The 17th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision

in co-operation with

EUROGRAPHICS

W S C G ' 2009

Communication Papers Proceedings

University of West Bohemia
Plzen
Czech Republic

February 2 - 5, 2009

Co-Chairs
Min Chen, Swansea University, United Kingdom
Vaclav Skala, University of West Bohemia, Czech Republic

Edited by
Min Chen
Vaclav Skala

WSCG'2009 Communication Papers Proceedings

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

Managing Editor: Vaclav Skala

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

Printed: University of West Bohemia, Plzeň, Czech Republic

Hardcopy: *ISBN 978-80-86943-94-7*

WSCG 2009

International Programme Committee

Adzhiev,V. (United Kingdom)
Baranoski,G. (Canada)
Bekaert,P. (Belgium)
Benes,B. (United States)
Bilbao,J. (Spain)
Biri,V. (France)
Bittner,J. (Czech Republic)
Bouatouch,K. (France)
Brodlie,K. (United Kingdom)
Buehler,K. (Austria)
Chen,M. (United Kingdom)
Chover,M. (Spain)
Coquillart,S. (France)
Crosnier,A. (France)
Csebfalvi,B. (Hungary)
Cunningham,S. (United States)
Daniel,M. (France)
de Geus,K. (Brazil)
Debelov,V. (Russia)
Feito,F. (Spain)
Ferguson,S. (United Kingdom)
Flaquer,J. (Spain)
Flusser,J. (Czech Republic)
Gallo,G. (Italy)
Groeller,E. (Austria)
Gudukbay,U. (Turkey)
Gutierrez,D. (Spain)
Havran,V. (Czech Republic)
Jansen,F. (Netherlands)
Kavan,L. (Ireland)
Klein,R. (Germany)
Klosowski,J. (United States)
Lee,T. (Taiwan)
Magnor,M. (Germany)
Mollá Vayá,R. (Spain)
Muller,H. (Germany)
Murtagh,F. (Ireland)
Myszkowski,K. (Germany)
Pasko,A. (United Kingdom)
Pedrini,H. (Brazil)
Peroche,B. (France)
Platis,N. (Greece)
Puppo,E. (Italy)
Purgathofer,W. (Austria)
Rojas-Sola,J. (Spain)
Rokita,P. (Poland)
Rosenhahn,B. (Germany)
Rossignac,J. (United States)
Rudomin,I. (Mexico)
Sakas,G. (Germany)
Schumann,H. (Germany)
Segura,R. (Spain)
Selim,B. (Turkey)
Skala,V. (Czech Republic)
Slavik,P. (Czech Republic)
Slusallek,P. (Germany)
Sramek,M. (Austria)
Stroud,I. (Switzerland)
Teschner,M. (Germany)
Theoharis,T. (Greece)
Triantafyllidis,G. (Greece)
Vergeest,J. (Netherlands)
Wu,E. (China)
Wuethrich,C. (Germany)
Zara,J. (Czech Republic)
Zemcik,P. (Czech Republic)

WSCG 2009
Board of Reviewers

Abas,M. (Malaysia)
Adzhiev,V. (United Kingdom)
Akleman,E. (United States)
Andreadis,I. (Greece)
Aspragathos,N. (Greece)
Aveneau,L. (France)
Baranoski,G. (Canada)
Battiato,S. (Italy)
Bekaert,P. (Belgium)
Bellon,O. (Brazil)
Benes,B. (United States)
Bilbao,J. (Spain)
Biri,V. (France)
Bittner,J. (Czech Republic)
Borchani,M. (France)
Bouatouch,K. (France)
Bouville,C. (France)
Brodlie,K. (United Kingdom)
Bruckner,S. (Austria)
Bruni,V. (Italy)
Brunnet,G. (Germany)
Buehler,K. (Austria)
CarmenJuan-Lizandra,M. (Spain)
Chaudhuri,D. (India)
Chen,M. (United Kingdom)
Chover,M. (Spain)
Chum,O. (Czech Republic)
Coquillart,S. (France)
Crosnier,A. (France)
Csebfalvi,B. (Hungary)
Cunningham,S. (United States)
Daniel,M. (France)
de Geus,K. (Brazil)
Debelov,V. (Russia)
Dingliana,J. (Ireland)
Drbohlav,O. (Czech Republic)
Duce,D. (United Kingdom)
Durikovic,R. (Slovakia)
Egges,A. (Netherlands)
Eisemann,M. (Germany)
Erbacher,R. (United States)
Feito,F. (Spain)
Ferguson,S. (United Kingdom)
Ferko,A. (Slovakia)
Fernandes,A. (Portugal)
Flaquer,J. (Spain)
Flusser,J. (Czech Republic)
Foufou,S. (France)
Gallo,G. (Italy)
Galo,M. (Brazil)
Ganovelli,F. (Italy)
Garcia-Alonso,A. (Spain)
Giannini,F. (Italy)
Gonzalez,P. (Spain)
Grammalidis,N. (Greece)
Groeller,E. (Austria)
Gudukbay,U. (Turkey)
Gumbau,J. (Spain)
Gupta,A. (United States)
Gutierrez,D. (Spain)
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)
Iwanowski,M. (Poland)
Janda,M. (Czech Republic)
Jansen,F. (Netherlands)
Joan-Arinyo,R. (Spain)
Jones,M. (United Kingdom)
Karabassi,E. (Greece)
Kavan,L. (Ireland)
Kimmel,B. (Canada)
Klein,R. (Germany)
Klosowski,J. (United States)

Knight,M. (United Kingdom)
Kohout,J. (Czech Republic)
Kolcun,A. (Czech Republic)
Krishnaswamy,A. (United States)
Lanquetin,S. (France)
Lee,T. (Taiwan)
Lewis,J. (New Zealand)
Lin,W. (Taiwan)
Liu,D. (Taiwan)
Maciel,A. (Brazil)
Magnor,M. (Germany)
Maierhofer,S. (Austria)
Mandl,T. (Germany)
Matey,L. (Spain)
Matkovic,K. (Austria)
Mattausch,O. (Austria)
Michoud,B. (France)
Mokhtari,M. (Canada)
Mollá Vayá,R. (Spain)
Montrucchio,B. (Italy)
Mudur,S. (Canada)
Muller,H. (Germany)
Murtagh,F. (Ireland)
Myszkowski,K. (Germany)
OliveiraJunior,P. (Brazil)
Papaioannou,G. (Greece)
Pasko,A. (United Kingdom)
Pasko,G. (Cyprus)
Paulin,M. (France)
Pedrini,H. (Brazil)
Peroche,B. (France)
Pettifer,S. (United Kingdom)
Platis,N. (Greece)
Přikryl,J. (Czech Republic)
Puig,A. (Spain)
Puppo,E. (Italy)
Purgathofer,W. (Austria)
Renaud,C. (France)
Ripolles,O. (Spain)
Ritschel,T. (Germany)
Rodeiro,J. (Spain)
Rojas-Sola,J. (Spain)
Rokita,P. (Poland)
Rosenhahn,B. (Germany)
Rossignac,J. (United States)
Rudomin,I. (Mexico)
Sakas,G. (Germany)
Sanna,A. (Italy)
Schumann,H. (Germany)
Segura,R. (Spain)
Selim,B. (Turkey)
Sellent,A. (Germany)
Sirakov,N. (United States)
Skala,V. (Czech Republic)
Slavik,P. (Czech Republic)
Slusallek,P. (Germany)
Solis,A. (Mexico)
Sousa,A. (Portugal)
Sramek,M. (Austria)
Stroud,I. (Switzerland)
SuarezRivero,J. (Spain)
Svoboda,T. (Czech Republic)
Teschner,M. (Germany)
Theoharis,T. (Greece)
Theußl,T. (Austria)
Torrens,F. (Spain)
Triantafyllidis,G. (Greece)
Tytkowski,K. (Poland)
Vanecek,P. (Czech Republic)
Vasa,L. (Czech Republic)
Veiga,L. (Portugal)
Vergeest,J. (Netherlands)
Viola,I. (Norway)
Vitulano,D. (Italy)
Wan,T. (United Kingdom)
Wu,E. (China)
Wuethrich,C. (Germany)
Yencharis,L. (United States)
You,S. (United States)
Zach,C. (United States)
Zachmann,G. (Germany)
Zalik,B. (Slovenia)
Zara,J. (Czech Republic)
Zemcik,P. (Czech Republic)
Zhu,Y. (United States)

Contents

Communication Papers

Knuth,M.,Kohlhammer,J.: A Hybrid Ambient Occlusion Technique for Dynamic Scenes 1

Vergeest,J.S.M., Song,Y.: Fitting freeform multi-parameter shapes to 3D data points - A case study 9

Yasmin,S., Talib,A.Z.: Shape Transformation of Multiple Objects Using Slices 17

Cabral,I.P.S., Gonçalves,L.M.G., Xavier,J.C.Jr.: WEB GIS by Ajax for Analysis and Control of Environmental Data 25

Lee,S.-H., Han,S.-I., Kang,M.G.: Object Detection in Low Illumination Environment 33

Puig-Centelles,A., Varley,P.A.C., Ripolles,O., Chover,M.: Automatic Terrain Generation with a Sketching Interface 39

Löffler,F., Rybacki,S., Schumann,H.: Error-bounded GPU-based terrain visualisation 47

Kim,K.R., Lee,B.G., Yoo,H.: Improvement of the DOI Technique for De-Interlacing 55

Zemčík,P., Maršík,L., Herout,A.: Point Cloud Rendering in FPGA 63

Mlích,J., Zemčík,P., Jiřík,L.: Trajectory classification using HMMs 67

Haindl,M., Hatka,M.: Generalized Roller 73

Yang,X., Xu,D.-Q., Zhao,L.: A fast SAH-based construction of Octree 81

Lefer,W.: Vision Aware Continuum Crowds 89

Beran,V., Herout,A., Řezníček,I.: Video-Based Bicycle Detection in Underground Scenarios 95

Garnier,L., Belbis,B., Foufou,S.: Conversion of biquadratic rational Bézier surfaces into patches of particular Dupin cyclides: the torus and the double sphere 101

Schoor,W., Seidl,T., Bollenbeck,F., Seiffert,U., Preim,B., Mecke,R.: Efficient Semiautomatic Segmentation of Plant Biological Objects Based on Live-wire Methods 109

Benger,W., Ritter,M., Acharya,S., Roy,S., Jijao,F.: Fiberbundle-based visualization of a Stir Tank Fluid 117

Sisojevs,A., Krechetova,K., Glazs,A.: 3D Modeling of Free-Form Objects 125

A Hybrid Ambient Occlusion Technique for Dynamic Scenes

Martin Knuth
Fraunhofer IGD
Fraunhoferstr. 5
64283 Darmstadt, Germany
martin.knuth@
igd.fraunhofer.de

Jörn Kohlhammer
TU Darmstadt
Fraunhoferstr. 5
64283 Darmstadt, Germany
Joern.Kohlhammer@
gris.informatik.tu-darmstadt.de

ABSTRACT

In this paper we present a hybrid technique for illuminating a dynamic scene with self shadowing. Our goal is to perform the necessary calculations with real-time or interactive frame rates. The main idea is to split up the self shadowing process into a global and a local part. This separation allows us to choose combinations of completely independent algorithmic approaches. The global part calculates the self shadowing information of the entire scene. Since it has to process the whole scene, it has a high computational cost; however, a coarse approximation can be created in a short time. In contrast the process for the local part deals with fine details, which have local impact on the scene. It only processes a relevant subset of the scene. Finally we present results based on an implementation using a GPU based self shadowing approximation combined with screen space ambient occlusion for solving the local part

Keywords Shadow Algorithms, GPU Programming, Ambient Occlusion

1. INTRODUCTION

Shadows are a very important part for the realism of a 3D scene. They give information about spatial object relations, allowing better immersion. Our goal is to illuminate a fully dynamic 3D scene with a high dynamic range (HDR) light probe surrounding this scene. Since the light probe describes colour and intensity for given directions, it is necessary to know the directional occlusion of the light for each position of the scene. Additionally the lighting condition described by a light probe creates a wide range of shadow frequencies. This is especially a problem for dynamic and deformable scenes, where no information about self occlusion can be precomputed. To cope with the complexity of the shadowing problem in large, dynamic scenes, we propose a new approach. The core of the problem is based on the fact, that all geometry and all lighting parameters may affect the lighting conditions in every single point in the scene. All approaches are approximations, working on specific samples of geometry, spatial areas or light sources (or combinations thereof), always resulting in a trade-off between computational speed and accuracy/realism. The strategy for our approach is to use two shadowing algorithms for two different computational domains: A global domain for the dependencies between the lighting conditions of the complete scene and a local domain for the small-scale dependencies especially focussing on the visible part of the scene. In the global domain the light exchange is approximated on a scale coarse enough to allow real-time rendering. The computational effort for the details of the scene is shifted to the algorithm operating on the local domain. The algorithms work and can be chosen independently from each other. The result of both algorithms is a ”shadow-mask” of the scene, defining the effect for its specific domain. We combine the two results, in the final step.

In our implementation section we present the implementation of a particular combination of algorithms. This combination shows self shadowing evaluation per pixel and the ability to trade speed against accuracy in

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

the global part. In the local part we decided to use a screen space ambient occlusion (SSAO) technique, which has the advantage of being independent from the complexity of the scene.

The presented implementation processes the all frequency self shadowing of the scene at interactive or real time frame rates on current GPUs. It is able to process non-manifold deformable objects within fully dynamic scenes without prior knowledge of the animation.

In section 2 we refer to work related to our approach. Additionally we refer to two previus works, that we use for the global and the local computation. Then, we explain our technique and considerations behind it. In section 5 we present the particular case we have chosen and implemented. In the next section we present a summary and results of this approach, followed by a future work section.

2. RELATED WORK

We will now present a collection of algorithms, which aim at similar goals. These algorithms can be classified as follows. We will start with algorithms, which deal with the shadowing problem of single light sources. A survey of real time shadowing methods is given in [HLHS03]. Some of the presented algorithms work with small area light sources, but can be computational costly if these lights become large. Fast shadow calculations for simple point or directional lights within dynamic scenes can be done inside modern GPU's. Mainly two different approaches are used: Shadow volumes [Cro77] and shadow maps [Wil78]. Shadow maps are fast to compute and need less fill rate than shadow volumes. On the other hand, care has to be taken of sampling artefacts. An approach for minimizing shadow buffer artefacts and softening of the borders of shadows can be found in [RSC87, DL06].

We will now refer to algorithms, which provide or store directional lighting information. If shadows are not necessary, the scene can be directly illuminated by an environment map. For this task a lot of fast environment mapping methods (like [Gre86, RH01, HS99])exist. The advantage of using environment map based approaches is based on their capability to consider all lights at once, which makes them independent from the number lights used. High dynamic range (HDR) environment maps [Deb98] represent the full variablility of natural lighting conditions, which can be used to bring together synthetic and real scenes. An environment map used for illuminating a scene is referred as lightprobe in this literature. Since this technique does not support directional shadowing directly it can be necessary to compute a carefully chosen set of light sources to approximate the lighting effect of a light probe. As an alternative methods can be used, that handle directional information directly. The represented occlusion or lighting data is then used to process the environment map. These methods store directional data, using spherical harmonics, or similar methods. The type of data used often require precomputations, so they cannot be used for animated geometry. This problem is addressed by Kautz et al. in [KLA04]. By using model hierarchies they accelerated the calculations to interactive frame rates. Limitations are small model sizes and low frequency shadows. In addition model hierarchies are costly to compute. A method with similar results is presented in [SSZK04]. It evaluates the visibility of a given set of directional light sources per vertex, taking into account the colour of the environment map, by sampling the map in an uniform way. Visibility calculation is done inside the GPU, and is read back to the CPU which is a bottleneck.

At last we will refer to ray casting and ray tracing methods, which too are well suited to answer occlusion questions. [WPS+03, WBS03, PBMH02]. Nowadays, real-time frame rates can be achieved by using a PC-cluster, the GPU or dedicated ray tracing hardware, which is still in development. Software solutions require the power of an efficient cluster system and at last GPU ray tracing needs to solve the bottleneck problem of reading back towards the CPU.

Utilising a real time ray tracer, global illumination can be simulated. In [CHL04] a direct approach towards global illumination on the GPU is presented, which works for small scenes. Another approach aimed for fast global illumination computations was presented by Keller et. al in [Kel97]. In this method single shadow casting lights are used to illuminate the scene.

Previous Work

We will now describe two techniques which we used as building blocks for the local and the global part in our implementation.

In [KF05] a fast GPU based method for evaluating the directional occlusion of a 3D scene is described. The method heavily relies on the capability of modern GPU to quickly perform shadow calculations. The HDR light probe is approximated by a set of light sources, which are used in conjunction with shadow mapping. The advantage of this approach lies in the possibility to tune its performance with the parameters "'light sources"' and "'shadow map resolution"'. Since the map resolution is directly related with the details preserved within the casted shadows this is a complement to the SSAO algorithm. However one drawback of the presented method is an effect called "'light bleeding"' which occurs, when using shadow

maps with low resolutions. To prevent aliasing artefacts the masking effect of the shadow buffer is applied a short way behind the occluder to not accidently shadow the occluder's front.

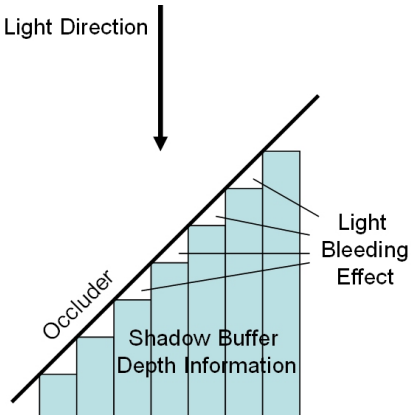


Figure 1: To prevent shadow texel aliasing effects the shadow effect is moved a bit behind the occluder. This results in a "'light bleeding"' effect for thin occluders, since the light travels a bit through the occluder, until it is masked out by the shadow buffer.

To deal with the local occlusion we use a SSAO technique. The key idea of (SSAO) is to interpret the depth information as a height field. For this height field the self occlusion is computed. The approach is decoupled from the scene geometry, since it is completely screen based. In [SA07] the authors present a technique for evaluating the ambient occlusion (AO) of a scene. They divide the ambient occlusion problem into a near AO part and a far AO part (these stand for the high and low frequency details of the scene). For the low frequency part they use an arrangement of spheres which allows a quick approximation of the ambient occlusion in a very coarse way. The authors evaluate the high frequency part by a SSAO technique. This algorithm uses the depth information of the current frame. However the depth information of the current scene is only a very lossy representation of the information defined by the 3D scene. The localness of the method is given by the range in which occluders are searched around the currently processed object.

In difference to this technique our presented combination is able to use a HDR lightprobe to model the incoming light. Ambient Occlusion assumes a steady over the area of the lightprobe. In combination with lightprobes representing an unbalanced intensity distribution this can lead to a wrong illumination.(figure 2)

At last our implementation makes massive use of deferred shading in order to separate the computations from the scene complexity. The idea of deferred shading is originally presented in [DWS+88]. The key idea behind deferred shading is to store rendering information in images once per frame. These images resample information of the rendered objects, like depth, normal, colour and other scene data. This allows to perform processes like "'lighting the scene"' without the need to touch the original geometry. Typically these images are rendered into a so called multiple render target (MRT), which allows the rendering of several images at once. Algorithms, which only need to access this data, run independently of the scene complexity.

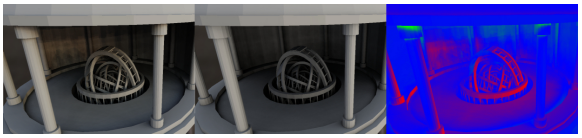


Figure 2: Difference between the occlusion of single light sources and ambient occlusion. Left: shadowing of the single light sources. Middle: Ambient occlusion. Left: difference image. The beach-cross lightprobe[Deb98] was approximated by 2048 light sources to create a high quality rendering

3. HYBRID OCCLUSION

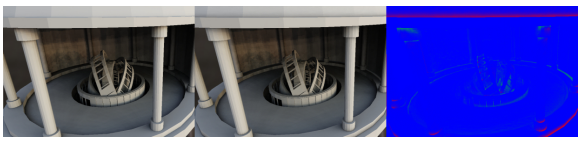


Figure 3: Difference between the occlusion of single light sources and an ideal hybrid directional/ambient occlusion combination. Left: shadowing of the single light sources. Middle: Result of an ideal hybrid rendering. Left: difference image. Shadows of the single light sources are computed on a coarse level to reduce computational costs. Details are processed by a local Ambient Occlusion technique. In Difference to 2 the difference image shows fewer artefacts, since Ambient Occlusion now only affects small regions.

Our key idea is to separate the complex self shadowing problem into two parts. The first step is the global shadow calculation on a level coarse enough to allow interactive frame-rates. We use a shadow map based approximation for the directional occlusion, but our hybrid approach is not restricted to this specific class of algorithms.

The benefit of this approach is mainly given by the characteristics of the local part. In the local part occluders are only tested near the current object of interest. This reveals high frequency details only and has the low computational costs of a local process. The

main strategy is the reduction of overall computational costs by reducing the costs in the global part far more then the costs appended by the local part.

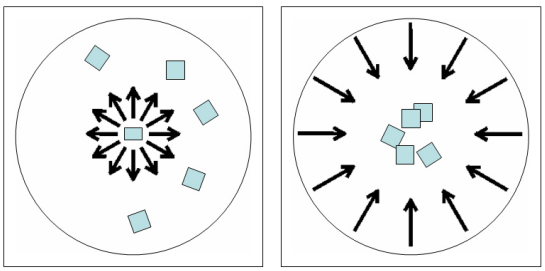


Figure 4: Differences between Local(right) and Global(left) Part. In the local part an object tries to figure out its occlusion. In the global part light from outside the scene is occluded by the scene.

The Global Part

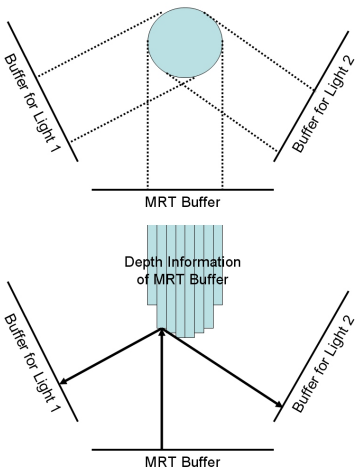


Figure 5: Principle functionality of our global part workflow. Upper image: We first render the whole scene in the scene/shadow buffers. Lower image: in the lighting stage only the data inside the buffers is needed to perform the shadow computations. The world position is calculated from depth and camera information. This position is then transformed into shadow map world space for the lookup.

Our intention within the global part is to calculate an approximation of the directional occlusion effects of the entire scene. The process starts by converting the light probe into a set of directional light sources, for which shadow maps are computed to approximate the directional occlusion. The conversion process has to be done once; since the light probe has a static property. However, the shadowing process itself is dynamic and can be performed completely inside the GPU. Additionally we intent to overcome the appearance of single shadow borders by by adding noise to the sample position of the shadow map. Unfortunately this increases the amount of light bleeding, since the higher threshold has to be chosen in order to not shadow the occluder accidently. This is due to virtually increasing the area of a texel in the shadow map, which introduces the same artefact like decreasing the shadow map resolution. The noising allows us to reduce the shadow map resolution, reducing the fill rate used to generate the shadow maps, saving computational time. Unfortunately the whole shadowing process will tend towards low frequency shadows, since the high frequency parts are blurred away, too. These two issues, light bleeding and missing details are then addressed in the local part.

The Local Part

The idea of the local part is to add missing details to the scene shadowing, and to correct artefacts, introduced by the global parts approximation. In this part we want to retrieve missing high frequency shadows and additionally we reduce the light bleeding artefacts. The basic approach is to apply an algorithm to the output image of the global part, which darkens concave areas of the scene with a local operation. Since this wanted property is similar to ambient occlusion it fulfils both purposes.

Combination

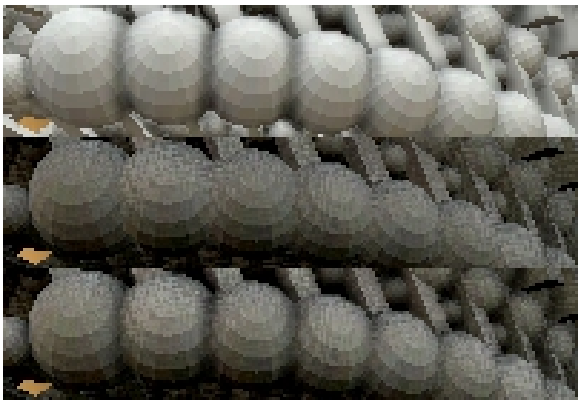


Figure 6: The effects of the details introduced by the local part: Top: local output. Middel: global part output. Down: the final combination.

In a final stage we have to combine the results of the two used methods. To simplify this we constructed the whole process around a deferred shading rendering system. The local part outputs an attenuation value, with which the result from the global pass can be modulated. Modulation is a sufficient operation in this case, since directional information provided by the global part can not be applied to the local part

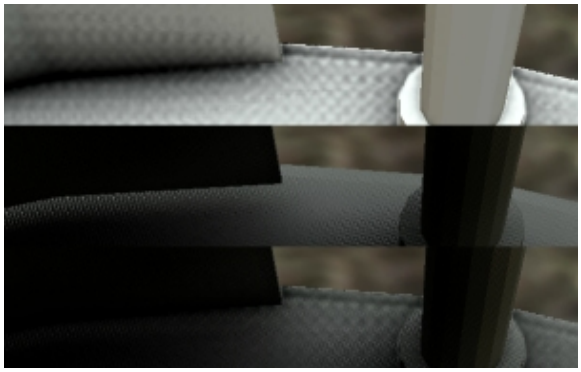


Figure 7: Light bleeding reduction introduced by the local part: Top: local output. Middel: global part output. Down: the final combination.

algorithm. So both outputs are combined in a final stage and not on a per lightsource level. Otherwise the output factors of both algorithms would have to be merged regarding their direction. By using an ambient occlusion technique in the local part we can work around this problem.

4. IMPLEMENTATION

We implemented our technique on a NVIDIA 8k Series graphics hardware. It is a recent Graphics processor, which is capable to use the same processing stages for vertex processing as well as fragment processing. We use this capability to perform most of the work on a per fragment level.

Global Part

Our experimental scene setup is illuminated by a large spherical area light source surrounding the scene - the light probe. We get our light setup by sampling the light probe uniformly.

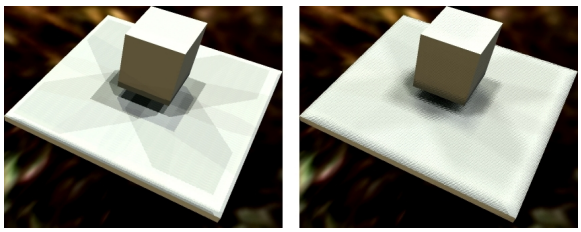


Figure 8: To smooth the transition between single shadows we introduce noise to the current sample position of the shadow map. Left image: shadow map sampling using only the map position, right image: shadow map sampling with noise, resulting in a smoother appearance.

These light sources are used in conjunction with shadow buffers to create the self shadowing effect. So we can control 2 parameters. The first is the number of light sources used; the second is the resolution of the shadow buffers used. To enhance the image quality of the borders between shadowed and non shadowed areas, we introduce noise to the shadow map sample position. (seen in figure 8.) The noise data is stored in a small tiled texture, which is repeated over the screen. This method introduces only low extra cost in difference to other smoothing or soft shadow methods. Additionally the shadow maps are applied in a screen space based method, which makes this part of the algorithm independent of the scene complexity. The cost of the shadowing process is only dependent of the number of shadows applied.

Local Part

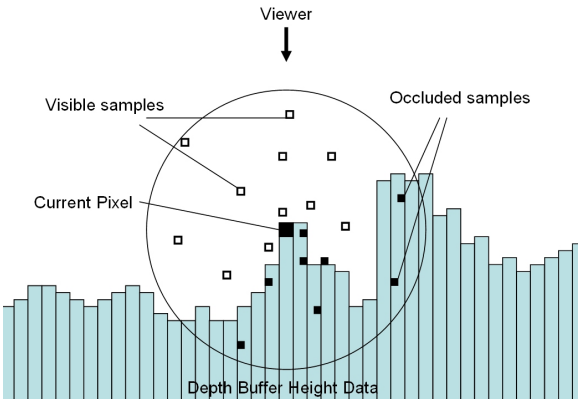


Figure 9: Self occlusion of the depth image, showing one line interpreted as height field: we test points inside a spherical region. The test results are related to the current sample point in the centre to get an approximation of its occlusion by its surroundings.

For the local self shadowing effects we have chosen a screen spaced ambient occlusion technique similar to the one presented in [SA07]. It only processes data of the depth component of the MRT buffer. In difference to the SSAO technique presented in [SA07] we used some simplifications. Our version samples the occlusion of the spherical area around the currently analysed pixel. We do not take into account the current normal. The distance attenuation of the importance of an occluder is automatically given by using a denser sampling towards the centre of the sphere. The amount of occlusion is simply approximated by averaging the test results of the sampled positions. Due to this sampling method, objects occluding less then the half of the sphere would appear brighter then usual. Since this is the case for all convex objects, we clamp the attenuation factor generated by the algorithm to the range between [0..1].

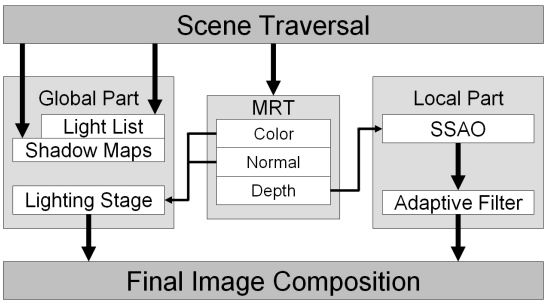


Figure 10: Overview of the used Rendering Architecture.

Architecture

The rendering system was implemented via a deferred shading approach. This allows a modular design and reduces the number of rendering passes for algorithms working on the current frame of the scene. An overview can be seen in figure 10. We start with a standard scene traversal and rendering of the scene. The result of the traversal is a light list, which is processed later in the pipeline. The scene is rendered to a multiple render target buffer wich holds the necessary information for the later deferred shading processes. In our case this buffer contains the colour component, the normal component and the depth component of the scene. We use the camera parameters to reconstruct the word coordinates of pixel data from the depth buffer information. The global pass is performed with the light list, an associated shadow map list and the colour + normal information from the MRT buffer. The local pass consists of the SSAO algorithm which works on the depth component of the MRT buffer. To reduce the number of texture reads inside the algorithm we make use of dithering. After the SSAO stage an adaptive filtering stage is applied to get a smooth appearance of the local pass output image. For the final combination the image from the lighting stage is attuned by the image from the local pass.

5. RESULTS AND DISCUSSION

We presented a hybrid technique for calculating the self shadowing of a fully dynamic scene with interactive to real-time frame rates. The technique splits the process into a local and a global method in order to combine the advantages of both. These methods can be chosen independently and are not limited to the ones we used. For the implementation of these two parts we presented a combination of a shadow buffer based self shadowing method and a screen spaces ambient occlusion algorithm. We have shown the capability of the local part to reduce light bleeding effects introduced by the method used in the global part. Additionally it adds shadowing details, which are not covered by the global part. Both parts are evaluated on the GPU using a deferred shading approach which is independent of the scene complexity. In our implementation we reach around 6 FPS when using an light probe approximation of 512 light sources and around 15 FPS when using an approximation of 128 lights.

6. FUTURE WORK

A benefit of the presented implementation is the possibility to use standard lighting models inside the global part. The presented screenshots use lambertian reflectance, but other reflection models can be used, too. A major problem of combining a directional occlusion and an ambient occlusion technique is over occlusion. Since both methods do not communicate with each other, light can be occluded several times. A future local approach should take into account the directional character of the incoming light. It would allow combining the directional occlusion information of both parts correctly. Additionally the lighting information of the light probe would also be taken into account in the local part. Another topic is the use of global illumination techniques inside the global and local part. The presented technique should be flexible enough to allow such modifications. However, an interesting aspect will be the combination of the local and the global part. With increasing complexity of the used algorithms it will be more and more necessary to exchange information between local and global part to get a seamless merge of the local and the global domain.

7. ACKNOWLEDGEMENTS

The high dynamic range environment maps used in this paper were made by Paul Debevec [Deb98].

8. REFERENCES

[CHL04] Greg Coombe, Mark J. Harris, and Anselmo Lastra. Radiosity on graphics hardware. In *GI '04: Proceedings of the 2004 conference on Graphics interface*, pages 161–168. Canadian Human-Computer Communications Society, 2004.

[Cro77] F.C. Crow. Shadow algorithms for computer graphics. *j-COMPGRAPHICS*, 11(2):242–248, July 1977.

[Deb98] Paul Debevec. Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH '98: Proceedings of the 25th annual conference*

on Computer graphics and interactive techniques, pages 189–198. ACM Press, 1998.

[DL06] William Donnelly and Andrew Lauritzen. Variance shadow maps. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 161–165, New York, NY, USA, 2006. ACM.

[DWS+88] Michael Deering, Stephanie Winner, Bic Schediwy, Chris Duffy, and Neil Hunt. The triangle processor and normal vector shader: a vlsi system for high performance graphics. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 21–30, New York, NY, USA, 1988. ACM.

[Gre86] Ned Greene. Environment mapping and other applications of world projections. *IEEE Comput. Graph. Appl.*, 6(11):21–29, 1986.

[HLHS03] Jean-Marc Hasenfratz, Marc Lapierre, Nicolas Holzschuch, and François Sillion. A survey of real-time soft shadows algorithms. In *Eurographics*. Eurographics, Eurographics, 2003. State-of-the-Art Report.

[HS99] Wolfgang Heidrich and Hans-Peter Seidel. Realistic, hardware-accelerated shading and lighting. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 171–178. ACM Press/Addison-Wesley Publishing Co., 1999.

[Kel97] Alexander Keller. Instant radiosity. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 49–56. ACM Press/Addison-Wesley Publishing Co., 1997.

[KF05] Martin Knuth and Arnulph Fuhrmann. Self-shadowing of dynamic scenes with environment maps using the gpu. In *WSCG'2005 Full Papers Conference Proceedings*, pages 31–38, 2005.

[KLA04] J. Kautz, J. Lehtinen, and T. Aila. Hemispherical rasterization for self-shadowing of dynamic objects. In *Proceedings Eurographics Symposium on Rendering 2004*, 2004.

[PBMH02] Timothy J. Purcell, Ian Buck, William R. Mark, and Pat Hanrahan. Ray tracing on programmable graphics hardware. *ACM Trans. Graph.*, 21(3):703–712, 2002.

[RH01] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 497–500. ACM Press, 2001.

[RSC87] William T. Reeves, David H. Salesin, and Robert L. Cook. Rendering antialiased shadows with depth maps. *SIGGRAPH Comput. Graph.*, 21(4):283–291, 1987.

[SA07] Perumaal Shanmugam and Okan Arikan. Hardware accelerated ambient occlusion techniques on gpus. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 73–80, New York, NY, USA, 2007. ACM.

[SSZK04] M. Sattler, R. Sarlette, G. Zachmann, and R. Klein. Hardware-accelerated ambient occlusion computation. In B. Girod, M. Magnor, and H.-P. Seidel, editors, *Vision, Modeling, and Visualization 2004*, pages 331–338. Akademische Verlagsgesellschaft Aka GmbH, Berlin, November 2004.

[WBS03] Ingo Wald, Carsten Benthin, and Philipp Slusallek. Interactive global illumination in complex and highly occluded environments. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 74–81. Eurographics Association, 2003.

[Wil78] Lance Williams. Casting curved shadows on curved surfaces. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 270–274. ACM Press, 1978.

[WPS+03] Ingo Wald, Timothy J. Purcell, Joerg Schmittler, Carsten Benthin, and Philipp Slusallek. Realtime ray tracing and its use for interactive global illumination. In *Eurographics State of the Art Reports*, 2003.

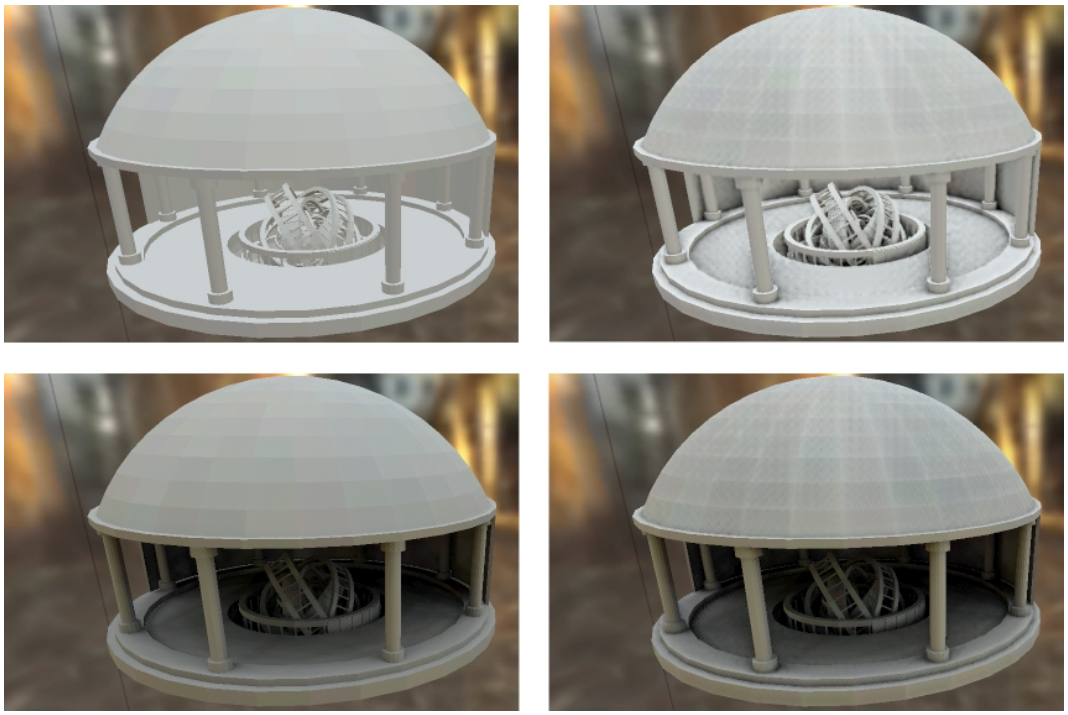


Figure 11: Comparison between the different effects of the presented technique. Upper left: no self shadowing. Upper right: local part. Lower left: global part. Lower right: combination of local and global part. The light probe was sampled using 512 light sources. The achieved an average of 6 fps.

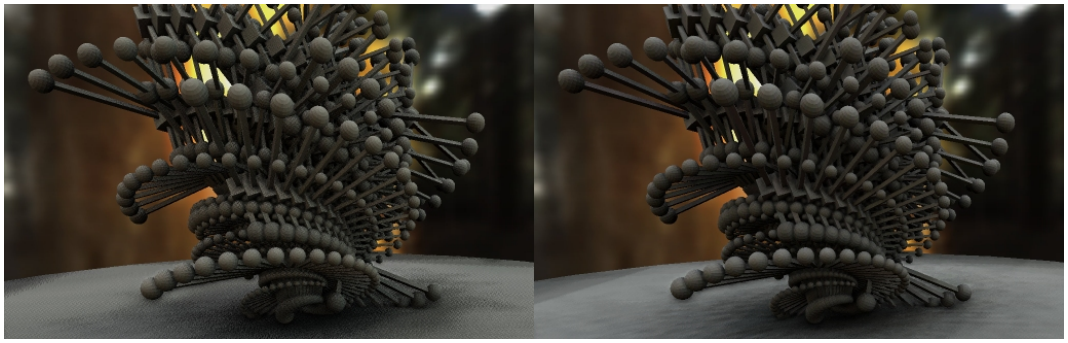


Figure 12: Comparison between different amounts of light sources in the global pass. Left image: 128 light sources at 15 fps. Right image: 512 light sources at 6 fps.

Fitting freeform multi-parameter shapes to 3D data points – A case study

J.S.M. Vergeest, Y.Song

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

j.s.m.vergeest@tudelft.nl

ABSTRACT

We present an approach of geometric fitting of freeform shapes to 3D data points, where the size of the fitting problem is relatively large. The shapes studied represent ship propeller blades of dimension up to 3m, with precision requirements of few mm. Moreover, the shapes are freeform and designed using a geometric model dependent on hundreds of numerical parameters. For the ship industry it is crucial to optimally deal with the shape parameters in order to judge whether or not a particular manufactured part fits within the tolerances. We have developed a method to evaluate the shapes numerically and we report on the approach we took. The sensitivity of a deviation objective function with respect to critical design parameters could be acquired. Also some procedures for automated optimization were explored.

Keywords

Propeller shapes, geometric design parameters, objective function, optimization, shape fitting

1. INTRODUCTION

Fitting of geometric shapes to predefined data points can be regarded as a special case of function fitting, where the function specifies a surface of dimensionality 2 in three-dimensional (3D) space. Shape fitting is a well developed technique applied in surface reconstruction for reverse engineering, or as part of a shape feature recognition process, for example [Chivate 1993, Bardinet 1998, Thompson 1999, Li 2000, Piegl 2001]. In most applications, the shape is modeled as function of shape parameters, for example when the shape is represented by B-spline functions (then the parameters are the control point coordinates), or as a form feature. In the latter case the parameters are called design parameters, since the parameters reflect a quantity which is relevant to the particular application or purpose of the shape.

Although designing new shapes starting from 3D range data is getting technically feasible [Vergeest 2003], the major purpose of 3D scanning and measurement is, historically, the verification of manufactured parts against design requirements. Reverse engineering of shape is mostly done in order to calculate and present spatial deviations between a physical object and a nominal geometric model of the object's surface.

In this paper we explore some ways to evaluate relatively complex shapes of ship propeller blades. These shapes have large dimensions (up to 3 meter each) and should meet high requirements of surface smoothness and geometric precision. The formal requirements of the shapes is evaluated according to quantities defined in ISO standard ISO 484 of manufacturing tolerances [ISO 1981]. To evaluate the shape of a propeller blade against the standard it is sufficient, in industry, to measure a relatively small number (about 100) of surface points. It is obvious that practicing the right methodology for shape evaluation is critical, since the rejection of a manufactured propeller would be extremely costly. In section 2 we describe the nominal model of the shape and the definition of the design parameters, and the general process of shape evaluation. In section 3 we investigate the sensitivity of the geometric deviation function on critical shape parameters. In section 4 we propose a semi-automatic method to determine whether or not a shape conforms to the ISO require-

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

ments. An outlook to application of the method to larger number of parameters is given in section 5.

2. DESIGN AND MEASUREMENT OF THE GEOMETRIC SHAPES

The purpose of a propeller is to deliver propulsion to a ship by pressing water backwards with its rotating blades. The shape of a blade is designed to move water particles into the direction of its axis of rotation. The distance that a water particle would theoretically be moved during one full revolution of the propeller is called the pitch of the blade. Once the requirements of the propeller are determined by the customer, the designer creates the shapes. The actual production is achieved by filling molds with a copper alloy. The inner shape of the molds is not exactly equal to the shape of the blade, since there should be compensation for shrinkage of the metal after cooling down. In this paper we will only consider the nominal model of the shape, as designed, and the shape of a physical propeller blade after its manufacturing (Figure 1).

