions can be replaced by unknown vectors $\mathbf{x}_j(t)$. Therefore, we formulate the objective function E as a summation of distance E_d, string E_s, and angle E_a error as in Eq. (5). Sequential quadratic programming (SQP) [GMW81] [Fle87][Gle98], is applied to solve this problem. Each error term is multiplied by the weighting constants α, β, and γ. In our experiment, we applied 0.2, 0.4 and 0.4 respectively.

$$E = \alpha E_d + \beta E_s + \gamma E_a \qquad (5)$$

Distance error: The square sum of the positional difference between the original joint position $\mathbf{x}^o_j(t)$ and an unknown simplified joint position $\mathbf{x}_j(t)$ is the distance error E_d as in Eq. (6). Constants f and n are the total number of frames and joints, respectively. Root joint (j =1) is excluded in the evaluation, since the root position of the simplified motion is constrained to be the same as the original position.

$$E_d = \sum_{t=1}^{f} \sum_{j=2}^{n} \left\| \mathbf{x}_j(t) - \mathbf{x}^o_j(t) \right\|^2 \qquad (6)$$

String error: The square sum of joint-to-joint length difference between $b_{c,p}(t)$ and $b_{c,p}(0)$ is the string error E_s as in Eq. (7). Function $b_{c,p}(t)$ is the length between joint c and p of the unknown simplified motion at frame t, where p is the first parent of c among joints not to be removed (non-frozen joint). Constant $b_{c,p}(0)$ is the same length at the key posture. Constant m is the total number of non-frozen joints. Root joint (c = 1) is excluded in the evaluation, since its parent joint does not exist.

$$E_s = \sum_{t=1}^{f} \sum_{j=2}^{m} \left\| b_{c,p}(t) - b_{c,p}(0) \right\|^2 \qquad (7)$$

Angle error: The square sum of approximated cosine difference between $\theta_r(t)$ and $\theta_r(0)$ is the angle error E_a, as in Eq. (8). Function $\theta_r(t)$ is the cosine value of joint r at frame t, where r is one of the frozen joints. Constant $\theta_r(0)$ is the same cosine value of the angle of joint r at the key posture. Constant n-m is the total number of frozen joints. In the case of non-frozen joints, the cosine value varies through the frame, and therefore is not considered here.

$$E_a = \sum_{t=1}^{f} \sum_{r=1}^{n-m} \| \theta_r(t) - \theta_r(0) \|^2 \qquad (8)$$

For the initial values, the original motion's joint position is used. The local transformation value of each joint is recovered from the optimization result $\mathbf{x}_j(t)$ and the hierarchical information of each joint. Our approach optimizes simplified motion from the motion data itself. The frozen joints are removed after fitting motion data into the meshed structure. Fig. 6 shows each level of simplified motion.

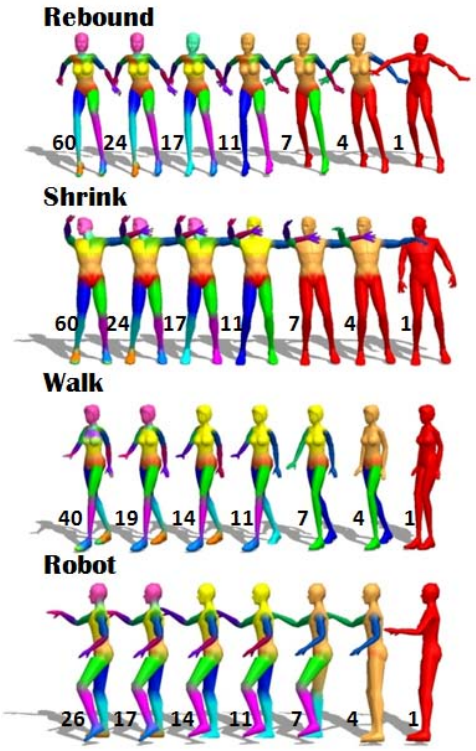


Figure 6. Motion LoD results. The number of joints is indicated on the left, for each LoD posture. Each color corresponds to a joint.

6. EXPERIMENTAL RESULTS

In this section, we describe our experiments that address the advantage of our proposed method. We have conducted four experiments - preprocessing time, motion quality, memory, and animation speed.

6.1. Preprocessing Time

The main advantage of our approach is that we use simple parameters in the optimization process. Since the previous works [JT05][RGL05][AOW06] include complex parameters such as velocity, acceleration, position, and normal of mesh, they give a burden of preprocessing time. As was described in Table 1, we analyzed the preprocessing time of four different methods. The number of joints, frames, and vertices (N_j, N_f, N_v) are described for each experiment. Each preprocessing time was measured by minutes (t_m). Motion analysis contributes to a fast convergence.

Methods	t_m	N_j	N_f	N_v
SMA	7.2	22	400	3030
LSA	29.4	11	287	2239
SQP	3.5	24	331	2679
ASQP	0.8	24	331	2679

Table 1. A comparison of preprocessing times; SMA [JT05]; LSA [AOW06]; SQP: Our method; ASQP: Our method with motion analysis

6.2. Motion Quality

By formulating constraints of bone length and angle, we succeed to attain reasonable quality of motion. In order to show this, we conducted an experiment on the average errors and variations of the bone length (see Fig. 7). Each motion is simplified into 8 levels of detail and *the sum of bone errors* is calculated for each posture (*i.e.* the sum of bone length differences in a frame). The blue and green bar is the average of *the sum of bone errors* over entire frames and levels of motion. In addition to the average bone error, we generated the variation of bone length as well. For each bone, the standard deviation of bone length is calculated over all frames of motion. For each level, we added standard deviations of all bones. The red and violet bar is the average standard deviation over all levels of motion. As the result, our approach shows better stability compared to the other approach. Moreover, the length variation is less than 3%, which is not perceivable with full attention [HRP04]. The enhancement of our approach is shown in Fig. 8.

6.3. Memory

A previous work on impostor [DHO+05] mentioned that 7 MB of memory are required for sampling a single frame of motion. Considering a 287-frame shrinking motion, more than 2 GB will be required. However, our approach needs only 10.3 MB for ten levels of motion (animation and geometry: 9.94 MB, texture: 0.36 MB). According to the experiment of polypostor [KDC+08], our approach also gives better efficiency. The memory cost will decrease more, if a temporal compression [Ari06] or a geometric LoD [PHB07] are applied into our framework.

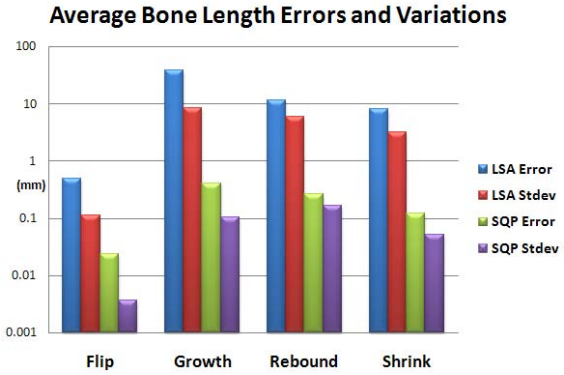


Figure 7. The comparison of average bone length errors and variations in logarithmic scale of mm.

Figure 8. Enhancement in terms of motion quality; Top: Growth motion; Bottom: Rebound motion

6.4. Animation Speed

To conduct an experiment on speed, 5,000 VHs with 200,000 joints are populated in the scene. Motion and mesh are simplified into ten levels. As described in Table 2, average animation speed is measured for four different types of animation with the same navigation path. During the navigation, our approach improves the speed by minimum two to maximum five times faster than the original. Moreover, owing to the GPU skinning technique, motion LoD is even faster than geometric LoD. We used the same simplification rate for both GLoD [Hop96][OZS+03] and MLoD. The hardware environment is GeForce 7800 GTX 512 MB and Pentium D 3.0 Ghz.

	Original	GLoD	MLoD	G+MLoD
Nav1	6.78	12.44	17.83	21.73
Nav2	12.49	20.62	26.44	30.43

Table 2. Average animation speed in FPS for two navigation paths (Nav1, Nav2); GLoD: Navigation with Geometric LoD; MLoD: with Motion LoD; G+MLoD: with Geometric and Motion LoD

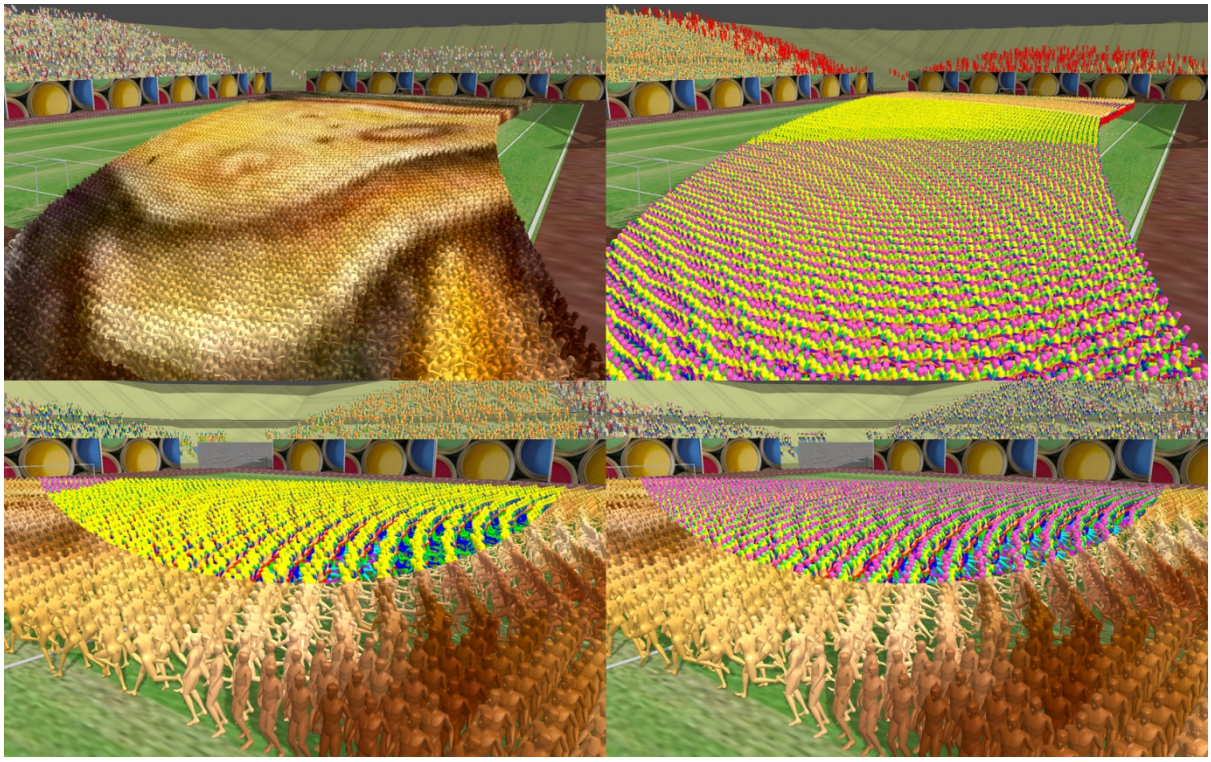


Figure 9. The Stadium: A massive crowd scene; Top left: A scene populated with 15,880 VHs, 805,189 joints, and 56,775,042 polygons; Top right: Top left scene with colored joints; Bottom left: Top right scene shown from the other camera view (motion LoD is applied from the top scene's camera view); Bottom right: Bottom left scene without motion LoD (for comparing visual quality of our motion LoD); The average frame rate is 5.23 FPS (original scene is 1.21 FPS)

Finally, as was depicted in Fig. 9, we have populated 15,880 VHs in a stadium environment to show the efficiency of our motion LoD approach. A total six different VH models and six different motions with about 1,240 frames were preprocessed by geometric LoD and motion LoD, respectively. The size of memory required for simplified VH was 41.7 MB (animation and geometry: 39.88 MB, texture: 1.82 MB). We were able to animate 805,189 joints and 56,775,042 polygons at 5.23 FPS with GeForce 8800 GTX 768 MB and Core2Duo 2.4 Ghz.

7. CONCLUSION

We have presented a new LoD framework that improves the performance of a real-time crowd environment. We automate the whole process and reduce the preprocessing time for the practical use. Moreover, the bone length and angle preservation improves the animation quality of the simplified motion. As the result, we verified that our approach can be adopted in a crowd animation framework. However, the proposed motion LoD doesn't consider the end effector or foot plant of the simplified motion. We assumed that the end effector is not a serious problem, since the detail of low level motion is indistinguishable in the scene. Our approach is not suitable for physics based animation, since the motion analysis and simplification is running on the preprocessing time. The proposed motion LoD is suitable for a real-time environment with captured motion. The error of LoD transition is not considered in our optimization. We tried to minimize the artifact by controlling LoD with projected pixel size of VH.

For improving the quality of optimization, some future works remain. The proposed optimization focuses on a fast preprocessing with affordable accuracy of the simplified result. Therefore, other important issues such as motion smoothness are not considered in the proposed objective function. By resolving the temporal coherence problem, the quality of simplified motion will surely be improved. Another work that should be considered is to derive multiple key postures, since the error of simplified motion strictly depends on the key posture. The proposed motion LoD technique is expected to be more useful in the applications of highly crowded environments such as an urban simulation and games, since a great number of virtual humans are often occluded or appear tiny in the entire scene. It can readily be surmised that the number of virtual humans and the complexity of a skeleton would greatly increase in the near future. Motion LoD will be more challenging than ever before.

8. ACKNOWLEDGMENTS

This research was supported in part by a grant from European Union (EU) FP7 funded IP project, CYBEREMOTIONS (Grant: 231323), from Swiss National Research Foundation (FNRS) funded project, AERIALCROWDS (Grant: CRSI20-122696), and from KAIST Institute for Entertainment Engineering (KIEE). The authors wish to thank Dr. Stéphane Gobron, Quentin Silvestre, Clément Marx, and Olivier Renault for the discussions and supports.