Figure 1. Example of a propeller blade. The marks on the surface indicate the locations of the verification measurements.

The model of the shape B is constructed from m profile curves G'_1, ..., G'_m, where G'_i defines the intersection of B with a cylinder of radius r_i with axis equal to the rotation axis of the propeller. However, G'_i is specified by the designer as the development G_i of the closed curve onto the xy-plane. Profile G_i is specified by n points g_{ij} with coordinates (x_{ij}, y_{ij}), i=1, ... ,m, j=1, ..., n. Here we assume that n is the same for all profiles. To each profile G_i the following quantities are assigned, which will determine the mapping of G_i into $\mathbb{R}^3$:

- r_i is the radius of the cylinder onto which the points g_{ij}, j=1,...,n will be placed.
- ϕ_i is the pitch angle designed for the profile. The pitch angle determines the pitch distance (defined below). A pitch angle can be designed (implemented) by rotating the profile in the xy-plane by that angle.
- p_i is the pitch distance, defined as the theoretical amount of shift due to a screw revolution of 360 degrees, or $p_i=2\pi r_i$ tg ϕ_i. and hence ϕ_i = atan $p_i/2\pi r_i$.
- σ_i is the skew distance designed for the profile. It is implemented by a shift of the profile in the surface of the cylinder in a direction perpendicular to its axis.
- ρ_i is the rake distance designed for the profile. It is implemented by a shift of the profile in the surface of the cylinder into a direction parallel to its axis.

Examples of a profile curve is shown in figure 2.

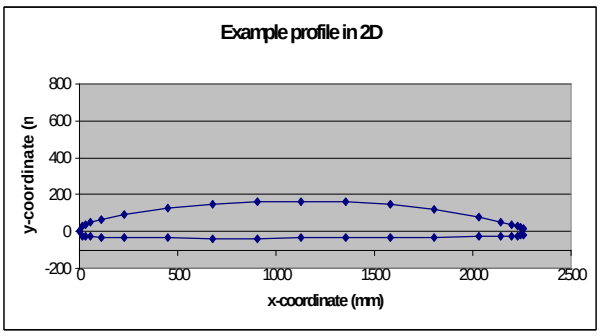


Figure 2. One of the profiles G_i of an example blade design.

The profiles G_i are mapped into space by applying the transformation T_i to each of its points g_{ij} as follows:

$$g'_{ij} = T_i(g_{ij}) = T_i(x_{ij}, y_{ij}), \qquad (1)$$

such that

$$g'_{ij} = \begin{pmatrix} x'_{ij} \\ y'_{ij} \\ z'_{ij} \end{pmatrix} = \begin{pmatrix} r_i \sin(f_{ij}) \\ r_i \cos(f_{ij}) \\ \rho_i + (x_{ij} - \sigma_i)\sin(\phi_i) - y_{ij}\cos(\phi_i) \end{pmatrix}$$

and

$$f_{ij} = (-y_{ij}\sin(\phi_i) - (x_{ij} - \sigma_i)\cos(\phi_i)) / r_i .$$

From the points g'_{ij} the mapped profiles G'_i can be derived for i=1,...,m and from these the nominal blade design B. Figure 3 shows the points g'_{ij} of an example blade design with m=25.

Equation (1) implies the coordinate system convention used, which differs from the one depicted in [ISO 1999].

From (1) we see that in 3D points g'_{ij} are located on cylinders with axes in the z-axis of the coordinate system.

 ISBN 978-80-86943-94-7

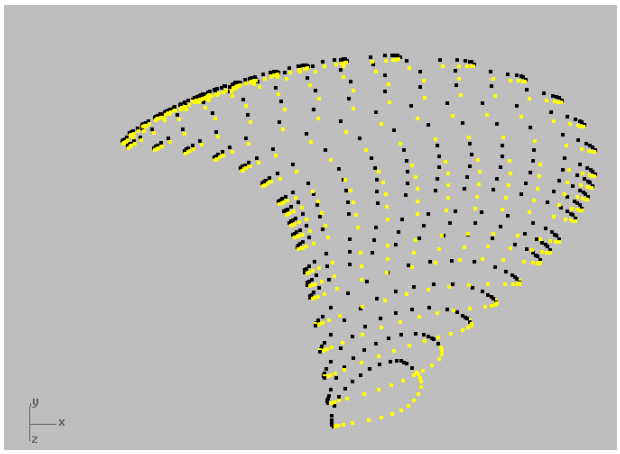


Figure 3. Profile points positioned in 3D space using equation (1).

In the simplest case, if rake, skew and pitch are vanishing, a profile point in the origin of the 2D plane is placed in 3D at the intersection of the *y*-axis with the cylinder. Points with positive *x*-values in the 2D plane will be positioned with negative *x*-coordinates in 3D, since f_{ij} is getting negative. Points with positive *y* in 2D will have lower *z*-coordinates in 3D. Now, if the pitch angle (or pitch distance) is set to a positive value, points in the profile having large *x*-values in the 2D plane move up into positive *z*-direction in 3D. In other words, due to increasing pitch the profile is rotated in 3D about the *y*-axis in the positive sense, *i.e.* counterclockwise, and thus influences the shape of the propeller. If the propeller is operated and be rotated clockwise about the *z*-axis, the profile would then be rotated clockwise about the *z*-axis and the water be pressed into the positive *z*-direction and thus the ship be propelled into the negative *z*-direction. Then the water is pushed by the pressure side, as it should for forward propulsion of the ship. The shape of the blade resulting from equation (1), see Fig. 4, should be interpreted in 3D relative to the ship as follows. The forward direction of the ship is in the negative *z*-direction, the *y*-direction is vertically upward and the *x*-direction is horizontal to the right as seen from an observer on the ship looking forward. The blade is orientated such that the line from axis of rotation toward the tip of the blade is roughly into the positive *y*-direction. As seen from behind the ship the blade is pointing upward and its pressure surface is visible. Due to forward operation the tip of the blade has a speed into the positive x-direction. The profile point located nearest to the origin of the 2D plane (as in Fig. 2) will in 3D have the largest x-value and will move into the positive *x*-direction and therefore is a point on the so-called leading edge of the blade. The concave part of the profile (the lower part in Figure 2) is facing the positive z-direction in 3D and is hence in the pressure surface. A drop of water near the pressure surface of the blade will observe the surface moving toward it and receive a momentum into the positive *z*-direction.

If the pitch would increase then the shape in Fig. 5 would be further twisted about *y*. If the rake of a particular profile gets positive, then in 3D the profile shifts into the positive *z*-direction, according to equation (1). When its skew increases, the profile moves into the positive *x*-direction in 3D. Obviously, rake and skew have a smaller effect on the performance of the screw than has the pitch.

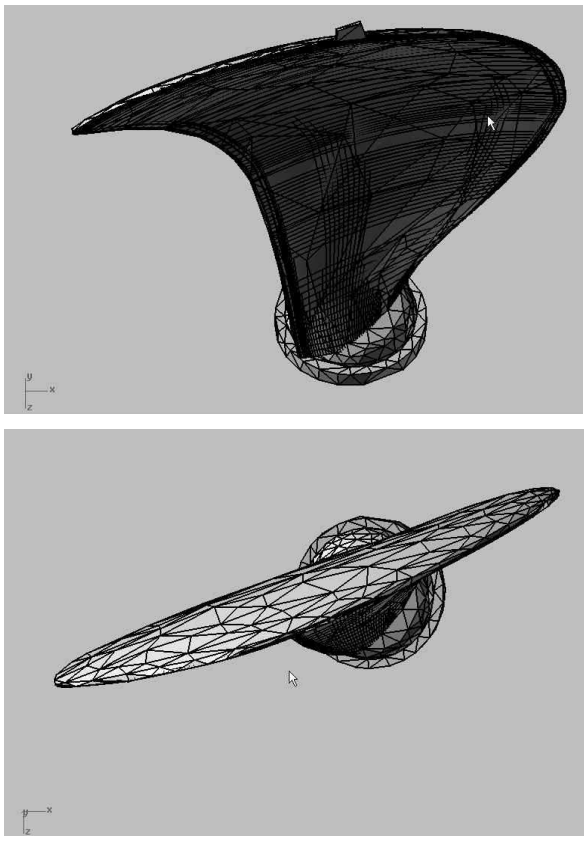


Figure 4. Shape of *B* in 3D space after applying equation (1) to the profile points.

The model depicted in Fig. 4 includes the foot of the blade, which was not represented by the profile points. We show this picture to provide a clear impression of the positioning of the blade relative to the propeller and the ship. However, from this point on, we will refer to a simpler shape design directly obtained from the profile points, see Fig. 5. This shape is used for the computation of distances between measured points on a surface and the designed surface later. The shape consists of two B-spline surfaces, one representing ∂B_p (the pressure surface as designed) and the other ∂B_s (the suction surface) . The pressure surface in Fig. 5 was obtained from the designed profile points g'_{ij} as follows. The profile points form two networks of $m \times n$ = 25×19 points

 ISBN 978-80-86943-94-7

each, one for the pressure surface, one for the suction surface. Using a tensor-product B-spline approximation method [VNI 2006] we determined the B-spline surface of polynomial degree 3 in both directions, having 15 control points in each direction approximating the profile point network of the pressure surface in a least-squares sense. The suction surface ∂B_s was obtained similarly.

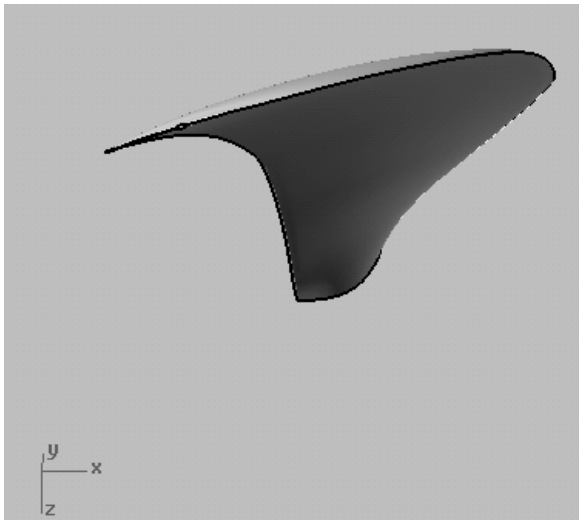


Figure 5. Shape of *B* consisting of two B-spline surfaces fitted to the profile points in 3D.

Once a blade *B* has been manufactured it is inspected as to verify whether its shape fits within the tolerances supplied by the customer. The current procedure at the Wärtsilä company includes the digitization of the shape of physical part *B*. Typically, about 100 data points are acquired of each of the two surfaces, ∂B_p and ∂B_s. Let us denote these two point sets by p_i and s_i, where p_i, $s_i \in \mathbb{R}^3$ and i is an index from 1 to the number of points in the set. The locations of the measured points p_i and s_i relative to the designed shapes are shown in Fig. 6 for the example model. Suppose that the data points p_i are all in front of the designed surface ∂B_p when viewed from an angle as in the upper picture of Fig. 6. Then, in case the deviation would exceed a tolerance it could be resolved by machining or polishing material away from the pressure side of *B*. However, when the data point would be behind the designed surface, there is a risk that *B* is too thin, and there were no correction possible by machining. It is therefore the general policy to produce *B* for which $p_{iz} < p'_{iz}$ for all i, where p_{iz} is the z-coordinate of p_i and p'_{iz} is the z-coordinate of p'_i, the point in ∂B_p closest to p_i. Similarly, $s_{iz} > s'_{iz}$ for all s_i and s'_i is the point in ∂B_s nearest to s_i. In the particular example shown in Fig. 6 the z-residuals have all have that property.

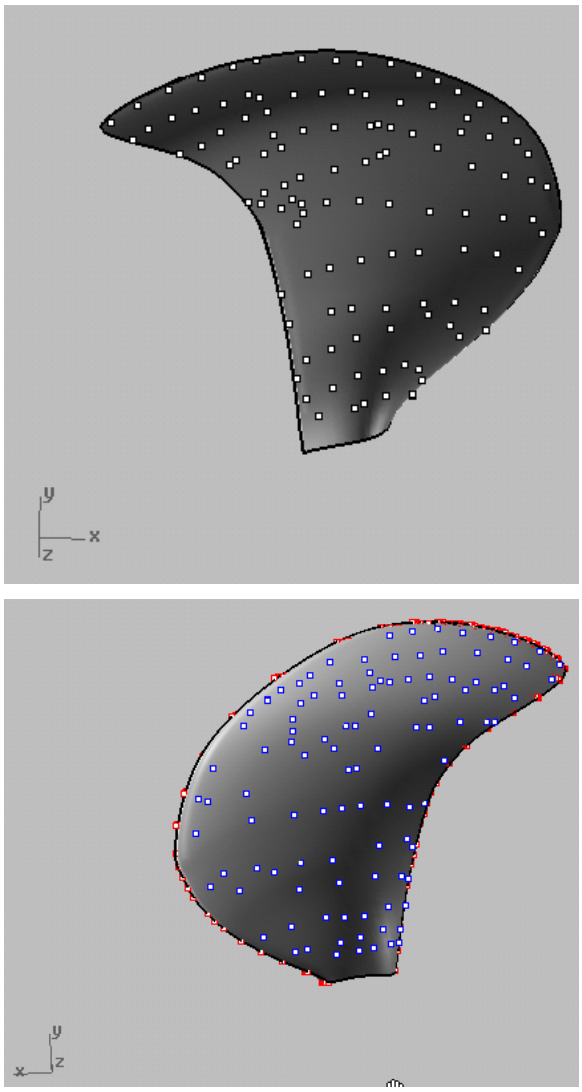


Figure 6. B-spline surface representing the pressure surface and points measured on it (top); same picture of the suction surface (bottom).

In Fig. 7 the distribution z-differences D_z, calculated as $p_{iz} - p'_{iz}$ and $s_{iz} - s'_{iz}$,have been plotted. The D_z against the radius of the data points, *i.e.* their distance from the z-axis, are shown in Figure 8. The data points appear in a vertical linear pattern, due to the fact that the surface was measured at about 5 positions on a profile of a particular cylindrical radius r.

3. SENSITIVITY ANALYSIS OF SHAPE PARAMETERS

Instead of correcting physical part *B* itself we can (at least theoretically) also modify the design *B* as to fulfill the measurement conditions $p_{iz} - p'_{iz} > 0$ and $s_{iz} - s'_{iz}. < 0$ (if necessary) and/or to decrease the mean error of the physical part against the modified nominal design model.

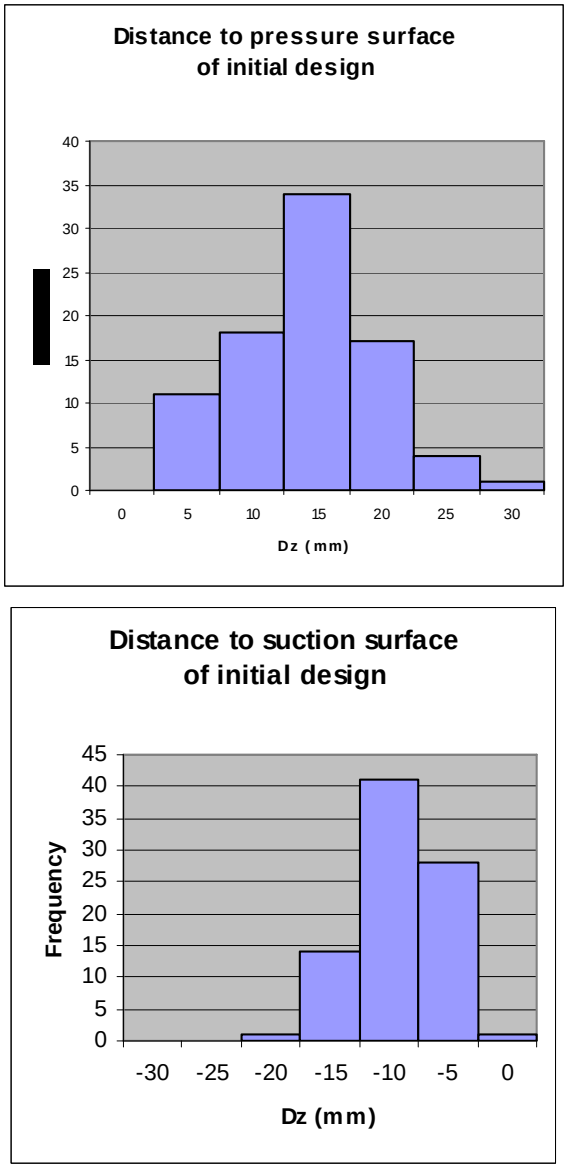


Figure 7. Distribution of signed *Z*-residuals of the measured data points to the nominal surface of the blade.

In doing so, the negotiation between customer and supplier is made in terms of design parameters, rather than in terms of spatial deviations of measured points from the nominal shape. We wish to gain understanding of the effect on the shape due to modification of shape parameters. The parameters which are the least critical to the performance of a propeller are the rake parameters ρ_i. Suppose that the design fulfills all requirements is still not optimal. Then modification of rake could improve the shape while remaining meeting the requirements. This offers an opportunity to optimize a manufacturing condition

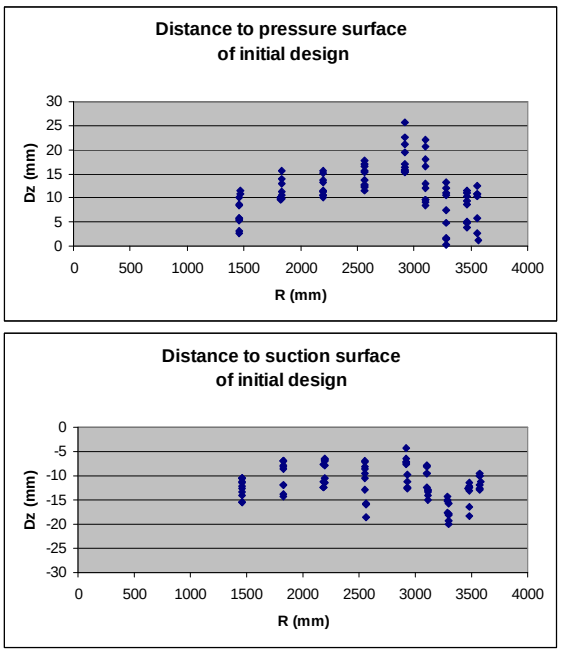


Figure 8. *Z*-residuals of the measured data points versus their radius.

From equation (1) it follows that if the rake ρ_i of some profile(s) G_i is increased by a positive amount a then the values p'_{iz} and s'_{iz} will increase by an amount between 0 and a, depending on the location of the surface point relative to the changed profiles. In terms of the two plots in Fig. 8 it means that points in both plots will move down when a is increased and will move up when decreased From Fig. 8 it can be reasoned that at r near 3500mm the rake should not be increased since it would make some of the D_{zi} for the suction side negative and thus bring some data points inside the blade's volume. A decrease of the rake would be admitted. The largest deviation between data points and suction surface occurs at $r \approx$ 3000mm, so an increase of rake in that region might improve the shape fit in a least-squares sense.

We performed a numerical sampling of rake values ρ'_i around the design values ρ_i by setting:

$$\begin{aligned} \rho'_i &= \rho_i + A_1 \text{ for } 1 \le i < 5 \\ \rho'_i &= \rho_i + A_2 \text{ for } 5 \le i < 9 \\ \rho'_i &= \rho_i + A_1 \text{ for } 9 \le i < 13, \end{aligned} \tag{2}$$

where we used step sizes of 5mm for A_1, A_2 and A_3. The index i=1 corresponds to the data points on the profile with largest radius, r_1 = 3650mm, and the highest, $i = m = 25$, for this particular blade design, the radius is r_{25} = 730mm. The A-parameters influence the rake values and thus the position of the design points g'_{ij} as follows. A_1 affects points in the region of approximately 3600mm < r < 3650mm, A_2 in the region 3100mm < r < 3600 and A_3 in the region

Table 1. Deviation of data points from the shape for three configurations.

Measure	Initial design	$A_{1,2,3}$ = (-20, -5, +5)	Free ρ_0 ... ρ_{11}
Dist. sq. P (mm^2)	15609.3	13644.7	16059.2
Dist. sq. *z* P (mm^2)	13103.8	10991.3	13565.8
Av. dist. P. (mm)	12.5	11.9	12.9
Max. dist. P (mm)	26.9	21.1	27.2
Dist. sq. S (mm^2)	18442.2	19684.8	15885.3
Dist. sq. *z* S (mm^2)	12391.2	13897.0	11223.5
Av. dist. S (mm)	13.8	14.5	12.7
Max. dist. S (mm)	27.4	27.0	25.9
Dist. sq. P and S (mm^2)	34051.4	33329.5	31944.5

2370mm < r < 3100mm. The best fit by sampling was obtained for A_1 = –20mm, A_2 = –5mm and A_3 = +5mm. We also performed a shape fit with ρ_0 ... ρ_{11} as independent free parameters. The results of the analysis are shown in the second data column of Table 1.

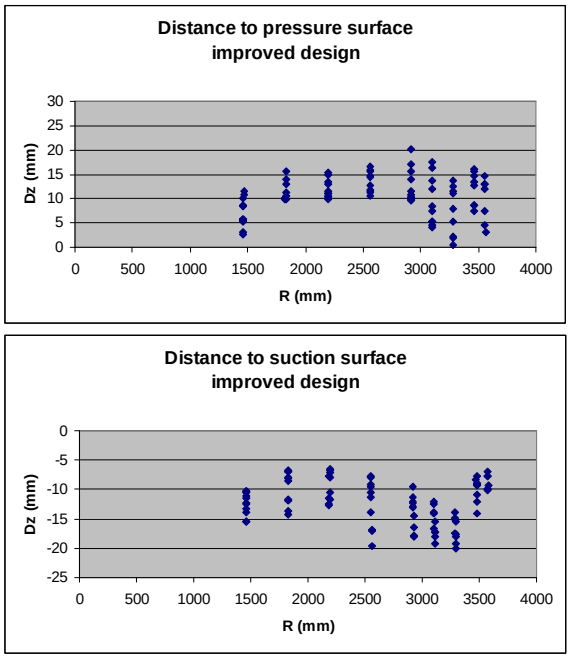


Figure 9. *Z*-residuals of improved shape design, for optimal A_1, A_2 and A_3.

The sums of the distances squared from the measurements of the pressure surface (P) and suction surface (S) are presented, as well as the average and maximum distances.

We performed a 12-parameter fit where the objective function was the sum of the two sums of distances squared, displayed in the lower row of Table 1. The result from the 12-parameter fit, listed in the rightmost column of Table 1, is not realistic, since the rake values deviate strongly (between 10 and 1000mm) from the original design, leading to a distorted shape of the blade (not shown). So, although the fitting criterion is well satisfied, the shape would be unacceptable. The method of stepwise varying sets of 3 adjacent rake values, as in equation (2) provides a more reliable result. The individual data points for this configurations are shown in Figure 9, and can be compared to Figure 8. One improvement of the design in Figure 9 is the reduction of the error at the pressure surface around r = 2900mm. It goes at the cost of a slightly larger sum of squared distances at the suction surface, but the total error, of both surfaces together reduces from 34051.4mm^2 to 33329.5mm^2.

We study the effect of changing rake more closely by varying A_3 around its best value A_3 = 5mm found during the rough stepwise sampling. The quantities defined for Table 1 are plotted as a function of A_3 in Figure 10.

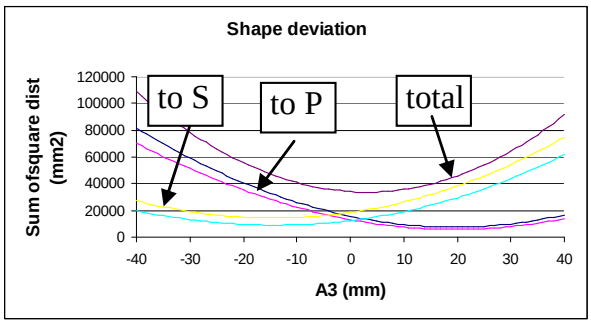


Figure 10. Deviation of shape from data points a function of deviation of rake with indices 7 to 11.

The sum of distances squared of the data points measured on the pressure surface (labeled "to P" in Figure 10) is minimal for $A_3 \approx 19$mm, compared to $A_3 \approx -14$mm for the suction side. The total sun of distances squared in the *z*-direction are shown as well. The total sum reaches its minimum at $A_3 \approx 3$mm.

4. POSSIBLE FITTING STRATEGY

The minimal total sum (as used for Figure 10) is a criterion for goodness of shape, since it implies that material should be removed from both sides of the physical part in order to achieve zero-error. However, care should be taken that the modified shape design never touches or encloses data points.

As mentioned, leaving each rake value as a free parameter in a fit does not work properly. The main reason for this is the instability of shape generation from the design points g'_{ij}. This problem does not occur during the regular design process, where the overall smoothness of the shape is controlled. For the purpose of the numerical experiments, however, we used a simpler, faster shape generation method as to keep the objective function efficient. In this simplified method it can occur that the shape becomes wavy when subsequent rakes are changed very unevenly. To avoid those fluctuations, we varied the rake values in sets, as in equation (2). The result of such a fit is shown in Table 2. Indeed the total sum of squared distances is reduced compared to the estimated values in the middle data column of Table 1, however at the cost of a larger distance at the pressure side of the blade.

The fitting result in Table 2 was obtained by setting the starting values for A_1, A_2 and A_3 to -20, -5 and 5. When choosing different starting values, the fit typically appear to be worse.

Table 2. Fit of shape to data points with free parameters A_1, A_2 and A_3.

Measure	**$A_{1,2,3}$= (–20.0, –5.2, 3.5)**
Dist. sq. P (mm^2)	15228.2
Dist. sq. *z* P (mm^2)	12345.6
Av. dist. P. (mm)	12.7
Max. dist. P (mm)	23.0
Dist. sq. S (mm^2)	17640.9
Dist. sq. *z* S (mm^2)	12381.9
Av. dist. S (mm)	13.7
Max. dist. S (mm)	26.2
Dist. sq. P and S (mm^2)	32869.1

The fitting strategy could be composed as follows:

1. Import the data points from measurements of the pressure side and the suction side of the propeller blade(s).
2. Calculate the deviation of the data points from the nominal design of the blade surface.
3. If any of the data points is contained in the volume of the nominal model or if any data point is too far off from the nominal design, issue a message.
4. Starting form the nominal design parameters, minimize the total of the sums of squared distances of data points to the shape, where a subset of the shape parameters are free in the fitting procedure.
5. Perform some sampling of the free parameters and conduct the action as described in step 4. If any shape is found better fitting the data points, then save these parameter values.
6. Verify whether this shape is conform the specifications of the customer. If not reiterate step 5.
7. Use the resulting shape to generate a milling or grinding process to achieve a physical part matching a design.

5. DISCUSSION

We have shown for a particular case that instead of the shape of the nominal design, an improved shape can be found to control the final machining process of the physical part. The new shape can provide a better balanced machining result because the shape is "positioned" halfway the data points of the pressure surface and those of the suction surface. Then the amounts of material to be removed from the two sides are approximately equal, making the machining process more efficient.

Using the simplified surface construction method, the objective function of the fitting procedure is sufficiently efficient as to generate new shape proposal in about 100s or less. However, as mentioned, there is a small deviation between the shape constructed using the company's high-end CAD system and the shape we obtained by a straightforward least-squares cubic B-spline surface fitting against the design points g'_{ij} of equation (1). In this B-spline fit, there are multiple options for the B-spline knot vector and the polynomial degree of the B-spline functions [VNI 2006]. There is also an issue in the computation of the distance between a data point to the B-spline surface. Finding the point in the B-spline surface closest to a given data points depends on a Quasi-Newton fit with the *u*- and *v*- parameters of the B-spline surface as free parameters. In some cases this fit produces a wrong local minimum, and thus fails to find the closest point. We could solve this issue by choosing 16 points evenly dispersed in the *uw*-space, and performing the distance computation with each of these 16 points as starting values.

As mentioned, the fitting procedure for shape against data points can produce unreliable results. The Quasi-Newton algorithm typically outputs a local optimum near the starting values supplied. Even if the number of parameters is kept as low as three (as we did with the A_i fits), finding the global optimum requires pre-sampling of the parameter space. Whereas this proved feasible for three parameters, it would become impractical for 10 or more parameters. A more sophisticated search algorithm, dedicated to the specific geometric characteristics of propeller blades should be developed. It is unlikely that such an algorithm would work on the data consisting of as few as about a hundred points on each side of the blade. A medium dense point set, of about 5000 points on each surface side is required.

As mentioned, the pitch is a more critical parameter than the skew and rake are. Still, very small changes to the pitch of the nominal design model based on measurements of the produced part, could improve the final machining process as we showed for changes of rake, that is better balanced removal of material. However, if the measurements would reveal that the pitch values are not as designed (and maybe even exceed the tolerances) then it can be attempted to create a new design model with improved pitch values, but still contained within the volume of the manufactured part.

6. REFERENCES

[ISO 1981] "International Standard ISO 484/1, Shipbuilding – Ship screw propellers – Manufacturing tolerances". ISO, Geneva, 1981.

[ISO 1999], Propulsion plants for ships – Terms and definitions – Part 1: Geometry of propellers. Draft International Standard ISO/DIS 3715-1, ISO, Geneva 1999.

[Li 2000] Li, C. L., and Hui, K. C., 2000, "Feature Recognition by Template Matching," Comput. Graph., **24,** pp. 569–583.

[Chivate 1993] Chivate, P. N., and Jablokow, A. G., 1993, "Solid-Model Generation From Measured Point Data," Comput.-Aided Des., **25**, pp. 587–600.

[Bardinet 1998] Bardinet, E., and Cohen, L. D., 1998, "A Parametric Deformable Model to Fit Unstructured 3D Data," Comput. Vis. Image Underst., **71**, pp. 39–54.

[Piegl 2001] Piegl, L. A., and Tiller, W., 2001, "Parametrization For Surface Fitting in Reverse Engineering," Comput.-Aided Des., **33**s8d, pp.593-603.

[Vergeest 2003] Vergeest, J. S. M., Spanjaard, S. and Song, Y., 2003, "Directed Mean Hausdorff Distance of Parameterized Freeform Shapes in 3D: A Case Study," Visual Comput., **19**, pp. 480–491.

[Thompson 1999] W.B. Thompson, J.C. Owen, H. James de St. Germain, S.R. Stark and T.C. Henderson, "Feature-based reverse engineering of mechanical parts", IEEE Tran. Robotics and Automation, Vol. 15, No 1, 1999, pp. 57-66.

[VNI 2006] Visual Numerics, IMSL C Numerical Library Version 6.0, Visual Numerics, Houston, 2006.

Shape Transformation of Multiple Objects Using Slices

Shamima Yasmin
School of Computer Sciences
Universiti Sains Malaysia
11800 USM Penang, Malaysia
shamima@cs.usm.my

Abdullah Zawawi Talib
School of Computer Sciences
Universiti Sains Malaysia
11800 USM Penang, Malaysia
azht@cs.usm.my

ABSTRACT

3D shape transformation is usually confined to transformation between a pair of objects. The objective of this paper is to look at shape transformation from a different perspective: instead of binding this concept between two objects, the technique is extended to the concept of incorporating the characteristics of a number of objects in one body at a time. Equal number of slices are generated from all objects. Slices may be parallel to each other or each slice may have different orientation. Traversal of a data along its longitudinal direction may generate slices which are differently oriented from each other. When multiple objects are transformed to one and is used as an influence shape, it also works as incorporating multiple influence shapes at a time during transformation between two objects. The paper shows the ease of implementation of this concept in sliced data and also discusses its extendibility.

Keywords:
Shape Transformation, Boundary interpolation, Surface Reconstruction.

1. INTRODUCTION

Generally shape transformation means incorporating the characteristics of two different bodies in one output. Our aim is to interpret the term in a different way: incorporating the overall characteristics of a number of different objects (more than two) in one output or incorporating the characteristics of a number of different objects in different proportions in one output. The idea of inventing and designing new models may have applications in show piece design, in pottery industry as well as in animation industry.

In this paper a shape transformation algorithm based on slices is discussed. A number of objects are collected and equal number of slices are generated from these objects. Slices from a particular object may be parallel to each other or each slice can have different orientations. The method works with minimal user intervention without producing any significant distortion to the eye. Again a number of different transformed output can be generated from the same number of objects and user has the right to choose the best one. This method particularly shows the extendibility, versatility of the sliced data and ease of its implementation.

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

2. BACKGROUND

Shape transformation algorithms can be classified into the following two broad categories: a) Surface-based: consists of continuous mapping of small pieces of polygonal surfaces of corresponding objects; b) Volume-based: modifies voxel values of a volume data set.

Surface-based approach uses user-defined control fields such as point fields, line fields etc. during transforma - tion to map key features of the input objects [Hong88, Kent92, Gregory98, Lazarous94, Lee99]. Surface-based methods are important because of its ability to morph between objects of different types of genus, but these methods also require a significant amount of user input. Another troubling feature of surface-based method is the problem of self-intersection. It cannot guarantee that polygonal surfaces will not pass through themselves, creating self-intersecting intermediate result as found in

[Hong88]. All these problems will become more prominent when more than two objects are involved.
Volume-based approach alleviates some of the problems mentioned above. Among them, the simplest approach is the cross-dissolving method [Hughes92,He -94] which at first transforms volume data from spatial domain to frequency domain, interpolates volume in frequency domain and again transforms back to spatial domain. Transformation of multiple objects based on fourier or wavelet can be easy but both methods will have difficulties in specifying slightly complex geometric transformations such as object rotation.
This problem can be alleviated by applying warping before interpolation as found in [Lerois95]. Here user-defined warp is applied on input objects to resemble each other. Instead of using point and line control fields, user-specified disk field [Chen96] can be used. Equal number of disks are applied on both source and target to establish correspondence between them. Each disk has its own normal direction which helps considering distortion of the body. But as the number of input objects increases, the amount of user intervention involved also increases in both of the above mentioned cases.
'Distance Volume' [Payne92] measured by computing the shortest distance of each voxel within the volume to the surface of the object or 'Level Set Method' [Breen01] where the way in which points on the surface moves is used to establish connection among all input objects may not scale well as the number of objects increases.
In shape transformation using implicit function [Turk99], implicit functions of each pair of 2-D slices are determined using a set of constraints i.e. location, weight, scalar values etc. When the number of objects increases, this method may become a bit complex. Pasko et al. uses CSG for blending a number of objects but in a very limited way [Pasko05].
From the above discussion, it is obvious that most of the shape transformation methods do not scale well as the number of objects increases. Our aim is to develop a shape transformation algorithm which optimizes user input, considers rotation/orientation of rigid body during transformation as well as scales better when the number of input objects increases.

3. PROPOSED ALGORITHM

The algorithm mainly consists of the following major steps as shown in Figure 1:

- Data Traversal and the Slicing of the Data;
- Boundary Extraction;
- Boundary Projection and Boundary Interpolation;
- Orientation and Translation of Boundaries;
- Surface Reconstruction.

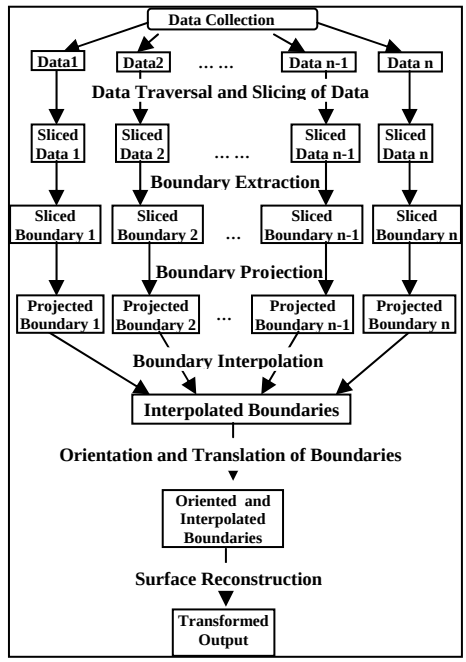


Figure 1: Flow Chart of the Proposed Algorithm.

Various steps of the algorithm are discussed in detail in the following sub sections.

3.1 Data Traversal and Slicing of the Data

A number of data is collected. The initial orientations along which the data are subdivided in the first step of the binary subdivision is defined along any of the directions of the Oriented Bounding Box (OBB) [Lin96]. An Oriented Bounding Box (OBB) is a bounding box that does not necessarily align itself along the coordinate axes. OBB is constructed from the mean and covariance matrix of the cells and their vertices that define the dataset. The eigen vectors of the covariance matrix are extracted, giving a set of three orthogonal vectors that define the alignment of the dataset. Figure 2 shows the difference between a normal bounding box and an oriented bounding box. No doubt, an oriented bounding box more closely fits the data than a normal bounding box. The purpose of choosing the oriented bounding box is to allow checking of the longitudinal direction of dataset from its oriented bounding box rather than from normal bounding box.
Eigen vectors describe the maximum, medium and mini –mum variance of concentration of point clouds. The

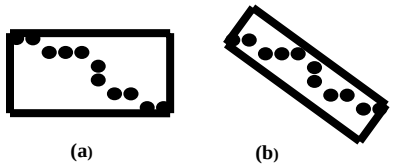


Figure 2: (a) Normal Bounding Box and (b) Oriented Bounding Box.

'maximum' direction shows the maximum amount of concentration of the cells of the data along that direction, whereas the 'medium' direction exhibits less amount of concentration than maximum direction and the 'minimum' direction shows the least amount of concentration or the least alignment of the cells along that direction. The first subdivision takes place along a plane centered at the center of the Oriented Bounding Box (OBB) of the object with normal along the initial alignment. This step, called 'step 0', divides the data into two end parts.

In the next step i.e. 'step 1', each of the two end pieces found from 'step 0' is wrapped with OBB and tested whether the longitudinal direction of the alignment of the sliced end is still within the maximum or medium direction of the OBB. If the alignment is still within maximum/ medium direction, a line joining the center of the previous cut plane and the center of the OBB is used as the direction of the cut plane normal for the ends in that step (Figure 3(a)). Otherwise ends are sliced along the cut plane normal found in the previous step which is used for any further subdivision of the ends and in the subsequent steps no further checking on the alignment is done. At the end of 'step 1', the data is divided into four parts i.e. two end parts and two middle parts.

Before further subdivision of the ends, if necessary, checking is done for the alignments of the two sliced ends. The procedure described in 'step 1' is followed for further subdivision of the two ends. For the middle parts, data is sliced along the plane with center as the center of the OBB and normal directed along the resultant normals of the two ends of the middle data (Figure 3(b)). Slicing is continued along the longitudinal direction until the desired number of steps is reached. In each subsequent step, the number of slices is doubled. The default longitudinal direction is the 'maximum' direction of the eigen vectors and the default number of steps for binary subdivision of the data is 'four'. To provide more flexibility, the initial longitudinal direction as well as the number of steps can be defined by the user. We indicate the maximum alignment as '0', the medium alignment as '1' and the minimum as '2'. Therefore, the users are allowed to vary the morphed output based on the initial alignment. Subdivision can also be forced to happen along any particular direction or along any of the axes i.e. x, y or z to generate parallel slices.

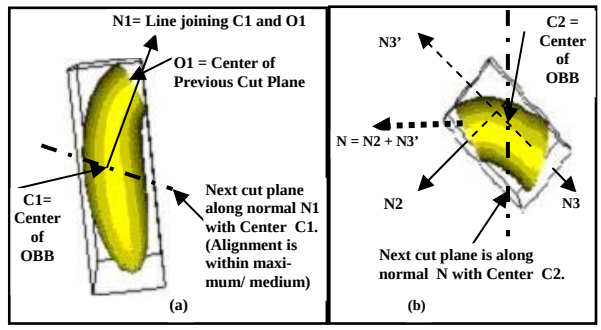


Figure 3: Division of (a) the End Data and (b) the Middle Data.

After reaching the desired number of steps, two ends are traversed along the tip. Usually no further checking for the alignments of the two ends are needed now as the current alignments of the sliced ends are usually not along the maximum or the medium direction of the OBB of the two sliced ends. Hence the normal is usually along the direction which was found at the step before the last checking step and traversing towards the tip is continued along that direction until it is close enough to the tip. At this stage two ends are again divi -ded into two parts.

3.2 Boundary Extraction

Only boundaries of the slices are extracted (Figure 4). As discussed above, in 'step 0', data is divided into two parts. In each of the subsequent steps, the number of slices are doubled. Hence in 'step 4', there are $2^{(4+1)}$ i.e. 32 slices. From 32 slices, 31 boundaries can be extracted. Then boundaries at the ends which are determined after the specified number of steps is reached are also added. Two such boundaries at the two ends result in a total of 33 boundaries each. Hence in 'step 4', the number of extracted boundaries (for each data) is computed as follows:

$$2^{(step+1)} + 1 = 2^{(4+1)} + 1 = 2^5 + 1 = 33.$$

3.3 Boundary Projection and Boundary Interpolation

All data boundaries are projected onto the XZ plane and centered at the origin. Each of the boundaries are traversed along the direction of their minimum X (X_{min}) to maximum X (X_{max}) with a traversal plane defined as (1,0,0). For each boundary, traversal spacing is deter-

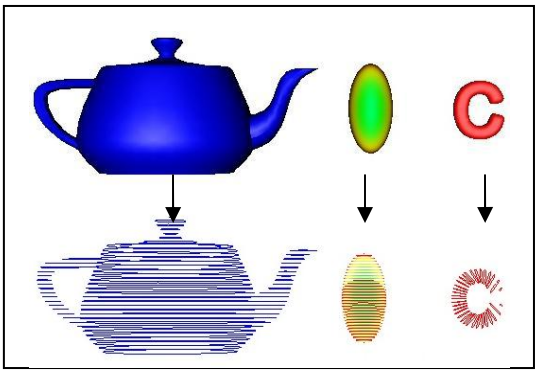


Figure 4: Extraction of Boundaries from a Number of Data.

mined separately. Equal number of traversals is perfor - med for all data. Traversal spacing is determined as follows:

$$Spacing = (X_{max} - X_{min}) / Number\ of\ Traversals$$

Only boundary points are extracted from the traversals. If the number of extracted points in any cut plane happens to be odd, it is made to be even. Next interpo - lation is performed onto the XZ plane. For simplicity, linear interpolation is used in our implementation. Here it should be noted that only one normal is extracted per boundary regardless of whether any particular boundary consists of multiple holes or empty spaces. Also each boundary has one center irrespective of the irregular geometric configuration of that particular boundary. Usually corresponding boundary point clouds are just interpolated. Sometimes enhancement of the interpola - tion process is carried out. When all boundaries have equal number of regions and there are more than one region in all of them, then each of the corresponding regions are interpolated so that interpolated point clouds also have equal number of regions (Figure 5). Region is an area where the number of points extracted by the cut plane is the same while traversing along the X axis.

Again when there are equal number of empty spaces in all boundaries, we have equal number of corresponding regions for the corresponding boundaries. Thus corres - ponding regions can be interpolated. When some of the boundaries contain empty space while other does not have any, number of empty spaces is counted from the boundary which contains the least number of empty spaces and equal number of empty spaces are inserted into the non-empty boundaries. Now all the boundaries contain empty spaces. When there are unequal number of empty spaces/ regions, rightward and leftward traver -sals are carried out from both ends until region in one of the boundaries is exhausted. Corresponding regions

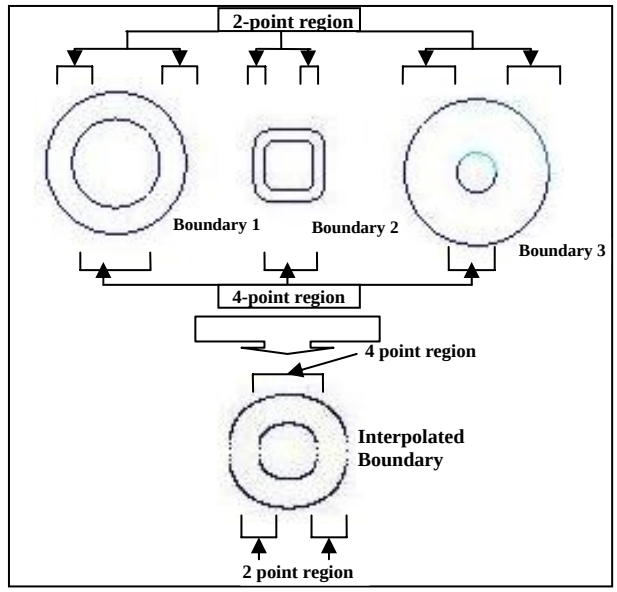


Figure 5: Interpolation of Boundaries after Region Separation.

during the traversal are just mapped and interpolated while the remaining regions can just be mapped if the exhausted side ends with an empty space. Otherwise number of empty spaces is counted from the non-exhausted side which contains the least number of empty spaces and a similar process is applied by inserting into the region of the exhausted side equal number of empty spaces.

3.4 Orientation and Translation of Boundaries

Each of the interpolated boundary already projected onto the XZ plane is oriented along the resultant normal of each of the source and target boundaries and trans lated- to the average center of each of the source and target boundaries (Figure 6). When all the interpolated boundaries are oriented as well as translated, we get the outline of the morphed output (Figure 7).

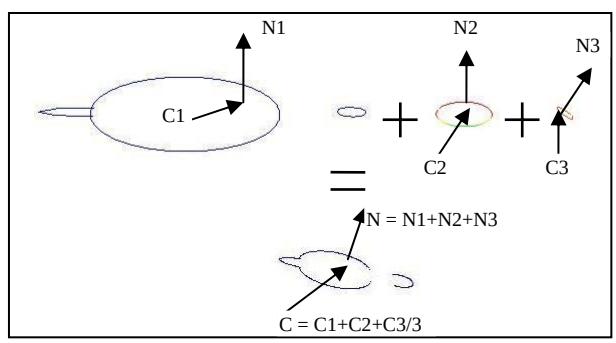


Figure 6: Orientation and Translation of a Single Inter - polated Boundary.

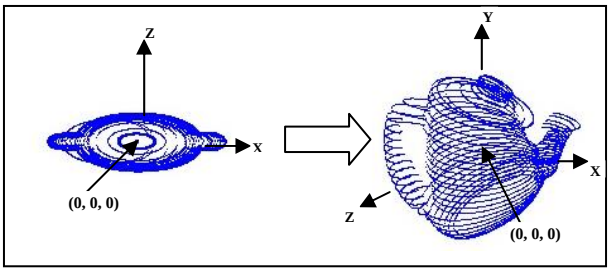


Figure 7: Orientation and Translation of All Interpolated Boundaries.

3.5 Surface Reconstruction

From the stack of oriented and translated boundaries, surface of the morphed object is constructed. Each of the boundaries merges with the next boundary by dividing the in-between space of the two consecutive boundaries into a number of cells. Surface reconstructtion is performed by only considering each of the two consecutive boundaries. This simplifies the overall surface construction process as where data is highly irregular, necessary modification among cell coordinates is limited to only two consecutive boundaries. Surface reconstruction in detail is discussed next.

3.5.1 Disconnected Region Separation

Each consecutive boundary may have regions which are disconnected from one another (Figure 8(a)). Nearest neighbor searching is carried out to find this kind of regions. The disconnected regions are laterally mapped (for better effect) and the other regions are to be mapped across the boundary. This is called vertical mapping (Figure 8(b)). The details of the vertical mapping are discussed in the next subsection.

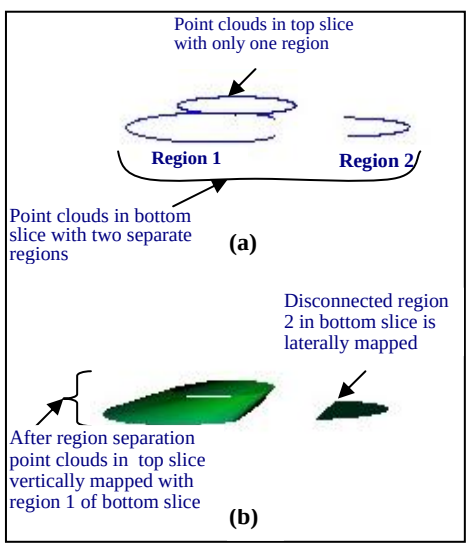


Figure 8: Separating Disconnected Regions between Two Consecutive Boundaries.

3.5.2 Basic Cell Construction

After region separation, two consecutive point/ cell arrays (representing two consecutive slices) are obtained and vertically mapped. The two arrays which contain the number of interpolated points at each index need to be compressed so that the process of mapping can be carried out in an easier and straightforward manner. Figure 9 shows the process of compressing two consecutive arrays. In the compression, the arrays are transformed into two new arrays each: region number array (where region numbers are stored) and number of occurrences array (where the number of occurrences of each region number are stored).

Firstly, the size of both arrays should be made equal using a heuristic approach. Sometimes some index values are dissolved and some are omitted in order to make the size of both arrays equal. Corresponding values of the number of occurrences arrays should also be made equal so that they are ready to be vertically mapped. In the case of unequal values, the larger of the two values is made equal to the smaller number by removing excess number of that particular number of occurrences value. Corresponding number in the two region number arrays should also be equal for the purpose of vertical mapping. If they are not equal, a further processing needs to be done. The process starts with finding the nearest matched index values of the region number arrays by traversing to the left and the right. The nearest matched values will ensure better continuity between different-numbered regions. Next the corresponding region numbers are split into two portions where the values of the region number of the first portion is derived from the continuous mapping of the nearest matched index values to the corresponding region number values and the values of the region number of the second portion are the remaining region numbers resulting from the split. In the example (Figure 9), the first discrepancy occurs at index number ‘2’ and the nearest matched values are at index number ‘1’ with a value of ‘4’ and ‘4’. The current values (i.e. 8 and 6) need to be split into two portions. The first portions are made equal to ‘4’ and the second portions are assigned the remaining values (8-4 = 4 and 6-4 = 2).

At the end of the entire processing, two sets of region number arrays are obtained. The top set (Figure 9(a)) now consists of equal region number and can therefore be vertically mapped whereas each of the bottom set (Figure 9(b)) is to be laterally mapped separately.

Enhancement is carried out in surface reconstruction when empty space is met or at the transition point between two different-numbered regions.

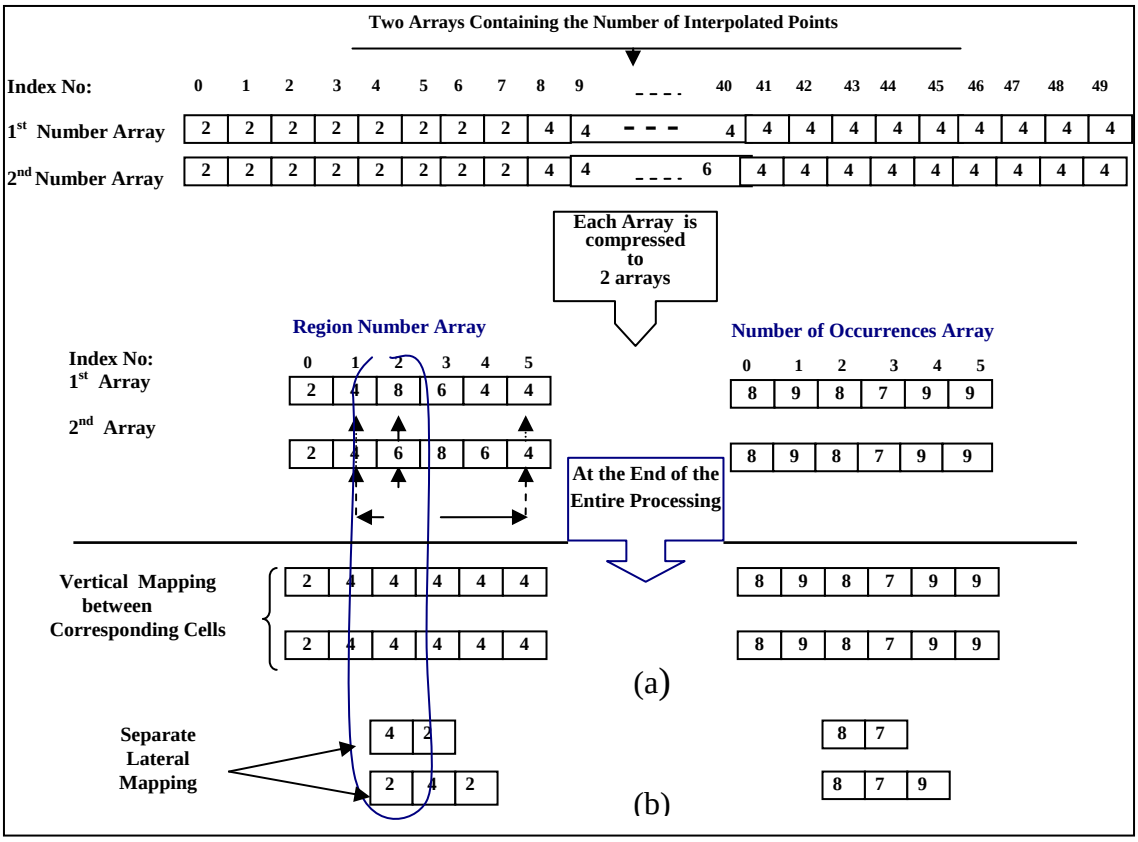


Figure 9: Basic Cell Construction between Two Consecutive Interpolated Boundaries.

4. IMPLEMENTATION AND RESULTS

The algorithm has been implemented using C++ with Visualization Tool Kit (VTK) as graphics platform. In Figure 10(a), transformed shape is constructed when the number of objects is three using '4' as the number of steps with the initial direction of traversal for the first data along the principal axis Y and the rest along the maximum direction of the eigen vector i.e. 'alignment = 0'. As the number of step increases, the number of slices is doubled which also increases the overall run time. Figure 10(b) shows the transformed output when the initial direction of traversal for all data is along the maximum direction of the eigen vector i.e. 'alignment = 0'. In Figure 10(c) transformed output is generated when the initial direction of traversal for all data is along the medium direction of the eigen vector i.e. 'alignment = 1' whereas in Figure 10(d), transformed output is generated with initial direction of traversal for all data along minimum direction of the eigen vector i.e. 'alignment = 2'.

Figure 11 shows the shape transformation when there are four input objects. In Figure 11(a), shape transformation is performed with the initial direction of traversal for all data along the principal axis Y and in Figure11(b) transformed output is generated with initial direction of traversal for all data along minimum direction of the eigen vector i.e. 'alignment = 2'.

Figure 12 compares the shape transformation as the initial direction of traversal for the same input object changes. In Figure 12(a), the initial direction of traversal for all data is along the maximum direction of the eigen vector i.e. 'alignment = 0' and in Figure 12(b), the initial direction of traversal for the first data along principal axis Y and the rest of the data is along the maximum direction of the eigen vector i.e.'alignment = 0'.

Instead of incorporating the overall characteristics of all the objects into one, different parts of different objects can be incorporated in different proportions during shape transformation or some portions of some objects can be kept unchanged. This may result in a totally different objects which is a blend of a number of objects. Figure 13 shows shape transformation by blending two objects in top and bottom portion while the middle portion is a blend of three objects. Figure 14 shows the gradual transformation between two objects

where the initial directions of traversal for both objects are along the minimum direction of the eigen vector i.e ‘alignment = 2’.

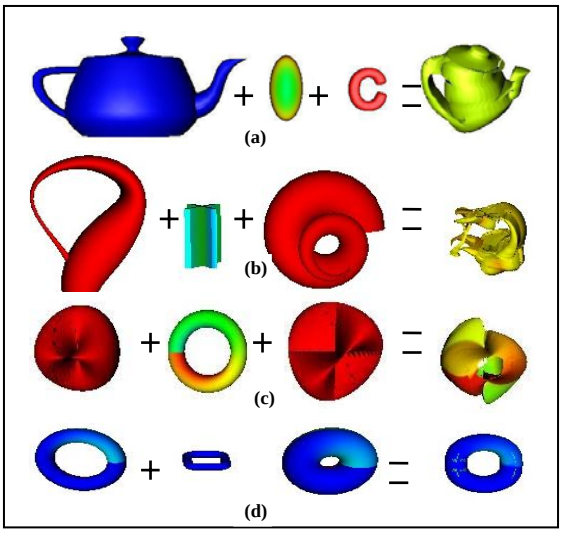


Figure 10: Shape Transformation when the Number of Input Objects is Equal to Three.

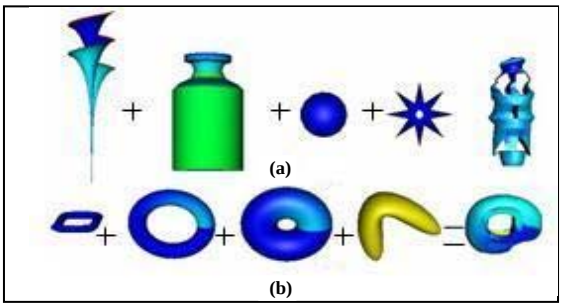


Figure 11: Shape Transformation when the Number of Input Objects is equal to Four.

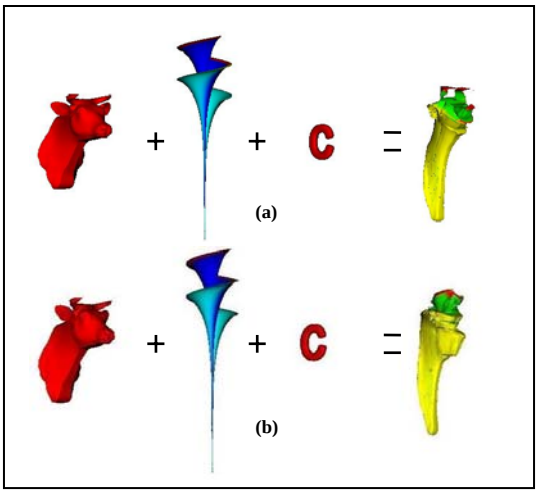


Figure 12: Shape Transformation for the same Input Objects with Different Initial Direction of Traversal.

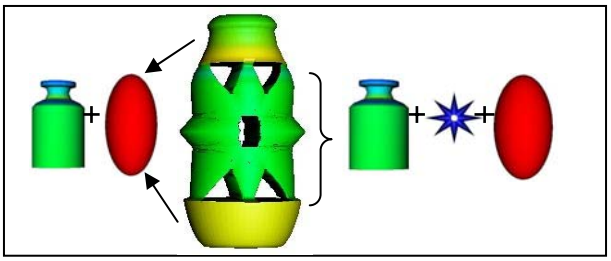


Figure 13: Shape Transformation by Blending Different Objects in Different Proportions.

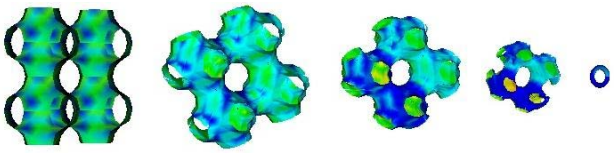


Figure 14: Gradual Shape Transformation between Two Objects (Leftmost and Rightmost).

The usual way of incorporating multiple influence shapes in shape transformation is to use different influence shapes at different stages of transformation in between two objects. Using multiple influence shape in this way does not incorporate the overall characteristics of all the influence shapes among all the intermediate objects. This shortcoming can be alleviated by using our technique. All the influence shapes are transformed first. Then the transformed object can be used as a single influence shape to produce a uniform blend of all the influence shapes during transformation (Figure 15).

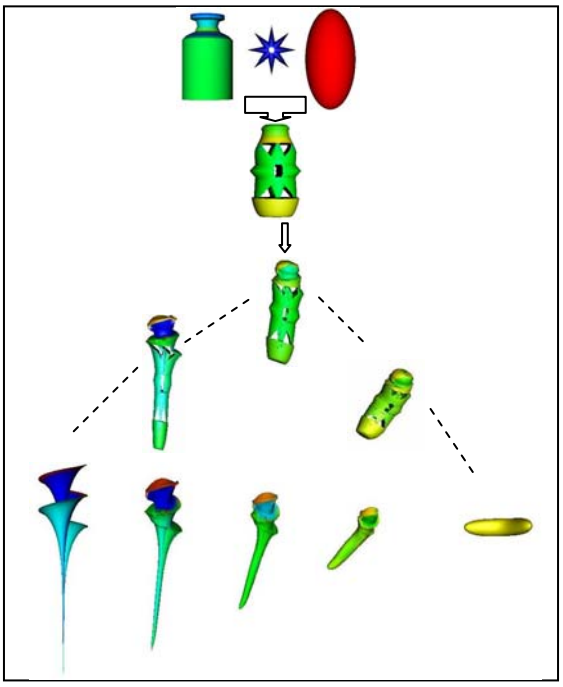


Figure 15: Transformation of Multiple Objects to One Single Object to Influence the Original Path of Morphing.

5. DISCUSSION

This section compares the proposed algorithm with some other existing shape transformation algorithms. Most surface-based methods consider the distortion/ rotation of the rigid body, but division of both source/ target into a number of corresponding patches or meshes is needed at the expense of a large number of user input and longer pre-processing stage [Kent92, Lazarous94, Gregory98, Breen01]. Hence surface-based methods do not scale well when the number of input objects increases. The proposed algorithm works without any user input with default initial settings ('number of steps = 4' and 'alignment = max'). If variations in the number of steps and alignments are desired, the user just needs to specify these two variables. The number of slices can also be reduced by varying the number of steps. This automated method of reducing the number of slices as well as run time is absent in most other algorithms. In most other existing algorithms, specific number of user-defined disk fields [Chen96] or point/line fields [Kent92, Lazarous94, Gregory98, Breen01] are used. Varying these fields involves a considerable amount of user intervention and longer pre-processing time: minor variation in these fields can generate a major variation in the output. The proposed algorithm automatically traverses the data along the user-defined initial alignment and is free from any inaccurate user intervention and at the same time if needed, allows user to specify the initial direction of traversal. This generates different transformed output for the same set of data. Scalability is another major characteristics of the algorithm. This algorithm scales well when the number of input objects increases. Transformed shape maintains the geometric features of the input objects. Rotation/ distortion of the rigid body is also considered which is absent in some volume-based methods that use discrete mathematical function in shape transformation [Hughes92, He94, Turk99]. Some volume-based algorithms alleviate this problems [Payne92, Breen01]. However they are highly sensitive to the user-specified initial overlapping of the objects. Use of CSG in blending of a number of shapes is a bit rigid and its application is also limited [Pasko05].

6. CONCLUSION AND FUTURE WORK

Simplicity and flexibility are two major characteristics of the algorithm which have made it more dynamic and extendible over other existing shape transformation algorithms. Future work includes exploitation of the me thod in parallel / distributed computing environment as simple data structure of sliced body and binary subdivision are suitable for both data and functional partitioning.

7. REFERENCES

[Breen01] D. E. Breen and R. T. Whitaker. A Level-Set Approach for the Metamorphosis of Solid Models. IEEE Transactions on Visualization and Computer Graphics, 7(2), 2001, pp. 173-192.

[Chen96] M. Chen, M. W. Jones and P. Townsend. Volume Distortion and Morphing Using Disk Fields. Computers and Graphics, 24(2), 1996, pp. 567-575.

[Gregory98] A. Gregory, A. State, M. Lin, D. Manocha and M. Livingston. Feature-based Surface Decomposition for Correspondence and Morphing between Polyhedra. Proceedings of Computer Animation, 1998, pp. 64-71.

[He94] T. He, S. Wang and A. Kauffman. Wavelet-based Volume Morphing. IEEE Visualization Proceedings, 1994, pp. 85-92.

[Hong88] T. Hong, N. Magnenat-Thalmann and D. Thalmann. A General Algorithm for 3D Shape Interpolation in a Facet-based Representation. Proceedings of Graphics Interface, 1988, pp. 229-235.

[Hughes92] J. F. Hughes. Scheduled Fourier Volume Morphing. ACM SIGGRAPH Computer Graphics, 26(2), 1992, pp. 43-46.

[Kaul92] A. Kaul and J. Rossignac. Solid-interpolating Deformations: Constructionand Animation of PIPs. Computers and Graphics, 16(1), 1992, pp. 107-115.

[Kent92] J. R. Kent, W. E. Carlson and R. E. Parent. Shape Transformation for Polyhedral Objects. Computer Graphics, 26(2), 1992, pp. 47-54.

[Lazarous94] F. Lazarous and A. Verroust. Feature-based Shape Transformation for Polyhedral Objects. Fifth Eurographics Workshop on Animation and Simulation, 1994, pp. 241-254.

[Lee99] A. W. F. Lee, D. Dobkin, W. Sweldens and P. Schroder. Multiresolution Mesh Morphing. Proceedings of ACM SIGGRAPH , 1999, pp. 343-350.

[Lerois95] A. Lerois, C. D. Garfinkle and M. Levoy. Feature-based Volume Metamorphosis. Computer Graphics, 29, "Annual Conference Series", 1995, pp. 449-456.

[Lin96] S. Gottschalk, M. C. Lin and D. Manocha. OBBTree: A Hierarchical Structure for Rapid Interference Detection. Computers and Graphics (30), "Annual Conference Series", 1996, pp. 171-180.

[Pasko05] G. Pasko, A. Pasko and T. L. Kunii. Bounded Blending for Function-based Shape Modeling. IEEE Computer Graphics and Applications, 25(2), 2005, pp. 36-45.

[Payne92] B. Payne and A. Toga. Distance Field Manipu - lation of Surface Models. IEEE Computer Graphics and Applications, 12(1), 1992, pp. 65-71.

[Turk99] G. Turk and J. F. O'Brien. Shape Transformation Using Variational Implicit Functions. Proceedings of ACM SIGGRAPH , 1999, pp. 335-342.

WEB GIS by Ajax for Analysis and Control of Environmental Data

Igor P. S. Cabral
Federal University of Rio Grande do Norte, Natal/RN, Brazil
igorpsc@yahoo.com.br

Luiz Marcos G. Gonçalves
Federal University of Rio Grande do Norte, Natal/RN, Brazil
lmarcos@dca.ufrn.br

João Carlos Xavier Junior
Federal University of Rio Grande do Norte, Natal/RN, Brazil
jcxavier01@gmail.com

ABSTRACT

The use of Geographic Information Systems (GIS) has becoming very important in fields where detailed and precise study of earth surface features is required. Environmental protection is such an example that requires the use of GIS tools for analysis and decision by managers and enrolled community of protected areas. In this specific field, a challenge that remains is to build a GIS that can be dynamically fed with data, allowing researchers and other agents to recover actual and up to date information. In some cases, data is acquired in several ways and come from different sources. To solve this problem, we propose a tool that includes a model for spatial data treatment on the Web, that we named System for Integrated Monitoring - *SIM*. The research issues involved in this tool start with the dealing of satellite images of protected areas. It is followed by acquisition and processing of different types of video and images as Small Format Aerial Images, or simply *SFAI*, acquired by a radio controlled helicopter, used for construction of mosaics to be fed in the GIS, and underwater geo-referenced pictures, acquired in-loco by researchers for better analyzing a given area. The SIM continues with the feeding and processing of environmental control data collected in-loco as biotic and geological variables and finishes with the presentation of all information on the Web. For this dynamic processing, we have developed tools that make *MapServer* more flexible and dynamic, allowing data uploading by the proper users. As example, on the top of SIM, we have developed a module that uses interpolation to aiming spatial data analysis. A very complex application that has validated our research is to feed the system with data coming from coral reef regions located in northeast of Brazil.

Keywords: Coral Reefs, GIS, Ajax, Interpolation.

1 INTRODUCTION

The use of geo-referenced information is a fundamental issue for enhancing the understanding, managing and ordering of territories, mainly in countries as huge as Brazil. The need of systems for adequate geo-information managing is crucial on the treatment of urban, non-urban and environmental problems. The availability of spatial data trough the Internet is also very important since managing of actual (trustable) and up to date information can be performed in a shared way by users on different locations. The great importance of Internet is related to its possibility of allowing access to information in a shared way including users remotely connected.

In this work, we use Web technologies developed based on the open-source paradigm in such a way that results can be achieved attending two main characteristics, low-costing and information sharing. The main idea is to allow different users, with distinct needs, to feed the system with data and to have access to all information by way of maps available in the Internet. These characteristics are essential in environmental managing actions, including monitoring.

In this context, we propose the System for Integrated Monitoring (SIM) destined to manipulation of geographic data, which has been tested, in a more specifically case study, with environmental coral reef data. Getting input data to the system involves dealing with satellite images, techniques as mosaic construction based on the acquisition of small format aerial images (SFAI), and underwater images acquisition, besides other ways of acquiring data (manual counting, water analysis etc), and the feeding and integrating of these data into the system, a dynamic Web GIS. On the top of this, tools for spatial analysis, as interpolation techniques, could be developed for presentation of data on the Web. The system also allows different users to have different views of the spatial information using user profiles.

So, the main contribution of this work is the development of a flexible platform that can be used by universities or organizations that may be interested in controlling environmental data on the Web. Also, another contribution is the development of mechanisms that allow

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

Plzen, Czech Republic.
Copyright UNION Agency – Science Press

more interaction user × system, in such a way that the user can work on-line, inserting, removing or changing geographic data in several formats, including *shapefile* insertion (standard file used in most GIS [1, 2]). An application of interpolation was also developed in order to generating graphics and other presentation objects that facilitate analysis of correlated data in environmental monitoring, which validates our proposal. Further, this work is a contribution not only to the GIS field but also to the communities involved in environmental areas management and protection, as public organisms (State Environmental Institute), tourists, fishermans, and other Internet users that may be interested in such information.

In order to validate the SIM methodology and technical proposals, we use a case study dedicated to environmental monitoring. We have chosen an area in the State of Rio Grande do Norte, northeast of Brazil, called *Parracho of Maracajaú* that is perfectly adequate to this need. The mentioned area has 9 kilometers by 3 kilometers. It is approximately 7 kilometers distant from the shore and partially emerged at low tides. Actually, *Parracho of Maracajaú* is a coral reef formed by rocks and biogenic living organisms.

The remainder of this paper is organized as follows. In section 2, some background is given about GIS and Internet. In section 3, a brief description about the methodology is explained. In section 4, the implementations for users upload and the interpolation method are described. Experiments and results that validate our proposal are presented and discussed in section 5. Finally, in section 6, we suggest conclusion of this paper and the future works.

2 DEALING WITH GEOGRAPHICAL DATA ON THE WEB

The use of Geographic Information Systems (GIS) has significantly increased in the last two decades. However, it is still necessary more researches in contribution to the development of this field. As described by Openshawn [3], exactly two decades ago but still true, the age of digital map in GIS is coming to an end due to the demand of more specific and advanced geographic analyzes. It is also believed that, in the future, it will be necessary to incorporate more adequate and linked analytical procedures with the existing features used for presenting geographic data.

Besides all of this evolution seems also to happen on GIS, or specifically on Web GIS, we could easy verify that these are, still, very closed applications, mainly with restricted permission for data insertion. Normally, such permissions are allowed only to the creators and managers of the system, presumable experts in geoinformatics, in most of the cases done off-line. So it is inherent and easy perceivable the lack of dynamism in several Web systems [4, 5, 6, 7, 8, 9].

In part, this is due to the fact that most systems use *MapServer*, that works with a single configuration file (*mapfile*) thus being incrusted to a single graphic representation of the map. Obviously the user is allowed to change the map representation, but such modification is made through standard control tools on the Web GIS, as changing the level of zoom, showing or hiding certain layers, between others.

As well described by Camara [10], a GIS generally uses the concept of layers to present maps. Each layer is composed by spatial objects and these are superposed forming the map image. The application environment known as *MapServer* also uses the concept of layers, which are described by a configuration file named *mapfile* [11]. Besides layers description, the *mapfile* also has all information necessary for *MapServer* to reproduce the map image referent to this *mapfile*. Each map image reproduced has a corresponding *mapfile* and each *mapfile* describes only a single map. Thus, the correct generation of *mapfile*s is fundamental for the development of a GIS [12].

A good definition of *mapfile* can be found in the literature [13]. The GIS component is evaluated on the server by an information conversion into a *mapfile*. The *mapfile* is a configuration tool of the map server and influences the appearance of the generated map. That *mapfile* and the GIS data (out of the database) generate the desired map. It is important to keep in mind that GIS data is not transferred between client and server. It is stored during the whole process in the database. Only the information about the location where the file is stored and the appearance of the GIS data is transferred.

Further, by analyzing the infinity of variations in personalization of geographic systems that can be represented on the Web, we can observe that the *MapServer*, working with a single *mapfile*, is limited. A solution would be creating and incrusting more than a *mapfile* into a single system. In fact, this is valid and is what we propose in this work. With the association of multiple files of the type *mapfile*, we can, for example, insert, alter, or remove data directly on the map disposed on the Web GIS. So, in order to attend the dynamical needs of the application case (environmental monitoring), a tool with some singular characteristics is developed here. It has to be easy of use, to allow input of data by the users themselves and to allow on-line processing and creation of layers to be presented on the Web.

2.1 Data interpolation for presentation

In order to come up with proposed modifications in the generally used, single *mapfile* approach, in this work we started by studying a well known problem, which is related to how simple data collection can be converted into information (knowledge). In many cases, data need to be converted into contour line maps, diagrams of per-

spective or images for graphically representing space variation of the data. In other cases, inferring values between given points need to be achieved. In both situations, an interpolation process is necessary [14].

Spatial analysis as part of a GIS offers a variety of tools for estimating unseen points [15]. The use of techniques for interpolation of the measured data allows representing natural phenomena through mathematical calculations that improve understanding of the variations of spatial data. In general, these techniques use models that aim at quantifying the spatial dependence among samples through the use of geo-statistics techniques that presuppose a homogeneous behavior of the structure of spatial correlation in the study area [16].

The single question that we faced is: how to allow users to insert data into the system, process these data (for example in the above interpolation procedures) and present them in a raster image form, on-line? The answer is: a *mapfile*, basis for the visualization, would have to be generated on-line. So we decided to open the black-box, generating our own *mapfile*, as necessary, in order to get this facility to all users of our SIM.

3 THE SIM METHODOLOGY

As depicted above, we propose a full methodology consisting of data acquisition, feeding the system, data processing and presentation of processed information on the Internet. Each one of these steps has an associated module in our system. Actually, this work is an integration of several researches that have been done at our Lab: a complete GIS including several visualization tools is developed for environmental monitoring, a spatial database is modeled for GIS applied to environmental monitoring, techniques are also developed for satellite image processing and for mosaic construction from small format aerial images, and other researches in which we have acquired in-loco data, in the case study region. This integration will be described here. So, a challenging problem solved in this work is to provide dynamic (on the Web) treatment and presentation for data coming from several sources, as hand annotated tables from researchers using scuba diving, satellite, aerial and underwater pictures, and videos.

3.1 Data Acquisition

Figure 1 shows a schema describing our acquisition system. We use satellite images (Landsat) at 30 m resolution for first classifying our interest areas, for visualization when working at low resolutions, and also for serving as basis for other layers coming from other kind of data.

For taking small format aerial images (SFAP) and videos, we have developed an acquisition system consisting of an air model helicopter, a JR GSR260 Voyager Gas, that has a video acquisition and transmission system mounted on a coupled platform under it.

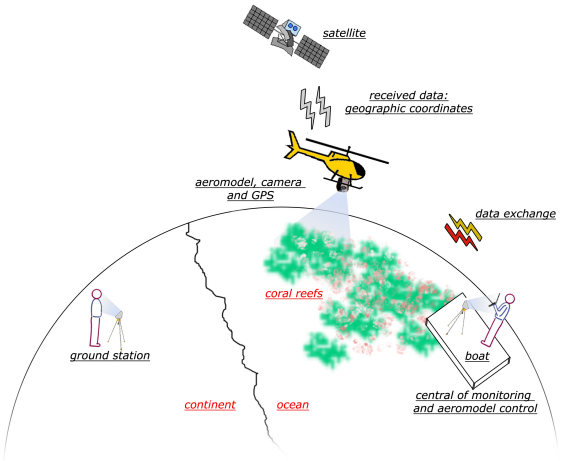


Figure 1: General vision of the acquiring data systems

The used air-model with the coupled platform is presented in Figure 2. Actually, it has a pan-tilt head with a camera on it, pointing down, with 360 degrees of motion in pan and 90 degrees of motion in tilt (so a complete hemisphere is possibly reached only by moving the head). A computer for receiving data, on-line, is also part of this system that works as a ground station. The platform is capable of taking data on the study region and transmitting it to the ground station, in which it can be processed and presented on-line (in real time). The RC helicopter has a GPS coupled to it that can be used for geo-referencing data. In fact, we have generated geo-referenced mosaics of our study areas using digital images acquired from this system [17]. Based on these mosaics, several visual analyses can be carried out and the results presented using the GIS visualization tools.

Other ways for acquiring digital images are also planned in areas where more detailing is necessary based on the visual analysis of the mosaics. For that, we use a digital camera, a Sony DSC-T10, protected by an underwater case. We have taken underwater videos and pictures. Also, hand annotated (geo-referenced) data serve as input to our system, collected by researchers working on the area such as biologists [18] and geologists [19, 20].

3.2 Data visualization tools (Web GIS)

With the improvement of Web platforms, there is an increasing number of amazing technologies that bring GIS visualization applications to new levels of power and usability. In this work, we use the open-source platforms *Apache Web Server* and *MapServer* as basis for visualizing geo-referenced data, on top of which we developed a GIS for showing spatial information on the Web. We use the PHP Mapscript module as programming interface and the free library PostGIS for adding support to geographical objects to the Post-

Figure 2: Our small helicopter with the pan-tilt image and video acquisition head

greSQL database manager, in such a way that it can be used as a spatial repository for geographical information. So, representation of the geometric data is compatible with the open GIS consortium (OGC) standards.

We use AJAX [21] to implement the Web GIS application. This is a powerful and new programming model that has incorporated together and brings several technologies. The XML, JavaScript, HTTP and XHTML are examples of well known and widely-used technologies. Google Maps is an example of a high performance AJAX based application. By integrating these new powerful technologies [22] into GIS systems, we get higher performance results with additional functionalities.

4 IMPLEMENTED TOOLS (SYSTEM ARCHITECTURE)

Figure 3 shows the architecture of the proposed system. Communication among system and *mapfile* (item 2 of Figure 3) normally does not occur in other systems as they do not generate *mapfile*. The SIM uses this communication to guarantee consistency, as a *mapfile* may be automatically generated during system execution. Also, with this characteristic, a system devoted to the coral reef case study could be created with possibility of having different views, by grouping users according to user profiles. Yet, data insertion by the users, themselves, can be done.

4.1 User view based on profile

The case study region involves different users (tourists, managers, researchers, fisherman etc), each one with a different need when visualizing data. So a different view for each class of user (based on a user profile) is initially adopted, which can be changed by the user once he (or she) is working in the system. Besides, for better directing and optimizing the system, the concept of profile access is adopted in this work. Basically, the following types of users have access to our system:

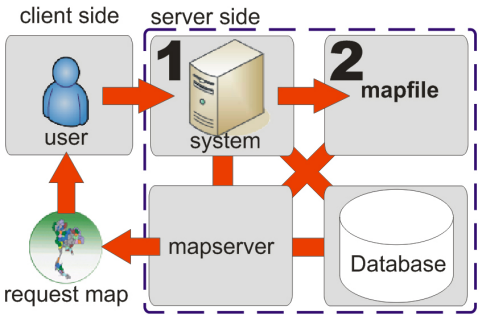


Figure 3: Data communication

- **Researcher**

  - **Biologist** with major interest, for example, in layers such as seaweed and other plants, fishes, live coral reefs, and other biotic species.

  - **Geologist** with interest in layers as coral reef, rocks, sand, and other types of bottom, water salinity, pollution, and in environmental aspects that may influence changes in the geological formation of the area.

  - **Oceanographist** with interest in layers as water visibility, quantity of fishes and other biotic organisms, and in variables such as level of degradation and tourism flow

- **Tourist** with interest in fishes and other alive organisms, coral reefs, water clarity, floating-boats position etc

- **Govern and other managers** with major interest in environmental variables as pollution, degradation level, amount of tourists per area, tax payments, accessibility and other control variables.

- **Local community** with interest in points where fishing is permitted, including lobster and shrimps, points where it is allowed to go with common boats etc.

In this way, the user initially inform to the system which type of user is intended, being directed to the most interesting layers. We not that this is just an initial indication, that can be changed by the user, mixing layers as desired.

It is important to remark that this treatment inside each profile does not involve use of multiple *mapfiles*, since distinct representations are based only on the concept of exhibition/occlusion of layers.

4.2 Multiple mapfiles, a tool for user inputting data

As described in subsection 2.1, we verified that a mechanism for allowing inserting/updating/removing geographic data by the users themselves is necessary in our

application. This has produced significant advances in the system in relation to user interaction and to an increasing in the amount of currently known information about the area. So the concept of multiple *mapfile* is implemented. Basically, a login is associated to each *mapfile*, that can be managed by the data-base management system (DBMS). A specific table stores user data including ids for the associated *mapfiles*.

For creating each *mapfile*, file writing libraries of PHP are used. A module written in PHP verifies in the data base which layers are already shown to the user and then adds new layers to these ones, as necessary, according to the following insertion manners:

1. Data insertion by *shapefiles* or raster image.
2. Data insertion of points and related values in the map with creation of a new table.
3. Data insertion of points and related values in the map in a previous existing table.
4. Data insertion using a file of points with related values.

Insertion by shapefiles or raster image This way makes possible to insert *shapefiles* or even raster images. The use of *shapefiles* is due to the fact that this is an extension standard broadly used in geographical systems in general. A primary advantage of *shapefiles* is that this simple file structure draws faster than a coverage does. This may be why the *shapefile* data structure was developed for the ArcView GIS, a software program that was originally designed for data viewing rather than analysis. In addition, *shapefiles* can easily be copied and does not require importing or exporting as they do the *.e00* format files. The *shapefile* specification is readily available, and a number of other software packages support it. These reasons have contributed to the emergence of the *shapefile* as a leading GIS data transfer standard [23].

In our system, we first define how the *shapefiles* are graphically represented, that is, attributes as layer name, group name, representation symbol (point, line, polygon, circle etc), *srid*, and filling and border color are set firstly. Then, the *shapefile* is uploaded. Each *shapefile* consists of at least three files: shapefile.shp, shapefile.shx, and shapefile.dbf. The latter is the *shapefiles* feature attribute table stored in dBASE format. A *shapefile* can contain only one table. The other two files (shapefile.shp and shapefile.shx) contain information about feature geometry [24]. With the *shapefiles* in the server side, a script runs automatically in order to insert them in the geographical database.

For raster images representation, the procedure is quite similar, differing only in some attributes definitions and in the upload of only one file in the *geotiff* format.

Insertion with creation of a new table Here, the user can insert points by directly clicking the mouse in the location in the map and entering the associated values. A table is created for the user data be stored. This is done dynamically as necessary by the user, which has to provide attributes as table name, group name, representation symbol (point, line, polygon, circle etc); *srid*; filling and border color. The definition of the table columns is also made by the user that determines its name, type (text, char, integer etc) and the size of the fields. Besides table creation, it is also realized the elaboration of the *mapfile* with representation characteristics (lines polygons, color etc) and the table name according with data given by the users.

After the definition of the table and the *mapfile*, the user performs mouse clicks on the map and enters for entering the points in the created table including its geographical localization and its n-values.

It is important to note that the *MapServer* uses a cache for storing data transmitted in the user session in such a way that the information flow between client $\times$ server is optimized. This mechanism avoids instantaneous visualization of new alterations in the map, once the system searches for previously transmitted data. In this way, it is necessary a mechanism (a script) for cleaning the session cache, allowing immediate visualization of new insertions.

Insertion in a previous existing table This function allows the user to enter more information in an existing table in the data base. Procedures for insertion of points work in the same way as in the previous case (insertion in a new table). It is important to note that, as it does exist a table and a *mapfile* associated to it, any insertion is performed only on the table and not in the *mapfile* anymore.

Insertion from a file This function is allows the user to insert data with known localization and n-values in the data base, without the need of using any software for generation of *shapefiles*. A window is opened in which the user enters the file name for upload, inserting its data automatically in the corresponding table in the database. The file extension is *.txt* and each line of the file have the position of the point (X and Y coordinates), and *n* attributes, as necessary, one for each row of the table.

The system also has mechanisms for changing and removing points and tables. The PHP allows the system to change the *mapfile*, removing the part of code that reference a table, after a table is removed.

4.3 Implemented interpolation method

To the end of the implementation of the above mechanisms, we could observe that the system has attended the needs for integration of collected data, since it allows, for example, an unknown user that could be realizing a given research in the region to enter with other

data in the system. This information can be visualized only by this user or by other ones, once the system has mechanisms for marking data as public or private. Based on these modifications, we could implement another mechanism that facilitates data analysis, using interpolation.

We adopt inverse distance weighting (IDW) as the basic interpolation method. We choose IDW because of its ease of use and low computation charge [25]. And furthermore IDW is a popular method used in several problems such as predicting of rainfall and temperature and mapping of crop spraying [26]. The main assumption of IDW is that values of locations closer to the unsampled location are more similar to the value to be estimated than values of locations far away [27]. IDW interpolations are of the form:

$$X_p = \frac{\sum_{i=1}^{n}(\frac{1}{d_i^2}X_i)}{\sum_{i=1}^{n}(\frac{1}{d_i^2})} \quad (1)$$

where X_p is the predicted variable; X_i is the particular measurements at the i-th neighbor point; and the weight function d_i is derived from the inverse square distance from the target point to the i-th neighbor point expressed by Equation 2:

$$d_i = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (2)$$

So, also as a way to validate the use of interpolation methods as a tool for coral reefs monitoring, the above interpolation module was implemented and incorporated to the system, which helps several types of users to monitoring the region of study.

5 EXPERIMENTS AND RESULTS

In order to validate our system, we present results from the system running, depicting its functioning. Figure 4 shows a view of the system for the biologist user.

We also made experiments using the implemented interpolation method. Basically, the interpolation module can be divided into four stages, as shown in Figure 5. The last stage of the interpolation module - stage 4 (Figure 5) - is responsible for representing the estimated values of each selected point chosen by the user. In order to improve the understanding of these values, they have been plotted as bar graphs. All three sets of graphics relate to the tables mentioned above. In the first set (fish set graphics), it is possible to have more sub graphics varying accordingly to the number of species in the region. Figure 6 shows the bar graphs of 3 different regions (a), (b) and (c).