9. REFERENCES

[ABT00] Aubel A., Boulic R., and Thalmann D., *Real-time display of virtual humans: Levels of detail and impostors*, IEEE Transactions on Circuits and Systems for Video Technology, 10(2): 207–217, 2000.

[AOW06] Ahn J., Oh S., and Wohn K., *Optimized motion simplification for crowd animation*, Computer Animation and Virtual Worlds, 17(3–4): 155–165, 2006.

[Ari06] Arikan O., *Compression of motion capture databases*, ACM Transactions on Graphics, 25(3): 890–897, 2006.

[AW04] Ahn J., and Wohn K., *Motion level-of-detail: A simplification method on crowd scene*, Computer Animation and Social Agents, pp. 129–137, 2004.

[BK04] Beaudoin J., and Keyser J., *Simulation levels of detail for plant motion*, Symposium on Computer Animation, pp. 297–304, 2004.

[CH97] Carlson D. A., and Hodgins J. K., *Simulation levels of detail for real-time animation*, Graphics Interface, pp. 1–8, 1997.

[CIF99] Chenney S., Ichnowski J., and Forsyth D., *Dynamics modeling and culling*, IEEE Computer Graphics and Applications, 19(2): 79–87, 1999.

[DHO+05] Dobbyn S., Hamill J., O'Conor K., and O'Sullivan C., *Geopostors: a real-time geometry / impostor crowd rendering system*, Symposium on Interactive 3D graphics and games, pp. 95–102, 2005.

[Dom01] Dominé S., *Mesh skinning*, NVIDIA, 2001. (http://developer.nvidia.com/object/skinning.html)

[Fle87] Fletcher R., *Practical methods of optimization (2nd edition)*, Wiley-Interscience, 1987.

[GJG+03] Giacomo T. D., Joslin C., Garchery S., and Magnenat-Thalmann N., *Adaptation of facial and body animation for mpeg-based architectures*, Cyberworlds, p. 221, 2003.

[Gle98] Gleicher M., *Retargetting motion to new characters*, SIGGRAPH, pp. 33–42, 1998.

[GMW81] Gill P., Murray W., and Wright M., *Practical optimization*, Academic Press, 1981.

[Hop96] Hoppe H., *Progressive meshes*, SIGGRAPH, pp. 99–108, 1996.

[HRP04] Harrison J., Rensink R. A., and van de Panne M., *Obscuring length changes during animated motion*, ACM Transactions on Graphics, 23(3): 569–573, 2004.

[JT05] James D. L., and Twigg C. D., *Skinning mesh animations*, ACM Transactions on Graphics, 24(3): 399–407, 2005.

[KDC+08] Kavan L., Dobbyn S., Collins S., Zara J., and O'Sullivan C., *Polypostors: 2D polygonal impostors for 3D crowds*, Symposium on Interactive 3D graphics and games, pp. 149–155, 2008.

[KPS03] hoon Kim T., Park S. I., and Shin S. Y., *Rhythmic motion synthesis based on motion-beat analysis*, ACM Transactions on Graphics, 22(3): 392–401, 2003.

[KRK08] Kim S., Redon S., and Kim Y. J., *View-dependent dynamics of articulated bodies*, Computer Animation and Virtual Worlds, 19(3–4): 223–233, 2008.

[LCR+02] Lee J., Chai J., Reitsma P. S. A., Hodgins J. K., and Pollard N. S., *Interactive control of avatars animated with human motion data*, SIGGRAPH, pp. 491–500, 2002.

[MT01] Musse S. R., and Thalmann D., *Hierarchical model for real time simulation of virtual human crowds*, IEEE Transactions on Visualization and Computer Graphics, 7(2): 152–164, 2001.

[OZS+03] Oliveira J., Zhang D., Spanlang B., and Buxton B., *Animating Scanned Human Models*, Journal of WSCG, 11(2): 362–369, 2003.

[PAB07] Pelechano N., Allbeck J. M., and Badler N. I., *Controlling individual agents in high-density crowd simulation*, Symposium on Computer Animation, pp. 99–108, 2007.

[PHB07] Payan F., Hahmann S., and Bonneau G.-P., *Deforming surface simplification based on dynamic geometry sampling*, IEEE International Conference on Shape Modeling and Applications, pp.71–80, 2007.

[PW99] Popović Z., and Witkin A., *Physically based motion transformation*, SIGGRAPH, pp. 11–20, 1999.

[Rey87] Reynolds C. W., *Flocks, herds and schools: A distributed behavioral model*, SIGGRAPH, pp. 25–34, 1987.

[RGL05] Redon S., Galoppo N., and Lin M. C., *Adaptive dynamics of articulated bodies*, ACM Transactions on Graphics, 24(3): 936–945, 2005.

[ST05] Shao W., Terzopoulos D., *Autonomous pedestrians*, Symposium on Computer Animation, pp. 19–28, 2005.

[TCP06] Treuille A., Cooper S., and Popović Z., *Continuum crowds*, ACM Transactions on Graphics, 25(3): 1160–1168, 2006.

[TLC02] Tecchia F., Loscos C., and Chrysanthou Y., *Image based crowd rendering*, IEEE Computer Graphics and Applications, 22(2): 36–43, 2002.

[TT94] Tu X., and Terzopoulos D., *Artificial fishes: physics, locomotion, perception, behavior*, SIGGRAPH, pp.43–50, 1994.

[YMP+09] Yersin B., Maïm J., Pettré J., and Thalmann D., *Crowd patches: populating large-scale virtual environments for real-time applications*, Symposium on Interactive 3D graphics and games, pp. 207–214, 2009.

Screen Space Ambient Occlusion for Virtual and Mixed Reality Factory Planning

Fabian Scheer
Daimler AG
Wilhelm-Runge-Str.11
89081 Ulm/Germany
fabian.scheer@daimler.com

Michael Keutel
Otto-von-Guericke-Universität Magdeburg
Hedwigstraße 5
04315 Leipzig/Germany
michael.keutel@googlemail.com

ABSTRACT

Ambient occlusion represents one aspect of global illumination, simulating the actual accessibility of surfaces for indirect illumination due to occlusion of nearby geometry. Recently the approximation in screen space made it feasible for realtime applications. We focus on the utilization of screen space ambient occlusion (SSAO) in virtual and mixed reality environments, especially considering scenes with a large spatial extent and massive amounts of data, typical for industrial factory planning scenarios. Therefore, a sophisticated screen space approach closer to the original definition of ambient occlusion is presented and compared with existing techniques considering the visual quality. Furthermore, we introduce a method to avoid the disappearance of the SSAO effect in depth for scenes with a large spatial extent. A user study compares the impact of our approach to standard SSAO and standard phong shading, regarding the several cues of human depth perception in a virtual and a mixed reality scenario. The evaluation underlines the benefit of our approach for pure virtual scenes and additionally illustrates a different perception of the cues regarding virtual scenes with a half transparent overlay-image of the real world, typical for mixed reality discrepancy check scenarios.

Keywords

Ambient Occlusion, Screen Space, global illumination, discrepancy check, mixed reality, factory planning.

1. INTRODUCTION

The approximation of global illumination (GI) effects in realtime has always been a major issue of computer graphics research. One technique simulating the occlusion term of indirect diffuse light is known as Ambient Occlusion (AO). Surface points being occluded by nearby geometry are shadowed, thus reflecting less light to the viewer, which can be observed in cavities or creases. Although AO is an indirect GI effect, the human visual system gains essential clues of depth, curvature and spatial arrangement from the resulting soft shadows [Lan99a]. Taking advantage of the evolved capabilities of modern graphics hardware [Mit07a] recently presented SSAO as an approximation of AO in image space. In contrast to former computationally expensive AO approaches, SSAO uses the Z-Buffer as a representation of the scene geometry and therefore involves significant advantages:

- computation independent from polygon count
- feasible for fully dynamic content
- calculation can entirely be done on the GPU

Due to an improved realism, SSAO nowadays is widely spread. Being independent from scene geometry and supporting fully dynamic content makes SSAO attractive for the visualization of mass data scenarios with a large spatial extent. Discrepancy checks in mixed reality environments, comparing digitally planned data with real build structures are an actual topic of research [Sch08a]. Typical scenes like complete facilities contain massive amounts of data [Bru06a], thus realtime GI effects are hard to achieve and sparsely considered. Screen space approaches can close this gap by improving the visualization of purely virtual environments and the process of discrepancy checks in mixed environments may also have a benefit. In this paper we present an alternative approach for SSAO, producing visually convincing results, especially in scenes with a large spatial extent. We give a comparison to existing approaches regarding the visual quality and present an evaluation of SSAO in mass data scenarios, showing how SSAO affects the several criteria of human depth perception in merely virtual and in mixed environments. Finally, our evaluation results for the use of SSAO in

discrepancy checks of industrial factory planning applications are described.

2. RELATED WORK

The idea of computing AO approximately in screen space by using a linear Z-Buffer originally evolved from Crytek [Mit07a]. They compute the occlusion value for every pixel *p* in the following way: The respective view space position *P* is computed to sample the vicinity with a set of 3D offset-vectors. Every obtained sample point is projected back to image space and an occluder is found if the depth stored in the linear Z-Buffer is closer to the viewer than the depth of the sample point itself [Kaj09a]. The final occlusion term represents the ratio of solid geometry to empty space in the vicinity of *P*. However, their method suffers from uniform shadowing of actually unoccluded faces and produces edge highlighting artefacts.

[Fil08a] present a similar approach, but apply an attenuation function to weight an occluder's influence dependent on the difference between the depth of the sample point and the one read from the Z-Buffer. The final occlusion value is formed by averaging the contribution of all samples. However, they do not assure that the detected occluders are actually located within the visible hemisphere of the point of interest. As a result they have to deal with self-occlusion occurring on unoccluded surfaces.

[Bav08a] interprete the Z-Buffer as an heightfield and search for the horizon, which is characterized by the slope in the heightfield surrounding a pixel. This horizon-based AO reveals visually pleasing shadows at interactive framerates. They further suggest to speed up computation time by rendering in half resolution and compensate the resulting low quality with a blur pass. We will give a comparison to this approach in section 4.

[Fox08a] consider the orientation of surface normals in the vicinity of a pixel to detect occlusion caused by creases. This approach is fast, but ignores low-frequency occlusion from distant objects.

[Sha07a] introduce an approach to calculate high- and low-frequency parts separately. They assume small spherical occluders to compute the high-frequency occlusion by nearby surface points entirely in screen space. Approximating scene geometry with spherical proxies is used to determine the low-frequency occlusion by distant geometry. Due to the need of approximating the scene geometry for distant occlusion, low frame rates for highly tessellated massive data scenes (e.g. > 20 Mio.) can be expected.

[Rei09a] address the limitation of viewpoint dependence. They perform a realtime voxelization of the scene geometry and compute occlusion by tracing sample rays through the voxelgrid, determining the amount of solid cells being hit. Thus, occlusion caused by hidden surfaces can be considered.

3. VECTOR BASED APPROACH

Referring to the definition of AO, occlusion at a point of interest *P* is caused by any surface point within its visible hemisphere, as this represents all directions of incoming light. We derive our approach from this background and determine occluders, by checking their position to be within the hemisphere of *P*.

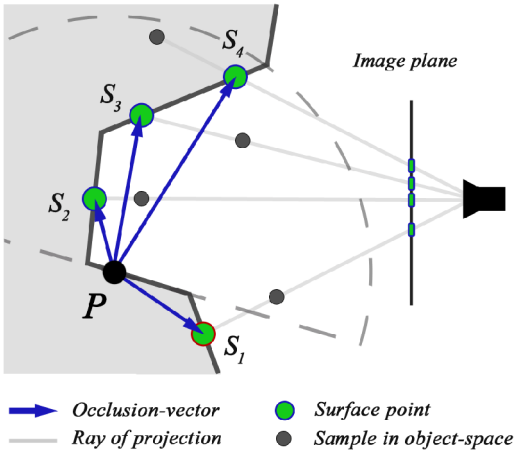


Figure 1. Concept of occlusion-vectors: Sample S_1 is located outside the visible hemisphere and does not contribute to occlusion of P.

Having a set of sampling positions on the image plane, we read the respective depth values from the linear z-buffer and further un-project the sample points into view-space. This gives us a set of surface points representing potential occluders. For any of them we finally create a so called occlusion-vector v_O, pointing from *P* to the respective occluder candidate. An occluder is found if the angle between the surface normal at *P* and v_O is less than 90°, checked by a simple dot-product (see figure 1).

The final occlusion value $o(v_O)$ is determined considering the length of v_O. This idea is straightforward, as long vectors representing distant objects occlude less than short vectors pointing to nearby objects. We apply a function similar to the one usually being used in computer graphics to model the attenuation of light:

$$o(v_O) = \frac{1}{1 + c \cdot \|v_O\|^2} \cdot \left\langle v_O^{norm}, n_P \right\rangle$$

The strength of the quadratic attenuation is controlled by an artistic parameter *c*. In addition a cosine weight expressed by the attached dot product is applied. This is based on [Lan02a] who applies Lamberts law on

the concept of ambient occlusion and concludes that the occlusion is maximal for surface points along the direction of the normal.

The utilization of occlusion-vectors is a simple but effective technique. We correctly ignore surface points not contributing to occlusion, represented by occlusion-vectors pointing outside the hemisphere. This enables us to detect fully unoccluded surface points and therefore avoids false shadowing effects on planar faces or the highlighting of edges, e.g. occurring in [Kaj09a]. For the same reason, our approach avoids the self-occlusion problem described by [Fil08a]. Finally, we acquire an enhanced visual quality of the AO with our approach. This is a result of a higher sample density effectively used for the occluder detection. This issue is clearly illustrated in figure 1, as we obtain two valid occluders for the sample points S_2 and S_3 in contrast to the approaches of [Kaj09a, Fil08a]. Furthermore, the length of the occlusion-vector is a more effective measure for the occlusion strength than the view-direction dependent depth deltas used by [Fil08a]. Hence, we are able to properly represent the distance from the point of interest to the respective occluder.