By analyzing the interpolated data, it is possible to identify the relationships between different aspects, such as: good clarity of the water versus high provision

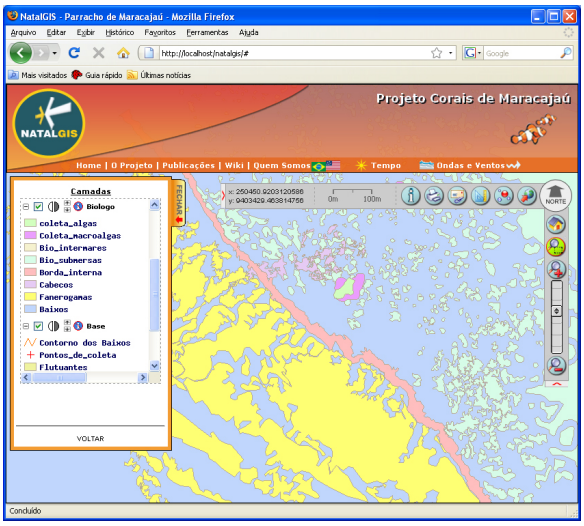


Figure 4: Biologist profile

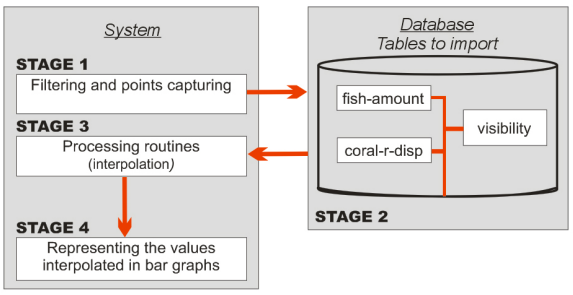


Figure 5: Stages of interpolation process

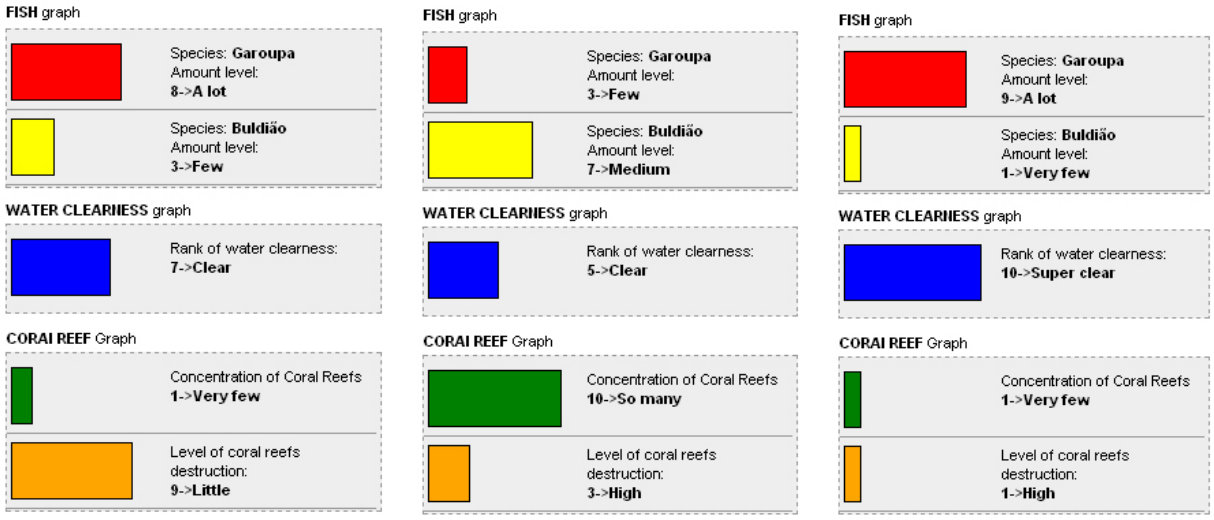
of fish in the region; low provision of fish versus high erosion of the coral reef area. For example, from Figure 6 we can infer some questions to be answered by specialists:

- Why does the level of destruction of the coral reefs is substantially increased on points in which the water clarity is better?
- Why does, differently from Budiao fish specie, the incidence of Garoupa is smaller in regions in which there is a larger concentration of coral reefs?

To verify the first question, Figure 7 shows a view of the map where interpolated data is used to serve as input for analysis of the amount of degradation in function of water clarity.

6 CONCLUSION AND FUTURE WORKS

This paper describes main results obtained by working on a Web based system that aims to help the scientific community. An integrated monitoring system is proposed that involves since data capturing until spatial data representation on the Web, including a procedure that helps data analyses based on interpolation.



(a) (b) (c)

Figure 6: Interpolation graphics

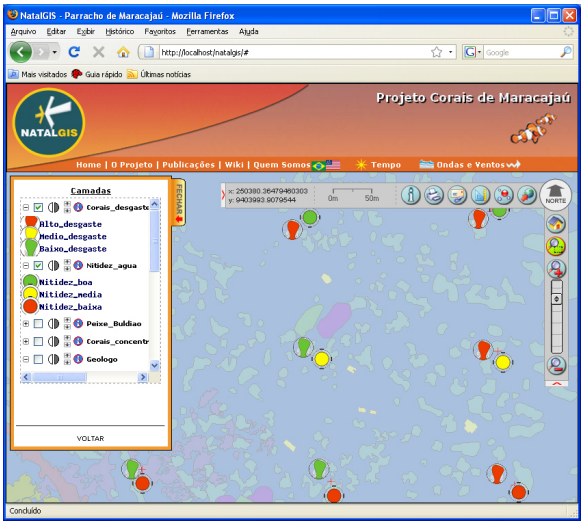


Figure 7: Destruction plus clarity

In this paper, to satisfy previously mentioned requirements, new integration methods are proposed and implemented. The method proposed in this paper for enhancing data integration has implemented the concept of multiple *mapfiles*. Each file, related to the user login, makes possible the insertion of dynamical data that can public and private through the use of *shapefiles*, tables, or even by clicking the position directly in the map.

We can cite several benefits that are generated by this mechanism. A greater amount of data is stored and, consequently, being presented to the user. Independence of systems devoted to the local machine, once all the creation of the system can be done in the server side. A better control mechanism of facts related to coral degradation. The user does not have to know how to create *mapfiles*.

Based on these advances, a method for interpolation, presented in the paper, could be implemented and used for analyses in the case study region. This makes possible to work with analytical data, with comparison of data collected on points in the field besides an estimation can be given to points where data was not collected.

As future work, the interpolation procedure implemented will be used not only for graph generation, but also for creation of polygons and raster data from the point samples. These can be visually represented thus enhancing analysis mainly for managers. For example, dark of clear colors can be used for representing places with more or less amount of fishes, as it happens when using a numerical terrain model in some tools. Yet, an agent architecture is already conceived and will be implemented as a way to determine the user profiles, in a more dynamical way. From user information, the system could infer a transaction set according to the characteristics of each user.

ACKNOWLEDGMENT

This work has the financial support of CNPQ and CAPES, Brazilian sponsoring agencies for research and teaching quality enhancement.

REFERENCES

[1] Clodoveu Davis and Alberto Laender, *Multiple Representations in GIS: Materialization Through Map Generalization, Geometric, and Spatial Analysis Operations*, Kansas city, USA: ACM GIS, 1999.

[2] ESRI, *ESRI Shapefile Technical Description*, Portugal: ESRI 2008.

[3] S. Openshaw, *Developments in geographical information systems*, Economic and Social Research Council, 1988.

[4] NMS, *Laysan Island*, 2008. Accessed on `http://www.hawaiireef.noaa.gov/visit/laysan.html`, in Sep 2008

[5] ICRAN, *International Coral Reef Action Network*, 2008. Accessed on `http://www.icran.org/`, in Sep 2008.

[6] BHP, *World distribution of coral reefs and mangroves*, 2008. Accessed on `http://www.unep-wcmc.org/marine/data/coral_mangrove/index.html`, in Sep 2008.

[7] John Seagethy, *Millenium Global Coral Reef Mapping Project Eastern Caribbean and Atlantic*, 2008. Accessed on `http://imars.marine.usf.edu/corals/maps/ec_reefs.html`, in Sep 2008.

[8] NPS, *Virgin Island National Park*, 2008. Accessed on `http://www.nps.gov/archive/viis/VIIS/index.html`, in Sep 2008.

[9] HCRN, *Coral Reefs of Hawai*, 2008. Accessed on `http://www.coralreefnetwork.com/reefs/default.htm`, in Sep 2008.

[10] Gilberto Camara, *Anatomia de Sistemas de Informação Geográfica*, Florianópolis, Brazil: UNICAMP, 1996.

[11] Mapserver, *MapServer Documentation Project*, 2004. Accessed on `http://www.gpsglobal.com.br/Artigos/Geoestat.html`, in Sep 2008.

[12] Carlos Mello and Geraldo Silva and Jano Souza, *Desenvolvimento de SIG para Web utilizando MDA*, São José dos Campos, Brazil: INPE, 2007.

[13] Hans Hagen and Gerhard Steinebacht and Michael Münchhofen, *DaMI - Data Management for Multimedial Information Systems*. IEEE, 2004.

[14] Eduardo C. Camargo,*Desenvolvimento, implementação e teste de procedimentos geoestatísticos (krigeagem) no sistema de processamento de informações georreferenciadas (spring)*, São José dos Campos, Brazil: INPE, 1997.

[15] Luciana C. Bucene and Célia L. Zimback, *Comparação de métodos de interpolação e análise espacial em dados de pH, em Botucatu-SP*, Botucatu, Brazil: Irriga Magazine, 2003.

[16] G. Câmara and C. Davis and A. Monteiro, *Introdução à Ciência da Geoinformação*, São José dos Campos, Brazil: INPE, 2001.

[17] Natal Cordeiro and Bruno Carvalho and Luiz Gonçalves, *Construção de Mosaicos Georreferenciados Usando Imagens Aéreas de Pequeno Formato para SIG*, Campos do Jordão, Brazil: Geoinfo 2007.

[18] Liana Mendes, *Diagnóstico da fauna da ponta de pirangibúzios, litoral do rio grande do norte*, Caxambu, Brazil: VIII Congresso de Ecologia do Brasil, 2007.

[19] Claude Santos and Helenice Vital and Venerano Amaro, *Mapeamento de recifes submersos na costa do Rio Grande do Norte, NE Brasil: Macau a Maracajau*, Revista Brasileira de Geofísica, 2007.

[20] Ricardo Amaral, *Mapeamento da Área de Proteção Ambiental dos Recifes de Corais - Fase Exploratória*, Relatório Interno Ilustrado, 2002.

[21] Dave Crane and Eric Pascarello, *Ajax in Action*, Manning Publications, 2005.

[22] Ahmet Sayar and Marlon Pierce and Geoffrey Fox, *Integrating AJAX Approach into GIS Visualization Web Services*, Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW 2006): IEEE, 2006.

[23] David Theobald, *Understanding Topology and Shapefiles*, ArcUser, 2001.

[24] Kai Lin and Bertram Ludäscher, *A System for Semantic Integration of Geologic Maps via Ontologies*, Sanibel Island, Florida, USA: Semantic Web Technologies for Searching and Retrieving Scientific Data (SCISW), 2003.

[25] F. Collins and P. Bolstad, *A Comparison of Spatial Interpolation Techniques in Temperature Estimation*, Third International Conference/Workshop on Integrating GIS and Environmental Modelling, 1996.

[26] M. Tomczak, *Spatial interpolation and its uncertainty using automated anisotropic inverse distance weighting (IDW)*, Journal of Geographic Information and Decision Analysis, 1998.

[27] Jun Gao and Peter Revesz, *Visualization of Temporal-Oriented Datasets*, Proceedings of the Geometric Modeling and Imaging¯ New Trends (GMAI'06): IEEE, 2006.

Object Detection in Low Illumination Environment

Suk-Ho Lee
Univ. Dongseo, Korea
petrasuk@gmail.com

Seung Il Han
Univ. Dongseo, Korea
ted4sim2@naver.com

Moon Gi Kang
Univ. Yonsei, Korea
mkang@yonsei.ac.kr

ABSTRACT

We propose a motion detection algorithm which works well in low illumination environments. By using the level set based bimodal motion segmentation, the algorithm obtains an automatic segmentation of the motion region and the spurious regions due to the large CCD noise in low illumination environment are removed effectively. Based on a noise analysis, we will show how to obtain the required parameters automatically. Experimental results verify the stableness of the proposed algorithm in low illumination conditions.

Keywords: Level set, motion detection, low illumination, surveillance system.

1 INTRODUCTION

Motion detection in low illuminance environment is one of the most difficult and important problems in surveillance camera applications. Conventionally, motion detection is performed by using thresholding based methods [7][8][9][10][11] or statistical methods such as Pfinder [12], W^4 [13], Mixture of Gaussians method [14], and so on. Such methods all use some kind of a threshold value that has to be determined a priori.
However, motion detection in low illumination environment are difficult due to the following facts: First, the difficulty of setting an a priori known threshold value for segmentation[7] increases in low illumination environment, since the range of the brightness value in the difference map is narrower than in day light environment. Second, the CCD noise, which becomes more dominant when the light intensity(or the SNR) is low, causes several spurious regions that can raise the false alarm rate in intelligent surveillance camera systems.

In this paper, we propose an algorithm that deals with these problems effectively. The algorithm performs a real-time motion detection in low illumination environment as well as in day light environment, without the need of setting a threshold value and with the capability of removing spurious regions caused by the noise.

The algorithm is based on two separate level set based bimodal segmentation algorithms [4]-[6], which process a level set function such that the processed level set function classifies the image frame into the motion and the non-motion regions. The first bimodal segmentation processes the level set function such that the level set function automatically sets the threshold value, and performs a pre-segmentation of the motion region, which contains also spurious regions. After that, a new difference map is formed using the geometric information of the pre-processed level set function, and the second bimodal segmentation processes this new difference map such that spurious regions caused by the CCD noise become entirely removed.

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

2 LEVEL SET BASED OBJECT DETECTION

In [4], we have proposed a model that implements a bimodal segmentation based on the competition of the brightness values in an image. The segmented regions are represented via a level set function ϕ which minimizes the following energy functional:

$$\begin{aligned} E(\phi) \;=& \lambda_1 \int_\Omega \left|u_0(\mathbf{r}) - ave_{\{\phi\geq 0\}}\right|^2 \phi(\mathbf{r}) H(\alpha+\phi(\mathbf{r}))d\mathbf{r} \\ -\;& \lambda_2 \int_\Omega \left|u_0(\mathbf{r}) - ave_{\{\phi<0\}}\right|^2 \phi(\mathbf{r}) H(\alpha-\phi(\mathbf{r}))d\mathbf{r} \quad (1) \end{aligned}$$

where λ_1, λ_2 are non-negative parameters,α is an arbitrary small positive value, $u_0(\mathbf{r})$ is the given image, Ω is the domain of the given image u_0, ϕ is the level set function, and $ave_{\{\phi\geq 0\}}, ave_{\{\phi<0\}}$ are the average values of $u_0(\mathbf{r})$ in the 2-D regions $\{\phi \geq 0\}$ and $\{\phi < 0\}$, respectively. Here, $H(\phi)$ is the one-dimensional Heaviside function with $H(s) = 1$ if $s \geq 0$, and $H(s) = 0$ if $s < 0$. In [4], the following theorem was proved.

Theorem 2.1 *Assume u_0 is a continuous function. Let,* $\Lambda_\phi(\mathbf{r}) := \left|u_0(\mathbf{r}) - ave_{\{\phi\geq 0\}}\right|^2 - \left|u_0(\mathbf{r}) - ave_{\{\phi<0\}}\right|^2$ *and*

$$sign(\Lambda_\phi(\mathbf{r})) := \begin{cases} 1 & \text{if } \Lambda_\phi(\mathbf{r}) > 0 \\ -1 & \text{if } \Lambda_\phi(\mathbf{r}) < 0 \\ 0 & \text{if } \Lambda_\phi(\mathbf{r}) = 0. \end{cases}$$

Then, if $\phi(\mathbf{r})$ is a minimizer for the energy functional $E(\phi)$ in (1), $sign(\Lambda_\phi(\mathbf{r}))\phi(\mathbf{r}) = -\alpha$, i.e., $\phi(\mathbf{r}) = -sign(\Lambda_\phi(\mathbf{r}))\alpha$, whenever $sign(\Lambda_\phi(\mathbf{r})) \neq 0$.

The theory states that the energy of the energy functional in (1) is minimized when every $\mathbf{r}$ at which $\left|u_0(\mathbf{r}) - ave_{\{\phi\geq 0\}}\right|^2 < \left|u_0(\mathbf{r}) - ave_{\{\phi<0\}}\right|^2$ is classified in the region $\{\mathbf{r} \mid \phi(\mathbf{r}) = -\alpha\}$, and every $\mathbf{r}$ at which $\left|u_0(\mathbf{r}) - ave_{\{\phi\geq 0\}}\right|^2 > \left|u_0(\mathbf{r}) - ave_{\{\phi<0\}}\right|^2$ is classified in the region $\{\mathbf{r} \mid \phi(\mathbf{r}) = \alpha\}$ with an arbitrary initial level set function. In the steady state, the ϕ function becomes a piecewise continuous function having either the value α or $-\alpha$ at each point, and the zero level set of ϕ becomes the contour that segments the image region into two regions based on the competition of the colors. Since the segmentation result depends on the competition of the colors, that is, on the relative relationship of the colors, the segmentation result is adaptive to the image.

The adaptive property of the bimodal segmentation makes it useful for motion detection. As mentioned in the introduction, the object is normally segmented by segmenting the background subtracted frame, where the threshold value varies from scene to scene. However, the bimodal segmentation algorithm segments the object without thresholding, and therefore the segmentation process becomes adaptive to the image and automatical, since there is no need to set an a priori known threshold value. In the bimodal segmentation scheme, we use instead of u_0 in (1), the background subtracted frame $\mathbb{D}u$.

When the SNR of the image becomes very low, then the noise becomes apparent in the difference frame, and some pixels, at which the intensity difference value exceeds the average intensity difference between the two regions, become classified in the counterpart region. Figure 1(a)-(c) shows the background frame, the current frame, and the difference frame of a scene obtained in the night. Figure 1(d) shows the segmentation result with the bimodal segmentation. Even though the bimodal segmentation segments the regions automatically, due to the noise, there appears many spurious regions in the segmented regions.

Figure 1(d) shows that the ϕ function has several outliers(dotty points), which result in falsely segmented regions. Hereafter, we refer to the noise as the outliers in the ϕ function, that is, the points which have falsely converged to α (or -α), instead of -α (or α). To eliminate such outliers, certain regularization terms which suppress spatial discontinuity in the ϕ function have to be used together with the bimodal term.

3 LEVEL SET BASED MOTION DETECTION WITH HARMONIC REGULARIZATION

Various energy functionals that use the magnitude of the gradient of the image can be used as the regularization term to suppress the noise. A well-known regularization term is the total variation regularization term. Using this term together, the energy functional becomes:

$$\begin{aligned} E(\phi) &= \lambda_1 \int_\Omega \left|\mathbb{D}u(\mathbf{r}) - ave_{\{\phi\geq 0\}}\right|^2 \phi(\mathbf{r}) H(\alpha + \phi(\mathbf{r})) d\mathbf{r} \\ &- \lambda_2 \int_\Omega \left|\mathbb{D}u(\mathbf{r}) - ave_{\{\phi<0\}}\right|^2 \phi(\mathbf{r}) H(\alpha - \phi(\mathbf{r})) d\mathbf{r} \\ &+ \beta \int_\Omega |\nabla\phi| d\mathbf{r}, \end{aligned} \tag{2}$$

where we denoted by $\mathbb{D}u$, the background subtracted frame, and where α and β are an arbitrary positive value, and Ω is the domain of the given image $\mathbb{D}u$.

The corresponding gradient descent flow equation with respect to ϕ is :

$$\begin{aligned} \phi_t &= \beta \nabla \cdot \left(\frac{\nabla\phi}{|\nabla\phi|} \right) \\ &- \lambda_1 |\mathbb{D}u(\mathbf{r}) - ave_{\{\phi\geq 0\}}|^2 \{H(\alpha + \phi(\mathbf{r})) + \phi H'(\alpha + \phi(\mathbf{r}))\} \\ &+ \lambda_2 |\mathbb{D}u(\mathbf{r}) - ave_{\{\phi<0\}}|^2 \{H(\alpha - \phi(\mathbf{r})) - \phi H'(\alpha - \phi(\mathbf{r}))\}. \end{aligned}$$

However, a faster algorithm than using the total variation term as the regularization term is the one with harmonic regularization. The harmonic regularization uses the L_2 norm of the magnitude of the gradient of the image and thus the smoothing speed is faster than the total variation regularization. The bimodal segmentation with harmonic regularization has the following energy functional form:

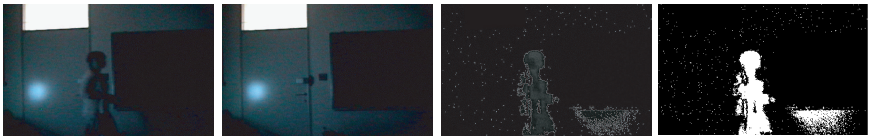
$$\begin{aligned} E(\phi) &= \lambda_1 \int_\Omega \left|\mathbb{D}u(\mathbf{r}) - ave_{\{\phi\geq 0\}}\right|^2 \phi(\mathbf{r}) H(\alpha + \phi(\mathbf{r})) d\mathbf{r} \\ &- \lambda_2 \int_\Omega \left|\mathbb{D}u(\mathbf{r}) - ave_{\{\phi<0\}}\right|^2 \phi(\mathbf{r}) H(\alpha - \phi(\mathbf{r})) d\mathbf{r} \\ &+ \beta \int_\Omega |\nabla\phi|^2 d\mathbf{r}, \end{aligned} \tag{3}$$

Here, the choice of β will depend on the noise structure of the difference image.

The corresponding gradient descent flow equation with respect to ϕ is :

$$\begin{aligned} \phi_t &= \beta \nabla^2 \phi \\ &- \lambda_1 |\mathbb{D}u(\mathbf{r}) - ave_{\{\phi\geq 0\}}|^2 \{H(\alpha + \phi(\mathbf{r})) + \phi H'(\alpha + \phi(\mathbf{r}))\} \\ &+ \lambda_2 |\mathbb{D}u(\mathbf{r}) - ave_{\{\phi<0\}}|^2 \{H(\alpha - \phi(\mathbf{r})) - \phi H'(\alpha - \phi(\mathbf{r}))\}. \end{aligned}$$

Normally, in image denoising, it is preferred to use the total variation regularization term, since it preserves edges, and the intensity value of large scale structures varies not much. However, in smoothing the ϕ function, the change in the magnitude of the ϕ value is not of interest as long as it does not change its sign, since it is the sign of the $\phi(\mathbf{r})$ value and not the magnitude value that determines the classification. Sharp edges in the ϕ function become smeared by the harmonic regularization term, but the smearing is not so critical an artifact as in image denoising, since the position of



(a) (b) (c) (d)

Figure 1: Showing the result of object detection with level set based bimodal segmentation without using regularization terms. (a) current frame (b) background frame (c) absolute difference frame (d) segmentation result (level set function).

the zero level set will not change much by the smearing process. Therefore, the harmonic regularization term can achieve similar classification result as the total variation regularization term, while being faster. Since, in video processing the speed is the more important factor we prefer (3) to (2).

Clearly, the regularization process should have a larger effect if the SNR becomes lower. Therefore, the parameter β is dependent on the SNR. Therefore, we give here a rough analysis on the relationship between the parameter β and the noise, and show how to choose the parameter β.

We denote by $\Psi[\phi](\mathbf{r})$ the integrand of the energy functional in (1):

$$\begin{aligned}\Psi[\phi](\mathbf{r}) := \quad & +\left|u_0(\mathbf{r}) - ave_{\{\phi\geq 0\}}\right|^2 \phi(\mathbf{r})\, H(\alpha+\phi) \\ & -\left|u_0(\mathbf{r}) - ave_{\{\phi<0\}}\right|^2 \phi(\mathbf{r})\, H(\alpha-\phi),\end{aligned} \tag{4}$$

As stated in theorem 2.1, in the steady state this integrand converges to α or to $-\alpha$ according to the intensity value at the pixel corresponding to the integrand. Now, together with the harmonic regularization term, the new integrand becomes,

$$\begin{aligned}\Psi[\phi](\mathbf{r}) := & +\left|u_0(\mathbf{r}) - ave_{\{\phi\geq 0\}}\right|^2 \phi(\mathbf{r}) H(\alpha+\phi) \\ & -\left|u_0(\mathbf{r}) - ave_{\{\phi<0\}}\right|^2 \phi(\mathbf{r})\, H(\alpha-\phi) + \beta|\nabla\phi|^2\end{aligned} \tag{5}$$

We will compute the minimum value of this integrand for a "one-pixel noise" model. In the "one-pixel noise" model it is assumed that there is no other noisy pixel in the neighborhood of the noisy pixel. Figure 4 shows this "one-pixel noise" model for the one-dimensional case. Here, the values $\phi_a = -\alpha$,$\phi_b = \alpha$, and $\phi_c = -\alpha$ are assumed to be the ϕ values at the steady state of (2) without the harmonic regularization term, where pixel 'b' corresponds to the noisy pixel, and 'a' and 'c' are the rightly classified pixels. By denoising, we mean that the ϕ value of 'b' drops below zero, so that sign of the ϕ values of 'a','b', and 'c' all become the same, i.e., negative.

First, we compute the minimizer ϕ value of 'b' with the harmonic regularization, i.e., the minimizer of (5), with β fixed. Then, we find the condition of β which makes the ϕ value of 'b' negative. Definitely, the minimizer at which the energy of the integrand reaches the minimum cannot be outside the interval $|\phi| < \alpha$, since this would make both the energy of the bimodal term and the harmonic regularization term increase. Therefore, the minimizer must lie in the interval $|\phi| < \alpha$, which is the point $\phi_b^{'}$ in Fig. 4. In this interval the two Heaviside functions in (5) become both 1, and the integrand becomes

$$\Psi[\phi](\mathbf{r}) := \Lambda(u_0,\phi)\phi(\mathbf{r}) \; + \beta|\nabla\phi|^2, \tag{6}$$

where $\Lambda(u_0,\phi) = \left[\left|u_0(\mathbf{r}) - ave_{\{\phi\geq 0\}}\right|^2 - \left|u_0(\mathbf{r}) - ave_{\{\phi<0\}}\right|^2\right]$. Even though $\Lambda(u_0,\phi)$ is a function of ϕ, the one point denoising has scarcely any effect on the change of $ave_{\{\phi\geq 0\}}$ or $ave_{\{\phi<0\}}$, therefore, to make the analysis simple, we regard $\Lambda(u_0,\phi)$ as a constant with respect to ϕ, and denote it by $\Lambda(u_0)$. We also use the discretized version of the harmonic regularization term, that is, we let

$$\begin{aligned}|\nabla\phi_{min}|^2 \quad & = (\phi_b^{'} - \phi_a^{'})^2 + (\phi_b^{'} - \phi_c^{'})^2 \\ & = 2(\phi_b^{'} - \phi_a^{'})^2 \\ & = 2(x - (-\alpha + \tfrac{x}{2}))^2 = 2(\alpha - \tfrac{x}{2})^2\end{aligned} \tag{7}$$

where ϕ_{min} is the minimum ϕ value of (6). Here, x is the solution we are looking for. We assumed that the change in magnitude of the ϕ value of point 'a' is twice those of point 'b' and 'c'. This assumption is supported by the fact that the noisy pixel 'a' has on both sides ϕ values that differ by 2α, whereas the pixels 'b' and 'c' have it only on one side. With this assumption, the ϕ value which makes the integrand in (6) smallest becomes

$$\phi_{min}(\mathbf{r}) = -\left(\alpha + \frac{\Lambda(u_0)}{4\beta}\right). \tag{8}$$

In our one-pixel noise example, denoising means that the sign of the current ϕ value has changed from a positive value to negative value. The condition for $\phi_{min}(\mathbf{r})$ to become a negative value is,

$$\begin{aligned}&\phi_{min}(\mathbf{r}) = -\left(\alpha + \tfrac{\Lambda(u_0)}{4\beta}\right) < 0 \\ &\Leftrightarrow \beta > -\tfrac{\Lambda(u_0)}{4\alpha}\end{aligned} \tag{9}$$

where it has to be kept in mind that $\Lambda(u_0)$ is negative in our example. The value of $\Lambda(u_0)$ is related to the amount of the noise in the image. It can be seen from (9) that if the noise becomes larger than β has to be larger to eliminate the noise. Intuitively, we see that

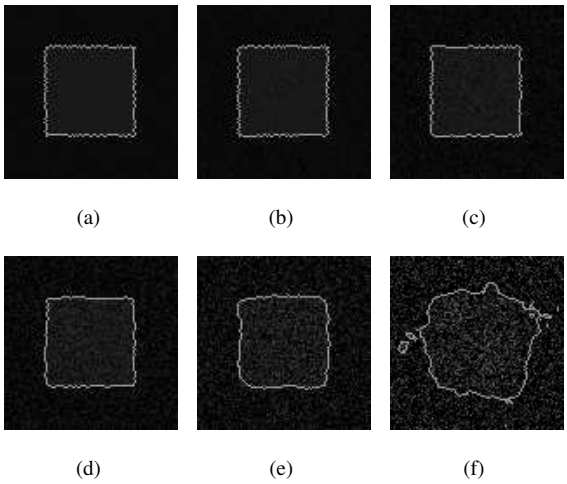


Figure 2: Result of automatically setting the β value, and applying the harmonic regularized bimodal segmentation on simulated images. (a) SNR =30 (b) SNR =25 (c) SNR = 20 (d) SNR = 15 (e) SNR = 10 (f) SNR = 5

$\Lambda(u_0)$ is somehow proportionally related to the variance of the noise. Therefore, after estimating the variance of the noise through some existing estimating method, we set the value of β to

$$\beta = 3\left(\frac{Var_{noise}}{4\alpha}\right). \tag{10}$$

Figure 2 shows the result of setting the β value as in (10) and then performing the harmonic regularized bimodal segmentation on simulated images with different SNR values. The intensity difference of the box in the middle and the outside is 8. It can be seen that the automatic setting of the β value results in good results when the SNR is large. However, as the SNR value decreases, the shape of the zero level set distorts much. Therefore, in the next section, we propose a method for eliminating the spurious regions due to the noise.

4 LEVEL SET BASED MOTION DETECTION WITH ELIMINATION OF SPURIOUS REGIONS

For a further elimination of spurious regions due to the noise, we can take extra elimination steps. These extra steps can also be directly applied on the level set function obtained by the bimodal segmentation without regularization term, to give good elimination results. The elimination of spurious regions is performed in the following three steps: First, calculate a mean curvature map from the processed level set function obtained by (3). Second, calculate a new difference map by normalizing the original difference map with respect to the mean curvature map, and taking the segmentation result in section 2 into account. Third, perform a regularized bimodal segmentation on the new difference map.

To calculate the mean curvature map, the level set function obtained from (3) is first binarized, where the binarized pixel values corresponding to the region $\{\mathbf{r}|\phi(\mathbf{r}) \geq 0\}$ are 1, and those corresponding to the region $\{\mathbf{r}|\phi(\mathbf{r}) < 0\}$ are 0. Then, regarding the binarized image as a two dimensional surface in the three dimensional space, with the binarized pixel value corresponding to the z axis, the mean curvature is computed at each pixel. The mean curvature value can be regarded as a measure that measures the amount by which the image structures deviate from being flat. As a result, small image structures like spurious regions due to the noise tend to have large mean curvature values. Therefore, if the region in the difference map corresponding to the motion region obtained by (3) is divided by the mean curvature map, then most of the spurious regions will have relatively small values in the new difference map. The new difference map is obtained by dividing the original difference map ($\mathbb{D}I$) by the mean curvature map $\mathbb{M}$ and normalizing it as follows:

$$\mathbb{D}I_{new}(\mathbf{r}) = \begin{cases} \frac{\mathbb{D}I(\mathbf{r})}{\mathbb{M}(\mathbf{r})\,\mathbb{D}I_{max}}, & \text{if } \phi(\mathbf{r}) \geq 0 \\ 0, & \text{if } \phi(\mathbf{r}) < 0 \text{ or } \mathbb{M}(\mathbf{r}) = 0, \end{cases} \tag{11}$$

where ϕ is the level set function obtained by (3), $\mathbb{M}(\mathbf{r})$ is the mean curvature value at the position $\mathbf{r}$, and $\mathbb{D}I_{max}$ is the maximum of all the $\mathbb{D}I_{new}(\mathbf{r})$ values.

Figure 5(h) shows the mean curvature map calculated from the binarized level set function, and Fig. 5(i) shows the corresponding normalized new difference map. The effect of dividing by the curvature map can be seen by comparing Fig. 5(f) and Fig. 5(i). The dense noise with relatively large intensity values in Fig. 5(f) appear to have become sparse in the new difference map, since small image structures have experienced a large decrease in their intensity values by the division through the mean curvature value. Therefore, if now a level set based bimodal segmentation with harmonic regularization is performed on the new difference map, the spurious regions due to the noise become entirely eliminated by the harmonic regularization.

The level set based bimodal segmentation with harmonic regularization term solves the following differential equation:

$$\frac{d\hat{\phi}}{dt} = \frac{1}{c_1+c_2}(\mathbb{D}I_{new} - c_2)^2 - \frac{1}{c_1+c_2}(\mathbb{D}I_{new} - c_1)^2 + \nabla^2\hat{\phi} \tag{12}$$

The equation is the same as (3), except for the harmonic regularization term ($\nabla^2\hat{\phi}$). The first and the second term in the righthand side are normalized with respect to $c_1 + c_2$ such that the harmonic regularization term can compete with these terms. The harmonic regularization term smooths out the level set function fast such that the sparse spurious regions due to the sparse noise

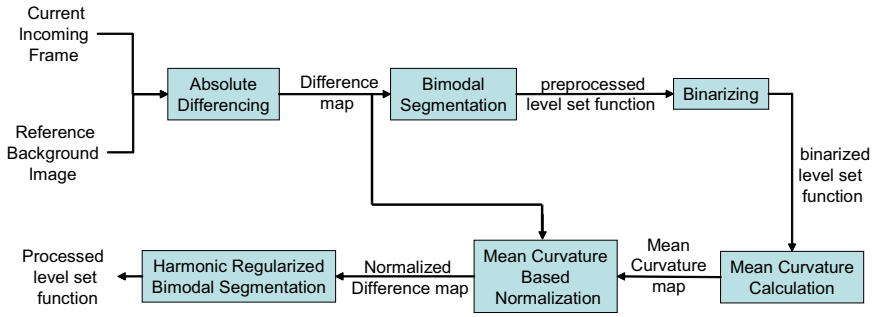


Figure 3: Overall block diagram of proposed method.

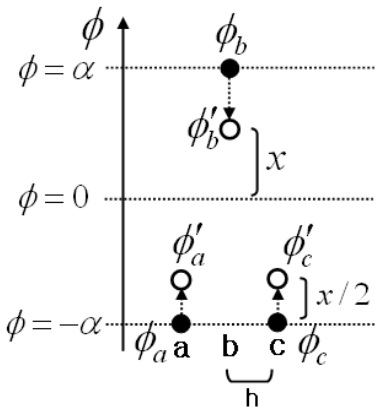


Figure 4: One-pixel noise model in 1-D

in the new difference map become entirely eliminated, while the valid motion regions survive due to the strong influence of the first two fidelity terms.

The segmentation result that results from the competition can be seen in Fig. 5(f), which shows the motion region represented by the region $\{\mathbf{r} \mid \hat{\phi}(\mathbf{r}) \geq 0\}$. The spurious regions have been effectively eliminated while motion regions are well preserved. For comparison, Fig. 5(g) illustrates how a simple application of a morphological filter can fail in the complete elimination of spurious regions. Figure 5(d) shows the result of using a threshold based method and Fig. 5(e) shows the result of the MOG (Mixture of Gaussian) method with threshold value 0.7 to segment the object. It can be seen that the a priori given threshold value can result in a loss of the object region or produce spurious regions. Figure 3 shows the overall block diagram of the proposed motion detection method. Figure 6 shows the object detection result of the proposed method. The proposed method works in real time on a 320×240 frame with 2.1GHz PC due to the fast bimodal segmentation algorithm, which takes just 3–5 iterations for the classification.

5 CONCLUSION

In this paper, we presented a motion detection scheme that sets its target especially for motion detection in low illuminance environment. It is shown that the proposed scheme can detect moving objects well even in very low illumination conditions and without the need of setting an a priori known threshold value and with the capability of eliminating spurious regions.

6 ACKNOWLEDGEMENT

This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD, Basic Research Promotion Fund) (KRF-2008-331-D00566).

REFERENCES

[1] P.L Rosin and T. Ellis. “Image difference threshold strategies and shadow detection,” In Proceedings of the 6th British Machine Vision Conference, pp. 347–356, BMVA Press, 1995.

[2] Osher, S. and Fedkiw, R., “Level Set Methods and Dynamic Implicit Surfaces,” Springer-Verlag, New York, (2002).

[3] Sethian, J.A., “Level Set Methods and Fast Marching Methods,” Second Edition, Cambridge University Press, Cambridge, UK, 1999.

[4] S.H. Lee and J.K. Seo, “Level Set-Based Bimodal Segmentation With Stationary Global Minimum,” IEEE Trans. on Image Processing, vol. 15, no. 9, pp. 2843–2852, Sep. 2006.

[5] Gibou, F. and Fedkiw, R., “A Fast Hybrid k-Means Level Set Algorithm for Segmentation,” 4th Annual Hawaii International Conference on Statistics and Mathematics, pp. 281–291, Stanford Technical Report, Nov. 2002.

[6] T.F. Chan and L.A. Vese, “Active contours without edges,” IEEE Trans. on Image Processing, vol. 10, no. 2, pp. 266–277, Feb. 2001.

[7] P.L Rosin and T. Ellis. “Image difference threshold strategies and shadow detection,” In Proceedings of the 6th British Machine Vision Conference, pp. 347–356, BMVA Press, 1995.

[8] G. Halevy and D. Weinshall, “Motion of disturbances: detection and tracking of multibody non-rigid motion”, Machine Vision and Applications, vol. 11, pp. 122-137, 1999.

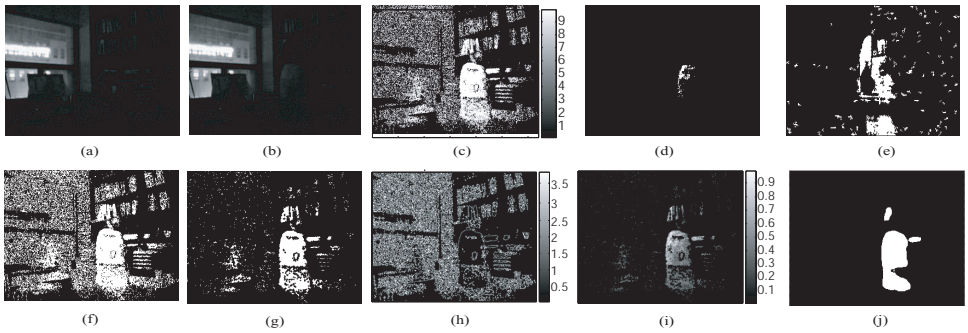


Figure 5: Segmentation result in low illumination environment: (a) reference background frame, (b) incoming current frame, (c) difference map, (d) result of thresholding the difference map (c) with threshold value=20, (e) result of MOG method (f) segmentation result using (3) in section 2, (g) result of eroding the image in (f) with morphological filter, (h) mean curvature map, (i) new difference map, (j) final segmentation result with proposed method.

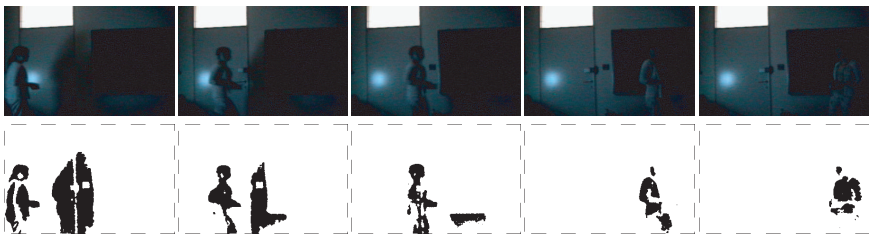


Figure 6: Object detection result with the proposed method.

[9] J. Heikkila and O. Silven, "A real-time system for monitoring of cyclists and pedestrians," Second IEEE Workshop on Visual Surveillance, pp. 74-81, July 1999.

[10] P. L. Rosin and E. Ioannidis, "Evaluation of global image thresholding for change detection," vol. 24, Pattern Recognotion Letters, pp. 2345-2356, 2003.

[11] E. Stringa,"Morphological change detection algorithms for surveillance applications," Proc.Brit.Machine Vision Conf., 2000.

[12] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-Time Tracking of the Human Body," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 19, pp. 780-785, 1997.

[13] I. Haritaoglu, D. Harwood, and L. S. Davis, "W^4: Who? When? Where? What? a real time system for detecting and tracking people", Third International Conference on Automatic Face and Gesture Recognition, pp. 222-227, April 1998,.

[14] C. Stauffer and W. E. L Grimson, "Adaptive background mixture models for real-time tracking," Computer Vision and Pattern Recognition, Fort Collins, Colorado Jan pp. 246-252, 1999.

Automatic Terrain Generation with a Sketching Interface

Anna Puig-Centelles
Universitat Jaume I,
Castellón, SPAIN
apuig@uji.es

Peter A. C. Varley
Universitat Jaume I,
Castellón, SPAIN
varley@uji.es

Oscar Ripolles
Universitat Jaume I,
Castellón, SPAIN
oripolle@uji.es

Miguel Chover
Universitat Jaume I,
Castellón, SPAIN
chover@uji.es

ABSTRACT

Virtual environments should offer the user a deep interactive experience with both large worlds to explore and a higher degree of perceived realism. The main goal of our work is to provide the final user with an easy-to-use accurate terrain generation application, which allows non-professional users to design their own desired terrain. In this paper we consider the creation of islands to be used in computer games. We introduce a simple terrain algorithm and we also consider its integration into a sketching application. The application will offer both a 2D and a 3D representation of the terrain, in order to simplify the interface and provide the user with more interactive feedback about the island that has been designed. Our framework offers real-time algorithms for both creating and modifying terrain features, thus improving the final results with more realism and greater customization by the user.

Keywords: Terrain Generation, Sketching Inteface, Islands, Game Environment.

1 INTRODUCTION

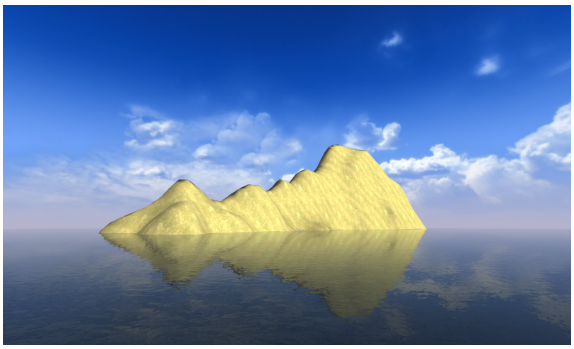


Figure 1: Example of a terrain obtained with our framework and imported into the Torque Game Engine.