Sampling technique

Correctly sampling the vicinity of a surface point forms the basis for a well approximated occlusion value. Regarding the space wherein the samples are created, view and image space sampling can be distinguished. View space sampling uses a set of 3D offset-vectors to sample the environment of the point of interest [Mit07a][Fil08a], while image space sampling utilizes 2D vectors to directly create sample positions on the image plane [Bav08a]. The latter makes the back-projection of sample points redundant and offers direct access to the depth- and normal-buffer.

However, we obtain an enhanced visual quality with view space sampling due to a better distribution of the sample positions. According to [Fil08a], we flip an offset-vector in the sampling process if it points outside the visible hemisphere of *P*, obtaining a higher density of sample positions within the hemisphere. Although the direction of the acquired occlusion-vectors is not predictable due to the un-projection with an unknown depth value, the higher initial sampling density significantly enhances the probability to find potential occluders. Figure 2 compares the AO obtained by both sampling methods. The higher noise using image space sampling is clearly observable.

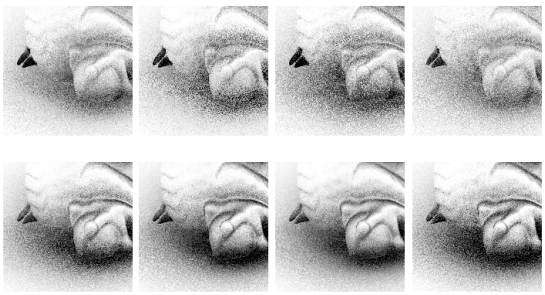


Figure 2. Comparison of image (top) and view space sampling (bottom). Four random sampling patterns applied to Stanford Dragon

Aliasing artefacts due to a constant sampling pattern, are avoided by reflecting the offset-vectors on a random plane as described by [Mit07a]. A Gaussian bilateral filter commonly used for SSAO is further applied to blur the noise, while preventing AO from bleeding through strong object edges [Pet04a].

Accounting for scene scale and extent

Virtual and mixed reality factory planning applications generally involve scenes of a large spatial extent, exhibiting geometry of different metric unit scales due to the individual creation process of the CAD authoring tools. These scenes reveal difficulties for a robust computation of occlusion. We therefore pay attention to the following issues:

1. Definition of the sampling radius
2. Normalization of the occlusion measure

In general the sampling pattern can be scaled by a radius multiplier s_{scene} to give an artistic control over the area being considered by AO. This sampling radius is defined to be constant in view space throughout all currently known approaches. Hence, the area of sampling decreases for surfaces being farther from the viewer, and the visible effect finally vanishes with distance. Therefore, we chose the sampling radius to linearly grow with increasing depth values, keeping its projection constant in screen space. Although this is physically incorrect, it reveals two advantages: First, the larger sampling area produces soft shadowing on distant objects as well and thus makes the AO effect observable within the entire scene (figure 5). An improved visual perception of depth has been verified in a user study presented in section 5. Second, setting the radius in relation to depth couples the radius to individual scene scales, making a further adjustment by the artist redundant. The relation between the initial radius r_B and the resulting sampling radius r_S is derived as follows:

$$r_S = r_B \cdot s_{scene} \cdot \frac{d}{z_P} \qquad d : \text{depth of image plane}$$

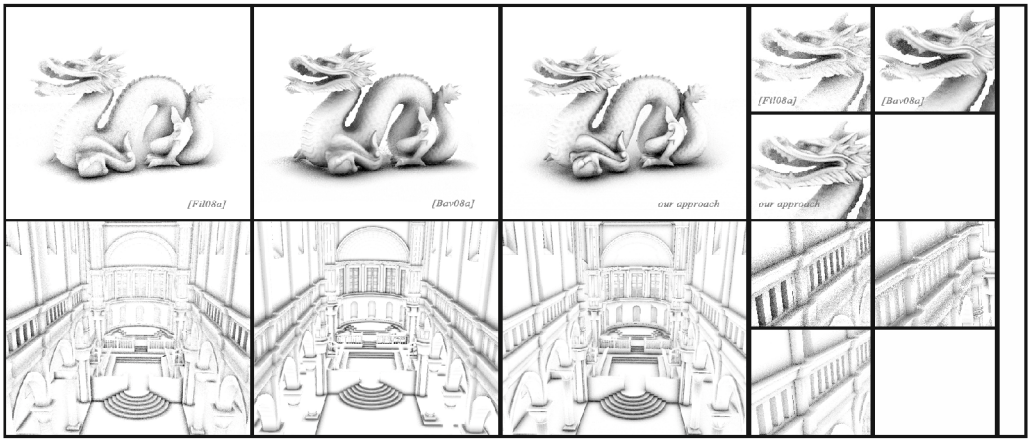


Figure 3. Comparison of our results with the approaches by [Fil08a] and [Bav08a]

In order to account for different scene scales, we normalize the measure of occlusion given by the length of the occlusion-vectors due to their dependence on the underlying scene unit. We hence avoid the individual adjustment of the artistic parameter *c* in extreme ranges of up to $[0,10^7]$. Yet the normalization mathematically requires the maximum length of the occlusion-vector to be known. As these vectors are formed by the un-projection with an afore unknown depth value, the maximum is not given in advance. We deal with this issue by taking the respective sampling radius in view space as an approximate measure and thus doing a quasi-normalization:

$$\|v_O\|' = \frac{\|v_O\|}{r_S}$$

This means the normalized length might exceed the range of [0,1]. Nevertheless, we observe robust results with this approach, being able to deal with different scene scales without the need of manual adjustment for every scene.

4. COMPARISON

In figure 3 we show the comparison of the SSAO approaches acquired with the Stanford Dragon (871K polygons) and the Sibenik Cathedral (80K polygons). 36 Samples were used at a resolution of 800x800 pixels. The unblurred effect of SSAO is shown to allow a proper evaluation. As SSAO generally is a strongly parameter-dependent effect, we attempted to appropriately set the parameter for every approach allowing a visible comparison. The analysis of the images reveals a high amount of noise for [Fil08a] in both scenarios. Surface details and contours are not properly underlined. [Bav08a] in contrast produces better results, tending to show less noise overall. Nevertheless, some regions exhibit a sudden volatile noise that can be observed in the close up illustrations at the chin and head of the dragon and around the row of small archs in the cathedral. However, we obtain a satisfying overall quality of occlusion. [Bav08a] indentified the following criteria affecting computation time of SSAO: screen resolution, resolution of AO computation, sample count and size of the applied blur kernel. We benchmarked the Sibenik Cathedral scenario using a constant resolution of 1280x1024 pixels while increasing the sample count stepwise from 16 to 128 samples per pixel. We captured the time for the mere calculation of the SSAO-Pass listed in table 1. As [Bav08a] and [Rei09a] favor a half resolution computation to significantly speed up, we consider this aspect as well.

Samples	**SSAO time [ms]**	
	Full Res.	Half Res.
16	12.4	1.6
36	24.6	3.5
64	47.0	6.1
128	96.9	12.4
Platform: Intel Core i7 920 2,67GHZ 3GB RAM, Nvidia Geforce GTX 285, OpenGL 2.1, GLSL		

Table 1. SSAO computation time and visual quality

Geo.	**SSAO**	**Blur**	**Application**
3.6ms	12.4ms	3.6ms (13x13)	19,6ms / **51.02FPS**
3.6ms	24.6ms	2.8ms (9x9)	31,0ms / **32.2FPS**

Table 2. Overall performance for 16 (top) and 36 Samples (bottom) per pixel

We gain an acceptable performance feasible for realtime applications and present the overall performance for 16 and 32 samples in table 2. The clear loss of quality applying half resolution

rendering is usually compensated by using big blur kernels. We do not favor this method, as satisfying quality is only obtained using high screen resolutions and the quality loss is too significant.

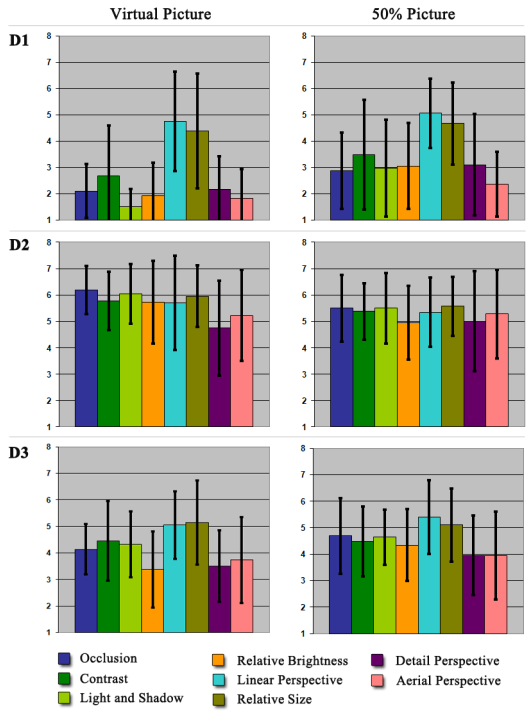


Figure 4. Mean values and standard deviation for test 1 (first col.) and test 2 (second col.)

5. EVALUATION

Even though our approach of keeping the sampling radius constant in image space produces results of enhanced visual quality for scenes with a large spatial extent, we intended to evaluate the impact on the several cues of human depth perception in detail. We further wanted to examine whether there is a change in the perceived criteria of depth using our approach in a mixed reality discrepancy check scenario in contrast to standard SSAO.

Discrepancy checks in industrial applications are performed in-situ at a factory floor with a tracked mobile device [Sch08a]. Therefore, the device is moved at the desired position, and the discrepancy is examined by interactively adjusting the transparency of a plane, textured with the real world video image laying in front of the virtual scene. The visualization is shown on a touch screen mounted on the device. Even if the tracked device can be moved, a single check is mostly done at a fixed position similar to an examination of a freezed image. Due to this setup, binocular or kinetic depth cues [Dra96a] like motion parallax and stereopsis can be neglected in our experiments, keeping focus on the perception of pictorial depth cues. The human perception of depth in virtual and augmented reality environments is a well researched topic in the community [Dra96a, Cut95a, Swa07a, Mur95a]. Nevertheless, the effect of an indirect illumination approximation like SSAO for the several cues of human depth perception has yet not been investigated. According to [Mur95a], we consider depth cues of a constructionist point of view, because the discrepancy check cannot fulfill the key point of the ecological approach regarding the impact of motion. [Mur95a] and [Dra96a] describe various depth cues and additional criteria that have a strong effect on the perception of depth cues. In our experiment we focused on the pictorial depth cues, listed as follows: *Occlusion*, *contrast*, *light and shadows*, *relative brightness*, *linear perspective*, *relative size*, *detail perspective* and *aerial perspective*.

During a period of one week we tested 22 participants (4 female, 18 male) recruited at the Daimler Research Center in Ulm, consisting of students, Phd candidates and scientific assistants aged between 23 and 47 years (M=31.05, SD=6.0). Their experience with virtual or augmented reality applications was asked and yielded a mean value of 4.14 (SD=2.57) on an ordinal scale from 1 to 7. The candidates sat in front of a Toshiba plasma tv-screen with an image diagonal of 42 inches and a distance to the viewer of approximately 55 inches.

Each experiment started by showing every participant three different renderings of a typical factory scene with a large spatial extent. We labeled the renderings *D1* for standard phong shading, *D2* for our approach to SSAO, and *D3* for standard SSAO with a decreasing size of the sample radius in depth. Additionally, the same scene was presented with a 50% alpha blended image taken in the real factory at a viewpoint matching the one of the virtual scene. The several depth cues were explained to the participants, being further instructed to take care of how the three rendering methods support their perception of the different cues, in both the mere virtual and the mixed 50% transparent representation. Next to the tv-screen we put an additional flat screen showing the definitions and sample pictures (not related to any of the test scenes) of the various depth cues for explanation and as a reminder during the tests.

Study Design

Overall, we ran two tests with five different factory planning scenarios. The scenes were randomized to eliminate any scene related effects. In the first test *D1*, *D2*, and *D3* were shown for a virtual scene (figure 5). The three scene representations were arranged side by side on the tv-screen. The resolution of *D1*, *D2* and *D3* was set to match the video resolution (1024x768) of the image processing

camera used for discrepancy checks to obtain comparable results. Since SSAO is a mere indirect effect of GI, and every individual has a different preference of the intensity of such an effect, we gave the participants the opportunity to switch between four slides showing *D2* and *D3* in a varying intensity of the effect, parameterized from weak to strong. *D1* remained unaffected. On a questionnaire the participants marked which of the three representations mostly fulfilled the depth cues as a whole. Next they evaluated their perception of the depth cues for each of the three representations on an ordinal point scale from 1 to 7, or marked an "X" if a conclusion was not found. The participants were asked to assign the points 1 to 7, representing how properly they could perceive the respective depth cue in the whole scene. In the second test *D1*, *D2* and *D3* where shown with a 50% transparent overlay of an image of the real factory, matching the viewpoint of the virtual scene (figure 5). Again the participants were asked to judge which representation mostly fulfills the various depth cues as a whole and assign points ranging from 1 to 7 or mark an "X". Due to former observations, we wanted to prove the hypothesis that some of the depth cues may be perceived differently in the 50% representation and that the general preference for one of the three representations might slightly shift in this scenario. The user study confirmed this assumption, as we will show in the next section. To prove any significance of our results we ran an analysis of variance (ANOVA) for repeated measures of pair wise comparisons with bonferroni correction, as well as a paired t-test to compare the results for the several cues between the virtual and the 50% scenario regarding the respective representation.

Study Results

For the virtual scene all participants marked *D2* as the representation mostly supporting the perception of the depth cues as a whole. For the 50% overlay scene 1 of the participants (4.54%) chose *D1*, 14 (63.63%) chose *D2* and 7 (31.81%) chose *D3*, indicating that SSAO is generally preferred in the discrepancy check scenario. However the advantage of perceiving depth with *D2* compared to *D3* is not that distinct as it was for the mere virtual scenario. The ANOVA for test 1 and test 2 revealed many main effects summarized in table 3 and table 4. The assessment of the several depth cues given by the participants is depicted in figure 4, including the mean value and the standard deviation. Comparing both representations of SSAO to pure phong shading, the ANOVA yields numerous main effects from highly significant ($p<.001$) over very significant ($p<.01$) to significant ($p<.05$), except the cue of linear perspective, where no significant change of the perception between the three representations could be found. Thus, the perception of the linear perspective does not significantly profit from the effect of SSAO. Furthermore, the cue of relative size is unaffected regarding *D1* and *D3*. Comparing *D2* and *D3*, the ANOVA revealed significant or highly significant results, except the cue of relative size yielding only a trend ($p<.10$). The reason for that lies in a stronger perceived effect of the blur and an increased shadowing in depth of our approach. The main effects for *D2* and the 100% selection of *D2* by the participants show that the essential pictorial cues for human depth perception like occlusion [Cut95a] are better perceived with our approach for pure virtual scenes with a large spatial extent. The ANOVA of test 2 revealed numerous main effects as well, but a slight shift in the perception of the several cues can be noticed. For the cue of contrast, we could only assess a trend between *D1* and *D3*, but no significance at all for the cues of detail perspective and relative size. Comparing both SSAO representations with *D1*, no effect was observable regarding the linear perspective, what proved that this cue in reference to test 1 does not take any advantage of the SSAO effect. The perception of the depth cues for the two SSAO representations revealed further interesting differences. Yet no significant change emerged for the cue of occlusion, relative brightness and relative size. The cue of contrast only reveals a trend. However the results for the cue of light & shadow and for the aerial perspective shift from highly significant to very significant, compared to the results of test 1. This could be an explanation why only 65% of the participants chose *D2*, whereas 32% preferred *D3*. Between *D2* and *D3* the most effective depth cue of occlusion [Cut95a] can not be perceived significantly different. The ANOVA of other important cues and additional criteria necessary for the perception of depth, like contrast, relative brightness and relative size also revealed no main effects. Nevertheless, 96% of the participants chose SSAO in general to be the preferred representation for the discrepancy check scenario. *D2* and *D3* do not differ in the perception of important cues, but a tendency to our approach *D2* is still noticeable. Additionally, we ran a paired t-test to evaluate significant changes in the perception of the several cues for *D1*, *D2* and *D3* between the two scenarios. The results are listed in table 5 and show various significant differences, indicating that the perception of depth seriously changes in the 50% overlay-image. According to the significance levels, the perceptual changes in the virtual and 50% scenario appear the most for *D1*. The perception of the cues for *D1* mainly improves in the mixed scene. For *D2* the important cues of occlusion and relative brightness change significantly, with lesser mean values for the

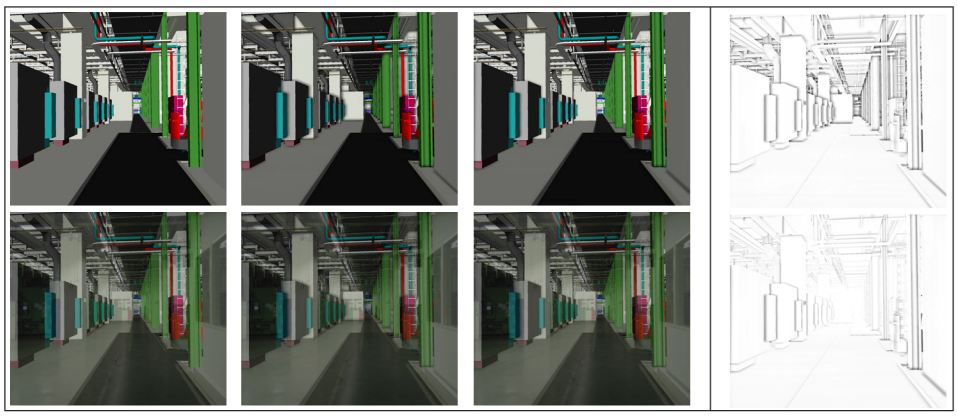


Figure 5. Sample Scene of the user study showing the pure virtual scene (top) and the 50% mixed scene (bottom). The occlusion term for D2 (top right) and D3 (bottom right) is also presented.

mixed scene, and even the clue of light and shadow shows a trends of being perceived worst. *D3* shows a trend of improving the perception for occlusion and also reveals a significant better perception of relative brightness. The strong changes in the perception of such important cues for *D2* and *D3* with *D3* showing a better benefit in the mixed scene explain the choice of 32% of the participants for *D3*. This observation is furthermore confirmed by the results of the ANOVA for test 2, where no main effects between *D2* and *D3* emerged for the important cues of occlusion and relative size.

6. Conclusion

We presented a more accurate realtime approach of SSAO according to the original definition of ambient occlusion. Keeping the sampling radius constant in screen space seriously improves the effect of SSAO for scenes with a large spatial extent. Even if the representation is physical incorrect, a user study confirmed the benefit for the human perception of depth among various cues for pure virtual scenes. Furthermore, a general preference of SSAO for the representation of virtual scenes was observed and compared for several depth cues. For mixed reality scenes with a 50% transparent overlay SSAO also proved to be the appropriate representation with a tendency to our approach. Nevertheless, mixed scenes reveal more balanced differences in perception, especially for *D2* and *D3*. Therefore, our future work will focus on extending SSAO with alternate forms of visualizations and their benefit for the perception in mixed scenarios.

7. ACKNOWLEDGMENTS

This work was partially funded by the AVILUS project of the german Bildungsministerium für Bildung und Forschung (BMB+F). The Dragon is courtesy of the Stanford University and the Sibenik Cathedral is courtesy of Marko Dabrovic

8. REFERENCES

[Bav08a] Bavoil, L., Sainz, M., Dimitrov, R. Image-Space Horizon-Based Ambient Occlusion, ACM SIGGRAPH 2008 talks, 2008.

[Bru06a] Brüderlin, B., Heyer, M., Pfützner, S.: Visibility Guided Rendering to Accelerate 3D Graphics Hardware Performance, in Course Notes EGSR 2006, ACM Eurographics, 2006.

[Cut95a] Cutting,. J.E., Vishton, P.M.: Perceiving Layout and Knowing Distances: The Integration, Relative Potency, and Conceptual Use of Different Information about Depth. In: Epstein, W., Rogers, S. editors. Perception of Space and Motion, p. 69-117, 1995.

[Dra96a] Drascic, D., Milgram, P.: Perceptual Issues in Augmented Reality, in: Proc. Stereoscopic Displays and Virtual Reality Systems SPIE 1996, p 123-134, 1996.

[Eng09a] Engel, W. Shader X7: Advanced Rendering Techniques, Charles River Media, 2009.

[Fil08a] Filion, D., McNaughton, R. Effects & Techniques, ACM SIGGRAPH 2008 classes, pp.133-164, 2008.

[Fox08a] Fox, M., Compton, S. Ambient Occlusive Crease Shading, Game Developer Magazine, 2008.

[Kaj09a] Kajalin, V. Screen Space Ambient Occlusion, in: Engel, W. (Editor): ShaderX7: Advanced Rendering Techniques, pp.413-424, 2009.

[Lan99a] Langer, M.S., Bülthoff, H.H. Perception of shape from shading on a cloudy day, Max-Planck-Institut für biologische Kybernetik, 1999.

[Lan02a] Landis, H. Production-Ready Global Illumination, ACM SIGGRPAH 2002 courses, 2002.

[Mit07a] Mittring, M. Finding next gen: CryEngine2, ACM SIGGRAPH 2007 courses, pp. 97-121, 2007.

[Mur94a] Murray, J.: Some Perspectives on Visual Depth Perception, ACM SIGGRAPH Computer Graphics, Vol. 28, Issue 2, 1994.

[Pet04a] Petschnigg, G., Szeliski, R., Agrawala, M. Digital Photography with flash and no-flash image pairs, ACM SIGGRAPH 2004, pp.664-672, 2004.

[Rei09a] Reinbothe, C., Boubekeur, T., Alexa, M. Hybrid Ambient Occlusion, EUROGRAPHICS 2009 Areas Papers, 2009.

[Sch08a] Schoenfelder, R. and Schmalstieg, D. Augmented Reality for Industrial Building Acceptance, Virtual Reality Conference VR 2008, IEEE Computer Society, 2008.

[Sha07a] Shanmugam, P., Arikan, O. Hardware accelerated ambient occlusion techniques on GPUs, I3D'07 (ACM) , pp.73-80, 2007.

[Swa07a] Swan II, J.E., Adam, J., Kolstad, E., Livingston, M.A. and Smallman, H.S. in: Egocentric Depth Judgements in Optical See-Through Augmented Reality, IEEE Transactions on Visualization and Computer Graphics, Vol. 13, No. 3, 2007.

	Test 1		**Test 2**	
Depth Cue	ANOVA	Bonferroni	ANOVA	Bonferroni
Occlusion	F(2;46) = 87.48	D1<D2, p < .001 D1<D3, p < .001 D2>D3, p < .001	F(2;46) = 21.22	D1<D2, p < .001 D1<D3, p < .001 D2>D3, NO
Contrast	F(2;46) = 20.77	D1<D2, p < .001 D1<D3, p < .001 D2>D3, p < .05	F(2;44) = 9.22	D1<D2, p < .01 D1<D3, p < .1 D2>D3, p < .1
Light and Shadow	F(2;46) = 101.5	D1<D2, p < .001 D1<D3, p < .001 D2>D3, p < .001	F(2;46) = 19.82	D1<D2, p < .001 D1<D3, p < .01 D2>D3, p < .01
Relative Brightness	F(2;46) = 53.16	D1<D2, p < .001 D1<D3, p < .01 D2>D3, p < .001	F(2;44) = 12.33	D1<D2, p < .01 D1<D3, p < .05 D2>D3, NO
Linear Perspective	F(2;42) = 2.41	D1<D2, NO D1<D3, NO D2>D3, NO	F(2;42) = 0.81	D1<D2, NO D1<D3, NO D2<D3, NO
Relative Size	F(2;44) = 6.57	D1<D2, p < .05 D1<D3, NO D2>D3, p < .1	F(2;44) = 7.08	D1<D2, p < .01 D1<D3, NO D2>D3, NO
Detail Perspective	F(2;42) = 21.0	D1<D2, p < .001 D1<D3, p < .01 D2>D3, p < .05	F(2;44) = 10.95	D1<D2, p < .01 D1<D3, NO D2>D3, p < .05
Aerial Perspective	F(2;44) = 39.25	D1<D2, p < .001 D1<D3, p < .001 D2>D3, p < .001	F(2;42) = 28.34	D1<D2, p < .001 D1<D3, p < .01 D2>D3, p < .01

Table 3. Statistical results of test 1 and test 2

Depth Cue	D1	D2	D3
Occlusion	T=-3.37, S1<S2, p<.01	T=3.07, S1>S2, p<.01	T=-1.78, S1<S2, p<.10
Contrast	T=-2.07, S1<S2, p<.10	T=1.06, S1>S2, NO	T=-0.12, S1<S2, NO
Light and Shadow	T=-4.18, S1<S2, p<.001	T=1.74, S1>S2, p<.1	T=-0.90, S1<S2, NO
Relative Brightness	T=-3.67, S1<S2, p<.01	T=2.83, S1>S2, p<.05	T=-2.83, S1<S2, p<.05
Detail Perspective	T=-2.39, S1<S2, p<.05	T=-0.53, S1<S2, NO	T=-0.87, S1<S2, NO
Aerial Perspective	T=-1.99, S1<S2, p<.10	T=-0.43, S1<S2, NO	T=-0.73, S1<S2, NO

Table 4. Statistical results for the t-test (S1/S2 being the mean value of test 1/test 2)

Evolving Time Surfaces in a Virtual Stirred Tank

Bidur Bohara, Farid Harhad
Department of Computer Science
Louisiana State University
Baton Rouge, LA-70803
bbohar1@tigers.lsu.edu,
fharhad@cct.lsu.edu

Werner Benger
Center for Computation & Technology
Louisiana State University
Baton Rouge, LA-70803
werner@cct.lsu.edu

Nathan Brener
S. Sitharama Iyengar, Bijaya B. Karki
Department of Computer Science
Louisiana State University
Baton Rouge, LA-70803
brener@csc.lsu.edu
iyengar@csc.lsu.edu, karki@csc.lsu.edu

Marcel Ritter
Department of Computer Science
University of Innsbruck
Technikerstrasse 21a
A-6020 Innsbruck, Austria
marcel.ritter@student.uibk.ac.at

Kexi Liu, Brygg Ullmer
Department of Computer Science
Center for Computation & Technology
Louisiana State University
Baton Rouge, LA-70803
kliu9@lsu.edu, ullmer@lsu.edu

Somnath Roy
Sumanta Acharya
Department of Mechanical Engineering
Louisiana State University
Baton Rouge, LA-70803
sroy13@tigers.lsu.edu, acharya@me.lsu.edu

ABSTRACT

The complexity of large scale computational fluid dynamic simulations demand powerful tools to investigate the numerical results. Time surfaces are the natural higher-dimensional extension of time lines, the evolution of a seed line of particles in the flow of a vector field. Adaptive refinement of the evolving surface is mandatory for high quality under reasonable computation times. In contrast to the lower-dimensional time line, there is a new set of refinement criteria that may trigger the refinement of a triangular initial surface, such as based on triangle degeneracy, triangle area, surface curvature etc. In this article we describe the computation of time surfaces for initially spherical surfaces. The evolution of such virtual "bubbles" supports analysis of the mixing quality in a stirred tank CFD simulation. We discuss the performance of various possible refinement algorithms, how to interface alternative software solutions and how to effectively deliver the research to the end-users, involving specially designed hardware representing the algorithmic parameters.

Keywords: visualization, CFD, large data, pathlines, timelines, surface refinement

1 INTRODUCTION

1.1 Motivation

Computational Fluid Dynamics (CFD) is a computationally based design and analysis technique for the study of fluid flow. CFD can provide high fidelity temporally and spatially resolved numerical data, which can be based on meshes that range from a few million cells to tens of millions of cells. The data from CFD can range to several hundred thousand time steps and be of sizes in order of terabytes.