In recent years, computer graphics have undergone an intense evolution as new graphics hardware offers a final image quality that was unimaginable just a few years ago. As a result, interactive graphics applications, such as computer games or virtual reality environments, now include more complex scenes offering very detailed environments. Terrain is therefore a key element that can lessen the sense of realism if it is not addressed correctly.

Terrain generation is a research area which has been active for many decades. The growing power of modern computers has made them capable of producing increasingly more realistic scenarios. Synthetic terrain generation is a process which creates elevation values throughout a two dimensional grid. The need for highly realistic scenarios often involves developing algorithms that can generate more realistic terrains with more user control over the final terrain that is created. Different terrain generation techniques that are capable of offering very realistic artificial terrains have been reported in the literature. Nevertheless, not many applications provide enough user control. On the contrary, those applications that provide user interaction are often too difficult to control.

Sketching is a tool that is well suited to the design of architectural elements and it provides the user with a considerable amount of control over the created elements. Research has produced prototype tools for interpreting sketches of abstract polyhedra [23, 11]. However, less work has focused on sketching the underlying terrain or extracting it from a photograph. Thus, buildings are often considered as the *foreground* and are taken into account properly, whereas terrain is seen as *background* which is often ignored.

Our aim is to develop convenient and simple ways to create computer models of terrain. In this paper we address the problem of creating models of islands for use in computer games. Our goal is similar to the idea given in [12], where the authors show that relatively simple algorithms can provide non-professional users with fast, successful results.

In this work we describe a terrain creation algorithm for islands based on *heightmaps*, which are regularly-spaced two-dimensional grids of height coordinates. The elevation of the terrain is automatically calculated from the coastline sketched by the user, who can also create hills and apply perturbations in order to achieve a more realistic and irregular terrain. Once again, it is

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

important to mention that our aim is to provide the user with more control over the terrain that is generated.

Furthermore, we also consider the integration of our algorithm into a sketching application. This application combines a 2D representation window and a 3D displaying window. First, the user creates a silhouette of the island in the 2D window and then, the user will be able to modify the terrain appearance in the 2D and the 3D windows. The terrain thus obtained will be outputted as a heightmap that may then be imported into a game engine.

This work is organized as follows. Section 2 contains the state of the art in terrain generation and sketching freeform surfaces. Section 3 describes our terrain generation algorithm. After that, Section 4 analyzes our sketching application. Later, Section 5 depicts our results and discusses a usability test. Finally, Section 6 presents our conclusions and future work.

2 RELATED WORK

In this section we analyze the different approaches that currently exist for terrain generation. After that we will take into account the different software tools which are available for creating artificial terrain. Finally, we will give some basic ideas on sketching and its application to our purposes.

2.1 Terrain Generation

The literature offers a wealth of research on synthetic terrain generation. Techniques can be grouped into three different categories:

Procedural Approaches. This category includes methods in which the terrain is generated automatically. These methods can be further separated into fractal techniques and physically-based techniques.

The most popular procedural approach is fractal-based terrain generation, which is efficient but difficult for users to control. It is possible to find a review of recent fractal approaches in [2].

Physically-based techniques simulate the effects of physical processes such as erosion by streams [8] or wind [24]. A recent technique that combines a non-expensive fluid simulation with an erosion algorithm is presented in [1]. It also supports effects like dissolving or sedimentation of material in the process of erosion.

Fractal landscape terrain generation and physical erosion simulation are both approaches that add terrain details through procedural refinement. Nevertheless, modifying their parameters to obtain a desired terrain may be a painstaking task.

Another proposal appears in [15], where the authors provide an alternative method for terrain generation that employs a two-pass genetic algorithm approach to produce a variety of terrain types using only intuitive user inputs. The process is efficient but very difficult for a user to control.

Real Terrain Information. This approach groups the techniques from the **Geographic Information Systems** (GIS), where elevation data come from real-world measurements [25]. Similarly, another source of information could be the study of the extraction of terrain from photographs [3]. All these approaches have the advantage of offering highly realistic terrains in very little time, but with little user control.

User Defined Approaches. This is the most flexible type of technique, in which a human artist creates the terrain manually, using an image editing program, 3D modeling software, specialized terrain editor programs or the editors that are included in game engines.

The authors of [10] allow the user to control the terrain generation process by receiving as input an image generated by an image editing program in order to perform the terrain generation. In [26], patches from sample terrain (represented as a height field) are used to generate new terrain and the synthesis is guided by a user-sketched feature map that specifies where terrain features occur in the resulting synthetic terrain. Later, [22] introduced a simple interface for sketching heightmaps of islands. This application was very simple but, although it offered a good amount of user control, it was difficult to use and the obtained terrain was not completely customizable.

2.2 Terrain Software

In this section we introduce some terrain tools for simulating artificial environments. There is a wide range of software available. In Terragen [19] and Terraineer [20], the user sets parameters and the program creates a pseudo-random landscape which meets those parameters. Terraineer offers the possibility of experimenting with different height generation algorithms. World Machine [17] additionally includes modeling of physical weathering processes. In all of these programs, terrain is modeled and imported/exported as a heightmap.

Further user interaction is offered in Nem's Mega 3D Terrain Generator [5], where the user is initially presented with a flat piece of terrain and has various options for modifying it. L3DT [21] is another software that generates artificial heightfields and exports its data to multiple formats.

2.3 Sketching

There have been some recent developments on the automated interpretation of freeform surfaces from sketches, but they either interpret the drawing as being that of a single solid object [7] or leave the freeform surface floating in mid-air as a patch, without continuing to the horizon in the manner of a landscape [18, 6].

However, interpreting a terrain sketch as a floating free-form patch presents some problems. Qin et al [18]

require the user to draw a grid of quadrilaterals to represent the surface, and a trained neural network is also needed to interpret it. In Kaplan and Cohen's approach [6] the boundary of the patch must be clear in the original input, either by virtue of its obvious contrast with the background or by being specified by the user. Their approach also requires user intervention to resolve ambiguities. The problem of boundary constraints can be overcome if we restrict ourselves to sketching islands. The boundary constraints for islands are simple: an island has a coastline, that is, a continuous bounding curve.

More recently, a system for designing freeform surfaces with a collection of 3D curves has been proposed in [13]. They create objects within an easy interface, which is based on drawing simple lines. By using a similar simple sketching interface, [27] introduce an over-sketching application for feature-preserving surface mesh editing. This application allows simple yet realistic mesh deformations to be obtained.

3 OUR TERRAIN GENERATION

The method that we present here for generating islands is based on the use of heightmaps and allows users to define and modify the coastline of the island. Furthermore, they can also create and reshape any number of hills, which will interact each other and with the existing terrain. Finally, we offer the possibility of applying filters to give the final terrain a more realistic appearance.

3.1 Reshaping the Coastline

The user can draw the silhouette of the island freely, but it will be necessary to delineate a continuous curve by sketching a closed shape, as shown in Figure 8. The shape of the coastline can be changed by redrawing, starting and ending at points near the existing coastline and drawing a continuous curve in any direction between those two points. Thus, it is possible to apply different operations in order to modify the existing coastline.

The user may decide to cut a piece of the island off. As a consequence, the terrain will be split into two areas. Depending on the direction of the cut, the algorithm will decide which one of these areas is to be rejected. The direction of the cut is understood as being the direction running from the initial to the finish point. The rejected area will be the one on the left-hand side of the cut. Figure 2 presents an initial coastline and the silhouettes obtained after performing cuts with the same start and finish points but following different directions.

Furthermore, the user can also add new pieces to the existing area. Again, the algorithm will behave differently depending on the direction of the sketched drawing. If the line has been sketched clockwise, the new area will be added to the existing one. In contrast, if the line has been performed in an counterclockwise direction, then this new area will be maintained and the old one will be rejected. In Figure 3 we can see an example of these possible ways of modifying the area by adding or subtracting a piece of terrain.

We must note that the algorithm will differentiate between cut and supplement operations by testing whether the line goes through the terrain area or not. In those cases in which a line is used to perform more than one operation, each point where the line intersects the silhouette will be interpreted as the finish and start points of the consecutive operations. In Figure 4 we depict an area that is being modified by two consecutive operations: the first one consists in an external clockwise supplement and the second one is a curved internal cut that is rejecting the piece on its left-hand side.

3.2 Updating the Terrain Height

Every time the coastline silhouette is modified, the terrain algorithm has to react adequately to those changes and recalculate the height of the terrain in order to offer a smooth continuous surface.

Since we are simulating the terrain of an island, we must take into account the level of the sea. We have to ensure that every single point within the sketched coastline is above sea level. As a consequence, when the coastline changes, it may be necessary to modify the elevations of some onshore points. Ideally, points close to parts of the old coastline which remain unchanged should also remain unchanged, but points close to parts of the new coastline should be elevated above sea level regardless of their previous height. Therefore, if we reshape the coastline then we have to check whether all the points contained inside the island have the appropriate height.

In order to obtain the new height values, we take into account the distance from each point to the nearest piece of new coastline D_n and to the nearest piece of old coastline D_o, both scaled to the range 0 to 1.

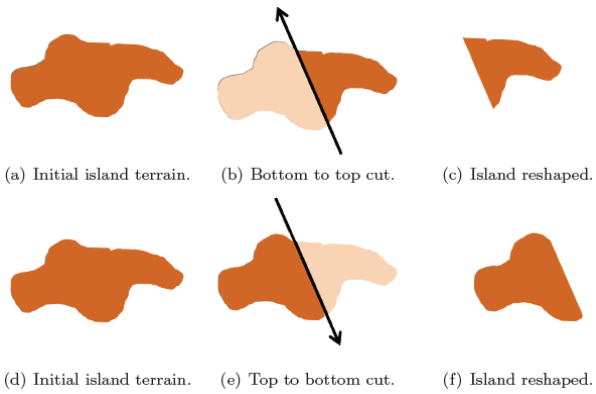
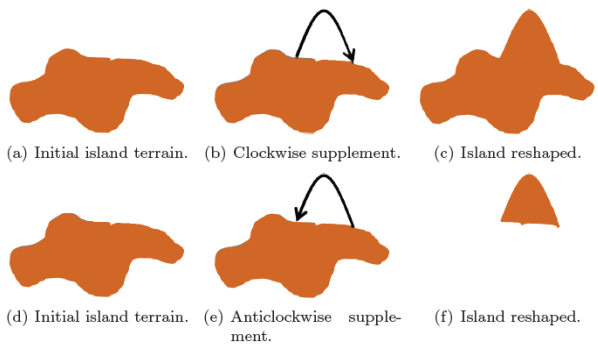


Figure 2: Cutting and reshaping the island.

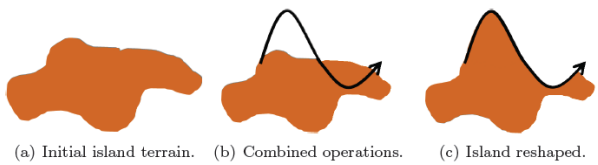
 ISBN 978-80-86943-94-7



(a) Initial island terrain. (b) Clockwise supplement. (c) Island reshaped.

(d) Initial island terrain. (e) Anticlockwise supplement. (f) Island reshaped.

Figure 3: Supplementing and reshaping the island.



(a) Initial island terrain. (b) Combined operations. (c) Island reshaped.

Figure 4: Two consecutive operations.

The height of each onshore point H_i is calculated as a weighted value between the old height H_o and the new one H_n, the latter being proportional to D_n. We implement the calculation of the heights with Equation 1,where the weight W is calculated by Equation 2.

$$H_i = H_o(1.0 - W) + H_n W \quad (1)$$

$$W = 0.5 + 0.5tanh((D_o)^2 - D_n) \quad (2)$$

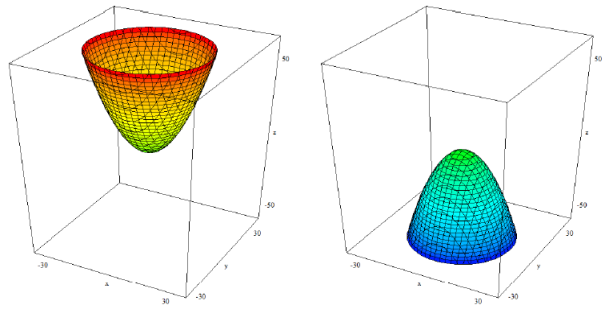
The hyperbolic tangent ($tanh$) function is chosen because it has the appropriate shape, which is close to -1 for points near the old coastline and close to 1 for points near the new coastline. Moreover, it never goes outside this range. D_0 is squared so that points close to neither coastline are treated as being closer to the old coastline rather than to the new one.

Whether a pixel is onshore or not is assessed by referring to a silhouette of the island which is recalculated after each change to the coastline by drawing the coastline on a blank array of pixels and using a flood-fill routine (starting from a point clearly outside the coastline) to distinguish sea from land.

3.3 Generating Hills

In previously presented methods for sketching islands [22], the orography was difficult to define. In our work we want to improve on the user definition of terrain. The idea is to allow the user to create multiple hills having the desired radii, height and location on the terrain.

In our algorithm we define hills as *elliptic paraboloids*. An elliptic paraboloid is shaped like an oval cup and can have a maximum or minimum point. In a suitable coordinate system, it can be represented by the equation:



(a) Elliptic paraboloid upwards. (b) Elliptic paraboloid downwards.

Figure 5: Example view of elliptic paraboloids.

$$\frac{z}{c} = \frac{x^2}{a^2} + \frac{y^2}{b^2} \quad (3)$$

considering that the *elliptic paraboloid* is centered on $0,0,0$ with radius a,b (along the x and y axes), being $a,b \in \Re$ and $a > b$.

Equation 3 represents an elliptical paraboloid which opens upwards and can be seen in Figure 5a. This quadratic surface will be used in our algorithm to define the hills. It is important to note that we allow the user to define valleys by means of elliptical paraboloids which open downwards, as can be seen in Figure 5b.

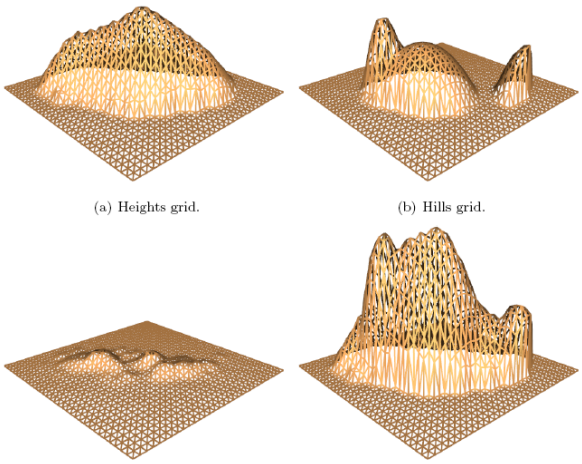
With this equation the user can introduce the central point, the radius and the height of each hill. Once we have this information, our algorithm will be able of calculate the height of each point affected by the hill. All those points are obtained with the central point and the radius that have been defined. The height of each single point will be modified by following Equation 3. We will add the new height to the previous one in order to obtain more realistic and integrated terrains. By so doing, we allow for the creation of valleys and volcanos.

3.4 Filtering the Terrain

In order to obtain a better appearance for the terrain being designed, we can introduce some fuzzy bumps to deform the regular surface. We have implemented a filter to introduce 'noise' into the previously defined rounded terrain. This filter can be applied as many times as the user desires and it will give us a number of perturbations that are proportional to the surface area of the island. These perturbations will also have an elliptical paraboloid shape, but they will be wider than taller and they will be produced upwards or downwards in a random manner.

3.5 Integrating the Processes

The final terrain that is visualized comprises three heightmaps that will be added one after the other.



(a) Heights grid. (b) Hills grid.

(c) Filters grid. (d) Final grid.

Figure 6: Sample composition of the *Final Grid* of heights by adding the previously updated grids.

These heightmaps are stored as grids (2D matrix) of floats:

- *Heights Grid*, which contains the heights of the terrain obtained after modifying the coastline.
- *Hills Grid*, which stores the increments or decrements in height, due to the hill volumes.
- *Filters Grid*, which holds the variations introduced by the filters that have been inserted.

As a consequence, with those three data structures we obtain the *Final Grid*, which stores the sum of the three previous ones. We assume that the *Heights Grid* defines the basic features of the terrain. Then, the other data structures will add more details. It is important to comment that the division of the terrain information into those three data structures simplifies the updating process of any of them. This way, for example, adding a hill only involves modifying the *Hills* grid.

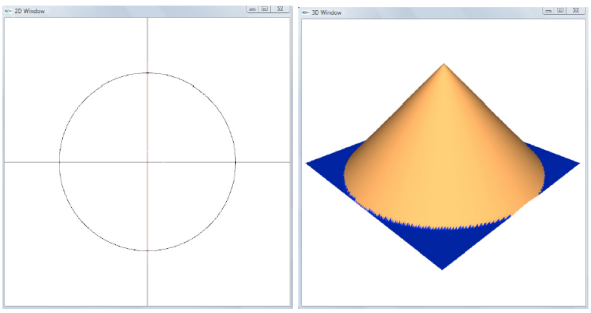
In Figure 6 we can see an example of the different grids that compose the *Final Grid*. The *Heights Grid* is obtained after defining the coastline of the island. The *Hills Grid* contains three different hills. Lastly, the *Filters Grid* stores the perturbations introduced by the user. Consequently, these three grids combined together give form to the final terrain.

This representation uses an internal triangle mesh for representing the 3D island. The 2D heightmap is properly linked with this 3D representation in order to allow fast and efficient updates. The implementation has been optimized to assure that each modeling operation entails to update the minimum amount of information, including both the heightmap and the vertices information: spatial coordinates, normals, colors, etc.

4 USER INTERFACE

This section describes our sketching application for terrain generation by using the ideas presented earlier. In paper [22], a simple interface for sketching heightmaps of islands was presented. This interface was close to the ideal of a modeless single-tool interface, with all of its major operations being controlled by a single device (pen or single-button mouse). A problem that appears in some of the most advanced sketching applications, like [16], is that they require a multi-modal push-button interface. Our intention is to maintain the original sketching objectives in order to keep our application as simple and natural to use as possible. Nevertheless, it has been necessary to develop a two-button mouse software application to integrate all the functionalities presented in the previous section.

Our proposed framework offers the user an interactive sketching application. This solution consists of two windows. The 2D window depicts the silhouette of the coastline of the island, as seen in Figure 7a. The 3D window represents the volumetric view of the whole island, as seen in Figure 7b. This 3D view presents a smooth surface which is automatically constructed with the information stored in the heightmap. When the implementation starts, the 2D window contains a circular coastline, as shown in Figure 7a. The 3D window, shown in Figure 7b, depicts a conical island, which is the initial terrain that the user will be able to modify.



(a) 3D window. (b) 2D window.

Figure 7: 2D and 3D Window on Startup.

In the following subsections we will provide a detailed description of the interaction with the aforementioned windows in our application. In Figure 8 we describe a step-by-step design of an island using our framework.

4.1 2D Modeling Operations

The 2D window allows the user to perform two basic sketching operations: defining the silhouette of the island and adding and modifying hills.

When interacting with the left mouse button in this window, the user is allowed to design the coastline of the island. It is possible to draw a free-form silhouette

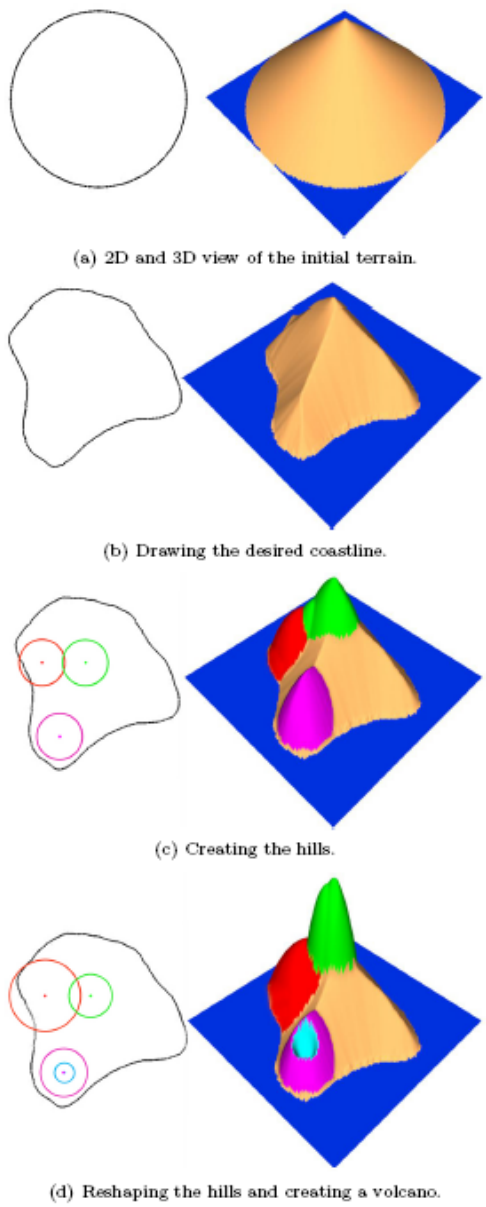


Figure 8: Designing a sample island in four steps.

interactively and the system will simultaneously update the terrain. As we have mentioned in previous sections, it is possible to cut and extend the existing terrain by defining lines that start and end on the coastline. Figure 8 shows how we could draw any irregular shape to delimit the terrain of our island. Our system includes an additional feature, which has proved popular with users: when a change is made to the coastline, the old coastline gradually fades away, taking about two seconds to do so. Pressing the right mouse button during this period removes the new coastline and recovers the old coastline.

In our application, when the user clicks with the right button either inside or outside the coastline, the application understands that the user is defining the central point of a hill. Then a colored circular line will appear on the terrain surface surrounding the central point that has just been created. This line represents the area influenced by that particular hill. The user may add as many hills as desired and each one will be depicted in a different color. It is important to note that the hills which are located outside the coastline affect the terrain in those areas where the hills overlap the already existing coastline. After defining the hills, the user will be able to modify the radii and the location of the hills inside the island. Right-clicking on the center point of the hill and dragging, allows the user to change the position of the hill. The user can eliminate a hill by dragging it outside the island until its radius is completely outside the coastline. Alternatively, right-clicking and dragging on the circular line allows the user to modify the hill radius.

The application includes the possibility of *zooming* in on the sketched island by clicking the right button of the mouse. Using the zoom can help the user to get a better overview of the terrain. We have to click outside the island in order to zoom, but always away from the coastline. This is because if the user clicks too close to the coastline, the application will interpret that the user wants to create or modify a hill.

4.2 3D Modeling Operations

In the initial version of the application, the height of the island was defined by the *cross section* [22] selected in the Plan Window and the input given in the Elevation Window. This interface was complicated to use and it did not give the user full control.

In our application, the 3D window shows a volumetric view of the terrain. The user will be able to click with the right button on any of the previously defined hills and can decide on the height of each hill by dragging the mouse up and down. If we drag upwards, then the height will be positive and we will create a hill, and if we drag downwards the terrain will be a valley. Furthermore, by dragging the mouse left and right, the user will be able to decrease and increase the size of the radius of the selected mountain.

The 3D window also allows for the use of filters, which the user can decide to apply to the whole terrain in order to introduce some fuzzy bumps. Clicking with the left mouse button on any point on the island and dragging upwards will add filters to the terrain. The more we drag upwards, the more bumps are created. On the contrary, if we drag downwards then the application will understand that we want to decrease the number of perturbations.

In addition, the 3D window offers two more functions. Clicking with the left button away from any hill and dragging, acts as a rotating function. Also, clicking with the right button away from the terrain and dragging, acts as a zoom function.

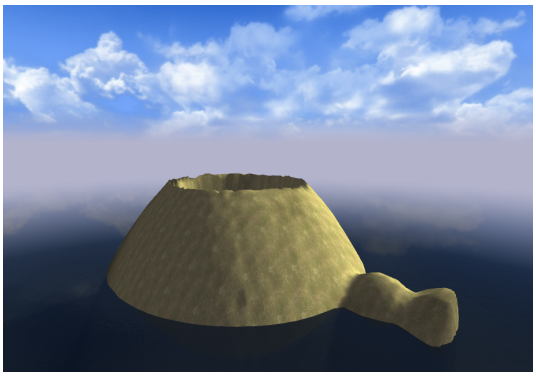


Figure 9: Examples of volcanic island in TGEA.

5 RESULTS

This section presents our results using a Pentium D 2.8 GHz processor with 2 GB. RAM and an nVidia GeForce 8800 GT graphics card. The framework was implemented in C++ with OpenGL.

In order to show the possibilities of our framework, we exported different heightmaps obtained from our terrain generation algorithm. Then we introduced those heightmaps as input to a game engine and we obtained several examples of islands. The game engine that we selected in our tests is the Torque Game Engine Advanced (TGEA) [4].

Figure 1 depicts an elongated island. The island is quite abrupt after having applied some filters in order to give the final terrain a more realistic appearance. In Figure 9 we have rendered a volcano with two tiny hills on one side. The volcano consists of a big hill but with a hole in the middle. This hole is performed by applying a slightly smaller negative-height hill, located in the middle of the first hill we created.

5.1 User Study

We consider important to perform a user study in order to evaluate the quality of our sketching application. We gave first-time users some basic indications in order to make them know how our application works. After that, we asked them to try to draw an island silhouette with the desired hills. They played with different coastline, hills and filters until they outlined the desired appearance of island and terrain.

Our informal user study considered the application from different perspectives. First, we ask our volunteers to grade from 1 to 10 the overall quality perception of the application, being 1 the worst and 10 the best. Secondly, we monitored their activity to record the time passed until they obtained a satisfactory terrain. Finally, we asked them to report any difficulty or mistake they could find in using our application.

Different studies have proven that 5 users are enough to assess the quality of software applications [14], although evaluating visualization results need a different amount of volunteers in order to obtain valuable results [9]. In our case we have conducted the test with 30 people. We have also grouped them in three groups depending on their expertise both in computer use in general and in computer design in particular.

In Table 1 we present the results obtained with university volunteers. Most of them were satisfied with their results after less than five minutes. It is important to mention that both computer scientist and designers found several problems that helped us to improve the application. Most of them found the application difficult to use at the beginning, although after some practice they started modeling their terrains. These initial usability problems helped us to modify some aspects of the application and also encouraged us to create a brief guide to explain how the application works. Moreover, most users found the application *funny* and played with it after acquiring a little expertise. Finally, we also encouraged some of them to include the terrain inside the Torque Game Engine [4], in order to give them the possibility to experience gaming over their modeled island and terrain.

6 CONCLUSIONS

This paper presents a method for terrain generation which is suitable for users who wish to have full control over the whole creation process. We have also presented a simple tool for creating solid models of imaginary islands. The tool is easy to use and requires only a minimal user interface, with all of its major operations being controlled by a two-button mouse. From this application, the user can add, remove and reshape existing hills interactively and the terrain will be updated accordingly. Moreover, the user is able to modify the silhouette of the island and add fuzzy bumps as desired.

With the images of islands that we have shown in the previous section, it can be seen how our approach is capable of offering very realistic terrains. The user can decide on the final appearance of the island, as it is possible to apply as any number of filters. Nevertheless, the user could choose not to apply filters in order to obtain a fairly rounded terrain which could be useful for a cartoon-like environment. The usability study, which was performed among persons with different computer skills, showed that the user interface we finally selected is comfortable and adequate in most cases.

Future lines of work on this application include the possibility of adding more tools at the expense of losing simplicity. In this sense, the authors are interested in allowing the user to include weather phenomena or vegetation and other decorative elements on their island.

ACKNOWLEDGEMENTS

This work was supported by the Spanish Ministry of Science and Technology with grant TSI-2004-02940,

Study Group	Number of volunteers	Overall Satisfaction	Avg. Required Time (min.)	Observed Usability Problems
Computer Scientists	12	8	3	12
Designers	7	6	5	7
Other Disciplines	11	9	4	2

Table 1: Results obtained with 30 university volunteers.

project TIN2007-68066-C04-02 and through the Ramon y Cajal programme. Also by Bancaja with project P1 1B2007-56.

REFERENCES

[1] M. Wacker B. Neidhold and O. Deussen. Interactive physically based fluid and erosion simulation. In *Eurographics Workshop on Natural Phenomena*, 2006.

[2] C. Dachsbacher. Interactive terrain rendering: Towards realism with procedural models and graphics hardware. Technical report, Fiedrich Alexander Universität, Germany. Thesis, 2006.

[3] S. Fefilatyev, V. Smarodzinava, L. O. Hall, and D. B. Goldgof. Horizon detection using machine learning techniques. *ICMLA*, 0:17–21, 2006.

[4] Garage Games. Torque game engine advanced. http://www.garagegames.com/, 2008.

[5] R. Gregg. Nem's 3d mega terrain generator. http://nemesis.thewavelength.net/, 2005.

[6] M. Kaplan and E. Cohen. Producing models from line drawings of curved surfaces. In *Eurographics Workshop on Sketch Based Interfaces and Modeling*, 2006.

[7] L.B. Kara and K. Shimada. Sketch-based design of 3d geometry. In *Eurographics Workshop on Sketch-Based Modelling*, pages 59–66, 2006.

[8] A. D. Kelley, M. C. Malin, and G. M. Nielson. Terrain simulation using a model of stream erosion. In *SIGGRAPH '88*, pages 263–268, 1988.

[9] R. Kosara, C. Healey, V. Interrante, D. Laidlaw, and C. Ware. User studies: Why, how, and when? *IEEE Comput. Graph. Appl.*, 23(4):20–25, 2003.

[10] Q. Li, G. Wang, F. Zhou, X. Tang, and K. Yang. Example-based realistic terrain generation. In *ICAT*, pages 811–818, 2006.

[11] M. Masry and H. Lipson. A sketch-based interface for iterative design and analysis of 3d objects. In *ACM SIGGRAPH 2007 courses*, page 31, 2007.

[12] Y. Mori and T. Igarashi. Plushie: an interactive design system for plush toys. In *SIGGRAPH 2007*, page 45, 2007.

[13] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Fibermesh: designing freeform surfaces with 3d curves. *ACM Trans. Graph.*, 26(3), 2007.

[14] J. Nielsen and T. K. Landauer. A mathematical model of the finding of usability problems. In *CHI '93: Proceedings of the INTERACT '93*, pages 206–213, 1993.

[15] T. J. Ong, R. Saunders, J. Keyser, and J. J. Leggett. Terrain generation using genetic algorithms. In *GECCO '05*, pages 1463–1470. ACM, 2005.

[16] S. Owada, F. Nielsen, K. Nakazawa, and T. Igarashi. A sketching interface for modeling the internal structures of 3d shapes. In *SIGGRAPH 2007 courses*, page 38. ACM, 2007.

[17] S. Schmitt. World machine. http://www.world-machine.com/about.html, 2006.

[18] D.K. Wright S. Lim U. Khan S.F. Qin, G. Sun and C. Mao. 2d sketch based recognition of 3d freeform shape by using the rbf neural network. In *Eurographics Workshop on Sketch Based Interfaces and Modeling*, 2005.

[19] Planetside Software. Terragen. http://www.planetside.co.uk/terragen/.

[20] K. Stachowski. Terraineer. http://terraineer.sourceforge.net/, 2006.

[21] A. Torpy. L3dt. http://www.bundysoft.com/L3DT/, 2008.

[22] P. A. C. Varley, M. Chover, and A. Puig-Centelles. Sketching islands for a game environment. In *5th Europ. Conf. on Visual Media Production*, 2008.

[23] P.A.C. Varley. Automatic creation of boundary-representation models from single line drawings. PhD Thesis, University of Wales, 2003.

[24] Kansas State University Wind Erosion Research Unit. Weru. wind erosion simulation models. http://www.weru.ksu.edu/weps.html, 2003.

[25] M. F. Worboys. *GIS: A Computing Perspective*. Taylor and Francis, 1995.

[26] H. Zhou, J. Sun, G. Turk, and J. M. Rehg. Terrain synthesis from digital elevation models. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):834–848, 2007.

[27] J. Zimmermann, A. Nealen, and M. Alexa. Sketching contours. *Computers & Graphics*, In Press, Accepted Manuscript, 2008.

Error-bounded GPU-supported terrain visualisation

Falko Löffler
University of Rostock, Germany
Albert-Einstein-Strasse 21
18055, Rostock
falko.loeffler@uni-rostock.de

Stefan Rybacki
University of Rostock, Germany
Albert-Einstein-Strasse 21
18055, Rostock
stefan.rybacki@uni-rostock.de

Heidrun Schumann
University of Rostock, Germany
Albert-Einstein-Strasse 21
18055, Rostock
schumann@informatik.uni-rostock.de

ABSTRACT

The interactive visualisation of digital terrain datasets deals with their interrelated issues: quality, time and resources. In this paper a GPU-supported rendering technique is introduced, which finds a tradeoff between these issues. For this we use the projective grid method as the foundation. Even though the method is simple and powerful, its most significant problem is the loss of relevant features. Our contribution is a definition of a view-dependent grid distribution on the view-plane and an error-bounded rendering. This leads to a better approximation of the original terrain surface compared to previous GPU-based approaches. A higher quality is achieved with respect to the grid resolution. Furthermore the combination with an error metric and ray casting enables us to render a terrain representation within a given error threshold. Hence, high quality interactive terrain rendering is guaranteed, without expensive preprocessing.

Keywords: GPU-Rendering, terrain rendering, projective grid, level of detail

1 INTRODUCTION

The interactive visualisation of digital terrain datasets is a complex and challenging problem. Usually highly accurate terrain datasets contain billions of elevation and colour values, a data volume that cannot be displayed in real-time. View-dependent approximation of the terrain is needed to achieve interactive rendering.

In general, interactive terrain rendering has to address three interrelated issues:

- quality of the final image,
- restrictions regarding available resources, and
- the real-time capability of the algorithm.

The approximation of terrain data with respect to these criteria and for a given application context is a current research challenge. The problem can be characterised as follows: Usually we seek high quality. This can be accomplished either by spending more time on rendering or by storing pre-calculated results. On the other hand, we have to keep an eye on the resources used. Using fewer resources either leads to lower quality or might require to forego the real-time capability. Hence, changes with respect to one criterion necessarily affect the other criteria. The challenge is to find a good compromise between quality, used resources, and rendering time.

There has been extensive research on terrain visualisation. Today's algorithms can be categorised based on their utilisation of graphics hardware into *CPU-based* and *GPU-based* algorithms. CPU-based approaches focus on high quality and as such spend much time on complex calculations on the CPU. To achieve real-time capability they use pre-computed data structures that consume additional resources. However, the communication between CPU and GPU is often a transportation bottleneck that usually leads to lower frame rates. Moreover and inversely, most CPU-based terrain rendering algorithms use advanced error metrics, which directly affects rendering quality and time.

GPU-based algorithms, on the other hand, focus on real-time rendering by exploiting the parallel architecture of the graphics hardware. The idea is to perform many rather simple operations instead of a few complex ones to achieve high performance. This is possible through programmable vertex- and fragment processors of current graphics hardware. Even though GPU-based algorithms do not take the topology of the terrain into account, they can produce high-quality images due to the high primitive throughput. However, GPU-based algorithms usually cannot guarantee an approximation within a freely adjustable error rate.

All these terrain rendering approaches are powerful and well-designed. But some problems still exist in particular scenarios. For instance, CPU-based algorithms are not suited for resource-limited environments or for applications where the terrain is subject to modification during runtime. Vice versa, GPU-based algorithms are not the best choice in cases where a representation within a given error threshold is required.

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

Our approach focuses on a compromise between the competing needs for high quality, low resource consumption, and real-time capability. We have developed an algorithm that avoids expensive pre-processing, ensures real-time rendering, and achieves high quality within a guaranteed error threshold. This is particularly useful in aerospace systems, where resources are limited and a high quality visualisation is strictly required. Other applications like games can also benefit from our approach if they make use of dynamic terrain.

As the basis for our approach we use the *projective grid method* [13]. Johanson employs this method for real-time rendering of water surfaces that are modelled as dynamic height fields. The algorithm is easily portable to the GPU and suitable for very large terrain datasets using cliptextures [20]. Even though the algorithm can be applied for direct high-quality view-dependent rendering of height fields in real-time without any pre-processing, some problems can be observed. For instance, while navigating a height field, visual artifacts are recognisable. These are due to inadequate sampling and filtering of the height field. They are also caused by not taking the height field's topology into account.

In our approach, we reduce these visual artifacts to achieve high quality while maintaining real-time capabilities. Furthermore, we guarantee the terrain representation's quality within a given error threshold. To achieve real-time rendering, we employ a view-dependent sampling of the height field that results in a view-dependent level of detail (LOD) representation of the terrain. We use a GPU-tailored grid resolution for the sampling to fully exploit the power of the GPU. This leads to higher quality during rendering. Additionally, we generate an error map that gives us error boundaries for each elevation sample. In turn, the error map is used to control an adaptive ray casting that is applied to those regions of the image whose errors exceed a desired threshold. This way, the rendering quality can be guaranteed to be always better or equal to the given error threshold.

In the remainder of the paper we explain in detail how we achieve this good compromise between rendering quality, used resources and real-time capability. In the next section, we discuss approaches related to our work. In Section 3, we introduce the projective-grid method and discuss its major problems when applied to terrain rendering. Based on this discussion, we present and evaluate our own approach in Section 4. Section 5 is dedicated to the discussion of results.

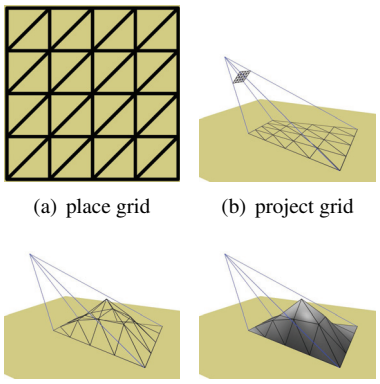
2 RELATED WORK

Terrain rendering algorithms can be categorised into *CPU-based* and *GPU-based* approaches.

CPU-based approaches construct, manage and select a proper approximation of the terrain data set using the CPU and the RAM. This allows utilising complex data structures and operations to construct terrain geometry. The composed geometry is then sent to the graphics hardware for rendering, which is often a bottleneck. The geometry of digital terrain data sets is usually described by triangles, which are directly supported by graphics hardware. Assembling a triangle mesh with regard to a sufficient triangle count leads to a good approximation of the terrain, provided that a proper triangulation algorithm is used. However, such meshes must be reassembled each frame to get a suitable view-dependent refinement of the original terrain data set. For example, [7, 24] apply a delaunay triangulation to limit the triangle count. This is also useful to improve the refinement and simplification of the terrain mesh and to reduce temporal aliasing. Whereas some approaches like [11, 12] do not constrain the triangulation process, other do so to generate and display hierarchies with multiple levels of detail. Many approaches use a regular network or quad-tree decompositions resulting in specialised and limited level-of-detail hierarchies. [17, 10] use binary trees to efficiently traverse and store the triangle hierarchy. Quad-tree triangulation is preferred by [2, 22]. The subdivision scheme from [19] subdivides the longest edge of a triangle to refine the terrain mesh. All these approaches extract a mesh on each frame, which restricts geometry caching and makes it difficult to utilise specialised techniques for efficient rendering. To solve this problem, [16, 22, 23] aggregate triangles to patches of different resolutions. At rendering time, patches of suitable resolutions are chosen to be combined and sent to the GPU. Hence, using patches accelerates the communication between CPU and GPU, but does not solve this problem entirely. Algorithms like [3, 4, 5, 26] store the patches in the graphics hardware's video memory. This significantly reduces data transmissions between CPU and GPU and hence increases rendering speed.

GPU-based approaches delegate the geometry processing to the GPU. These algorithms perform many simple operations rather than a few complex ones to achieve high performance through the parallel architecture of the GPU. In [1, 6, 9, 15, 14, 21] approaches are presented that can be implemented on today's programmable GPUs. A progressive geometry transmission is applied in [26] to reduce CPU to GPU communication. Warping and resampling of the underlying grid according to the viewpoint is done in [8]. This approach also adds procedural detail after resampling. Most GPU-based approaches use static levels of detail: the stitching of different resolutions is a common problem.

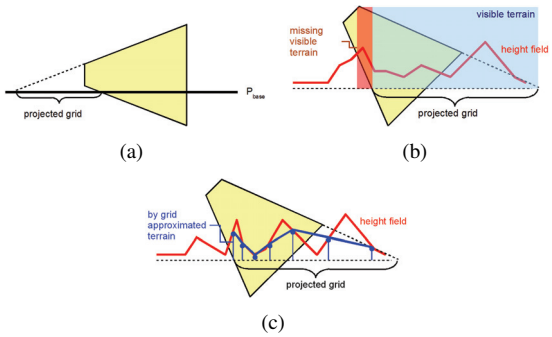
Another alternative for height field visualisation is the projective grid method. The method was first introduced by Johanson in [13] and was later applied to dynamic height field visualisation. Livny applied



(a) place grid (b) project grid

(c) displace grid points (d) use grid for rendering

Figure 1: Steps of the projective grid method



(a) (b)

(c)

a) backfiring projection when looking above the horizon
b) intersection of terrain data peaks with view frustum
c) undersampled terrain and resulting grid

Figure 2: Visual artifacts caused by the projective grid method

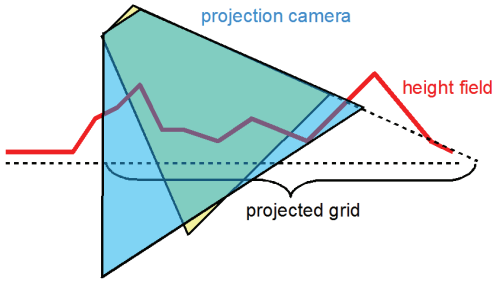


Figure 3: Projection camera with increased field of view to solve backfiring and intersected terrain

the approach to terrain rendering and combined it with clipmaps (see [27]) to support very large terrain datasets [20]. Instead of handling the geometry on the CPU, the grid is cached on the GPU and the programmable hardware is used to project and render the grid. This reduces CPU to GPU communication to a minimum. In [25], Schneider et al. use the projective grid method to display theoretically infinite terrain in high detail. Instead of precalculating height fields, they are generated at runtime.

Whereas Schneider et al.'s approach can not be used for predefined height fields, Livny does not guarantee rendering quality within a given error threshold.

We extend the projective grid method of Johanson in such a way that it is applicable to arbitrarily predefined or dynamic height fields. Furthermore we also ensure rendering quality within a given error threshold.

3 PROJECTIVE GRID METHOD

In this section we give a brief overview of the idea behind the projective grid method and describe the problems to be solved for its application to terrain rendering.

Basic idea

The projective grid method has been developed for interactive water rendering based on a dynamic height field. The principle of this method is simple and powerful. The basic idea is to cover the currently visible area of a height field and just this area, with a grid of fixed size which is placed onto the view plane. The size of the grid determines the quality of the terrain approximation and can be adjusted with respect to the capabilities of the used graphics hardware. The grid is projected onto the terrain's ground plane. Each projected point of the grid is displaced in the direction of the ground plane's normal by a fetched height value. The resulting grid is a view-dependent approximation of the original height field and can be used for rendering (see Figure 1).