Therefore, a key challenge here is the ability to easily mine the time dependent CFD data; extract key features of the flow field; display these spatially evolving features in the space-time domain of interest. In this work, we present an interdisciplinary effort to generate and visualize time surfaces of the fluid flow from the time dependent CFD data. The implementation of time surfaces, such as an evolving surface of a sphere, for analyzing the flow field is more relevant in context of a stirred tank system. The integration of surfaces over time generates an evolving surface that can illustrate key flow characteristics such as how matter injected in a stirred tank disperses, and in what regions of the tank is the turbulence high. Such observations are crucial to identifying the best conditions for optimal mixing.

The CFD dataset was obtained from a large eddy simulation (LES) of flow inside a stirred tank reactor (STR). The simulation is performed on 200 processors (64 bit 2.33 G Hz Xeon quadcore) where each time-step is calculated in approximately 36 seconds. Stirred tanks are the most commonly used mixing device in chemical and processing industries. Improvements in the design of stirred tanks can translate into several billion dollar annual profit. However, better designs of stirred tanks require detailed understanding of flow and mixing be-

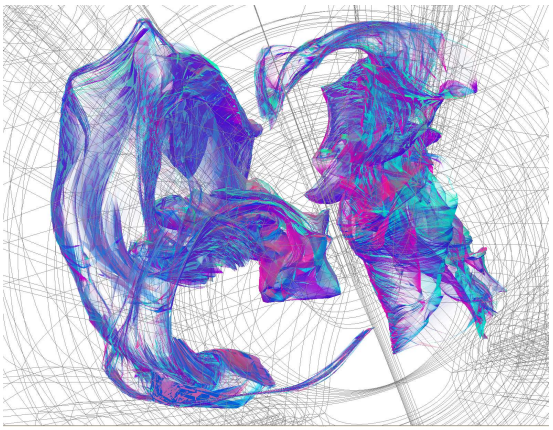


Figure 1: Two evolving spheres visualized just before their mixing in the Stirred Tank simulation system.

havior inside the tank. The present study focuses on analyzing the dynamics of mixing inside the tank. Turbulent flow inside the stirred tank was solved numerically using LES to resolve small-scale turbulent fluctuations and the immersed boundary method (IBM) in order to model the rotating impeller blade in the framework of a fixed curvilinear grid representing the tank geometry. The grid is distributed over 2088 blocks and comprised of 3.1 million cells in total. Flow variables like velocity and pressure are defined at the center of each cell and computed for each time step over a total of 5700 time steps representing 25 complete rotations of the impeller. The handling and processing of these voluminous, multi-block, non-uniform curvilinear datasets to generate time surfaces and track set of particles in the fluid flow is the main challenge addressed in this paper.

1.2 Related Work

One of the earliest works related to this problem is the generation of stream surfaces, in particular Hultquist's attempt to generate a triangular mesh representation of streamsurfaces. Hultquist introduced an algorithm that constructs stream surfaces by generating triangular tiles of adjacent streamlines or stream ribbons. In Hultquist's algorithm, tiling is done in a greedy fashion. When forming the next triangle, the shortest leading edge is selected out of the two possible trailing triangles and appended to the ribbon. Each ribbon forming the stream surface is advanced until it is of equivalent length to its neighboring ribbon along the curve they share [13]. Particles are added to the trail of the stream surface by splitting wide ribbons, and particles are removed from the stream surface by merging two narrow (and adjacent) ribbons into one. Note that Hultquist's algorithm was developed for steady flows. Also, advancing the front of the stream surface requires examining all the trailing ribbons.

Along the same lines, Schafhitzel et al. [15] adopted the Hultquist criteria to define when particles are removed or added, but they derived a point-based algorithm that is designed for GPU implementation. In addition to rendering a stream surface, they applied line integral convolution to show the flow field patterns along the surface.

Rather than remeshing a stream surface when the surface becomes highly distorted, von Funck et al [23] introduced a new representation of smoke in a flow as a semi transparent surface by adjusting opacity of triangles that get highly distorted and making them fade. Throughout the evolution of the smoke surface, they do not change the mesh, but rather use the optical model of smoke as smoke tends to fade in high divergent areas [23]. However, the authors report that this method does not work well if the seeding structure is a volume structure instead of a line structure.

Core tangibles [21] we use in this paper are physical interaction elements such as Cartouche menus and interaction trays, which serve common roles across a variety of tangible and embedded interfaces. These elements can be integrated to dynamically bind discrete and continuous interactors to various digital behaviors. Many toolkits support low-level tangible user interface design, allowing designers to assemble physical components into hardware prototypes which can be interfaced to software applications using event-based communication. Notable examples include PHidgets [10], Arduino [2], iStuff [1], SmartIts [3] etc. Core tangibles focus on tangible interfaces for visualization, simulation, presentation, and education, often toward collaborative use by scientist end-users [21].

2 MATHEMATICAL BACKGROUND

In the domain of computer graphics one distinguishes four categories of integration lines $q \subset M$ that can be computed from a time-dependent vector field $v \in \mathscr{T}(M)$, mathematically a section of the tangent bundle $T(M)$ on a manifold M describing spacetime: *path lines*, *stream lines*, *streak lines* and *material line*s. Each category represents a different aspect of the vector field:

path lines (also called *trajectories*) follow the evolution of a test particle as it is dragged around by the vector field over time.

stream lines (also called *field lines*) represent the instantaneous direction of the vector field; they are identical to path lines if the vector field is constant over time.

streak lines represent the trace of repeatedly emitted particles from the same location, such as a trail of smoke.

material lines (also called time lines) depict the location of a set of particles, initially positioned along a seed line, under the flow of the vector field.

Each of these lines comes with different characteristics: stream lines and path lines are integration lines that are tangential to the vector field at each point

$$\dot{q} \equiv \frac{d}{ds} q(s) = v(q(s)) \tag{1}$$

Since the underlying differential equation is of first order, the solution is uniquely determined by specifying the initial condition $q(0) = q_0$ by a seed point $q_0 \in M$ in spacetime. Neither stream lines nor path lines can self-intersect (in contrast to e.g. geodesics, which are solutions of a second order differential equation). However, a path line may cross the same spatial location at different times, so the spatial projection of a path line may self-intersect.

In contrast to stream and path lines, streak and material lines are one-dimensional cuts of two-dimensional integration surfaces $S \subset M$, $dim(S) = 2$. This surface is constructed from all integral lines that pass through an event on this initial seed line $q_0(\tau)$:

$$S = \{q : \mathbb{R} \to M, \dot{q}(s) = v(q(s)), q(0) = q_0(\tau)\}$$

The resulting surface contains a natural parametrization $S(s,\tau)$ by the initial seed parameter τ and the integration parameter s. It carries an induced natural coordinate basis of tangential vectors $\{\vec{\partial}_\tau, \vec{\partial}_s\}$, with $\vec{\partial}_s \equiv \dot{q} = v$. For a streak line, the initial seed line $q_0(\tau)$ is timelike as new particles are emitted from the same location over time, $dq_0(\tau)/dt \neq 0$, for a material line the seed line is spacelike $dq_0(\tau)/dt = 0$, a set of points at the same instant of time. The respective streak/time line is the set of points of the surface $q(t) = S_{t=const.}$ for a constant time. If the integration parameter is chosen to be proportional to the time $s \propto t$, for instance when performing Euler steps, then the original seed line parameter τ provides a natural parameter for the resulting lines, i.e. each point along a time line is advanced by the same time difference dt at each integration step.

Refinement of lines by introducing new integration points is mandatory to sustain numerical accuracy of the results. The ideas of the Hultquist algorithm [12] and its improvements by Stalling [17] could be applied also to the spatio-temporal case, however such would result in the requirement to perform timelike interpolation of the vector field. For data sets that are non-equidistant in time such as adaptive mesh refinement data generated from Berger-Oliger schemes [6] finding the right time interval for a given spatial location this becomes non-trivial. For now we refrain from non-equidistant temporal refinement (such as done in [14]), though this is an option – if not requirement – for future work.

A time surface is the two-dimensional generalization of a time line, a volumetric object in spacetime. The Hultquist algorithm, if applied to a spatio-temporal surface, discusses criteria on refining one edge, whereas here we have a much richer set of possible surface characteristics that may trigger creation or deletion of integration points. Some options are to refine a surface at locations where

- a triangle's edge
- a triangle's area
- a triangle's curvature
- a triangle degeneration ("stretching")

becomes larger than a certain threshold. Section 5.1 reviews our results experimenting with different such criteria.

3 SOLUTION

3.1 Data Model

We use the VISH [4] visualization shell as our implementation platform. It supports the concept of fiber bundles [8] for the data model. The data model consists of seven levels, each of which is comprised of compatible arrays that represent a certain property of the dataset [5]. These levels, which constitute a Bundle, are Slice, Grid, Skeleton, Representation, Field, Fragment and Compound. The Field represents arrays of primitive data types, such as int, double, bool, etc., and the collection of Fields describes the entire Grid. The Grid objects for different time slices are bundled together and are represented as a Bundle. As an example of our implementation, each Field contains values of a property such as coordinates, connectivity information, velocity, etc. The collection of these Fields is a Grid object, and the collection of Grid objects for all time slices is the Bundle of the entire dataset.

The dataset used for visualizing the features of fluid flow contains numerical data for 2088 curvilinear blocks constituting the virtual stirred tank. The input vector field is fragmented and these fragments are the blocks of the Grid. The input dataset for each time slice consists of coordinate location, pressure and fluid velocity for each grid point in the entire 2088 blocks. These properties are stored as Fields in the Grid object for each time slice, and these Grid objects are then combined into a Bundle.

When a multi-block is accessed for the first time, a Uniform-Grid-Mapper is created which is a uniform grid having the same size as a world coordinate aligned bounding box of the multi-block. For each cell of the Uniform-Grid-Mapper a list of curvi linear block cells (indices) is stored which intersect the Uni-Grid-Mapper cell by doing one iteration over all curvilinear grid cells and a fast min/max test. When computing the local multi-block coordinates the corresponding Uni-Grid-Mapper cell is identified first which then selects a small number of curvilinear cells for the Newton iteration.

Uni-Grid-Mapper objects are stored in the Grid object of the vector field and can be reused when accessing the same multi-block again later.

3.2 Out of Core Memory Management

The original approach taken while visualizing the features of fluid flow is to keep the entire vector field data in the main memory and integrate over the vector field to extract the features. However, with the necessity of visualizing the time-dependent 3D vector field, the original approach has restrictions, such as the size of the time-dependent data can easily exceed the capacity of main memory of even state of the art workstations. In [24], the authors present the concept of an out-of-core data handling strategy to process the large time dependent dataset by only loading parts of the data at a time and processing it. Two major strategies presented for out-of-core data handling are Block-wise random access and Slice-wise sequential access. The authors emphasize the Slice-wise sequential access strategy for handling the data given in time slices, however, we have implemented both Block-wise access and Slice-wise access of time-dependent data while generating the time surfaces for visualizing the fluid flow.

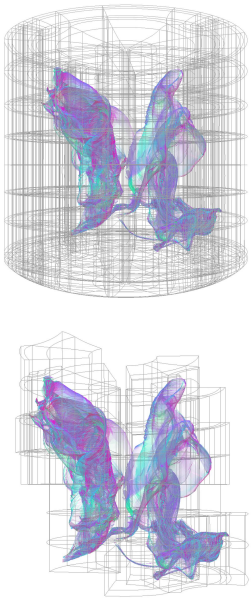


Figure 2: Time surface computed from a vector field given in 2088 fragments (curvilinear blocks) covering the Stirred Tank Grid (top). Only those fragments that affect the evolution of the time surface (bottom) are actually loaded into memory.

The virtual stirred tank system has 2088 blocks, and each block has vector field data for every time slice. The data for each time slice is accessed only once as a Grid object from the input Bundle and processed to generate the time surface at that particular time. The integration of the time surface does not process all the blocks, instead only the blocks that are touched at the given time slice are loaded and processed.

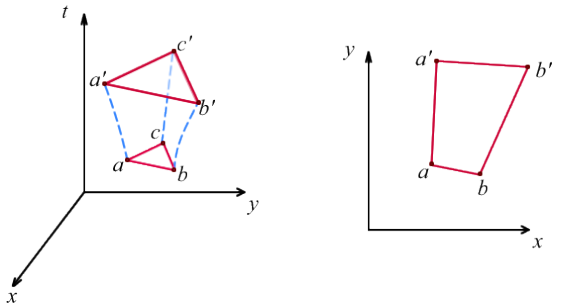


Figure 3: Particle advection of a 2-dimensional element vs. a 1-dimensional element. In our case, our surface element is in 3-dimensional space spanned over time.

At every time slice two Grid Objects are handled, one containing the input data of the vector field and the other consisting of seed points and connectivity information among the seed points. The connectivity information is used to generate the triangle mesh for surface generation. In the case of no surface refinement, the connectivity information is constant throughout the time slices and is stored once and used multiple times. This conserves the memory and also reduces the memory access. However, with surface refinement the number of points and their connectivity changes over time resulting in an increase in memory usage.

3.3 Particle Seeding and Advection

Our set of particle seeds q_{i,t_0} for $i = 0,\ldots,n-1$, lie on a sphere. At any given time $t > t_0$, the time surface is represented as a triangular mesh formed by the particles $q_{i,t}$ that have been advected using equation 1. Figure 3 illustrates the difference between our seeding approach versus Hultquist's where we are evolving a surface element (a triangle) over time as opposed to spanning a surface out of a line segment element.

3.4 Triangular Mesh Refinements

As time elapses, the triangular mesh of particles enlarges and twists according to the flow field. To preserve the quality of the mesh, we refine it by adding new particles and advecting them while updating the mesh connectivity. Of the possible refinements criteria mentioned above, we have implemented the following:

Edge length: If the distance between pairwise particles of a triangle is larger than a threshold edge length, we insert a new midpoint and subdivide the triangle accordingly.