Problem discussion

Even though the grid projection seems straight forward, there are three special cases which it needs to be adjusted in (see [13]):

- Looking above the scene's horizon results in *Backfiring*, which means that grid points will be projected behind the scene camera (see Figure 2(a)).
- In case of terrain data with *high amplitude*, peaks outside of the projected ground plane may intersect the view frustum (see Figure 2(b)).
- Undersampling can lead to a *loss of relevant features*, e.g., peaks and dips in the terrain. (see Figure 2(c)).

To solve the first two problems Johanson introduced the concept of an additional *projection camera*. This camera is aligned with respect to the viewing camera, but it never looks above the horizon. Moreover, to consider terrain that possibly extends into the view frustum, the projection camera's field of view is increased (see Figure 3). The problem of losing relevant features is not addressed by Johanson, because it can be ignored when rendering water surfaces. However, when applying the projective grid method for terrain rendering this problem has to be solved.

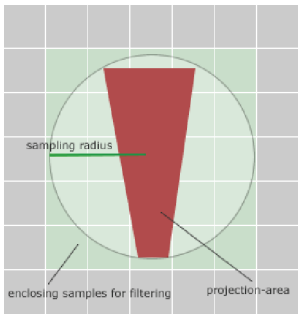


Figure 4: a non-uniformly shaped projection area leads to inadequate filter values due to the choice of the enclosing sampling radius.

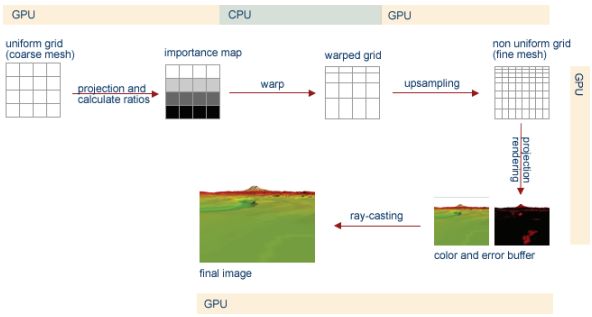


Figure 5: Scheme of our method's rendering process.

For this purpose, filtering of the height field has to be carried out. In [20], different resolutions of the height field are generated and the proper resolution depending on the sampling radius of a projected grid point is used. A drawback of this approach is that in situations where the view direction is close to the horizon, the projection of a single point in screen space onto the height field leads to a trapezoid area strongly elongated in the view direction, but narrow in the transverse direction (see figure 4). Because of the enclosing sampling radius used to determine the LOD, a filtered elevation value is chosen that does not approximate the underlying height field in a proper manner.

Undersampling as well as inaccurate filtering of the height values lead to a loss of relevant features, depending on the current view parameters and the resolution of the grid.

4 OUR METHOD

In this section we present our algorithm for interactive terrain rendering that addresses the problems described in the previous section. The general procedure can be described as follows: First, we generate a sample grid whose resolution depends on the capabilities of the graphics hardware. Thereby, we can guarantee the highest quality that is possible with respect to a given output device. Like in [20], we cache the grid in video memory, thus projection and rendering can be performed on the programmable graphics hardware. In contrast to previous approaches, we define the grid on the view plane depending on the current view in such a way that the projection of grid points results in a better approximation of the original terrain surface. This alleviates undersampling problems and helps achieve better image quality with respect to a given grid resolution. The projection is performed in a straight-forward manner. But contrary to known approaches, we compute an approximation error for each grid point using an extended MipMap hierarchy for the height field. The error values are used to generate an *error buffer*. During rendering the buffer is deployed for an adaptive per-pixel displacement mapping in regions where the error threshold is exceeded. This guarantees a representation within a given error threshold. A scheme of the rendering process is shown in figure 5. In the following, we will discuss the individual steps in more detail.

Grid Definition

The grid definition is a crucial step of the projective grid method. An accurate approximation of the original terrain surface implies a proper grid point distribution on the view plane. Earlier approaches used a fixed, pre-defined grid point distribution, leading to visual artifacts in particular situations (see Section 3).

These artifacts occur due to the fact that the projected grid points do not correspond to the original grid points of the terrain data. To alleviate this problem, we use a non-uniform, view-dependent grid point distribution. The grid points are defined in the view plane in such a way that the projection of the grid leads to almost quadratic grid cells. Thus, stretched grid cells caused by specific viewing conditions are avoided. This implicates that the region of influence of a projected grid point is also almost quadratic. As a result, artifacts caused by inadequate filtering are reduced. However, finding a good distribution is not a trivial task, because we need knowledge about the projection and perspective distortion. To define such a view-dependent grid point distribution in the view plane, a two step method is carried out:

First, a uniform grid is defined in the view plane and is projected onto the terrain's ground plane. The aspect ratio of each grid cell is calculated. This gives us a measure for the distortion of the grid cells. The aspect ratio is a sufficient measure, because it depends on the grid resolution as well as on the current view parameters. In the second step we use this measure to distort the uniformly distributed grid in the view plane, resulting in a non-uniformly distributed grid.

Whereas the first step is straight-forward, the second step can be implemented with the help of the importance-driven warping technique introduced in [8]. The warping function distorts the grid in such a way that more grid points are placed in regions

with high importance, while grid points are removed in other regions. This is exactly the behaviour that accomplishes our problem.

The required importance map is computed based on the aspect ratio of the projected grid cells. Regions with aspect ratios less than one are considered as very important, whereas regions with aspect ratios greater than one are declared as less important. This prompts the warping technique to relocate grid points from regions marked as unimportant to those declared as important. Hence, this results in the desired non-uniformly distributed grid.

This calculation is expensive and must be carried out on the CPU (see [8]) and therefore cannot be applied to the entire high-resolution grid. To reduce the calculation overhead, we use a coarse grid defined in the view plane. After applying the warping algorithm, we use the programmable GPU to refine the grid as far as possible with respect to the power of the graphics hardware.

Our procedure does not result in an optimal grid point distribution, but nonetheless, it leads to much better results than fixed, view-independent grid point distributions. Thereby, we are able to reduce visual artifacts and to achieve a better quality (see section 5).

Projection

After defining the non-uniform grid in the view plane, the grid points are projected using the algorithm introduced by Johanson. However, to reduce aliasing artifacts and to avoid a loss of relevant features, we calculate the height values of grid points with regard to their regions of influence on the ground plane.

To calculate proper height values, we filter the height field. We construct a multi-level texture pyramid of the height field, similar to [20], as follows: Starting from the original (finest) level, each level is constructed from the previous one by applying an average filter followed by halving its size in each dimension. The algorithm determines the level in the pyramid which a value is selected from depending on the region of influence. Similar to previous approaches, we calculate the farthest distance *dist* between adjacent projected grid points and use this distance to calculate the level in the texture pyramid as follows:

$$level = max(0, \log_2 dist) \tag{1}$$

In contrast to other approaches, our grid definition guarantees an almost uniform distance between adjacent neighbours of a grid point on the ground plane. This leads to more accurately filtered height values. The result is a better approximation of the original terrain surface (see Figure 6) with respect to the grid resolution.

Even though the projected grid could now be rendered using a simple texture mapping into the *colour buffer*, further enhancements are necessary to guarantee a high quality representation within a given error threshold.

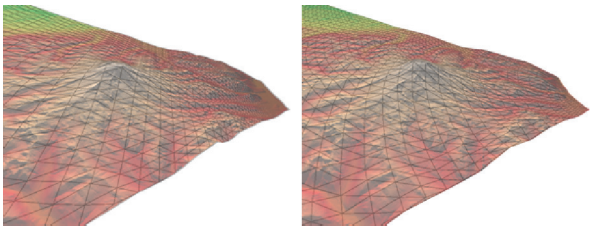


Figure 6: The left image shows the result of a uniform grid while the right image is generated using the view-dependent grid point distribution.

Error Metric

In our approach we want to guarantee a representation within a given error threshold. For that purpose, we use the following two error types:

- screen-space error
- object-space error

During the projection phase, the object-space error $\delta_{i,j}$ for each grid point $p_{i,j}$ is calculated. The object-space error depends on the chosen filtered height value h_{avg} as well as on the local minima h_{min} and maxima h_{max} in the region of influence of $p_{i,j}$. It is calculated as follows:

$$\delta_{i,j} = max(h_{max} - h_{avg}, h_{avg} - h_{min}) \tag{2}$$

To gather local minima and maxima we generate a *min* and *max* filtered texture pyramid similar to the previously generated average texture pyramid. In this way, average, min, and max height values can be fetched in unified manner from the texture pyramids. The fetching can be carried out in the projection step and the object-space error can be calculated using Equation 2. The object-space error is then projected back to the view plane, resulting in a screen-space error $\rho_{i,j}$. Since this can be computationally inefficient (see [18]), we use a simple metric:

$$\rho_{i,j} = \lambda \frac{\delta_{i,j}}{\|p_{i,j} - e\|} \tag{3}$$

with $\lambda = \frac{w}{\phi}$, where w is the number of pixels in the field of view ϕ and e the view position (see [18]).

The screen-space error $\rho_{i,j}$ can now be compared to the user-defined screen-space error threshold γ. If $\rho_{i,j} > \gamma$ we displace the grid point $p_{i,j}$ by h_{max} to preserve local maxima. Furthermore, the error is stored for each projected grid point $p_{i,j}$ and is used in the rendering pass to guarantee a representation within the error

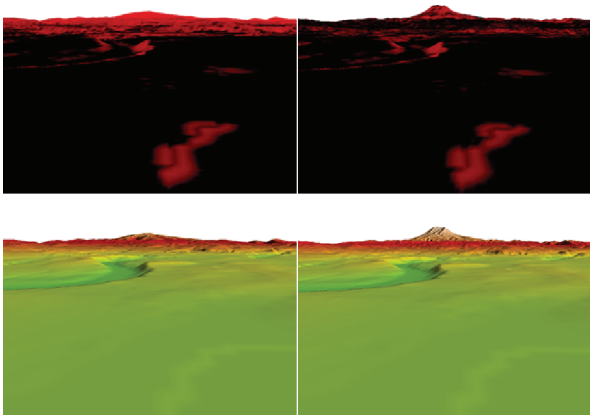


Figure 7: The error buffer for a 256x512 grid resolution. Red means high error, while black represents errors within the user-defined threshold. The left image shows the error buffer for a uniform grid point distribution. The right image was generated using the non uniform grid point distribution. Note the high detail and the minimised error in far-away regions.

threshold γ. For that purpose, we normalise the error values to the range $[0,1]$ as follows:

$$p_{i,j}.error = \begin{cases} 0 & \rho_{i,j} < \gamma \\ 1.0 - \frac{\gamma}{\rho_{i,j}} & else \end{cases} \quad (4)$$

Finally, the grid is rendered with the error value as colour attribute, resulting in an *error buffer* (see Figure 7) containing an interpolated error value for each visible pixel.

Rendering

During rendering our goal is to keep the per-pixel error below a given error threshold. Previous GPU-based approaches generated high quality images only by rendering huge numbers of primitives. But this does not guarantee any error rates. Therefore, we follow a different strategy. We perform adaptive ray casting in selected regions with errors that exceed the user-defined threshold. Hence, we are able to guarantee a chosen quality. The adaptive approach reduces calculation costs compared to applying ray casting to the entire height field. Ray casting is performed on the GPU as follows: For each pixel in screen space, the error is retrieved from the error buffer generated in the previous step (see Section 4). Ray casting calculates exact colour and precise depth values for a pixel in screen-space and replaces the less accurate ones in the colour and depth buffer (see Section 4). The final image can then be rendered using a *deferred shading* approach. We prefer deferred shading because it decouples shading from ray casting. Without deferred shading, to perform ray casting, we would require knowledge about the shading algorithm.

5 DISCUSSION AND RESULTS

Our approach can be summarised as follows:

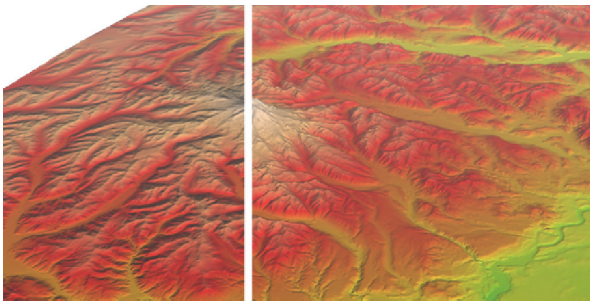


Figure 8: Ray casting of the terrain on selected areas. The left image shows terrain rendering without ray casting. On the right image ray casting is turned on.

The grid definition: defines a view-dependent, non-uniform grid on the view plane, which is novel compared to previous approaches. A uniform grid is warped with the help of an importance-driven method. The importance is defined by the aspect ratio of projected grid cells. This results in a non-uniform grid point distribution. Due to the view-dependent grid definition, we achieve a better approximation of the original terrain surface with respect to the grid resolution.

The projection: projects the non-uniform grid onto the ground plane and fetches proper height values for each projected grid point. The grid definition guarantees that the projected grid cells are almost quadratic, which leads to more accurately filtered height values. To avoid undersampling, the projection uses an average MipMap representation of the height field to fetch proper height values for each grid point.

The error measure: is used to gather approximation errors during the projection of grid points. In this step, a min and max MipMap representation of the height field is utilised. Based on the MipMaps, an object-space error is calculated for each grid point. The object-space error is projected back onto the view plane defining the screen-space error. The error is compared to a user-defined threshold and is normalised. An error buffer is rendered containing the interpolated normalised errors for each visible pixel.

The rendering process: performs adaptive ray casting utilising the error buffer in regions with high errors. The ray casting approach guarantees a representation within the user-defined error threshold.

The MipMaps reduce calculation time during the different steps. They can be generated in an offline process, but it is also possible to execute this during runtime, because the calculations are very simple and fast. Ray casting allows for a representation with a per-pixel error below a given error threshold. In fact, this can not

	fps				error	
grid size	uniform	non-uniform	ray casting (uniform)	ray casting (non-uniform)	uniform	non-uniform
1024x512	68.72	61.48	33.43	36.42	0.20	0.03
512x256	251.34	217.82	64.26	70.17	0.21	0.06
600x600	98.14	90.69	41.05	43.64	0.20	0.05
300x900	131.47	119.53	43.27	45.81	0.19	0.04
400x1900	48.61	45.80	23.52	25.34	0.18	0.02

fps: average frames per second for 8000 frames
error: average normalised error per grid point

Table 1: Speed and quality comparison between the standard method from [20] and our technique using different grid resolutions.

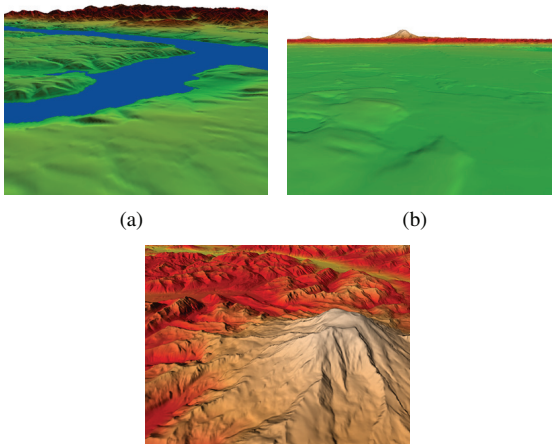
guarantee a fixed frame rate, as the original approach, but it is a good compromise between quality, time and resources. Indeed, the generation of the MipMaps consumes resources, but on the other hand, it enables us to guarantee a representation's quality. High quality is guaranteed by performing adaptive ray casting on selected areas, which, however, consumes time. But we keep ray casting to a minimum, by using an improved non-uniform grid point distribution on the view plane. This distribution is computed by a CPU-based warping technique, which again consumes time. However, except for the warping technique, all other calculations are performed on the GPU, which guarantees real-time and high quality terrain visualisation.

Our approach has been implemented using OpenGL 2.0 and requires graphics hardware supporting shader model 3.0 or higher. We use the vertex shader to define the grid on the view plane as well as for the projection and displacement of the grid points. The programmable fragment pipeline enables ray casting on the GPU. For the purpose of evaluation, we use the real-world 4k Puget Sound data set provided by Lindstrom with the original scaling factors having a peek at mount Rainier with ca. 4.400 metres.

It is also possible to support very large terrain using clipmaps as presented in [20]. Since Livny's and our approach use the same projection procedure, only a few modifications would be necessary.

The results we report in this section have been achieved on a PC with a Core 2 Duo 2.0 GHz processor, 1GB of memory and a GeForce 8800 GTX graphics card. Table 1 shows average frame rates (fps) as well as the average screen-space error per grid point, during a flight over Puget Sound with and without ray casting turned on (see figure 9). We tested various grid resolutions using the standard method and our technique, with a fixed screen size of *1024 x 800*. The uniformly distributed grid rendering corresponds to Livny's approach (see [20]).

As Table 1 shows, the usage of a non-uniform projection grid leads to a better approximation of the underlying terrain and reduces the average screen-space error per grid drastically. For instance, using a low grid resolution like *512x256* and a non-uniform projection grid generates an average error of *0.06* where a uniform grid with a four times higher resolution with *1024x512* still generates an average error of *0.20*.



(a) (b)

(c)

a) the start of the flight
b) near ground in the middle of the flight
c) close up at the end of the flight

Figure 9: The flight over Puget Sound. We tested various camera perspectives, from flight near ground till closeups.

	error pixels		avg error		max error	
grid-size	uniform	non-uniform	uniform	non-uniform	uniform	non-uniform
1024x512	10.7	7.5	1.60	1.42	18.0	6.77
512x256	15.6	12.1	1.70	1.40	23.1	6.9
600x600	11.9	8.5	1.83	1.53	22.0	8.3
300x900	13.0	9.0	1.68	1.44	19.3	6.7
400x1900	11.0	7.4	1.67	1.51	14.9	6.7

error pixels: the number of error pixels in % of all visible pixels
avg error: average screen-space error of all visible pixels
max error: max screen-space error of a all visible pixels

Table 2: Statistics on the errors of visible pixels. Our method minimises the regions, which ray casting must be performed in. Thus, we reduce the calculation time to achieve a representation within a defined error threshold. For performance issue see Table 1.

Comparing the frame rates of the standard method with our approach the time needed for warping is recognisable when using low grid resolution. The higher the grid resolution is, the more the frame rates converge. Looking at the grid resolution *400x1900*, the frame rate difference between the standard method and ours is very small and can be neglected.

Table 2 displays the percentage of error pixels in relation to the screen resolution (corresponding performance measurements are shown in Table 1. These regions must be handled by ray casting to guarantee rendering quality within the error threshold. Furthermore, the average error of all visible pixels as well as the maximum screen-space error have been captured. Comparing the maximum screen-space error of both techniques

shows that our technique approximates the original surface much better. Moreover, our method also minimises the regions with high errors. Hence, a lower resolution can be chosen, which still results in nearly the same number of error pixels, in contrast to the original approach. For instance, a *600x600* grid resolution generates fewer error regions with our technique than a grid resolution of *400x1900* with the classic approach. The results of Table 1 and Table 2 show that a compromise between time, resources and quality has been achieved.

6 CONCLUSION

We have introduced a GPU-supported approach for terrain rendering, using the projective grid method. We have shown how to reduce visual artifacts caused by inaccurate filtering of height values. Furthermore, we gather approximation errors that help us determine regions that need to be rendered using adaptive ray casting. Ray casting guarantees a representation within a given error threshold. We see the scope of future work in improving the view-dependent definition of the grid distribution in the view plane. Moreover, ray casting should replaced by a GPU-based subdivision algorithm utilising the shader model 4.0. This algorithm can be controlled by the error metric, and can be processed during the projection step. This will also increase the performance.

REFERENCES

[1] A. Asirvatham and H. Hoppe. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley Professional, 2005.

[2] X. Bao, R. Pajarola, and M. Shafae. Smart: An efficient technique for massive terrain visualization from out-of-core. In *VMV*, 2004.

[3] A. Brodersen. Real-time visualization of large textured terrains. In Stephen N. Spencer, editor, *GRAPHITE, Proc. of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia 2005, Dunedin, New Zealand, November 29 - December 2, 2005*, pages 439–442. ACM, 2005.

[4] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno. BDAM – batched dynamic adaptive meshes for high performance terrain visualization. *Computer Graphics Forum*, 22(3):505–514, September 2003. Proc. Eurographics 2003.

[5] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno. Planet-sized batched dynamic adaptive meshes (p-bdam). In *IEEE Visualization*, pages 147–154, 2003.

[6] M. Clasen and H.-C. Hege. Terrain rendering using spherical clipmaps. In Beatriz Sousa Santos, Thomas Ertl, and Ken Joy, editors, *EUROVIS - Eurographics /IEEE VGTC Symposium on Visualization*, pages 91–98, Lisbon, Portugal, 2006. Eurographics Association.

[7] D. Cohen-Or and Y. Levanoni. Temporal continuity of levels of detail in delaunay triangulated terrain. In *VIS '96: Proc. of the 7th conference on Visualization '96*, pages 37–42, Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.

[8] C. Dachsbacher and M. Stamminger. Rendering procedural terrain by geometry image warping. In *Rendering Techniques 2004 (Proc. of Eurographics Symposium on Rendering)*, pages 103–110, 2004.

[9] W. de Boer. Fast terrain rendering using geometrical mipmapping. %urlhttp://www.flipcode.com/tutorials/geomipmaps.pdf, October 31 2000.

[10] M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, and M. B. Mineev-Weinstein. ROAMing terrain: Real-time optimally adapting meshes. In *IEEE Visualization '97 (VIS '97)*, pages 81–88, Washington - Brussels - Tokyo, October 1997. IEEE.

[11] J. El-Sana and A. Varshney. Generalized view-dependent simplification. *Computer Graphics Forum*, 18:83 – 94, 1999.

[12] H. Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. In *VIS '98: Proc. of the conference on Visualization '98*, pages 35–42, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[13] C. Johanson. Real-time water rendering - introducing the projected grid concept. Master's thesis, Lund University, 2004.

[14] Y. Kryachko. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley Professional, 2005.

[15] B. D. Larsen and N. J. Christensen. Real-time terrain rendering using smooth hardware optimized level of detail. *Journal of WSCG*, 11(2):282–9, feb 2003. WSCG'2003: 11th International Conference in Central Europe on Computer Graphics, Visualization and Digital Interactive Media.

[16] J. Levenberg. Fast view-dependent level-of-detail rendering using cached geometry. In *VIS '02: Proc. of the conference on Visualization '02*, pages 259–266, Washington, DC, USA, 2002. IEEE Computer Society.

[17] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. Faust, and G. A. Turner. Real-time, continuous level of detail rendering of height fields. In *SIGGRAPH '96: Proc. of the 23rd annual conference on Computer graphics and interactive techniques*, pages 109–118, New York, NY, USA, 1996. ACM.

[18] P. Lindstrom and V. Pascucci. Visualization of large terrains made easy. In *IEEE Visualization*, August 2001.

[19] P. Lindstrom and V. Pascucci. Terrain simplification simplified: A general framework for view-dependent out-of-core visualization. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):239–254, 2002.

[20] Y. Livny, N. Sokolovsky, T. Grinshpoun, and J. El-Sana. A gpu persistent grid mapping for terrain rendering. *Vis. Comput.*, 24(2):139–153, 2008.

[21] F. Losasso and H. Hoppe. Geometry clipmaps: terrain rendering using nested regular grids. *ACM Trans. Graph.*, 23(3):769–776, 2004.

[22] R. Pajarola. Large scale terrain visualization using the restricted quadtree triangulation. In *VIS '98: Proc. of the conference on Visualization '98*, pages 19–26, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[23] A. A. Pomeranz. Roam using surface triangle clusters (rustic). Master's thesis, University of California at Davis, 2000.

[24] B. Rabinovich and C. Gotsman. Visualization of large terrains in resource-limited computing environments. In *VIS '97: Proc. of the 8th conference on Visualization '97*, pages 95–102, Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[25] J. Schneider, T. Boldte, and . Westermann. Real-time editing, synthesis, and rendering of infinite landscapes on GPUs. In *Vision, Modeling and Visualization 2006*, 2006.

[26] J. Schneider and R. Westermann. Gpu-friendly high-quality terrain rendering. *Journal of WSCG*, 14(1-3):49–56, 2006.

[27] C. C. Tanner, C. J. Migdal, and M. T. Jones. The clipmap: a virtual mipmap. In *SIGGRAPH '98: Proc. of the 25th annual conference on Computer graphics and interactive techniques*, pages 151–158, New York, NY, USA, 1998. ACM.

Improvement of the DOI Technique for De-Interlacing

Kyeong Ran Kim
Dongseo University
San 69-1, Jurye 2-Dong
Sasang-gu
617-716, Busan, S.Korea
d0076040@dongseo.ac.kr

Byung Gook Lee
Dongseo University
San 69-1, Jurye 2-Dong
Sasang-gu
617-716, Busan, S.Korea
lbg@dongseo.ac.kr

Hoon Yoo
Sangmyung University
Hongji 7-Dong,
Jongno-gu
110-743, Seoul, S.Korea
hunie@smu.ac.kr

ABSTRACT

This paper presents an improved method for the direction-oriented interpolation (DOI) method. The technique of DOI is considered to be a very strong tool for intraframe-based deinterlacing in the literature. However, the DOI still have a problem to use wrong edge direction. To remedy this problem, we embed three steps in the DOI method to interpolate missing lines more efficiently and robustly than existing methods. In the proposed method, the spatial direction vector (SDV) data are reused and processed to prevent them utilizing in next interpolation step, resulting in the more accurate de-interlacing method. We carry out experiments to compare the proposed method with the existing methods including the edge-based line averaging (ELA) and DOI. Experimental results show that the proposed method gives better performance in objective and subjective quality than the existing de-interlacing methods.

Keywords

Direction-oriented interpolation, direction vectors, ELA, de-interlacing.

1. Introduction

In order to reduce bandwidth for broadcasting, television system decides to adopt interlaced video signal. The interlaced video signal means delete odd lines in one frame and delete even lines in the next frame alternatively. We call frame without odd lines or even lines as field. It means capacitor of interlaced signal will become half of the original signal. Moreover the technique of CRT can display interlaced signal directly in alternative scanning lines. It will not exhibit flicker obviously. However, flat display can't show interlaced signal instantly. Interlaced signal must be recovered the original progressive signal before displayed. Therefore the technique of recovery is called de-interlacing.

De-interlacing methods have been proposed ever since the introduction of interlacing. With the advancement of technology, progressive displays have become feasible and that the importance of de-interlacing has increased. The methods have been suggested vary greatly in complexity and performance, but they can be segmented into two categories: intraframe and interframe. Intraframe methods only use the current field for reconstruction, while interframe methods make use of the previous or subsequent frames as well. Note that, all the methods attempt to reconstruct the missing scan lines without the knowledge of the original progressive source.

One well-known special domain interpolation method is the edge-based line-averaging (ELA) algorithm [10]. The algorithm utilizes directional correlation to interpolate a missing line between two adjacent lines of the even or odd filed. The ELA method provides a good result in regions where the edge can be estimated correctly. Introducing several methods in this paper is based on ELA.

In this paper, we proposed improved DOI method that added 3 steps. Expand the edge because after that the edge will be stronger and clearer. The method used SDV. Then, added the direction that can't be found and removed the noise produced after interpolation. In the section 2, will explained ELA and Directional Oriented Interpolation (DOI). In the section 3, we will explain our proposed method and next section we will explain the comparison our propose method with the previous method. In the last section, we will explain the conclusion.

.

2. Conventional De-interlacing Methods

2.1 Edge-based Line Averaging

The ELA method which utilizes directional correlations among the neighboring pixels

interpolates the missing pixels linearly. A 3x3 window is used as shown in Fig.1.

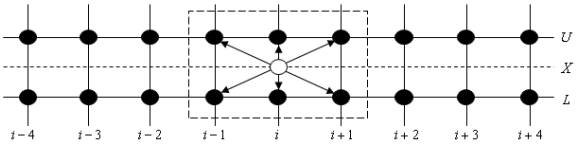


Figure 1 3×3 window for ELA method

According to Fig.1, the ELA method can be described in the equation (1), (2) and (3). The $C(k)$ denote a directional correlation measurement,

$$C(k) = \left|U(i+k) - L(i-k)\right|,\ k = -n, ..., -1, 0, 1, ..., n \quad (1)$$

The $C(k)$ is then used to determine the direction of the highest spatial correlation. The edge direction θ is determined as:

$$\theta = \arg\ \min\{C(k)\}, -n \le k \le n \quad (2)$$

The current pixel $x(i)$ is then interpolated by:

$$x(i) = \frac{U(i+\theta) + L(i-\theta)}{2} \quad (3)$$

The ELA method provides good performance in the most cases, however, due to misleading edge directions, interpolation errors often become larger in the areas of high frequency components. These unwanted artifacts significantly deteriorate visual quality.

2.2 Directional Oriented Interpolation

The DOI method interpolated the missing pixels using spatial direction vectors (SDVs) [13]. A spatial direction vector (SDV) is introduced so that finer resolution and higher accuracy of the edge-direction can be obtained. In figure 2, the center block of size 3-by-2 slides on the two upper reference lines or on the two lower reference lines for block-matching purpose.

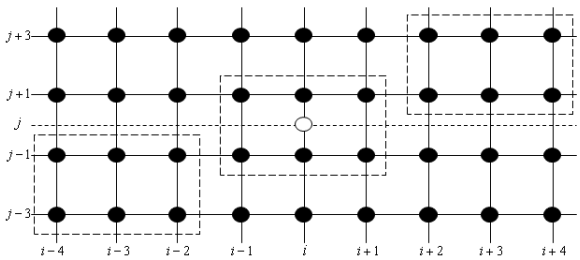


Figure 2 Illustration of the DOI

The USDV and the LSDV are determined as:

$$S_u(k) = \sum_{j=-1}^{1} (\left|U_0(i+j) - U_1(i+j+k)\right|^2 + \left|L_0(i+j) - U_0(i+j+k)\right|^2) \quad (4)$$

$$S_L(k) = \sum_{j=-1}^{1} (\left|U_0(i+j) - L_0(i+j+k)\right|^2 + \left|L_0(i+j) - L_1(i+j+k)\right|^2) \quad (5)$$

Using (4) and (5), the horizontal components of the USDV and the LSDV are obtained from:

$$sdv_u = \arg\ \min\{S_u(k)\}, -R \le k \le R \quad (6)$$

$$sdv_L = \arg\ \min\{S_L(k)\}, -R \le k \le R \quad (7)$$

The current pixel $x(i)$ is then interpolated by:

$$X(i) = \frac{U_0(i+i_u) + L_0(i+i_L)}{2} \quad (8)$$

$$\begin{aligned} i_u &= sdv_u / 2,\ i + i_u \\ i_L &= sdv_L / 2,\ i + i_L \end{aligned} \quad (9)$$

If the two index of the reference pixel i_U and i_L are not integer, the reference pixel can be obtained using interpolation in the horizontal direction. Detailed explain this method, give the current line and 4 reference lines, the magnitude of the difference between $U_0(i)$ and $L_0(i)$ is compared with a threshold T. If the magnitude of the summation of the two SDVs is larger than one, $X(i)$ is interpolated using linear interpolation in the vertical direction. Otherwise, the reference pixels are founds and $X(i)$ is interpolated using (12).

This de-interlacing method gives better performance in objective and subjective quality than ELA, Line Doubling, Line Averaging, A-ELA and E-ELA. However, even though DOI is more robust than the above methods, but DOI may produce annoying artifacts in periodic structures.

3. Proposed method

In this paper, we propose an improved approach to intra-frame de-interlacing which is based on direction-oriented interpolation (DOI) method as figure 3. We proposed method find the SDV, and then extend the SDV boundary within edge. Next, fill up the SDVs space in edge. Finally, we delete the wrong direction.

3.1 Calculate the intensity value

This step is previous step, we need the intensity value of edge in order to interpolate edge of correct direction. Intensity value of edge can estimate whether the direction edge is correct or not. Intensity value of edge is shown in the following:

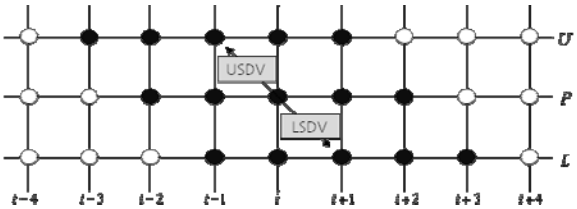


Figure 3 Calculate the intensity value

If the blank points are edges, current pixel calculates the intensity edge of value which is using SDV. And if the difference of USDV and LSDV is small, the edge has high possibility. So, intensity value of edge is large. Otherwise, if the difference is large, and the intensity value of edge is small. The equation of intensity edge of value is in the following:

$$
\begin{aligned}
&Case\ 1: |U_0[i] - L_0[i]| \geq T \\
&\quad Case\ 1.1: |P_{USDV}[i] + P_{LSDV}[i]| \leq 1 \\
&\qquad \rightarrow \text{ robust edge direction} \\
&\quad Case\ 1.2: |P_{USDV}[i] + P_{LSDV}[i]| > 1 \\
&\qquad \rightarrow \text{ weak edge direction} \\
&Case\ 2: |U_0[i] - L_0[i]| < T \\
&\qquad \rightarrow \text{nondirection}
\end{aligned}
\tag{10}
$$

3.2 Extension of edge area

The first step is how to extend the edge area which used SDV. Extension of edge area is showing in the figure:

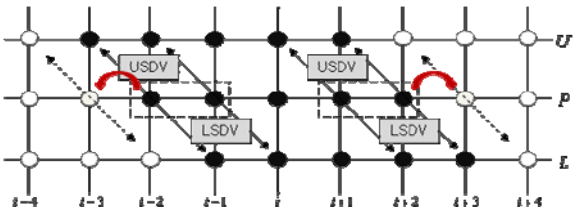


Figure 4 Extension of edge area

If black points are edges, we should compare SDVs of left and right pixels. The SDV in comparison, if there is no difference and similar, the SDV value is inserted at side pixel. The extension of edge area is determined as:

$$
\begin{aligned}
&CASE\ 1: D[i+1] = 2 \text{ and } D[i+2] = 2 \text{ or} \\
&\qquad D[i-1] = 2 \text{ and } D[i-2] = 2 \\
&\quad CASE\ 1.1: |P_{SDV}[i+1] - P_{SDV}[i+2]| = 0 \text{ and} \\
&\qquad |P_{USDV}[i+1] + P_{LSDV}[i+1]| = 0 \\
&\qquad \rightarrow \text{extend the left SDVs} \\
&\quad CASE\ 1.2: |P_{SDV}[i-1] - P_{SDV}[i-2]| = 0 \text{ and} \\
&\qquad |P_{USDV}[i-1] + P_{LSDV}[i-1]| = 0 \\
&\qquad \rightarrow \text{extend the right SDVs} \\
&CASE\ 2: D[i-1] < 2 \text{ and } D[i-2] < 2 \\
&\qquad \rightarrow \text{maintain the SDVs}
\end{aligned}
\tag{11}
$$

Through this step, the edge is robust and sharp.

3.3 Elimination of edge hole area

The result of previous step, we can get higher quality image than using the general SDV method. But it still can't find SDV. In this step, we can find the direction and add the direction of the edge using the near SDV, and we also can find direction using USDV, LSDV.

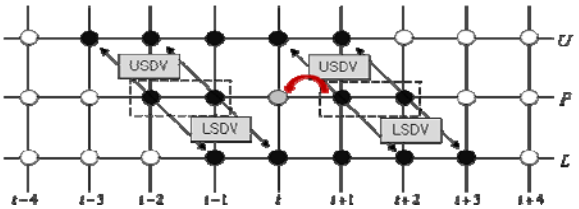


Figure 5 Elimination of edge hole area

For elimination of edge hole area, we use the neighborhood pixel of hole area. If intensity values of side pixels are large and SDVs(LSDV, USDV) of side pixels are similar, we can consider that the neighborhood pixels are in the same direction. So SDV of hole area pixel inserts SDV of next pixel, and then we can eliminate the edge hole area. The equation of elimination of edge hole area is as following:

$$
\begin{aligned}
&CASE\ 1: D[i+1] = 2 \text{ and } D[i+2] = 2 \text{ or} \\
&\qquad D[i-1] = 2 \text{ and } D[i-2] = 2 \\
&\quad CASE\ 1.1: |P_{SDV}[i-1] - P_{SDV}[i-2]| = 0 \text{ and} \\
&\qquad |P_{USDV}[i-1] + P_{LSDV}[i-1]| = 0 \text{ and} \\
&\qquad |P_{SDV}[i+1] - P_{SDV}[i+2]| = 0 \text{ and} \\
&\qquad |P_{USDV}[i+1] + P_{LSDV}[i+1]| = 0 \\
&\qquad \rightarrow \text{fill up the SDVs} \\
&CASE\ 2: D[i-1] < 2 \text{ and } D[i-2] < 2 \text{ or} \\
&\qquad D[i-1] < 2 \text{ and } D[i-2] < 2 \\
&\qquad \rightarrow \text{maintain the SDVs}
\end{aligned}
\tag{12}
$$

3.3 Correction of misdirection

The image of the last step which we calculated usually is wrong. Because it has the hopeless image, so in this step, we can get the improved quality of image when the information of SDV is wrong. How to remove the wrong SDV, the method is following:

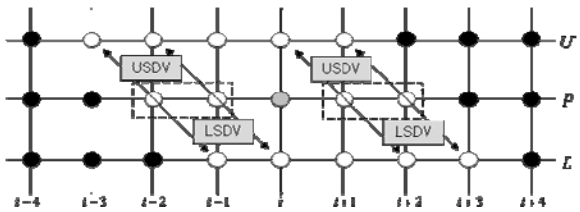


Figure 6 Correction of misdirection

To correct misdirection, we use the neighborhood pixels. If intensity value of neighborhood pixels is lowly and SDV of neighborhood pixels are similar.

So we consider that current pixel is the misdirection. We need modify SDV of current pixel. The correction of misdirection is shown in following equation 13:

$$
\begin{aligned}
&CASE\ 1: D[i+1] \le 1 \text{ and } D[i-1] \le 1 \\
&\quad CASE\ 1.1: \left|P_{SDV}[i-1] - P_{SDV}[i-2]\right| \le 1 \text{ and} \\
&\qquad \left|P_{USDV}[i-1] + P_{LSDV}[i-1]\right| \le 1 \text{ and} \\
&\qquad \left|P_{SDV}[i+1] - P_{SDV}[i+2]\right| \le 1 \text{ and} \\
&\qquad \left|P_{USDV}[i+1] + P_{LSDV}[i+1]\right| \le 1 \\
&\qquad \rightarrow \text{delete the wrong direction SDVs} \\
&CASE\ 2: D[i+1] > 1 \text{ and } D[i-1] > 1 \\
&\qquad \rightarrow \text{maintain the SDVs}
\end{aligned}
\tag{13}
$$

Through these steps, we can use the correction of misdirection to decrease the noise of image and improve the quality of image.

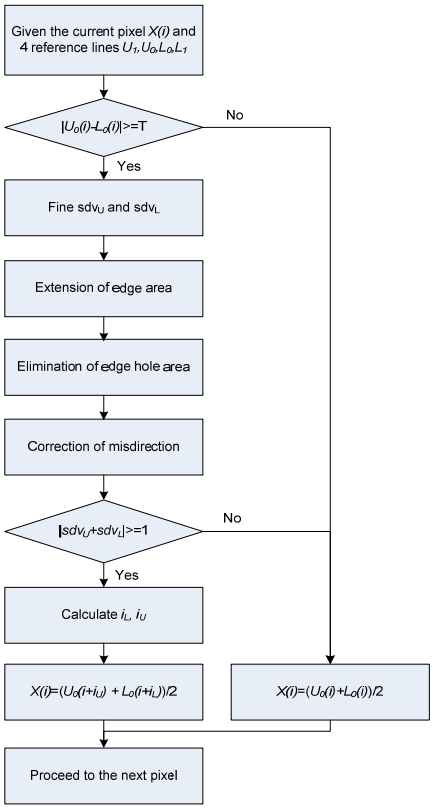


Figure 7 Flow diagram of the proposed method

4. Experimental Results

In order to obtain fair empirical evaluation of intraframe de-interlacing methods, it is important to use a set of well-known images for experiments. In this paper, seven intraframe-based de-interlacing methods are chosen to evaluate their performance. Here we set the threshold t to 10, and we set the search rage R to 16 as DOI.

Table 1 shows the performance of these methods for various images. We compare the proposed method with ELA, Block-Based Directional, Edge Interpolation (BDEI), Edge-Based Median Filtering (EMF), Anti-aliasing Interpolation filter (AAIF), Modified ELA (MELA) and Directional Oriented Interpolation (DOI). Compare to proposed method with other method, the propose method most performance is good. PSNR results show a higher value of 0.03dB to 0.47dB. Subjectively, the quality of result image is similar to the DOI, and the result images are good as shown figure 8 and figure 9.

5. Conclusion

In this paper, we proposed an improved directional oriented interpolation algorithm. We proposed method is based on DOI. We apply the three steps in DOI algorithm. We obtain the more sharp result image using extend SDVs step. And proposed method improves image quality for the SDV of wrong direction modified. Using the SDV method, the edge's boundary will be expanded. So, we can get the clearer image. And also we get the high quality image because of reducing noise when modify the wrong direction. The result of our experiment showed objectivity using PSNR. The value is almost same with the other methods and even better in some cases. Experimental results show that the proposed method gives better performance in objective and subjective quality than the existing de-interlacing methods.

6. References

[1] J.Kovacevic, Reobert J. Safranek, and Edmund M.Yeh. Deinterlacing by Successive Approximation. *IEEE Transactions on Image Processing*, Vol. 6, No. 2, pages 339-344, February 1997

[2] O.J.Kwon, Kwanghoon Sohn, and Chulhee Lee. Deinterlacing using Directional Interpolation and Motion Compensation. *IEEE Transactions on Consumer Electronics*, Vol. 49, No. 1, pages 198-203, February 2003

[3] S.F.Lin, Yu-Ling Chang, and Liang-Gee Chen. Motion Adaptive Interpolation with Horizontal Motion Detection for Deinterlacing. *IEEE Transactions on Consumer Electronics*, Vol. 49, No. 4, pages 1256-1265, November 2003

[4] M.K.Park, Moon Gi Kang, Kichul Nam, and Sang Gun Oh. New Edge Deinterlacing Algorithm Based on Horizontal Edge Pattern. *IEEE Transactions on Consumer Electronics*, Vol. 49, No. 4, pages 1508-1512, November 2003

[5] Y.L.Chang, Shyh-Feng Lin, Ching-Yeh Chen, and Liang-Gee Chen. Video De-Interlacing by Adaptive 4-Field Global/Local Motion Compensated Approach. *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 15, No. 12, pages 1569-1582, December 2005

[6] Q.Huang, Wen Gao, Debin Zhao, and Huifang Sun. An Efficient and Robust Adaptive Deinterlacing Method. *IEEE Transactions on Consumer Electronics*, Vol. 52, No. 3, pages 888-895, August 2006

[7] L.Yu, Jintao Li, Yongdong Zhang, and Yanfei Shen. Motion Adaptive Deinterlacing with Accurate Motion Detection and Anti-Aliasing Interpolation Filter. *IEEE Transactions on Consumer Electronics*, Vol. 52, No. 2, pages 712-717, May 2006

[8] Y.Shen, Dongming Zhang, Yongodng Zhang, and Jintao Li. Motion Adaptive Deinterlacing of Video Data with Texture Detection. *IEEE Transactions on Consumer Electronics*, Vol. 52, No. 4, pages 1403-1408, November 2006.