Triangle area: If the area of the triangle formed by the new positions of the particle triplet is larger than a threshold area, we insert three midpoints and subdivide the triangle to a new set of four triangles.

4 ALTERNATIVE APPROACHES

In order to verify and compare our results with other implementations, we also investigate alternative implementations. Paraview [11] is one of the well known and widely used visualization tools in the scientific community. It addresses issues pertaining to the visualization of large scale data-sets using high-performance computing environments. It can be perceived as a framework around the well known Visualization Toolkit [16] library. It not only provides a GUI to Visualization Toolkit(VTK), but also provides a convenient environment for intuitive visual programming of the visualization pipeline.

Paraview has implicit mechanisms for handling scale, both in terms of data and computation [7]. It achieves this by providing generalized abstractions for parallelization and distribution. Therefore a scientist using Paraview can switch from visualizing smaller data-sets on a desktop computer to a much larger data-set utilizing a large HPC infrastructure, with minimum effort.

We describe ongoing work and approaches to porting and visualizing the given F5 (fiber-bundle) data-set, as described in 3.1, in Paraview.

4.1 Porting Fiber-bundle (F5) to Paraview

The 500GB fiber-bundle data-set is provided in the F5 format. This format has no native support in Paraview and some form of conversion would be required to utilize the data. One approach to solve this problem is to use a format converter and separately convert the entire file to a natively supported format. However, this approach causes redundant data and can waste considerable amount of space on the storage disk. An alternative solution is to write a custom reader into Paraview such that the data is read and mapped into internal VTK data-structures. This approach adds an additional computation time into the visualization pipeline and can cause unnecessary slowdown of the visualization process.

An ideal solution would be a combination of the above mentioned approaches such that both space and time optimization can be achieved. Such a solution is possible in our case due to a certain characteristic of the F5 format (explained shortly) and the use of XDMF (eXtensible Data Model and Format) [9] which is supported in Paraview. An F5 format is characteristically a specific description or organization of the HDF5 data format. All HDF5 readers and commands which typically work on HDF5 formats also work on F5. The XDMF data format is an XML format for data generally known as a "light data". It provides light weight descriptions of the "heavy data" which is typically a HDF5 file containing the actual data. A XDMF file can thus be seen as an index into the HDF5 file and is usually much smaller in size, taking very less time to get generated.

Paraview is supplied with the generated XDMF file through which it can access the data in the corresponding HDF5 (or F5) file. No other reader or converter is necessary. An added advantage of this approach is that parallel file readers (if supported) and other parallel algorithms can be used to quickly access and process very large data-sets. We thus leverage on the parallel and distributed framework already provided in Paraview.

5 RESULTS

5.1 Surface Refinement

We benchmarked our implementation with a 30-timestep subset (85 MB per timestep) of the stirred tank data and on a 64-bit dual core (2GHz each) pentium laptop machine with 4GB of RAM. We advected one sphere for the first 30 timesteps of the simulation. Due to the small size of our test data, we could not notice a difference in time surface meshing quality from the visualization itself, but from the data in tables 1 and 2, we notice a slight performance improvement of the area criteria over the edge length criteria. Though the number of particles is slightly higher in the second case, this suggests that the quality of the surface with the area criterion is better.

threshold	tot points	avg time/slice	tot time
0.005	4269	6.480	200.868
0.01	822	1.519	47.1
0.02	258	1.165	36.101

Table 1: Timing Analysis (in seconds) for the Edge Length Criteria

threshold	tot points	avg time/slice	tot time
0.005	4269	6.864	212.785
0.01	837	1.454	45.08
0.02	258	1.150	35.646

Table 2: Timing Analysis (in seconds) for the Triangle Area Criteria

From either tables 1 or 2, picking a threshold too small compared to the characteristic of the triangle being examined, results in maximum refinement, while a large enough threshold leads to no refinement at all.

5.2 Timing Analysis

For the overall integration and refinement of the time surface, we used a larger dataset of size 12GB with 150 timesteps. We ran the implementation on a 64bit quadcore workstation with 64 GB of RAM. We used the edge length criterion with a threshold of 0.01.

The listing in the Table 3 is for 12 GB of input data from an initial time of 0 to a final time of 150. Initially the number of points is 516, which increases over time as more points are generated for surface refinement. As the number of points increases, the computation time

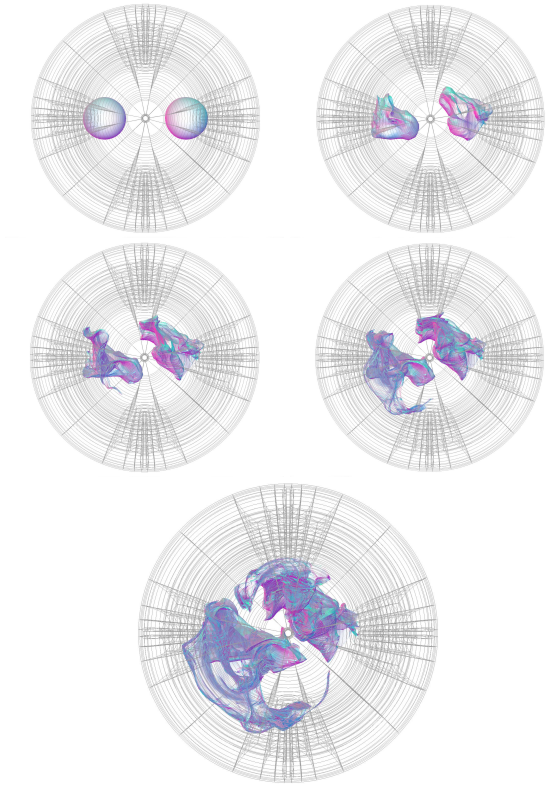


Figure 4: Images showing evolution of two spheres at time slices 0, 50, 100, 125 and 150, respectively from left-top to bottom, as seen top-view of the stirred tank. First image shows the seed spheres, and the last image shows two sphere just before the surfaces are about to mix.

time	no. of points	time/slice(s)	time/point(ms)
0	516	0.4	7.0
50	3468	2.0	5.9
100	15822	7.4	4.8
125	41574	18.8	4.7
150	129939	49.7	4.0

Table 3: Timing Analysis for Threshold=0.01

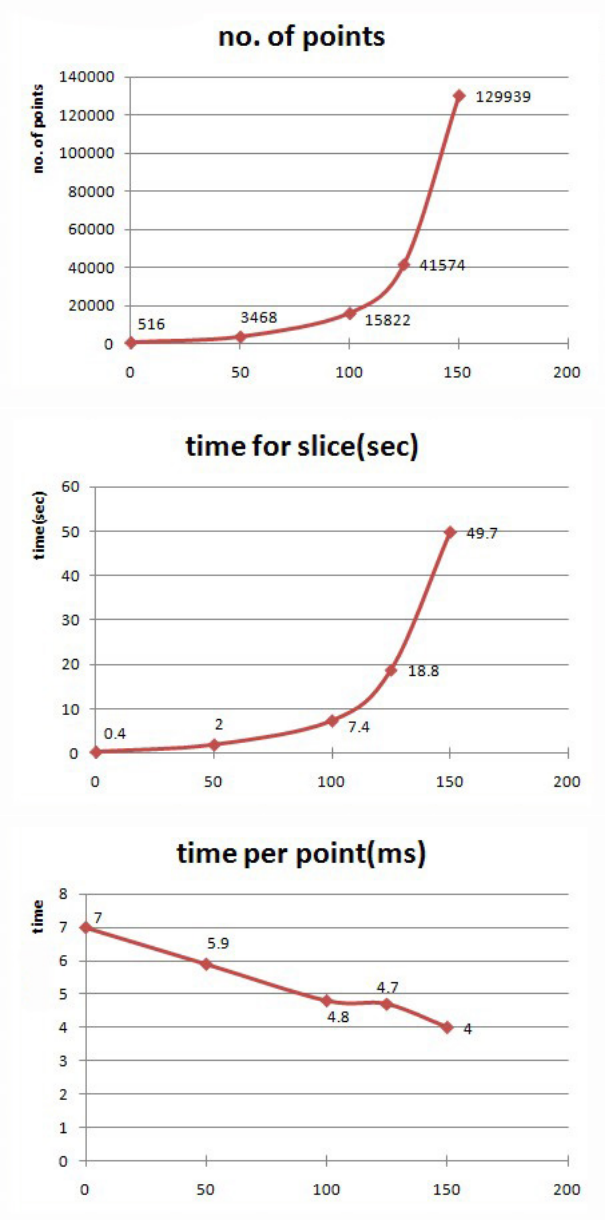


Figure 5: First two graphs shows the increase in no. of points and thus increase in processing time per slice over the time. Third grpah shows the decrease in time per point as number of point increases.

for the next time slice increases. However, the time per point seems to be slowly decreasing, as seen in third graph of Figure 5. This may be because more and more points tend to locate in the same block and the data of one block is shared by many points, resulting in less memory access per point.

6 DEPLOYMENT TO END USERS

Results of the algorithm can be investigated better if we explore the entire time evolution of the surface interactively, by navigating through space and time. In most visualization environments, the graphical user interface is tightly coupled with the underlying visualization functionality. One feature of VISH is that it decouples the interface from the underlying visualization application. At least in principle, this makes it as easy to couple VISH to a CAVE immersive environment, a web based distributed interface, or physical interaction devices as to the provided traditional 2D graphical user interface. As an example of this, we have based a significant portion of our interaction with the present large dataset from stirred tank with "viz tangible" interaction devices. An example of this is pictured in Figure 6. Earlier stages of this work have been described in [22, 20, 19, 18].

An application programming interface (API) is under development which supports coupling tangibles to VISH and other visualization environments. In this API, when interaction control messages are sent (trig-

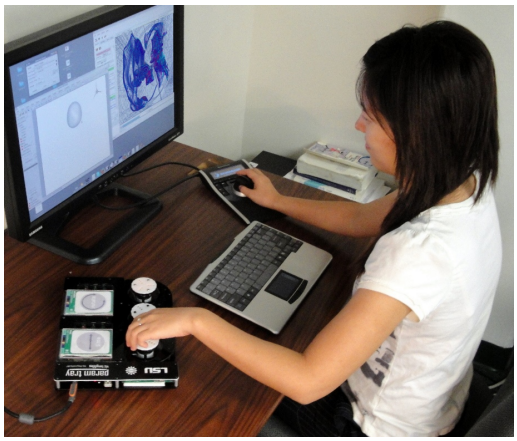


Figure 6: User physically manipulating VISH application through "viz tangibles" interaction devices

gered by physical events, such as RFID entrance/exit or the turning of a knob), they trigger corresponding methods in VISH. We use cartouches – RFID-tagged interaction cards [19, 20] – as physical interactors which describe data and operations within the VISH environment. Users can access, explore and manipulate datasets by placing appropriate cartouches on an interaction tray (Figures 6, 7), and making appropriate button presses, wheel rotations, etc.

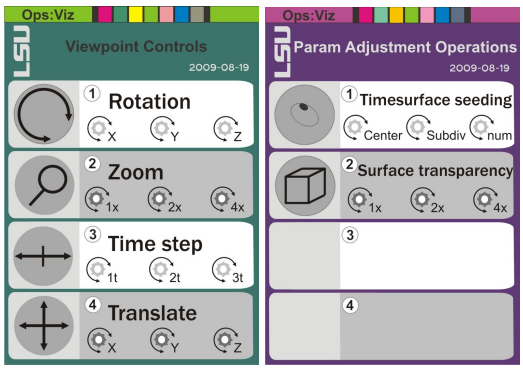


Figure 7: Cartouche cards for viewpoint control and parameter adjustment operations

In our present implementation, we have used two classes of cartouche objects. These are summarized below:

1. *Viewpoint operations:* Specific supported view point controls include rotation, zooming, and translation. In the case of rotation and translation, individual wheels are bounds to the (e.g.) x, y, z axis. In the context of zooming or time step navigation, wheels represent different scales of space and time navigation.
2. *Parameter Adjustment operations:* Our current implementation includes time surface seedings and surface transparency adjustment. For time surface seedings, we steer center of seeds, number of subdivisions, etc. to parameter wheels. Within surface transparency adjustment, wheels are bounded to different scales of surface transparency.

In future, we hope quantities in high dimensional parameter space such as curvature and torsion of the surface can also be explored effectively with the integration of "viz tangibles" and the API.

7 CONCLUSION

While most of the previous visualization techniques for fluid flow have concentrated on flow streamlines and pathlines, our approach has been directed towards generating the time surfaces of the flow. The interdependencies of integration over a vector field require random access to amounts of data beyond a single workstation's capabilities, while at the same time requiring shared memory for required refinements. This limits available hardware and impacts parallelization efforts. The evolution of a seed surface required refinement of its corresponding triangular mesh to preserve the quality of the time surface over time. From the results we noticed a slight superior quality of the area refinement criterion over the edge length criterion.

8 ACKNOWLEDGMENTS

We thank the VISH development team, among them Georg Ritter, University of Innsbruck, and Hans-Peter Bischof, Rochester Institute of Technology, Amitava Jana and Sanjay Kodiyalam from Southern University, Baton Rouge, for their support. This research employed resources of the Center for Computation & Technology at Louisiana State University, which is supported by funding from the Louisiana legislature's Information Technology Initiative. Portions of this work were supported by NSF/EPSCoR Award No. EPS-0701491 (CyberTools), NSF MRI-0521559 (Viz Tangibles) and IGERT (NSF Grant DGE-0504507).

9 ADDITIONAL AUTHORS

Additional Authors: Nikhil Shetty, Vignesh Natesan, and Carolina Cruz-Neira, Center of Advanced Computer Studies, University of Louisiana at Lafayette; nikhil.j.shetty@gmail.com; vigneshn85@gmail.com; carolina.louisiana@gmail.com.

REFERENCES

[1] R. Ballagas, M. Ringel, M. Stone, and J. Borchers. iStuff: a physical user interface toolkit for ubiquitous computing environments.

[2] M. Banzi. *Getting Started with Arduino*. Make Books - Imprint of: O'Reilly Media, Sebastopol, CA, 2008.