[9] W.Kim, S Jin, J Jeong, "Novel Intra Deinterlacing Algorithm Using Content Adaptive Interpolation", *IEEE Transactions on Consumer Electronics*, Vol. 53, No. 3, pages 1036-1043, AUGUST 2007.

[10] T.Doyle and M.Looymans, Progressive scan conversion using edge information, *Signal Processing of HDTV II*, L. Chairiglione, ED. Amsterdam, The Netherlands: Elsevier, pages 711-721, 1990

[11] Y.Kim, Deinterlacing Algorithm based on sparse wide vector correlations, SPIE Optical Engineering, vol.2727, pages 88-99, 1996

[12] H.S.Oh, Y.Kim, Y.Y. Jung, A.W. Morales, and S.J. Ko, Spatio-temporal edge-based median filtering for deinterlacing. *Digest of the Int. Conference of Consumer Electronics*, pages 52-53, June 2000

[13] H.Yoo and Jechang Jeong. Directional-oriented interpolation and its application to de-interlacing. *IEEE Trans. Consumer Electronics*, Vol. 48, Issue 4, pages 954-962, November 2002

[14] G. de Haan and E. B. Bellers. Deinterlacing of video data. *IEEE Transactions on Consumer Electronics.* Vol.43, No. 3, pages 819-825, August 1997

[15] R.Li, B.Zeng, and M. Liou. Reliable motion detection/compensation for interlaced sequences and its applications to deinterlacing. *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 10, No. 1, pages 23-29, February 2000

[16] C.Ballester, Marcelo Betralmio, Vicent Caselles, Luis Garrido, Adrian Marques, and Florent Ranchin. An Inpainting-based Deinterlacing Method. *IEEE Transactions on Image Processing*, Vol. 16, No. 10, pages 2476-2491, October 2007

[17] T.Chen, Hong Ren Wu, and Zheng Hua Yu, "Efficient Deinterlacing algorithm using edge-based line average interpolation," *Optical Engineering*, Vol. 39, No. 8, pp. 2101-2105, Aug. 2000.

[18] http://sipi.usc.edu/database/index.html

Table 1 PSNR result of different intraframe de-interlacing methods for various images

Image	LA	ELA	MELA	BDEI	EMF	AAIF	DOI	Proposed method
Lena (512×512)	36.82	35.99	36.07	37.09	36.81	38.05	37.32	37.35
Airplane (512×512)	31.82	31.12	30.41	31.59	31.55	32.22	31.69	31.72
Baboon (512×512)	23.50	22.97	23.00	23.49	23.71	23.07	23.43	23.47
Bike (400×500)	24.40	23.60	22.92	24.26	24.51	24.19	25.36	25.59
Gold (720×576)	33.30	32.73	32.94	33.44	33.57	33.73	33.73	33.80
Hotel (720×576)	32.88	32.15	31.75	32.42	32.37	33.39	32.94	33.07
Target (512×512)	17.13	18.98	17.35	16.46	17.16	16.54	18.60	18.75
Table Tennis (352×240)	26.83	27.05	27.27	26.90	26.98	25.19	29.53	30.10
Light House (512×768)	30.27	29.38	30.30	29.87	29.69	24.01	30.40	30.86
House (256×256)	31.83	31.88	31.79	31.73	31.68	22.59	32.55	32.62
Couple (512×512)	27.93	27.43	27.42	27.75	27.88	22.28	27.91	27.94
Average	28.79	28.48	28.29	28.64	28.72	26.84	29.41	29.57

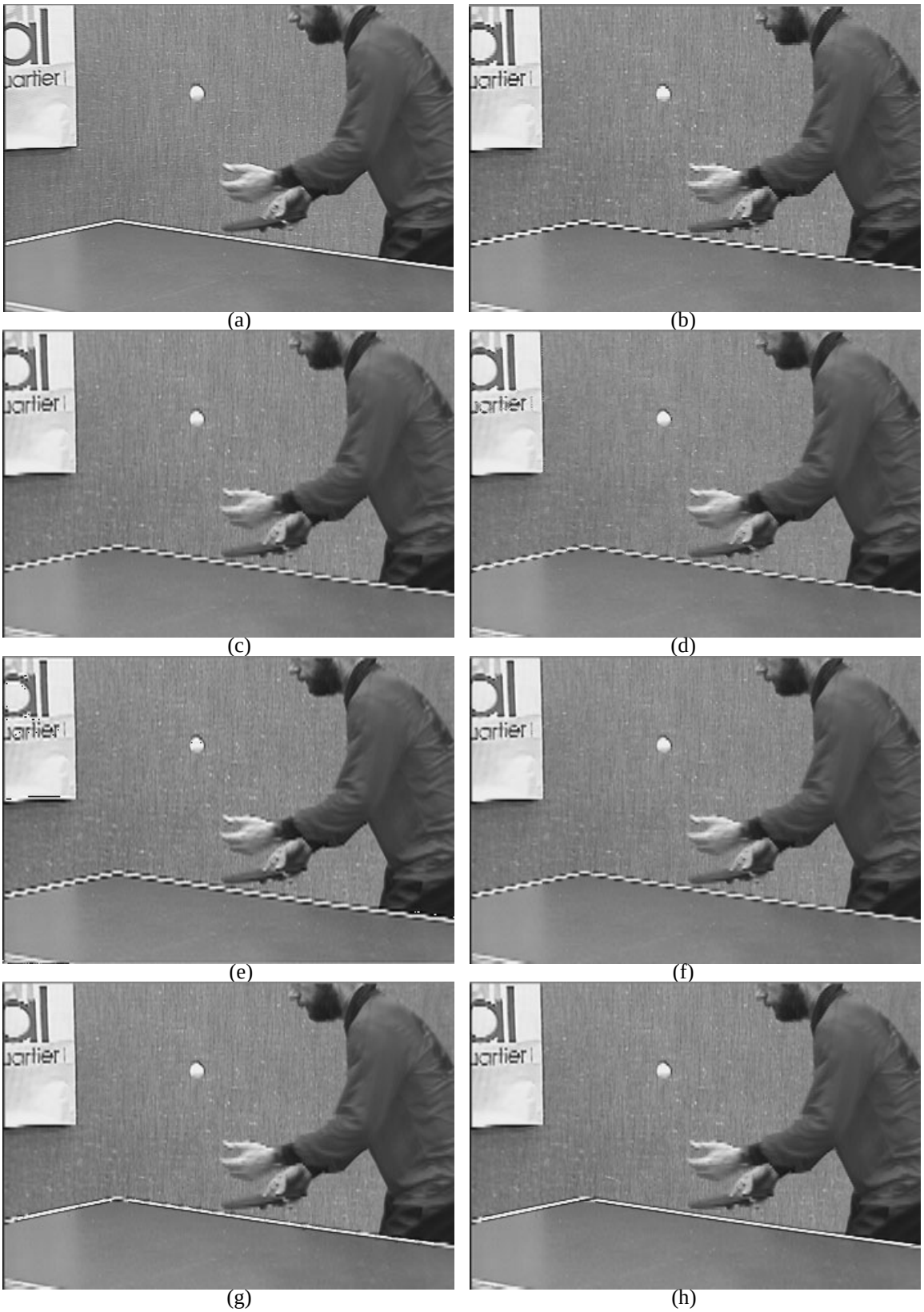


Figure 8 (a) Result of Table Tennis image, (b)-(h) Interpolated: (b) line-doubling (c) line-averaging (d) ELA (e) AAIF (f) MELA-2 (g) DOI (h) proposed method

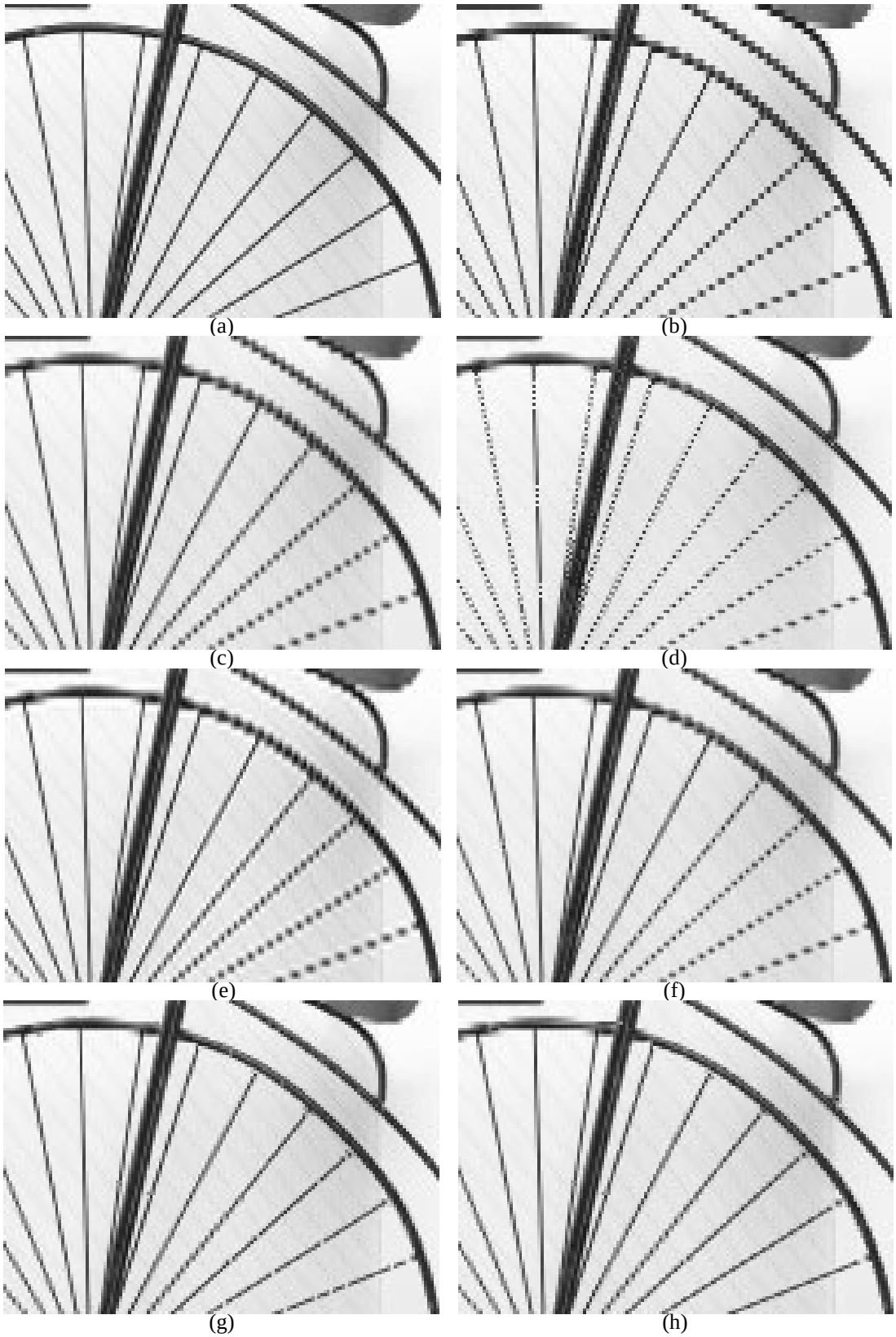


Figure 9 (a) Zoomed results of Bike image, (b)-(h) Interpolated: (b) line-doubling (c) line-averaging (d) ELA (e) AAIF (f) MELA-2 (g) DOI (h) proposed method

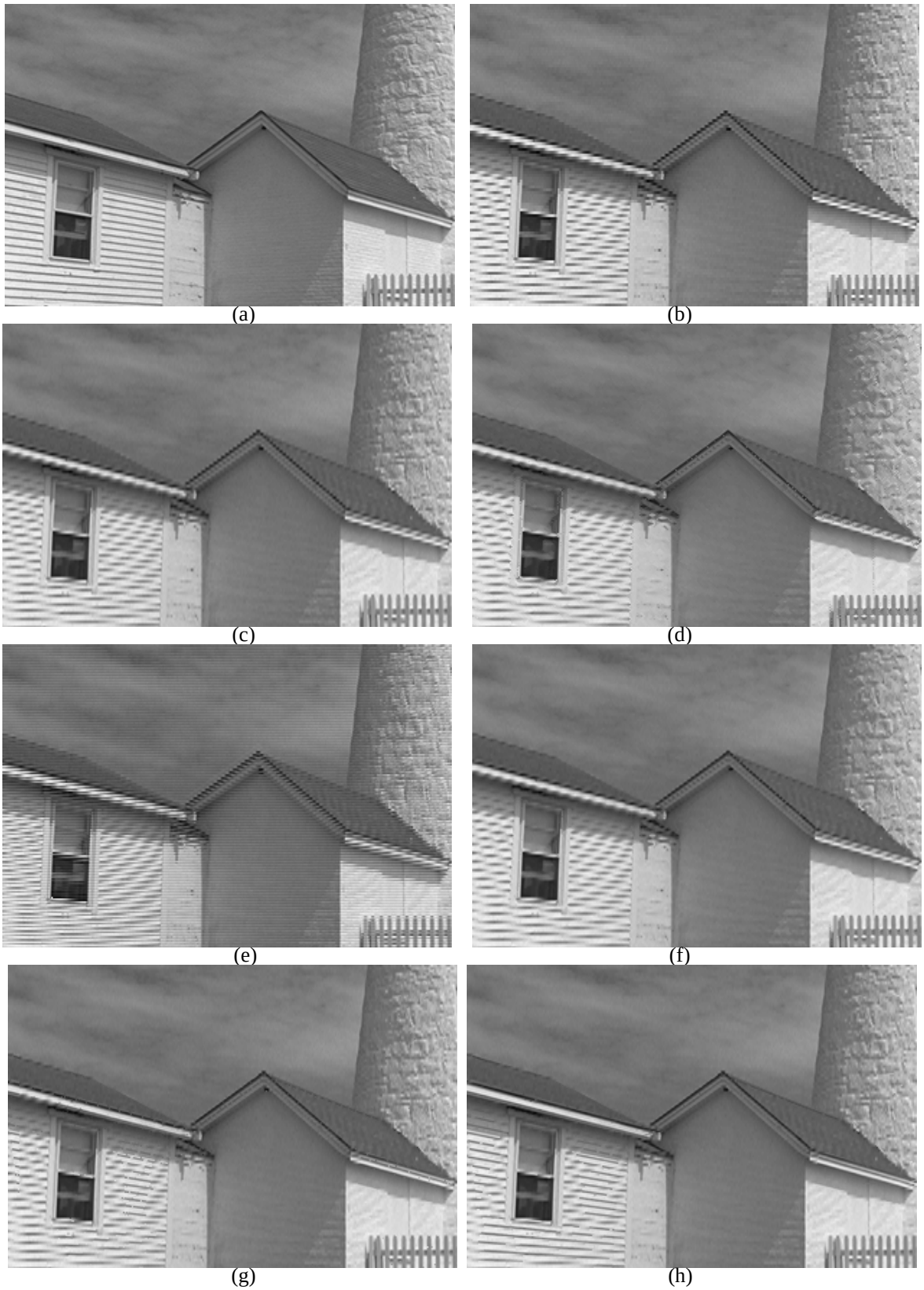


Figure 8 (a) Result of Light house image, (b)-(h) Interpolated: (b) line-doubling (c) line-averaging (d) ELA (e) AAIF (f) MELA-2 (g) DOI (h) proposed method

Point Cloud Rendering in FPGA

Pavel Zemčík
Faculty of Information Technology
Brno University of Technology
Božetěchova 2
Czech Republic, 612 66, Brno
zemcik@fit.vutbr.cz

Lukáš Maršík
Faculty of Information Technology
Brno University of Technology
Božetěchova 2
Czech Republic, 612 66, Brno
xmarsi03@stud.fit.vutbr.cz

Adam Herout
Faculty of Information Technology
Brno University of Technology
Božetěchova 2
Czech Republic, 612 66, Brno
herout@stud.fit.vutbr.cz

ABSTRACT

This contribution describes recent development in ongoing work focused on point cloud rendering algorithm implementation usable in environments containing programmable or custom hardware. The approach described in this paper is based on the idea that direct point cloud rendering, which is in the principle not too complicated, can be efficiently implemented in programmable or custom hardware. Such implementation can be useful not only for its performance but especially for the possibility to include it into solutions that require 3D graphics output in non PC environments and in embedded solutions with low power consumption, etc. This contribution describes the overall approach and the current results.

Keywords

Point cloud, programmable hardware, FPGA, hardware rendering.

1. INTRODUCTION

3D point-clouds [Rusi01][Gros02] represent a modern trend in computer graphics. They can be used for an alternative representation of graphics scenes instead of traditional entities, such as planar polygons or triangles, surface patches, etc. Geometrically, point-clouds are "internally unorganized" sets of points and as such they are very easy to handle during various graphics operations. On the other hand, point-clouds require large amount of memory to represent common objects and that is the reason why they have not been widely used historically. In recent few years, the situation changed. Not only the recent computer systems, including embedded and DSP systems, can be are equipped with cheap and large memory, but also the today affordable 3D scanning devices can create object models based on point clouds directly. To visualize these point clouds, however, methods to convert them into surface patches are still often used and/or GPUs are mostly used to render these structures although for direct point cloud rendering, the GPUs are often inefficient. The proposed approach offers an alternative based on a simple engine for direct point cloud rendering based on custom hardware circuits in the form of programmable hardware or custom chip. This alternative will most probably not be of wide use in contemporary PCs whose graphics performance, thanks to the modern GPUs, is very high and at the same time the PCs are relatively cheap. However, the approach, if successful, can serve as a model for future high performance implementation, e.g. as a module in some future piece of hardware or it can be used in embedded systems where exploitation of GPUs is not feasible at all and also in systems with limited power consumption, where exploitation of GPUs is also very problematic.

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

The recent development in programmable hardware, specifically Field Programmable Gate Arrays (FPGAs) [Curd07][Hite05] offers a good platform for power efficient, cost efficient, and also high performance implementation of point cloud rendering, which is not quite supported by the traditional computer graphics manufacturers

The proposed solution is characterized by using low amount hardware resources (system logic and fast memory) – offering possibility to embed numerous rendering engines on a chip, low power consumption, low heat emission and other features important for non-desktop applications.

The solution describer in this paper is an extension to previous work on FPGA accelerated point cloud rendering and adds higher performance due to both

technology and solution itself, parallelism, and extended color model [Zemc03][Zemc04].

2. POINT ELEMENT

The point cloud rendering algorithms described in this paper relies on rendering of the single oriented points forming the point cloud.

Point Element Shape

One of the feasible geometrical representations of the scene element – oriented point – is an oriented circle whose projection is a general ellipsis. See Figure 1.

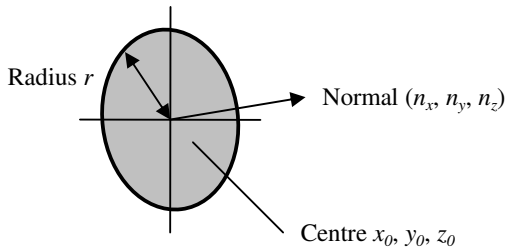


Figure 1: Point element projection

The rendering algorithm can be subdivided into several principal parts [Zemc04][Hero05] [Tisn02][Mars08]:

1. Projection of the elements' positions into 2D screen and Z space and computation of the corresponding elements' projected normal and radius.

2. Evaluation of the elements' color (lightness) based on the projected normal vector, element local lighting model (material), and the light sources' and observer's parameters.

3. Rendering of the elements (ellipses) into the image frame buffer (visibility solved using Z-buffer).

In the proposed approach, all the part 1 is performed through the host processor (DSP), part 2 is performed partially in the DSP through access to the pre-calculated reflection and diffusion tables and the final color evaluation is done in the FPGA, and the part 3 is performed in the FPGA.

Representation of Shapes

The point shapes are pre-calculated. The idea of pre-calculation is based on the fact that the normal vector can be converted into two dimensions according to the below equation and quantized and thanks to the fact that radius can be quantized as well.

$$(1) \quad \vec{n} = (n_x, n_y, n_z) = k(n'_x, n'_y, 1) = k\vec{n}'$$

The shapes of ellipses (circle projections) can then be stored in tables, indexed by quantized point size and normal vector, the size of the table is 17x64^2 (n'_x, n'_y are quantized as 64 values, the particle size as 17).

An efficient method is used to compress the point bitmaps is used, which uses namely the fact that the point shape is convex and symmetrical. Based on experiments on practical data and hardware implementation issues, the bitmap size is defined to be 8×8 pixels. For small points (whose shape can fit into the 8×8 raster), symmetry is exploited to minimize the representation size. Experiments show that majority of points rendered by the system meets the extent of small points and their processing is thus efficient. However, even larger point shapes appear in the rendered set, whose encoding takes larger number (inherently not limited, but practically 2×2) of fractional bitmaps. See Figure 2 and Figure 3 for examples of small and large point shapes.

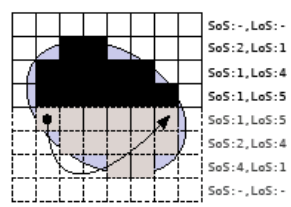


Figure 2: Small point shape evaluation

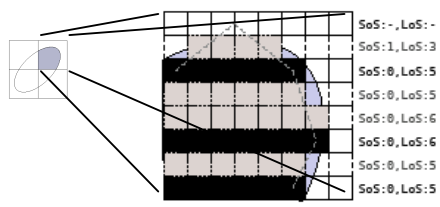


Figure 3: Large point shape evaluation

Representation of Colors

As for the color model, the proposed solution allows for Phong model with diffuse and specular parts but limited to a single level of specular reflection and single color of light sources presumably white.

$$(2) \quad \hat{I} = \hat{I}_0 + \hat{k}_{d,Material} I_{Diffuse} + \hat{k}_{s,Material} I_{Specular}$$

where *Diffuse*, *Specular*, and *Material* are parameters sent to the rendering engine for every element while color $\hat{I}_0$ and both $\hat{k}_d$ and $\hat{k}_s$ color tables are stored in the rendering engine.

The elements' properties, as evaluated by steps 1. and 2. described above through the pre-calculated table, form a code-word of 64 bits which enters the rendering Engine. See Figure 4 for the code-word's structure. Note, please, that the *y* co-ordinate is missing from the code word as the engine generates it implicitly. Detailed description of the rendering machine is in the following section of the paper.

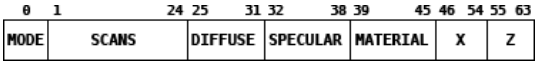


Figure 4: Point code word

3. SYSTEM ARCHITECTURE

The rendering engine is experimentally implemented on a PCI board placed in a host PC.

System board

The board (UNI1P-VUT manufactured by CAMEA, Brno) contains PCI controller, DRAM memory, and programmable hardware including FPGAs to control the data flow. The board is capable of carrying up to four DSP/FPGA computational engines (DX64), each with a DSP (TI C6416), FPGA (Xilinx VIrtex II), and DRAM (128MB). Block diagram is shown in the Figure 6.

Each of the DSP/FPGA boards implements single rendering engine.

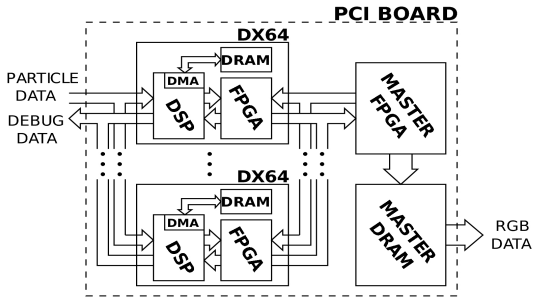


Figure 6: System architecture block diagram

Rendering engine

Each of the point elements could be rendered very quickly, just through interpreting the key word that is simple and the interpretation can easily be done in parallel except for the natural bottleneck lying in the number of pixels affected by each point element that can be up to 64 (8x8 blocks).

In our approach, however, this bottleneck is overcome through subdividing of the raster image memory and Z-buffer memory, used for rendering output, into 8 parts so that all the lines of the element image (8x8 pixels) can be rendered in parallel. As the shape of the element is encoded in very simple way (see above), the rendering lies merely in updating the Z-buffer and conditional writing of a color value into the output raster. As the Z-buffer and raster memory have 4 pixels per word (oriented horizontally), 8 pixels can be updated within 3 accesses in the memory as the data is not always word aligned.

Because the size of the raster and Z-buffer memory in the FPGA is limited, our implementation slides a narrow window (e.g. 16 pixels high and as wide as the image) across the output image line by line so that only that narrow window of raster and Z-buffer are present in the on-chip memory in the FPGA. The already processed part of the image is flushed out of the rendering engine. This approach allows for evaluating only those elements whose *y* co-ordinate is in the centre of the sliding window. This fact enforces sorting of the elements according to the *y* co-ordinate and sending them into the rendering engine in the *y* order. While the sorting operation seem to cause high computational complexity, the fact is that the *y* range is limited and as the co-ordinates are anyway integer, the sorting can be seen as merely subdividing the point cloud (set) into several subsets with linear complexity.

See Figure 7 for the illustration of sliding window. Note, please, that 8 pixels high window would be sufficient for rendering but the extra lines are suitable for buffered flushing of the output and buffered loading of the input (initialized raster and Z-buffer).

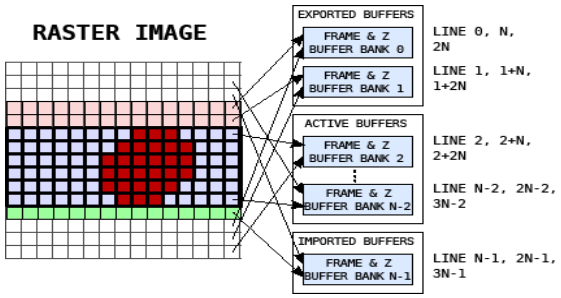


Figure 7: Sliding window over the raster

Rendering chain

The above described approach is well parallelizable. As the rendering engine is capable of uploading of the initialized image and Z-buffer and flushing the processed output with relatively low delay (processing of 16 lines of the window), it is possible to chain the rendering engines as shown in Figure 8 and when randomly data distributed among the engines lead into nearly linear speedup.

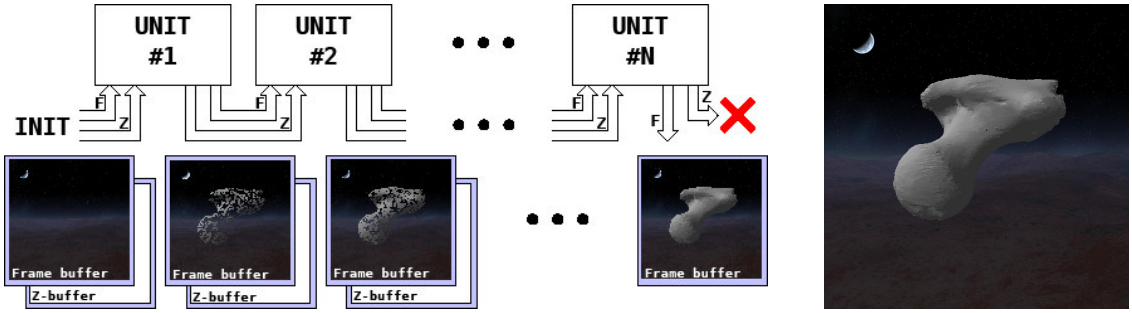


Figure 8: Processing chain block diagram and the rendering output

The block diagram of the engine itself is shown below. Note, please, that the "writer block" contains the raster and Z-buffer on-chip memory.

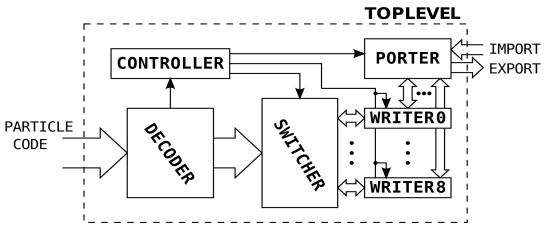


Figure 9: Single engine block diagram

4. PERFORMANCE

The performance of the approach can be evaluated in terms of the point elements rendering speed and in terms of the resource consumption.

Resource consumption

The consumption of FPGA resources on the Virtex II chip is shown in Table 1 below. Overall, the design consumes approximately 60 percent of the small Virtex II-250 chip.

```
Device utilization summary:
---------------------------

Selected Device : 2v250fg256-4

   Number of Slices:                    1167  out of   1536    76%
   Number of Slice Flip Flops:           960  out of   3072    31%
   Number of 4 input LUTs:              2150  out of   3072    70%
   Number of IOs:                         38
   Number of bonded IOBs:                 37  out of    172    21%
   Number of BRAMs:                       17  out of     24    70%
   Number of GCLKs:                        1  out of     16     6%
```

Table 1: Consumption of the FPGA chip resources

Rendering speed

The achievable rendering speed is affected by the clock speed of the FPGA structure. Currently, the achieved clock speed is 100 MHz and 4 clock cycles are needed to evaluate single element. At the same time, 4 rendering engines run in parallel so the raw rendering speed of 100 million (10^8) elements per second. The useful rendering speed is affected by the idle times. If e.g. video is generated, the worst idle time is 0.02 s. The rendering speed for N elements is:

(3) $$t_{\mathrm{Im}age}(s) = \max\left(0.02, N/10^8\right)$$

5. CONCLUSIONS

It has been demonstrated that the direct point cloud rendering engine in FPGA and DSP is feasible and that it can be built using relatively simple resources in programmable hardware. While the raw speed of the single engine does not reach extremely high figures, the approach is easily scalable and parallelizable and can be also used in embedded environments.

6. ACKNOWLEDGMENTS

This work has is supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research program LC-06008 (Center for Computer Graphics), and by the research project Security-Oriented Research in Information Technology, MSM0021630528.

7. REFERENCES

[Curd07] Curd, D.: Power Consumption in 65 nm FPGAs, White Paper: Virtex-5 FPGAs, February 1, 2007, XILINX, San Jose, CA, USA

[Gros02] Gross, M.: Point Based Computer Graphics, Proceedings of SCCG 2002, Budmerice, Slovakia, 2002

[Hero05] Herout, A., Zemčík, P.: Hardware Pipeline for Rendering Clouds of Circular Points, In: Proceedings of WSCG 2005, Plzeň, Czech Republic, 2005

[Hite05] Hitesh, P.: Synthesis and Implementation Strategies to Accelerate Design Performance , White Paper: Virtex-4 and Spartan-3 Families, July 6, 2005, XILINX, San Jose, CA, USA

[Mars08] Maršík, L.: Image processing in FPGA, Bc. Thesis, Brno University of Technology, Brno, Czech Republic, 2008 (in Czech)

[Rusi01] Rusinkiewicz, S.: Surface splatting, Proceedings of SIGGRAPH 2001, USA, 2001

[Tisn02] Herout, A., Tišnovský, P.: Vector Field Calculations on a Special Hardware Architecture, In: East-West-Vision 2002 Proceedings, Graz, TUV, Austria, 2002

[Zemc03] Zemčík, P., Tišnovský, P., Herout, A.: Particle Rendering Pipeline, Proceedings of SCCG 2003, Budmerice, Slovakia, 2003

[Zemc04] Zemčík, P., Herout, A., Crha, L. Tupec, P. Fučík, O.: Particle rendering pipeline in DSP and FPGA, In: Proceedings of Engineering of Computer-Based Systems, Los Alamitos, USA, IEEE CS, 2004

Trajectory classification using HMMs

Jozef Mlích, Pavel Zemčík, Leoš Jiřík
imlich@fit.vutbr.cz ,zemcik@fit.vutbr.cz, xjirik02@stud.fit.vutbr.cz
Department of Computer Graphics and Multimedia
Faculty of Information Technology, Brno University of Technology
Božetěchova 2, Czech Republic, 612 66, Brno

ABSTRACT

This paper focuses on evaluation of motion of objects through classification of their trajectories. The objects used for evaluation of the presented method are objects tracked in video sequence but the method is quite general and can be used for more general trajectories. The main potential application of the presented method is detection of abnormal behavior of humans, hand gesture detection and recognition, etc. Hidden Markov Models (HMMs) and Gaussian Mixture Models are used as the base of the the classification mechanism. The paper contains description of the method, description of experiments and their results, and conclusions.

Keywords

Trajectory classification, Gaussian Mixture Models, Hidden Markov models, video sequences.

1.INTRODUCTION

Understanding dynamic behavior of objects is still very open problem. One task, which is a small subset in the wide objects dynamics field, is classification of short fragments of motion of objects. This task is particularly interesting for human computer interfaces, computer applications in surveillance tasks, computer-assisted communication between humans, general computer graphics and computer vision, etc. Human computer interfaces can benefit specifically from understanding human hand gestures and human body motion. Surveillance applications can exploit detection of "abnormal" behavior of humans in e.g. public places. Computer-assisted communication between humans can benefit from understanding the gestures and body motion, too, specifically in case the users cannot see or even hear each other or in cases where the communication is otherwise limited. Other applications in computer graphics and computer vision include e.g. human motion understanding in navigation in virtual 3D space, animation of objects and humans, or motion capture assistance.

Previous Work

The generally used approach to analysis of dynamic behavior of objects is focused on understanding of video content. This leads to decomposition of problem to a couple levels of analysis. The first level of decomposition usually consist of image processing e.g. object recognition. This part also includes temporal element e.g. object tracking. The problem of object recognition and tracking is discovered in [Hra06], [Jir06], [Jir08]. The second level of decomposition is information interpretation. Decomposition of problem and solution of parts is widely discussed in [Mli08]. The [AC99], [CRCZ05] summaries the approach with focus on human motion analysis.

More general low level video processing which tends to obtaining complex information from scene could be based for example on texture analysis [CV05], important points analysis (SIFT) or areas analysis (SURF) [Bay08]. However, it is very difficult to describe behavior patterns in general, therefore proposed approach use only informations about well defined objects in scene.

The result of content focused analysis is set of features with describes variable count of objects in time. The special case of these features in time is the trajectory. For detailed analysis of object behavior is essential to describe all possible behaviors (trajectories) and assign them the most appropriate interpretation, which is discussed in [Mli08a]. In some cases, few well defined "strong types of behavior" could be sufficient for scene understanding.

The Gaussian Mixture Models (GMM) method is suitable for the trajectory classification task as it is capable of recognition of pattern in temporal variable data. The GMM statistically describes whole feature vector in time. Other well known approach, Dynamic Time Wrapping (DTW), describes changes of feature vector in time. The generalization of these two methods is done by Hidden Markov Models (HMM) which was used for sign language

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

recognition [Sta95] and gesture recognition [Rig97], [Bor08].

The method of gesture classification through the alternative Conditional Random Fields is discussed in [WLD06].

Data and Trajectory Classes

Based on the results of the previous work, we examined some data and methods for dynamic gesture processing and human motion trajectories. The data was available from CareTaker [Car08] in the form of moving humans in a public environment (underground stations) and AMIDA [Ami08] projects in the form of meeting data (humans sitting by a table and having a meeting). Since the number of all possible classes in data and their inner-class variability is too high, we have chosen and limited our solution to only few classes of trajectories. These classes included so-called Speech Supporting Gestures (SSG) and Abnormal Human Trajectories (AHT). The choice was made with the motivation of improvement of the speaker identification in the meeting data and of identification of generally abnormal behavior of humans in the public areas.

Our approach can be subdivided into several parts. The first part of the approach was to obtain the object trajectories through video sequence processing. Next part is dynamic (pre)processing of the trajectories. Finally, the training and classification is the last part of the processing chain.

2.OBJECT TRAJECTORIES

The aim of this part is to find positions and sizes of the objects that we can affirm to correspond to body parts. This approach has been widely investigated as mentioned above but let us briefly mention the approaches used in the experiments.

Object Localization

The localization task starts with the color-based segmentation. A Single Gaussian Method trained from two sets (skin and non-skin pixel colors) outputs the Skin Probability Image (SPI) where connected components are possible occurrences of ROIs. Every region area is evaluated and only the ones with the largest size are considered.

Once these regions are found we need to assign them to the body parts presence of which is expected in the image. For the available meeting data, simplified approach has been taken so the number of persons in every meeting data sequence is assumed to be two as well as it is assumed that in the beginning of each one, both persons remain in regular positions (head above hands and hands not switched while each person occupies his/her “half” of the image). In the public environment data, every moving object is considered to be a moving human. This approach has proven to be sufficient for the further presented steps. (See Figure 1 for the meeting data approach.)

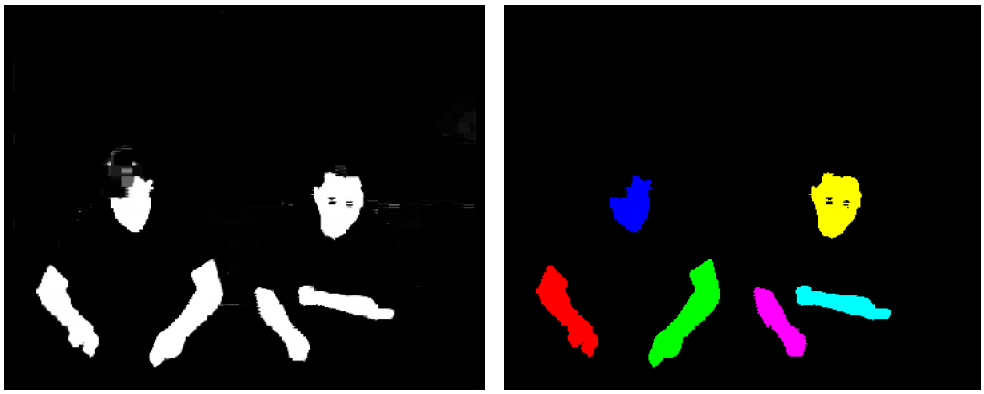


Figure 1. Skin probability and connected components evaluation

Object Detection

The positions of participants’ hands can be refined by localizing palms instead of hands (entire skin-colored regions). Let us assume the sub-image containing hand which has been multiplied by appropriate SPI. In such a sub-image only a flat region of skin color will occur corresponding to the elbow (and potentially arm) and a region with a certain number of protrusions as a consequence of inter-finger distance and shadows. Then, local maxima clusters feature finger region in this sub-image convolved with kernels. (See Figure 2 for illustration of the approach.)

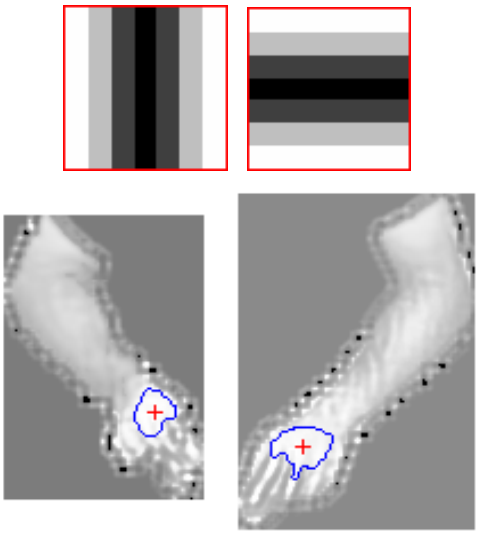


Figure 2. Convolution kernels and hand localization

Object Tracking

Once the regions are identified, they are consequently tracked so their trajectories are found.

From all of the inquired algorithms for object tracking, the Overlapping Boxes Method [Wah07] was chosen as it showed the best results on the meeting data. The essence of this method is that in two subsequent images, the two occurrences of the same object lie so close to each other that the bounding boxes overlap.

However, this algorithm cannot solve the occlusion problem. The problem of occlusion could be solved by modeling of motion by KLT filter presented e.g. in [Hra06]. In the future work, more tracking algorithms should be investigated; however, for the experiments shown below, the simple tracking algorithms are sufficient.

In case of public areas, the input data in the form of positions of humans were available along with the data set so the positions of the humans in the video were known in advance and no need for application of tracking algorithms existed.

3.TRAJECTORY PREPROCESSING

The trajectory can be described as a timed sequence of tuples [*x*, *y*] *x*, and *y* represent positions of the object. For the classification purposes, it is not clear, whether the motion of objects is characterized by its absolute position, speed of changes of the position (first derivative of position), or perhaps even acceleration (second derivative of the position). Additionally, co-ordinate system might be of interest in this context. In the presented approach, the quadruple [*x*, *y*, *dx*, *dy*], where *x*,*y* represents positions of object and *dx*, *dy* represent its velocity, was chosen as a good compromise.

Classification process selects trajectories (represented here through a sequence of the above mentioned quadruples) that do belong to a certain class (or classes). However, as we need to classify only parts of trajectories that are characteristic for the class (classes) of interest. A problem can occur in situation where the trajectory is relatively long and/or in cases where the characteristic motion is preceded or bound with other motion. This fact can lead in a need to cut the trajectories.

The method of choosing the best position of cut is unknown in general, but it depends on the application. The usual way is to divide the trajectory on uniform overlapping parts. Another approach assumes its division according to quiescent state of trajectory (trajectory has small energy).

By examining the trajectories a certain degree of jitter was discovered. This might cause unsatisfactory result in the recognition stage. So the Double Exponential Filtering was worked in. This filter makes the positions of tracked objects stable when in a rest phase preventing unwanted sub-trajectories from being segmented out by the Activity Measure Method (AM). In a dynamic phase (hand movement, etc.) it ensures that a motion trend from previous images is taken into account resulting in smoother trajectory.

In case, when we want to process position coordinates of the trajectory, it is necessary to normalize it. The normalization process consists of alignment of all coordinates according to mean position.

4.TRAINING AND CLASSIFICATION

As mentioned in the previous part, the purpose of classification is to decide whether a trajectory or a sub-trajectory belong to a certain class. Such a sub-trajectory may represent e.g. gesture.

For the experimental purposes, a set of trajectories was manually segmented and modeled using one Gaussian Mixture Model M for each of the two basic classes we had decided to work with: one for those led in horizontal direction (M_H) and one for those in vertical (M_V). The basic models of this situation are shown in Figure 3.

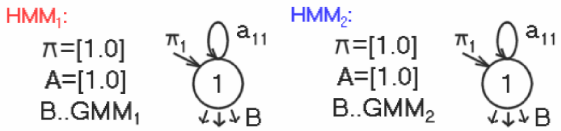


Figure 3. Hidden Markov Model corresponding with single Gaussian Mixture Model

The Figure describes Hidden Markov Model with the unitary transition probabilities and the emitting probability given by Gaussian Mixture Model.

For every unknown potentially infinite sequence $O(n) = (o_1, o_2 .. o_n)$ the log-likelihood of being emitted by the horizontal or vertical model can be computed as follows:

$$p(O|M_X)=\sum_{i=1}^{n}\frac{-\log p(o_i|M_x)}{n}$$

As an auxiliary metrics for validation of O belonging in SSG class we have defined periodicity of O in this manner: suppose some subsidiary vector $w = (w_1, w_2 .. w_Z)$ elements of which (w_i) are indices of winning distributions in model M_X for o_i where M_X stands for the model with higher $p(O \mid M_X)$. The periodicity is then the number of sub-sequences ($w_i, w_{i+1} .. w_j$) for which the following conditions are met:

$i \geq 1, j \leq Z, j - i \geq C$

$$\forall k=i..(i-1): w_k=w_{k+1}$$

$w_k = w_{k+1}$ where $k = i .. j - 1$

$w_{i-1} \neq w_i$

$w_{j+1} \neq w_j$

In the first condition we can find a constant C which defines the minimal length of such a sub-sequence (period).