[3] M. Beigl and H. Gellersen. Smart-its: An embedded platform for smart objects. In *Smart Objects Conference (sOc)*, volume 2003. Citeseer, 2003.

[4] W. Benger, G. Ritter, and R. Heinzl. The Concepts of VISH. In 4^{th} *High-End Visualization Workshop, Obergurgl, Tyrol, Austria, June 18-21, 2007*, pages 26–39. Berlin, Lehmanns Media-LOB.de, 2007.

[5] W. Benger, M. Ritter, S. Acharya, S. Roy, and F. Jijao. Fiberbundle-based visualization of a stir tank fluid. In *WSCG 2009, Plzen*, 2009.

[6] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.

[7] J. Biddiscombe, B. Geveci, K. Martin, K. Morel, and D. Thompson. Time dependent processing in a parallel pipeline architecture. *IEEE Transactions on Visualization and Computer Graphics*, 13:2007.

[8] D. M. Butler and M. H. Pendley. A visualization model based on the mathematics of fiber bundles. *Computers in Physics*, 3(5):45–51, sep/oct 1989.

[9] J. A. Clarke and R. R. Namburu. A distributed computing environment for interdisciplinary applications. *Currency and Computation: Practice and Experience*, 14, Grid Computing environments Special Issue:13–15, 2002.

[10] S. Greenberg and C. Fitchett. Phidgets: easy development of physical interfaces through physical widgets. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 209–218. ACM New York, NY, USA, 2001.

[11] A. Henderson. Paraview guide, a parallel visualization application, 2005.

[12] J. P. Hultquist. Constructing stream surfaces in steady 3d vector fields. In *Visualization '92*, pages 171–178. IEEE Computer Society, 1992.

[13] J. P. M. Hultquist. Constructing stream surfaces in steady 3d vector fields. In *VIS '92: Proceedings of the 3rd conference on Visualization '92*, pages 171–178, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press.

[14] H. Krishnan, C. Garth, and K. I. Joy. Time and streak surfaces for flow visualization in large time-varying data sets. *Proc. IEEE Visualization '09*, Oct. 2009.

[15] T. Schafhitzel, E. Tejada, D. Weiskopf, and T. Ertl. Point-based stream surfaces and path surfaces. In *GI '07: Proceedings of Graphics Interface 2007*, pages 289–296, New York, NY, USA, 2007. ACM.

[16] W. Schroeder, K. Martin, and W. Lorensen. The visualization toolkit: An object oriented approach to 3d graphics, 1996.

[17] D. Stalling. *Fast Texture-Based Algorithms for Vector Field Visualization*. PhD thesis, Free University Berlin, 1998.

[18] C. Toole, B. Ullmer, R. Sankaran, K. Liu, S. Jandhyala, C. W. Branton, and A. Hutanu. Tangible interfaces for manipulating distributed scientific visualization applications. *Submitted to Proc. of TEI'10*, 2010.

[19] B. Ullmer, Z. Dever, R. Sankaran, C. Toole, C. Freeman, B. Casady, C. Wiley, M. Diabi, A. J. Wallace, M. Delatin, B. Tregre, K. Liu, S. Jandhyala, R. Kooima, C. W. Branton, and R. Parker. Cartouche: conventions for tangibles bridging diverse interactive systems. *Submitted to Proc. of TEI'10*, 2010.

[20] B. Ullmer, A. Hutanu, W. Benger, and H.-C. Hege. Emerging tangible interfaces for facilitating collaborative immersive visualizations. *NSF Lake Tahoe Workshop on Collaborative Virtual Reality and Visualization*, 2003.

[21] B. Ullmer, R. Sankaran, S. Jandhyala, B. Tregre, C. Toole, K. Kallakuri, C. Laan, M. Hess, F. Harhad, U. Wiggins, et al. Tangible menus and interaction trays: core tangibles for common physical/digital activities. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 209–212. ACM New York, NY, USA, 2008.

[22] B. Ullmer, R. Sankaran, S. Jandhyala, B. Tregre, C. Toole, K. Kallakuri, C. Laan, M. Hess, F. Harhad, U. Wiggins, and S. Sun. Tangible menus and interaction trays: core tangibles for common physical/digital activities. In *Proc. of TEI '08*, pages 209–212, 2008.

[23] W. von Funck, T. Weinkauf, H. Theisel, and H.-P. Seidel. Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization 2008)*, 14(6):1396–1403, November - December 2008.

[24] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Feature flow fields in out-of-core settings. In H. Hauser, H. Hagen, and H. Theisel, editors, *Topology-based Methods in Visualization*, Mathematics and Visualization, pages 51–64. Springer, 2007. Topo-In-Vis 2005, Budmerice, Slovakia, Sept. 29 - 30.

Complex Geometric Primitive Extraction on Graphics Processing Unit

Mert Değirmenci
Department of Computer Engineering,
Middle East Technical University, Turkey
mert.degirmenci@ceng.metu.edu.tr

ABSTRACT

Extracting complex geometric primitives from 2-D imagery is a long-standing problem that researchers have had to deal with. Various approaches were tried from Hough transform based methods to stochastic algorithms. However, serial implementations lack sufficient scalability on high resolution imagery. As sequential computing power cannot pace up with the increase in size of datasets, researchers are compelled to exploit parallel computational resources and algorithms. In this study, we have merged parallelization capability of GPUs with inherent parallelism on genetic algorithms to cope with the problem of detecting complex geometric primitives on high resolution imagery. We have implemented ellipse detection on commodity graphics processing unit and showed that our GPU implementation achieve high speed-up relative to state of the art CPU by experimental results.

Keywords

Geometric primitive extraction, Genetic algorithm, GPU.

1. INTRODUCTION

Most of the tasks in image analysis involve geometric primitive extraction as a preprocessing step. Therefore, efficiently detecting primitives is an important research topic in the domain of computer vision. Researchers have proposed numerous solutions to detect geometric primitives [Tia96], [Rob98], [Rot94].

Hough transform, HT, based techniques have been widely used for geometric primitive extraction. The idea of classical Hough transform was to perform a mapping from image space to parameter space in order to obtain a function. Optima's of this function correspond to instances of primitives. HT is powerful for detecting simple primitives such as lines. However, computational and memory complexity of classical Hough transform increase exponentially along with the number of parameters [Pen99].

To alleviate problems of HT, randomized Hough transform, RHT, has been proposed by Kultanen et al. [Kul90]. RHT performs converging mapping to parameter space by randomly sampling a number of pixels from image space that will ensure convergence to one point.

RHT is not the only approach that tried to reduce the parameter space of classical Hough transform. Researchers have proposed partitioning parameter space to smaller subspaces by using features of specific complex geometric primitives. Si-Cheng et al. proposed real time ellipse detector that uses analytical properties of ellipses and edge gradient information to divide 5-D parameter space of classical Hough transform [Sic05].

Stochastic approaches were also tried to detect geometric primitives. Ever since geometric primitive extraction has been shown to be an optimization problem [Rot93], researchers have tried optimization algorithms to extract primitives [Rot94], [Pen99].

Genetic algorithm, GA, is one of the most popular stochastic approaches to extract geometric primitives. In the context of natural evolution, genetic algorithm simulates parallel evolution of individuals to find approximate solutions to optimization problems for which no efficient algorithm is known to find the exact solution otherwise. Since primitive extraction can be regarded as an optima search problem, genetic algorithm fit well into the problem of detecting imperfect instances of geometric primitives. Thus, many researchers have developed genetic algorithm oriented methods to detect geometric shapes [Yao05], [Kaw98], [Rot94].

GA based techniques use the feedback obtained from population to cluster solutions around optima's, whereas HT based methods exhaustively search image space irrespective of distribution in parameter space. Although genetic algorithms has inherent strengths over HT based methods, extracting all instances of a geometric primitive cannot be directly translated into problem space of classical genetic algorithm.

Detection of single most observable geometric primitive, on the other hand, can be converted into unimodal optimization problem. However, in real life applications, we need to detect multiple primitives that qualify certain criteria. This corresponds to multi-modal optimization. Researchers have proposed variety of multi-modal optimization stochastic algorithms to extract geometric primitives from 2-D imagery.

Lutton et al. proposed sharing genetic algorithm, SGA, where diversification of population is maintained by promoting fitness of local optima [Lut94]. Fitness of similar individuals is shared to decrease the clustering around global optima while guiding search towards uninhabited areas.

Yao et al. proposed multi-population genetic algorithm [Yao05]. They suggest island oriented model of population where each individual is prompted to live on a matching island. New island generation is also possible if an individual is not close to any existing island.

In this study, we have implemented multi-population genetic algorithm for extraction of complex geometric primitive on graphics processing unit. We have chosen Nvidia's compute unified device architecture, CUDA, as a development platform. Our genetic algorithm implementation on GPU has been tested for detection of ellipses and shown successful improvement over an optimized serial implementation.

2. GEOMETRIC PRIMITIVE EXTRACTION ON GPU

Our implementation of geometric primitive extraction starts by detecting edges of image. Edge image is then partitioned into tiles on which a copy of our genetic algorithm runs. In CUDA , each block works on its corresponding isolated island, and each thread is responsible for an individual on that island, as it can be seen in figure [1]. After loading data, threads behave as individuals who mate with other individuals to produce fitter offsprings for the next generation. However, these responsibilities are distributed after coalesced loads of tiles into shared memory. Current graphics processing units have sixteen kilobytes of shared memory available per block. To comply with limitations of graphics hardware, edges are encoded in bit string.

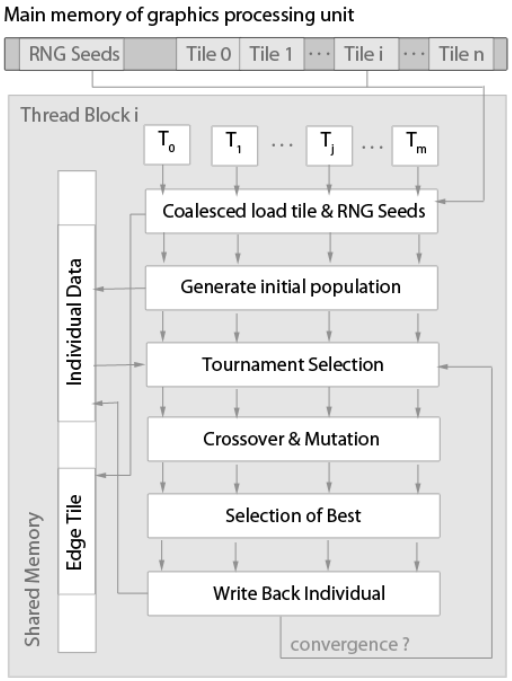


Figure 1. Evolution scheme in compute unified device architecture.

Genetic algorithm needs a random number generator. We have chosen park-miller pseudo-random number generator, RNG, due to its simplicity. Each individual has to load the seed for park-miller RNG from global memory of the device that was previously set using Mersenne Twister RNG. As seen in figure [1], each thread block loads initialization data to shared memory in a coalesced manner. RNG seeds, on the other hand, are kept in the local memory of the threads.

After initialization, genetic algorithm is carried out in parallel. When to stop algorithm is a challenging question in general. An indication of convergence can be checked to terminate the process [Yao05]. More practical applications employ predefined number of iterations before termination [Won09]. This is a tradeoff between accuracy and computational complexity. We have chosen to terminate the genetic algorithm after fixed number of epochs determined by empirical results.

Last step is to merge the results of thread blocks to find potential primitives. Individuals write the best result they have obtained to global memory of graphics processing unit. Results are then copied back to CPU memory, where they are combined to find candidate primitives sequentially. Merging of results is required only if segmentation is overlapped on image. If not, thread blocks can threshold the population and output a fixed number of qualified individuals.

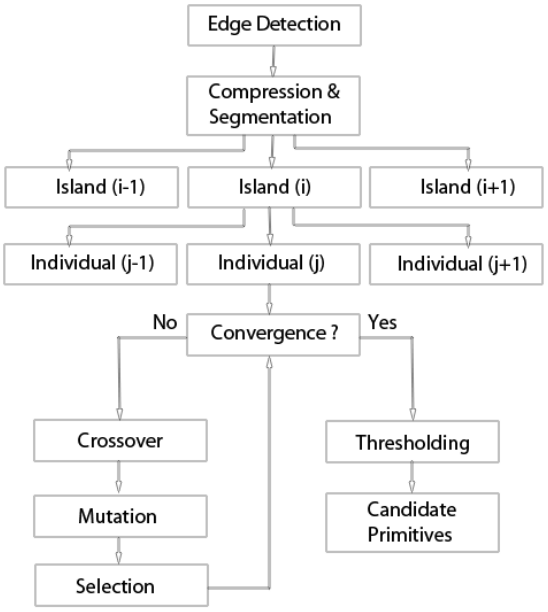


Figure 2. Architecture of primitive extractor on graphics processing unit.

Figure [2] shows phases of geometric primitive extraction on graphics processing unit. We will explain details of each step in the following sections of this document.

Edge detection

Edge detection is an important part of our process. Falsely detected edges lead to unreliable primitive extractor. There are various edge detectors available. Since we do not have enough memory for contour images, our expectation from an edge detector is to produce minimal response to a true edge.

Canny's edge detection algorithm is commonly used in the domain of computer vision. Its aim is to produce single response for an edge while maintaining the purpose of detecting all edges in the image. In this study, Canny's algorithm is selected for GPU implementation.

Yuancheng et al. have already implemented Canny's algorithm on CUDA [Yua08]. They have released source code of their implementation. However, their implementation is partial since it makes fixed number of iterations to find the connected components at the last stage of Canny's algorithm. They have tested their edge detector implementation on Lena, and reported limited improvement on edges for more than four iterations to find connected components. As it can be seen from figure [3], one needs more adaptive implementation to detect all edges.

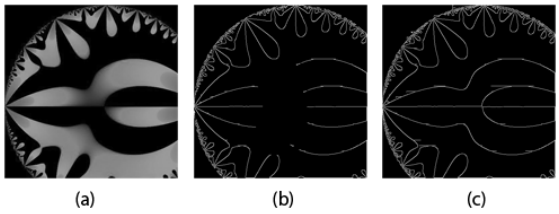


Figure 3. Original 512x512 image (a), edge image after 4 iterations (b), after 90 iterations

In our implementation of Canny's algorithm, we have used parallel breadth first search algorithm to detect connected edges, proposed by Pawan et al. [Paw07]. This adaptation has also accelerated Yuancheng's implementation, which is illustrated in figure [5]. Details of algorithm to find connected components are given in figure [4].

```
1 : For each definite edge do in parallel
2 :     Set frontier [ edge ] <- True , visited [ edge ] <- False
3 : End For
4 : For each edge do in parallel
5 :     If frontier [ edge ] is True
6 :         Set frontier [ edge ] <- False , visited [ edge ] <- True
7 :         For each neighbour,potential, and not visited edge
8 :             Set frontier [ not visited edge ] <- True
9 :         End For
10 :    End If
11 : End For
```

Figure 4. Algorithm to find connected edges on graphics processing unit.

Individual representation

There are different proposals for chromosome encoding of an individual for geometric primitive extraction. Yao et al. suggested encoding minimal number of points to uniquely define geometric primitive [Yao05], whereas Lutton et al. proposed storing parameters of the equation of geometric primitive [Lut94].

Both approaches have their own advantages and disadvantages. The former needs re-computation of parameters of primitive on each fitness evaluation, while keeping search focused on existing edge points and primitives. The latter, on the other hand, is computationally efficient on fitness evaluation while spending considerable time on non-existent primitives.

Since we have implemented ellipse detection on graphics processing unit, computation of ellipse parameters is negligible compared to time spent on fitness evaluation. Therefore, we have decided to encode a chromosome of an individual with five points. General ellipse equation has five coefficients to be determined. Note that, the latter approach should be considered on more complex shapes, where number of unknowns is large.

Initial population generation

One possible population generation is each thread composing its own individual based on edge points on block's shared memory. A thread can compose an individual by randomly sampling edge points from block's tile. One exceptional case is where there is not enough number of edge points on the tile. In that situation, block is terminated immediately since there is not enough evidence to hypothesize the existence of geometric primitive on the tile. Roth et al. has applied the same scheme of initial population generation, but noted that approach usually leads to ill-defined representation of entire geometric primitive [Rot94].

An alternative generation of initial population is total random point generation irrespective of edge existence on that point as suggested by [Pen99]. Although this approach is less computationally demanding, it does not consider existing edges to compose a candidate geometric primitive. However, number of points on chromosome is insignificant considering the points on the primitive it represents.

We have chosen second approach since first results in bank conflicts between threads. Thus, each thread generates a random chromosome consistent with tile dimensions.

Fitness Evaluation

Most of the proposed methods for fitness evaluation of ellipse simply matches template on the boundaries of it. Variations exist, however, in calculation.

Mainzer suggests punishment of displacement from boundary of an ellipse [Man02]. It is intuitive to distinguish edges near the primitive and the ones far from it. Value of the fitness function is given in eq.(1), where c is 0.7 and E(x,y) is 1 if there exists an edge pixel on point (x,y) 0 otherwise.

$$\sum_{x,y}\left(MAX(E(x+i,y+j)-(|i|+|j|)/c)_{\forall\, i,j}\right) \quad (1)$$

Yao et al. suggest two measures of fitness concurrently converging on the optima [Yao05]. They name these fitness measures as similarity and distance. Similarity determines how much the actual pixels match the perimeter of an ideal complete ellipse. Distance, on the other hand, is a measure of how far or close the actual pattern to the ideal ellipse is.

We have implemented fitness evaluation suggested by Mainzer [Man02]. If we had the capability of storing contour of tile on the shared memory of our GPU, this approach would be less time consuming since contour images can supply the distance from edge. In our implementation, we have checked neighborhood of boundary edge of ellipse. Since most time consuming task of genetic algorithm is fitness evaluation, contour image usage on larger shared memory can directly reduce bank conflicts, effectively increasing bandwidth of transaction between shared memory and co-processors.

Evolution

Selection and diversification dictate the main process of evolution. Selection eliminates individuals of low fitness value, promoting fitter individuals. However, it is important not to cluster all solutions around global optima while using elitism for selection. Since our genetic algorithm implementation divides the population into islands, side-effects of elitism have been eliminated.

Diversification is realized by crossover and mutation. Crossover mates two individuals to produce fitter offsprings. In CUDA, every thread selects her own mate to crossover randomly. We have implemented tournament selection on GPU as in Pospichal's study [Pos09]. However, their adaptation of tournament selection is deterministic since every thread mates with one next to it. In our implementation, we have used RNG to select a mate, which is more intuitive for stochastic algorithms but problematic in terms of bank conflicts.

Although other selection algorithms are possible, they are more computationally demanding on CUDA. For example, roulette wheel selection requires that every thread branch divergently in a loop, which immediately causes divergence of whole warp.

Mutation, in our implementation, simply changes a random bit of a chromosome with low probability of occurring. There are different mutation methods proposed for primitive extraction. Yao et al. suggested localized mutation operator that utilizes a trace tracking algorithm to find potential ellipse [Yao05]. Yin, on the other hand, proposed flipping a bit of a pixel such that it remains in the image [Pen99]. We have adapted Yin's implementation of mutation due to its lower processing requirements.

3. EXPERIMENTS & RESULTS

In this section, we inspect various aspects of our implementation and describe our results. Most of our experiments are focused on comparison of sequential and parallel implementations of ellipse detection.

A moderate improvement on canny edge detector has been achieved in our study. CUDA implementation of Yuancheng has been improved in terms of reliability and efficiency. Figure [5] shows efficiency gained by our adaptation of Pawan's parallel breadth first search algorithm for finding connected edges [Paw07]. Difference between two GPU implementations stems from work distribution among threads. In Yuancheng's implementation, each thread executes its own breadth first search,

while in our implementation every thread is responsible for a single edge.

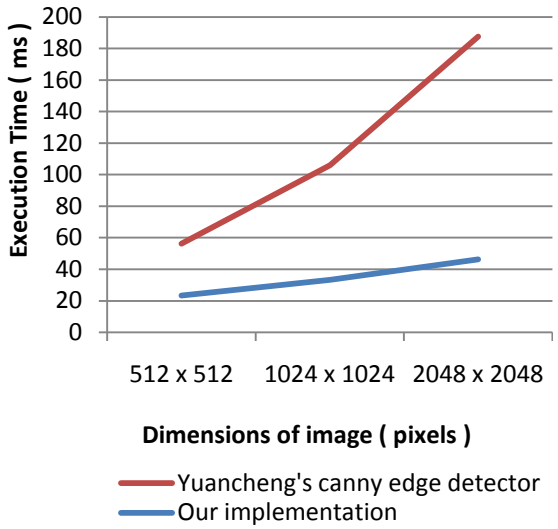


Figure 5. Comparison of our canny edge detector implementation with Yuancheng's implementation on CUDA.

Our ellipse detector is effective against high resolution imagery. Ellipse detection algorithm has been tested with over 600 images contained in our database. CPU implementation has been tested on Intel Core™ i7 at 2.67 GHZ, while GPU implementation has been tested on Nvidia GeForce GTX 260 graphics card. Figure [6] shows us that sequential computational resources could not exhibit scalability accomplished by our parallel implementation of ellipse detection. Note that GPU is not fully utilized for a 512 x 512 image since there are more processors than image segments.

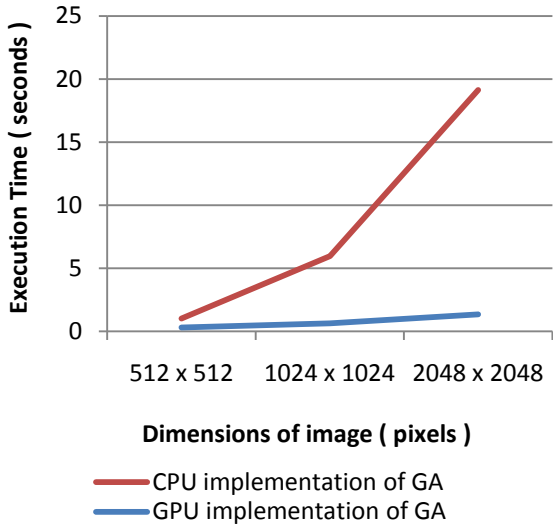


Figure 6. Comparison of sequential implementation of ellipse detection versus parallel implementation on CUDA.

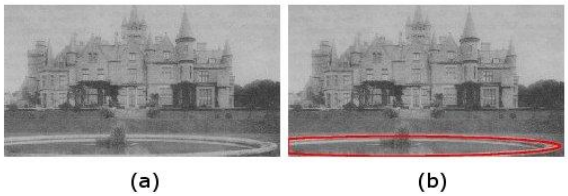


Figure 7. 1024 x 1024 image from our test database (a), image with highlighted ellipse (b).

To test our ellipse detector, we have constructed image database from both synthetic and real world images. Figure [7] shows a real-world image contained in our database along with result of the experiment. To test the accuracy, geometric properties of contained ellipses have been recorded. Table [1] shows statistical results obtained from ellipse detection experiments on GPU.

As input size is multiplied by four, average running time is doubled for small inputs. The reason is under utilization of GPU on low resolution images. Also note that, accuracy of our computation decrease as we increase number of ellipses. This is the result of clustering on global optima. Although we have adapted multiple island model of genetic algorithm, test cases where more than one solution falls into single island did not produce accurate results. To overcome this problem, variations of genetic algorithm, such as sharing genetic algorithm, can be implemented on CUDA.

Dimensions of Images (pixels)	Number of Images	Average Number of Ellipses	Average Running Time (s)	Accuracy (%)
512x512	124	1.3	0.311	82.4
1024x1024	241	2.4	0.627	81.9
2048x2048	248	5.2	1.340	77.3

Table 1. Statistical results obtained from experiments on our ellipse detector.

4. CONCLUSION

In this study, we have shown an implementation of geometric primitive extraction on graphics processing unit. Genetic algorithm has been fully utilized on GPU side, while CPU's computation time is saved. We have achieved up to 15x speed up relative to our sequential implementation of genetic algorithm on state of the art Intel CPU. Main

problem of our GPU implementation is low shared memory per multiprocessor. Usage of bit strings to represent image has produced bank conflicts during fitness evaluation, which is the most costly process of genetic algorithm. Advantage of our sequential implementation is the use of contour images to store edges. Such a data structure to store edge images is expected to accelerate fitness evaluation on GPU side. But current memory limitation has forced us to use bit strings. This problem might be alleviated by image compression techniques. Hardware solutions, on the other hand, are also possible for this kind of problem. As graphics processing hardware scales rapidly, more efficient ellipse detector can be implemented easily.

Experimental results confirmed the trend in parallelization of algorithms in the domain of computer vision. Scalability of current parallel architectures transforms many domains, including computer vision, into an era of parallel computation. Our study was aimed to contribute to this transformation.

5. ACKNOWLEDGMENTS

The author acknowledges the support of METU-TAF Modsimmer, and thanks Prof. İşler for his rigorous assistance to this paper.

6. REFERENCES

[Tia96] Tianzi Jiang; Song De Ma, Geometric primitive extraction using tabu search, Pattern Recognition, 1996., Proceedings of the 13th International Conference on , vol.2, no., pp.266-269 vol.2, 25-29 Aug 1996.

[Rob98] Robert A. McLaughlin, Randomized Hough Transform: Improved ellipse detection with comparison, Pattern Recognition Letters, Volume 19, Issues 3-4, Pages 299-305, ISSN 0167-8655, March 1998.

[Rot94] Roth, G.; Levine, M.D., "Geometric primitive extraction using a genetic algorithm," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol.16, no.9, pp.901-905, Sep 1994.

[Pen99] Peng-Yeng Yin, A new circle/ellipse detector using genetic algorithms, Pattern Recognition Letters, Volume 20, Pages 731-740, Issue 7, July 1999.

[Kul90] Kultanen, P.; Xu, L.; Oja, E., "Randomized Hough transform (RHT)," Pattern Recognition, 1990. Proceedings., 10th International Conference on , vol.i, no., pp.631-635 vol.1, 16-21 Jun 1990.

[Sic05] Si-Cheng Zhang, Zhi-Qiang Liu, A robust, real-time ellipse detector, Pattern Recognition, Volume 38, Issue 2, Pages 273-287, February 2005.

[Rot93] G. Roth and M. D. Levine, "Extracting geometric primitives," Comput. Vision. Graphics Image Processing: Image Understanding, vol. 58, pp. 1-22, 1993.

[Yao05] Yao J., Kharma N., Grogono P.,A multi-population genetic algorithm for robust and fast ellipse detection , Pattern Analysis & Applications, vol. 8 pp. 149-162,2005.

[Kaw98] Kawaguchi, T.; Nagata, R.-I., "Ellipse detection using a genetic algorithm," Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on, vol.1, no., pp.141-145 vol.1, 16-20 Aug 1998.

[Lut94] Lutton E, Martinez P, A genetic algorithm for the detection of 2D geometric primitives in images. In: Proceedings of the 12th international conference on pattern recognition, Jerusalem, Israel, 9–13 October 1994.

[Yua08] Yuancheng Luo; Duraiswami, R., Canny edge detection on NVIDIA CUDA, Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on, vol., no., pp.1-8, 23-28 June 2008.

[Paw07] Pawan Harish and P. Narayanan. Accelerating large graph algorithms on the gpu using cuda. pages 197–208. 2007.

[Man02] Mainzer T., Genetic algorithm for shape detection, Technical report no. DCSE/TR-2002 06, University of West Bohemia, 2002.

[Pos09] Pospichal P. , and Jaros J. , GPU-based Acceleration of the Genetic Algorithm, from contest GPUs for Genetic and Evolutionary Computation, 2009.

[Won09] Wong M. L. , Parallel multi-objective evolutionary algorithms on graphics processing units, Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference, pages 2515-2522, 2009.