As it has been observed that the higher the number of periods is the more probably the unknown sequence will be a SSG class representative.

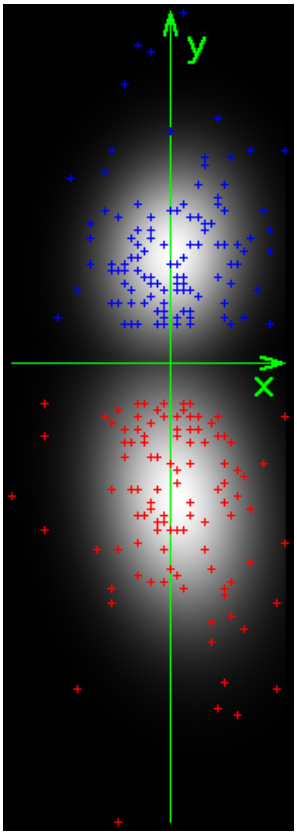
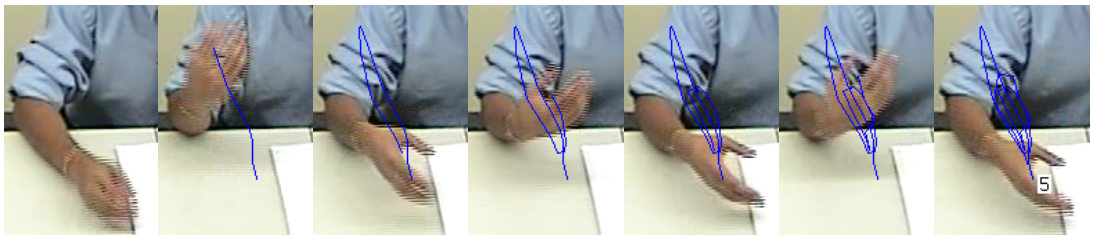


Figure 4. Gaussian Mixture Model for "vertical gestures" sub-class

The Gaussian Mixture Models describing two SSG and non-SSG are shown in Figure 4. The red colored cluster (in bottom part of picture) denotes non-SSG clusters and blue cluster denotes SSG gestures.



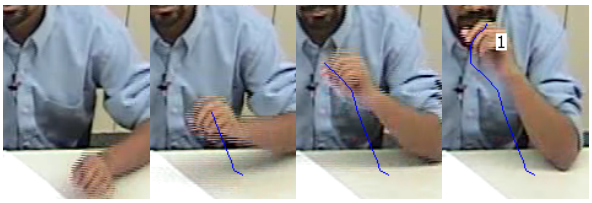


Figure 5. Examples of Speech Supporting Gesture and no-gesture

The training and evaluation of speech supporting gestures was done on data set containing about 30 trajectories with average length of 20 coordinates. The accuracy of classification was about 62%.

However, the single state HMM cannot differ more complex gestures. The second experiment was done with data from a public environment. For single scene was defined five simple classes which denoted different person behavior. For example person going thru turnpike (e.g. entering platform) and person going around turnpike.

The classification algorithm for Hidden Markov Model M is realized by the Viterbi Algorithm [Cer03] and it is defined as finding of path thru model M with best response:

$$P^*(O|M) = \max_X P(O, X|M)$$

The probability of passing of Object O thru a model M by way X is described by following equation, whereas $a_{k,j}$ denotes transition probabilities between states. Except first and last state, states are emiting or generating output probability density function $b_j(o(t))$.

$$P(O, X|M) = a_{x(o)x(1)} \prod_{n=1}^{T} b_{x(t)}(o_t) a_{x(t)x(t+1)}$$

For this case it was chosen more complex HMMs and the topology of each of them is shown in Figure 5.

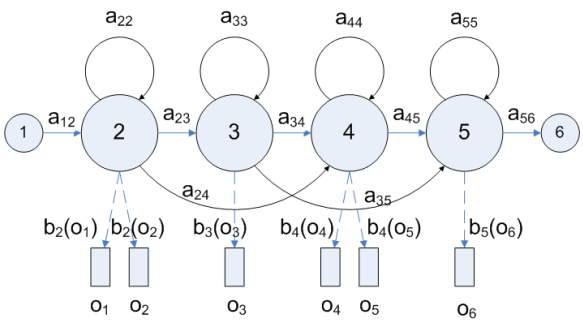


Figure 5. Example of non trivial Hidden Markov Model

Training and evaluation was performed on about 800 trajectories. The trajectory correct class was recognized with accuracy about 90% on the given data.

5.CONCLUSIONS

The work presented in this paper focused on classification of hand gestures and human motion. It has been shown that the classification of the trajectories through the HMM classifier is feasible and can lead into good results.

The single state HMM is well for recognition of gesture parts or simple gestures with single direction. Opposite, more complex gestures are very interesting from point of human behavior analysis and understanding. However, definition and training of complex gestures is more difficult.

However, we believe that the presented approach is reasonably well working and generally usable.

Further work is needed specifically in the field of refinement the sub-trajectories selection and also in refinement of representation of the trajectories.

The performance of the presented algorithm is affected by the segmentation and tracking methods. And the best working methods to be used in the presented approach have yet to be evaluated.

The experiments has shown, that the HMM based approach is suitable for sequential data like are gesture trajectories.

6.ACKNOWLEDGMENTS

This work has been supported by AMIDA - Augmented Multi-party Interaction with Distance Access, EU-6FP-IST, IST-033812-AMIDA, by the Ministry of Education, Youth and Sports of the Czech Republic under the research program LC-06008 (Center for Computer Graphics), and by the research project Security-Oriented Research in Information Technology, MSM0021630528.

7.REFERENCES

[AC99] Aggarwal, J. K., Cai, Q.: Human motion analysis: a review.Comput. Vis. Image Underst., 73(3):428–440, 1999.

[Ami08] Web site. Project: Augmented Multi-party Interaction with Distance Access (AMIDA), 2008. http://www.amiproject.org/

[Bay08] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. 2008. Speeded-Up Robust Features (SURF). *Comput. Vis. Image Underst.* 110, 3 (Jun. 2008), 346-359.

[Bis06] Bishop, C. M. 2006. Pattern Recognition and Machine Learning(Information Science and Statistics). Springer-Verlag New York,Inc., Secaucus, NJ, USA.

[Bor08] Borza P.-V.: Motion-based Gesture Recognitionwith an Accelerometer, Bachelor's Thesis in Computer Science, Babeş-Bolyai University of Cluj-Napoca, Romania, 2008

[Car08] Web site. Caretaker. Information Society Technologies, 2008. http://www.ist-caretaker.org/.

[Cer03] Černocký, J. 2003. Hidden markov models – an introduction.Tech. rep., Brno University of Technology, Faculty of Information Technology.

[CV05] Antoni B. Chan and Nuno Vasconcelos. Mixtures of dynamic textures. In ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, pages 641–647, Washington, DC, USA, 2005. IEEE Computer Society.

[Has06] N. Hassink and M.G. Schopman, "Gesture recognition in a meeting environment," Feb. 2006; http://essay.utwente.nl/56211/.

[Hra06] Hradiš Michal, Juránek Roman: Real-time Tracking of Participants in Meeting Video, In: Proceedings of The 10th Central European Seminar on Computer Graphics, Budměřice, SK, 2006, p. 5

[Jir08] Jiřík, L.: Recognition of poses and gestures. Master's thesis, Brno University of Technology, Faculty of Information Technology, 2008.

[Jir06] Jiřík, L.: Detection Human Body Parts. Bachelor's thesis, Brno University of Technology, Faculty of Information Technology, 2004.

[Mli08a] Mlích, J., Chmelař, P.: Trajectory classification based on hidden markov models. In Proceedings of 18th International Conference on Computer Graphics and Vision, pages 101–105. Lomonosov Moscow State University, 2008.

[Mli08] Mlích, J:. Object tracking in video sequences. Master's thesis, Brno University of Technology, Faculty of Information Technology, 2008.

[Nam96] Y. Nam and K. Wohn, "Recognition of space-time hand-gestures using hidden Markov model," ACM Symposium on Virtual Reality Software and Technology,pp. 51–58, 1996.

[Sta95] Starner, T.: Visual recognition of American Sign Languageusing hidden Markov models. Master's thesis, 1995 .

[Rig97] G. Rigoll, A. Kosmala, and S. Eickeler, "High Performance Real-Time Gesture Recognition Using Hidden Markov Models," In Proc. Gesture Workshop,, pp. 69--80, 1997.

[Row97] Roweis, S., Ghahramani, Z.:A unifying reviewof linear Gaussian models. Tech. rep., 6 , King's College Road,Toronto M5S 3H5, Canada, 1997.

[Sta00] Stauffer, C., Grimson, et al.: Learning patterns of activity using real-time tracking. IEEE Transactions onPattern Analysis and Machine Intelligence 22, 8, 747–757. 2000.

[You06] Young, S., Odell, et al.: The HTK Book, University Of Cambridge, UK, 2006

[WLD06] Wang S.B, Quattoni A., et. at.: Hidden Conditional Random Fields for Gesture Recognition, Computer Science and Artificial Intelligence Laboratory, MIT , 2006.

[Wah07] Wahde, M. a kol. Computer Vision Based System for Dynamic Gesture Recognition. Göteborg, Technical Report, Sweden, 2007.

Generalized Roller

Michal Haindl & Martin Hatka
Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic
CZ18208, Prague, Czech Republic
{haindl,hatka}@utia.cz

ABSTRACT

This paper describes a generalization of our previously published simple roller method for seamless enlargement of colour textures such as natural bidirectional texture functions (BTF) that realistically represent appearance of given material surfaces. The generalized roller allows automatic detection of major texture periodicity directions which do not need to be aligned with coordinate axes. The roller texture synthesis method is based on the overlapping tiling and subsequent minimum error boundary cut. One or several optimal double toroidal BTF patches are seamlessly repeated during the synthesis step. While the method allows only moderate texture compression it is extremely fast due to complete separation of the analytical step of the algorithm from the texture synthesis part. The method is universal and easily implementable in a graphical hardware for purpose of real-time rendering of any type of static or dynamic textures.

Keywords
Texture Enlargement, Texture Sampling.

1 INTRODUCTION

A realistic physically correct visualization of virtual objects with real material surfaces require to map 3D shapes with genuine nature-like colour textures. However, the appearance of real materials dramatically changes with illumination and viewing variations. Thus the only reliable way for a material visual properties representation is to capture its reflectance in as wide range of light and camera position combinations as possible. This is the principle of the recent most advanced texture representation the Bidirectional Texture Function (BTF) [DvGNK99, HF07]. The BTF textures which have rugged surfaces do not obey the Lambert law and their reflectance is illumination and view angle dependent. The purpose of a synthetic BTF texture is to reproduce and enlarge a given real digitized texture image so that ideally both natural and synthetic texture will be indiscernible under any illumination or viewing condition. BTF types of textures which occur in virtual scenes models can be either digitized natural textures or textures synthesized from an appropriate mathematical model.
However modelling of a natural texture is a very challenging and difficult task, due to unlimited variety of possible surfaces, illumination and viewing conditions simultaneously with the strong discriminative functionality of the human visual system.

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

The related BTF texture modelling approaches may be divided primarily into intelligent sampling and model-based-analysis and synthesis [HF07], but no ideal method for texture modelling exists. Each of the existing approaches or texture models has its advantages and simultaneously limitations and thus the optimal texture modelling method always depends on the application and material to be modeled.

Model-based texture synthesis [Bes74, Kas81, Che85, BK98, BK99, Hai91, HH98, HH00, ZLW00, HH02, GH03a, HGS*04, HF07] requires non-standard multi-dimensional (3D for static colour textures and even 7D for static BTFs) models. If such a texture space can be factorized then these data can be modeled using a set of less-dimensional random field models [HF07], but in any case such models are non trivial and they suffer with several unsolved problems which have to be circumvented. Among such possible models the Gaussian Markov random fields [HF03] are advantageous not only because they do not suffer with some problems of alternative options (see [Hai91, Hai00, HH00] for details) but they are also relatively easy to synthesize and still flexible enough to imitate a large set of natural and artificial textures. Unfortunately real data space can be decorrelated only approximately, hence this approach suffers with some loss of image information. Alternative full nD models allow unrestricted spatial-spectral correlation modelling, but its main drawback is large amount of parameters to be estimated and in the case of Markov models also the necessity to estimate all these parameters simultaneously. Model-based methods are also mostly too difficult to be implemented in modern graphical card processors.

Intelligent sampling approaches [DB97, EL99, EF01, HB95, XGS00, LLX*01, LYS01, DC02, TZL*02] rely on sophisticated sampling from real texture measurements. Given a randomly selected starting block of texture in the image, they propagate out from it selecting new texture blocks. For each new block in the image, all neighboring blocks that have already been generated are checked and the example image (or images) is searched for similar textures. The k best such matches are found and then randomly chosen the corresponding new texture patch from among them. The methods [EF01, EL99, WL01] all vary in how the blocks are represented, how similarity is determined, and how the search is performed.

Intelligent sampling approaches, such as the presented method, are based on some sort of original small texture sampling and the best of them produce very realistic synthetic textures, usually better than model-based methods. However these methods require to store original texture sample, often produce visible seams, they are mostly computationally demanding, they cannot generate textures unseen by the algorithm, and they cannot approach the large compression ratio of model-based methods. This paper describes a generalization of our previously published simple roller method [HH05] for seamless enlargement of colour textures. The generalized roller allows automatic detection of major texture periodicity directions which do not need to be aligned any more with the coordinate axes.

The rest of the paper is organized as follows. The following section describes a simple sampling approach based on the repetition of a double toroidal tile carved from the original texture measurement. The algorithm is summarized in the section 3. Results are reported in the section 4, followed by conclusions in the last section.

2 DOUBLE TOROIDAL TILE

BTF is typically measured in the form of several thousand images covering many combinations of illumination and viewing angles, however, such measurements have huge size what has prohibited their practical exploitation in any sensible application until recently. The BTF data space for a single material (e.g. wood Fig.7) represents giga bytes of data for recent BTF measuring setups [MMK03]. Even though any sampling based method cannot approach compression ratio of alternative probabilistic BTF methods, it is still important to select double toroidal tiles as small as possible to compress these huge original measurements. We assume mutually well registered data of the size $N \times M$ for fixed viewing angle and changing illumination angle hence the algorithm is repeated for every viewing angle but produces simultaneously identical tiles for all illumination angles.

Periodicity Detection

The methods starts with the detection of two major texture periodicity directions. We compute the Fourier amplitude spectrum (Fig.2) from the grey-scale version of the perpendicularly illuminated input multispectral texture component. Two largest Fourier coefficients with angular difference larger than 10 degrees specify bisectors of two sectors Fig.1-bottom. Using the maximum correlation approach in these two sectors we find magnitude and direction $\vec{u} = (u_1, u_2), \vec{v} = (v_1, v_2)$ for each major texture periodicity. If there are several local maxima in a given sector we choose the smallest periodicity maximum. These two directional vectors simultaneously specify two edges of the rhomboid to which is the double toroidal tile inscribed. The minimal rhomboid to which the tile is inscribed is limited not only by the sample spatial frequency content but also by the size of BTF measurements and the number of toroidal tiles we are looking for (n).

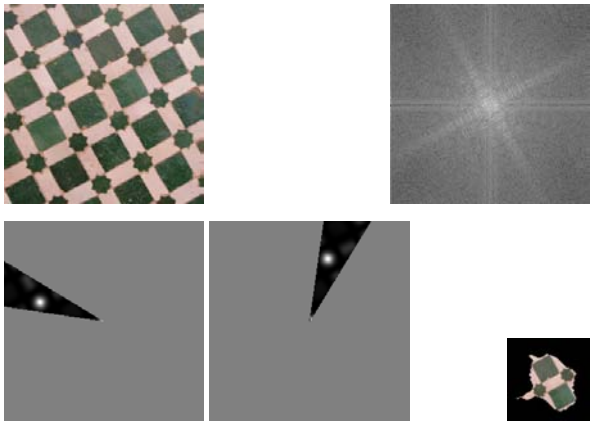


Figure 1: Original measured texture and its amplitude spectrum (upper row), detected spatial correlation sectors (bottom row) and the resulting toroidal tile.

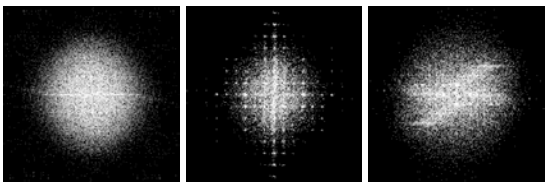


Figure 2: The amplitude Fourier spectra of the corduroy (Fig.8), proposte and wool textures, respectively.

Overlapping

The double toroidal tile (see Figs.1-bottom right,3-middle,) is limited by the selected minimal rhomboid to be inscribed in from the original texture measurement. The texture tile is assumed to be indexed on the regular two-dimensional toroidal lattice. The optimal lattice searched by the algorithm allows for seamless repetition in both $\vec{u}$ and $\vec{v}$ directions, respectively.

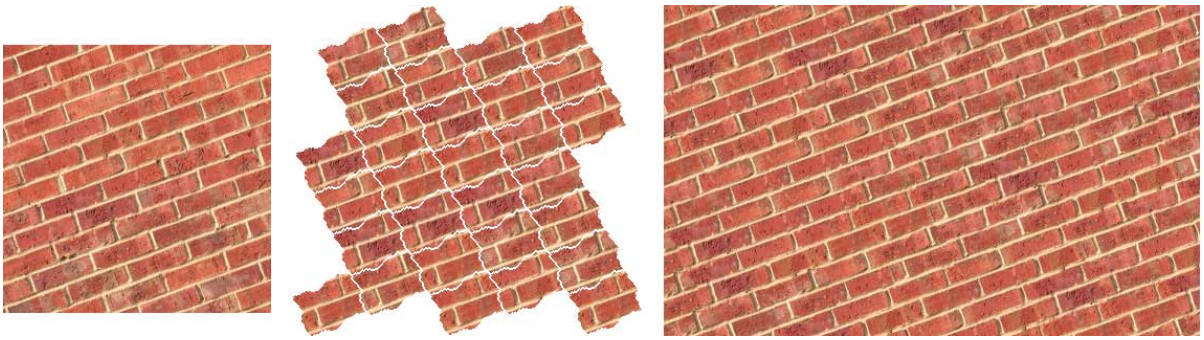


Figure 3: The roller principle - left input texture (left), combined toroidal tiles (middle), and the result (right), respectively.

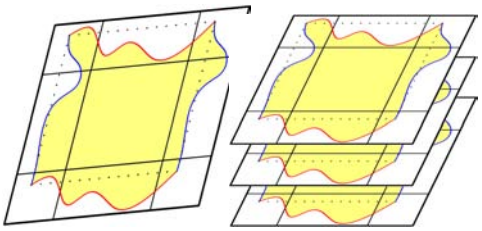


Figure 4: The optimal tile cuts in both directions (left) and the multiple tiles cut (right).

The multiindex r has two components $r = [r_1, r_2]$, the first component is row and the second one column index, respectively. Let us define the overlap error for a pixel r as follows:

$$\psi_r^u = \left(Y_r - Y_{r+[N,0]-\vec{u}}\right)^2 \quad \forall r \in I_u \ , \tag{1}$$

$$\psi_r^v = \left(Y_r - Y_{r+[0,M]-\vec{v}}\right)^2 \quad \forall r \in I_v \ , \tag{2}$$

where Y_r denotes a multispectral pixel indexed on the $N \times M$ underlying lattice and the index sets I_h, I_v are defined

$$I_u = (1,\ldots,u_2) \times (1,\ldots,M) \ ,$$
$$I_v = (1,\ldots,N) \times (1,\ldots,v_1) \ .$$

Optimal Cut

The optimal cuts for both the $\vec{u}$ and $\vec{v}$ edge is searched using the A^* algorithm. This suboptimal search method allows fast selection of both tile cuts. However for most applications the fast synthesis is prerequisite while the computation time for separately solved analytical part is of no significant importance. Both optimal cuts have to minimize the overall path error

$$\Psi_r^u = \psi_r^u + \min\left\{\Psi^u_{r-[1,1]}, \Psi^u_{r-[0,1]}, \Psi^u_{r+[1,-1]}\right\} \ ,$$
$$\Psi_r^v = \psi_r^v + \min\left\{\Psi^v_{r-[1,1]}, \Psi^v_{r-[1,0]}, \Psi^v_{r+[-1,1]}\right\} \ .$$

The combination of both optimal directional cuts creates the toroidal tile as is demonstrated on the Fig.4.

Multiple Tiles

Some textures with dominant irregular structures cannot be modeled by simple repetition of only one tile without clearly visible and visually disturbing regularly repeated effects. These textures are modeled by random changing of several tiles Figs.5,6 which have identical tile borders but different content. Similar tile rhomboids are searched using the correlation measure and such rhomboids are stacked and simultaneously optimally cut.

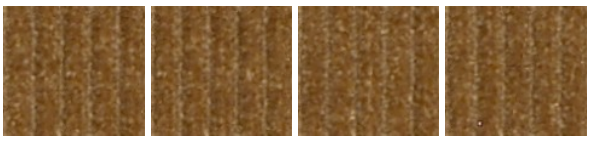


Figure 5: Multiple tiles for the BTF corduroy material.

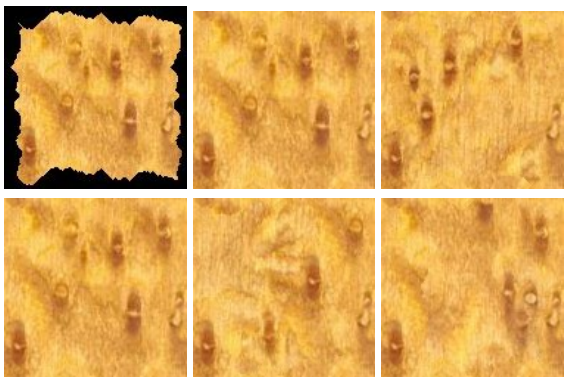


Figure 6: A tile and multiple rectangular tiles for the BTF wood material.

Synthesis

The synthesis of any required BTF texture size for a single tile case is simple repetition of the created double toroidal tile in both directions until the required texture is generated. There is no computation involved in this step hence it can be easily implemented in real time or inside the graphical card processing unit (GPU). In the

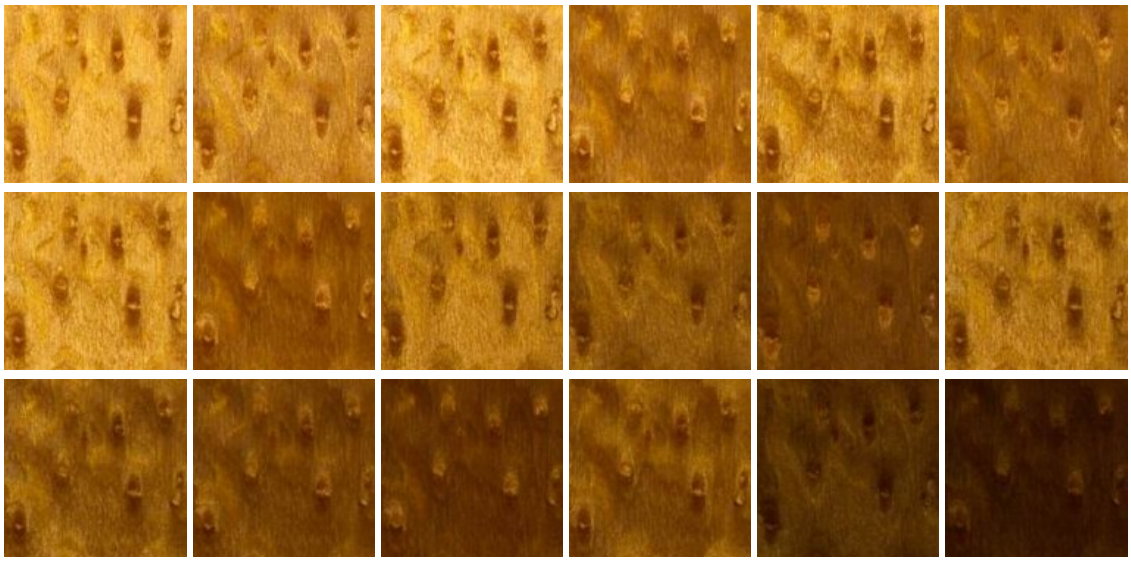


Figure 7: BTF wood rectangular tiles for different illumination angles.

case of several mutually interchangeable tiles we need a uniform random generator to decide which tile will follow. This additional computation is very simple and can be realized inside the GPU as well.

3 THE GENERALIZED ROLLER ALGORITHM

The completely automatic roller algorithm is as follows:

- Analysis
  1. Find the minimal inscribed rhomboid(s).
  2. Find the optimal vertex r^*.
  3. Search for optimal major periodicities directional cuts starting from r^* with final points in the corresponding rhomboid vertices.
  4. Create the double toroidal texture tile(s).
- Synthesis

The number of tiles is the only parameter specified by the user. The analytical part is completely separated from the synthesis. The most time consuming part of the analysis is the minimal tile specification together with its position in the input texture sample. In the worst case ($f_{r_1} = f_{r_2} = 2$) it is proportional to

$$T \propto MN^2 \frac{u_2}{2}(N - \frac{u_2}{2}) + NM^2 v_1 (M - \frac{u_2}{2}) \ .$$

The optimal cuts search time requirement is proportional to

$$T \propto v_1^2 \frac{u_2}{2} N - 2 v_1^2 \frac{u_2}{2}^2 + M v_1 \frac{u_2}{2}^2$$

and interior replacement for multiple tiles needs:

$$T \propto (M - n_{col})(N - n_{row})(n_{col} \frac{u_2}{2} - n_{row} v_1) \ .$$

The optimal tile size evaluation time is in most graphical applications not important while the important synthesis step contains either no computations at all or only uniform number generation.

4 RESULTS

We have tested the algorithm not only on BTF textures (e.g. Figs.8,9,10) but also on several hundred colour and grayscale textures from the VisTex database [PGM*95], Noctua database, Brodatz textures [Bro66] and mainly from our extensive texture database, which currently contains over 1000 colour textures. Tested textures were either natural such as bark, wood, plants, water, etc., or man-made knitwear, upholstery, brick wall Fig.11, textiles, food products and many others.

Figure 8: Enlarged BTF corduroy material.

 ISBN 978-80-86943-94-7

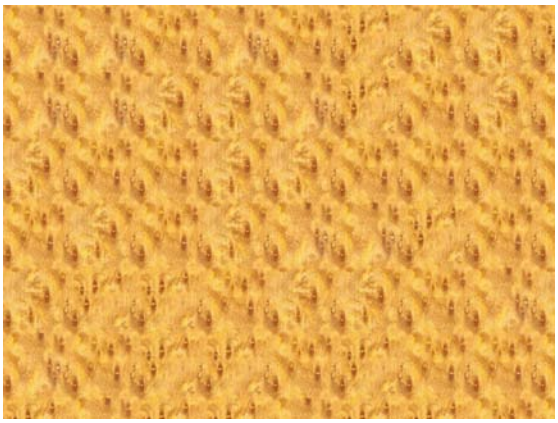


Figure 9: Enlarged BTF wood material.

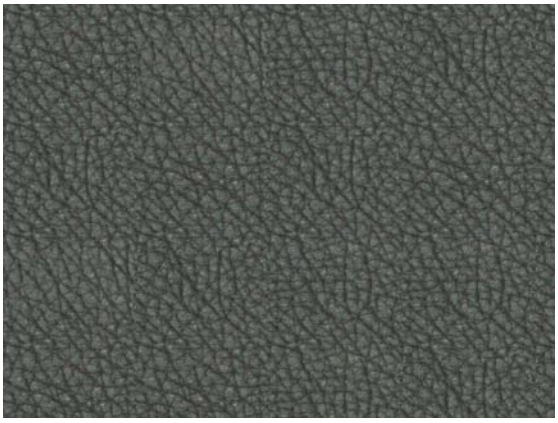


Figure 10: Enlarged BTF foil texture.

Several of these results (brick wall Fig.11, rattan Fig.12, leafs Fig.13, jeans cloth Fig.14, tiles Fig.16, text Fig.17, and plastic fence Fig.18) are demonstrated in the following images. Each image shows the input texture, one toroidal tile example, and the resulting synthetic texture, respectively. Such unusually extensive testing was possible due to simplicity and efficiency of the algorithm and it allowed us to get insight into the algorithm properties. BTF data we use are from the University of Bonn [MMK03]. We have tested the algorithm on BTF colour measurements such as upholstery, lacquered wood, knitwear or leather textures. Each BTF material sample comprised in the Bonn database is measured in 81 illumination and viewing angles, respectively and has resolution 800×800. Figs.8,9,10 demonstrate synthesized results for three different materials: fabric, wool and leather. Fig. 9 illustrates the lacquered wood smooth BTF synthesis.

Resulting textures are mostly surprisingly good for such a very simple algorithm. For example our results on the text texture (Fig.17) are indistinguishable (see [EF01]) from results on the same texture using much more complicated and slower image quilting algorithm [EF01].

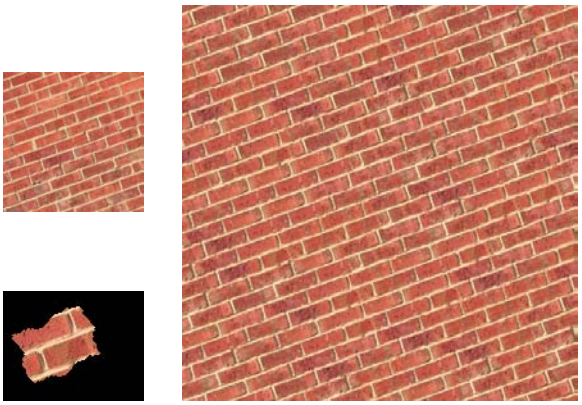


Figure 11: Enlarged brick texture.

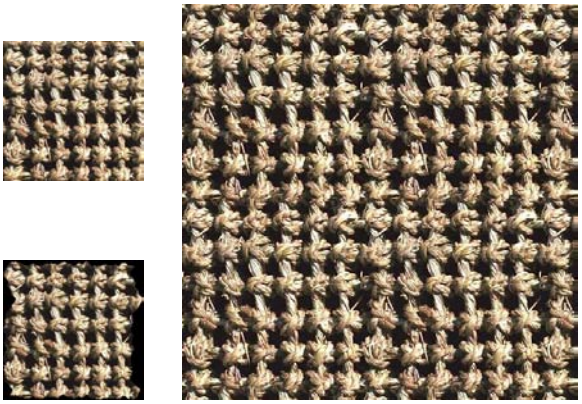


Figure 12: Enlarged rattan texture.

Figure 13: Enlarged leaves texture.

Obviously there is no optimal texture modelling method and also the presented method fails on some textures as can be seen on the following metal sheet example (Fig.19). But, this texture can be successfully modeled using the alternative probabilistic Markovian model [HH00]. The method also fails on textures with distinctive perspective distortion. However on most of our failure examples also most alternative intelligent sampling methods tested failed (e.g., [EF01] failed on the same metal sheet example on Fig.19 as well).

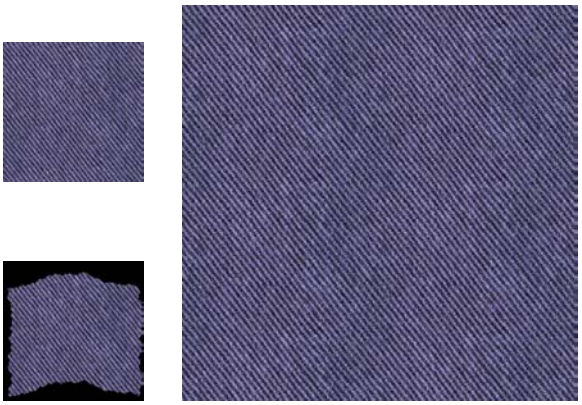


Figure 14: Enlarged jeans fabric texture.

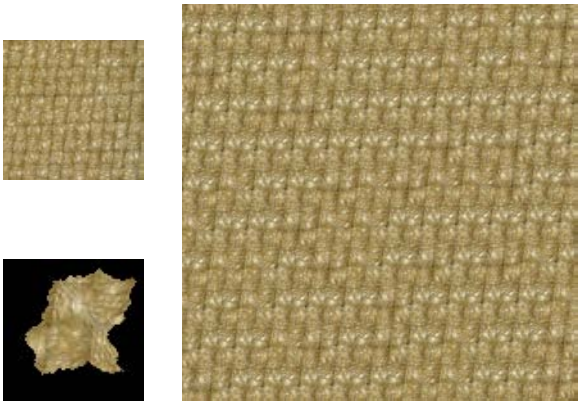


Figure 15: Enlarged woven matting texture.

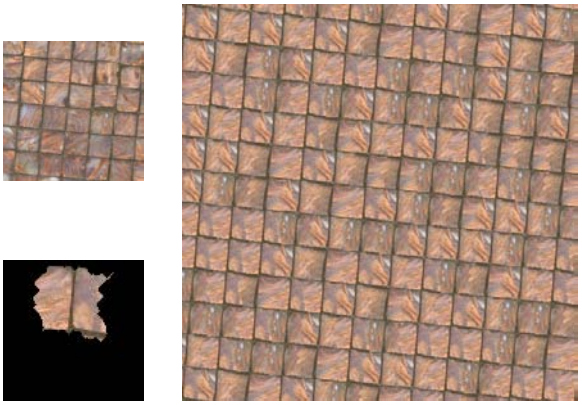


Figure 16: Enlarged shiny square tile texture.

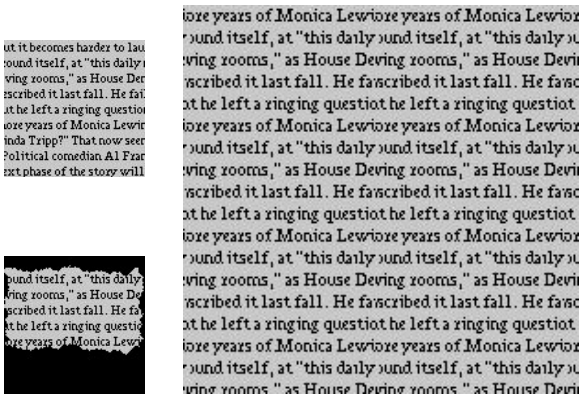


Figure 17: Enlarged newspaper texture.

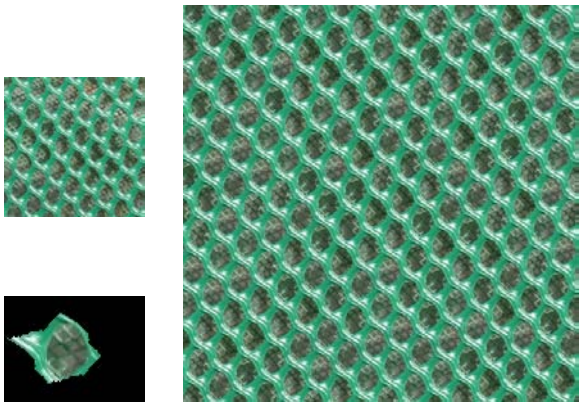


Figure 18: Enlarged plastic fence netting texture.

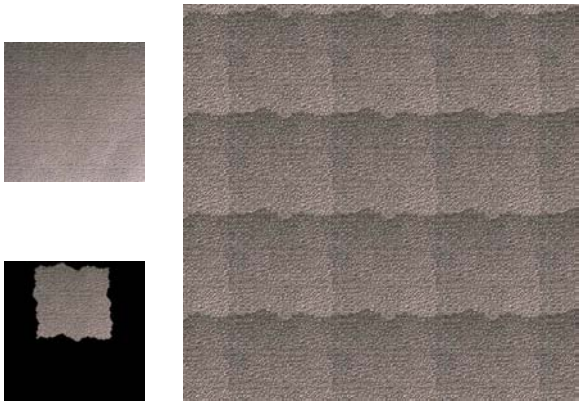


Figure 19: Metal sheet texture enlargement failure.

5 CONCLUSIONS

The test results of our algorithm on available BTF data are visually indiscernible for all our BTF data and also for most of tested colour textures. The test results of our algorithm on our extensive natural texture collection are encouraging. The presented method is extremely fast, fully automatic, very simple and easily implementable even in the graphical processing unit. The method offers only moderate compression ratio ($\approx 1:4$) for transmission or storing texture information while it has negligible computation complexity. Another drawback of the method is that it does not allow a BTF data space restoration or modelling of unseen (unmeasured) BTF space data unlike some probabilistic BTF models.

The roller method can be used for easy and fast seamless synthesis of any required texture size for many natural or man made BTF textures and their use in complex virtual reality scenes (Fig.20). The method's extension for alternative texture types or some other spatial data such as the reflectance models parametric spaces is straightforward.

6 ACKNOWLEDGEMENTS

This research was supported by the project no. 102/08/0593 of the Grant Agency of the Czech Repub-

Figure 20: Chair retreat with several synthesized textures (left), detail (middle), and detail with tiled textures (right), respectively.

lic and partially by the MŠMT grants 1M0572 DAR, 2C06019.

7 REFERENCES

[Bes74] BESAG J.: Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B B-36*, 2 (February),pp.192–236, 1974.

[BK98] BENNETT J., KHOTANZAD A.: Multispectral random field models for synthesis and analysis of color images. *IEEE Trans. on Pattern Analysis and Machine Intelligence 20*, 3 (March),pp.327–332, 1998.

[BK99] BENNETT J., KHOTANZAD A.: Maximum likelihood estimation methods for multispectral random field image models. *IEEE Trans. on Pattern Analysis and Machine Intelligence 21*,No. 6, pp.537–543, 1999.

[Bro66] BRODATZ P.: *Textures: A Photographic Album for Artists and Designers*. Dover Publications, 1966.

[Che85] CHELLAPPA R.: Two-dimensional discrete gaussian markov random field models for image processing. In *Progress in Pattern Recognition 2* (North-Holland), Kanal L., Rosenfeld A., (Eds.), Elsevier, pp.79–112, 1985.

[DB97] DE BONET J.: Multiresolution sampling procedure for analysis and synthesis of textured images. In *ACM SIGGRAPH 97*, ACM Press, pp. 361–368, 1997.

[DC02] DONG J., CHANTLER M.: Capture and synthesis of 3d surface texture. In *Texture 2002* , vol. 1, Heriot-Watt University, pp. 41–45, 2002.

[DvGNK99] DANA K., VAN GINNEKEN B., NAYAR S., KOENDERINK J.: Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics 18*, 1 (January), pp.1–34, 1999.

[EF01] EFROS A. A., FREEMAN W. T.: Image quilting for texture synthesis and transfer. In *ACM SIGGRAPH 2001*, Fiume E., (Ed.), ACM Press, pp. 341–346, 2001.

[EL99] EFROS A. A., LEUNG T. K.: Texture synthesis by non-parametric sampling. In *Proc. Int. Conf. on Computer Vision (2)* (Corfu, Greece), pp. 1033–1038, 1999.

[GH03a] GRIM J., HAINDL M.: Texture modelling by discrete distribution mixtures. *Computational Statistics Data Analysis 41*, 3-4 (January), pp.603–615, 2003.

[Hai91] HAINDL M.: Texture synthesis. *CWI Quarterly 4*, 4 (December), pp.305–331, 1991.

[Hai00] HAINDL M.: Texture modelling. In *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics* (Orlando, USA, July 2000), Sanchez B., Pineda J. M., Wolfmann J., Bellahse Z.,, Ferri F., (Eds.), vol. VII, International Institute of Informatics and Systemics, pp. 634–639, 2000.

[HB95] HEEGER D., BERGEN J.: Pyramid based texture analysis/synthesis. In *ACM*

SIGGRAPH 95, ACM Press, pp. 229–238, 1995.

[HF03] HAINDL M., FILIP J.: Fast BTF texture modelling. In *Texture 2003. Proceedings* (Edinburgh), Chantler M., (Ed.), IEEE Press, pp. 47–52, 2003.

[HF07] HAINDL M., FILIP J.: Extreme compression and modeling of bidirectional texture function. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, No.10, pp.1859–1865, 2007.

[HGS*04] HAINDL M., GRIM J., SOMOL P., PUDIL P., KUDO M.: A gaussian mixture-based colour texture model. In *Proceedings of the 17th IAPR Int. Conf. on Pattern Recognition* (Los Alamitos), Kittler J., Petrou M.,, Nixon M., (Eds.), vol. 3, IEEE Press, pp. 177–180, 2004.

[HH98] HAINDL M., HAVLÍČEK V.: Multiresolution colour texture synthesis. In *Proceedings of the 7th International Workshop on Robotics in Alpe-Adria-Danube Region* (Bratislava), Dobrovodský K., (Ed.), ASCO Art, pp. 297–302, 1998.

[HH00] HAINDL M., HAVLÍČEK V.: A multiresolution causal colour texture model. In *Advances in Pattern Recognition, Lecture Notes in Computer Science 1876*, Ferri F. J., Inesta J. M., Amin A.,, Pudil P., (Eds.). Springer-Verlag, Berlin, ch. 1, pp. 114 –122, 2000.

[HH02] HAINDL M., HAVLÍČEK V.: A multiscale colour texture model. In *Proceedings of the 16th IAPR International Conference on Pattern Recognition* (Quebec City), Kasturi R., Laurendeau D.,, Suen C., (Eds.), vol. I, IEEE Press, pp. 255 – 258, 2002.

[HH05] HAINDL M., HATKA M.: A roller - fast sampling-based texture synthesis algorithm. In *Proceedings of the 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision* (Plzen), Skala V., (Ed.), UNION Agency - Science Press, pp. 93–96, 2005.

[Kas81] KASHYAP R.: Analysis and synthesis of image patterns by spatial interaction models. In *Progress in Pattern Recognition 1* (North-Holland), Kanal L., A.Rosenfeld, (Eds.), Elsevier, 1981.

[LLX*01] LIANG L., LIU C., XU Y.-Q., GUO B., SHUM H.-Y.: Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics (TOG) 20*,No.3, pp.127–150, 2001.

[LYS01] LIU X., YU Y., SHUM H.-Y.: Synthesizing bidirectional texture functions for real-world surfaces. In *ACM SIGGRAPH 2001*, Fiume E., (Ed.), ACM Press, pp. 97–106, 2001.

[MMK03] MESETH J., MÜLLER G., KLEIN R.: Preserving realism in real-time rendering. In *OpenGL Symposium*, Reiners D., (Ed.), Eurographics Association, Switzerland, pp. 89–96, 2003.

[PGM*95] PICKARD R., GRASZYK C., MANN S., WACHMAN J., PICKARD L., CAMPBELL L.: *VisTex Database*. Tech. rep., MIT Media Laboratory, Cambridge, 1995.

[PS00] PORTILLA J., SIMONCELLI E.: A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision 40*, No.1, pp.49–71, 2000.

[TZL*02] TONG X., ZHANG J., LIU L., WANG X., GUO B., SHUM H.-Y.: Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics (TOG) 21*,No. 3, pp.665–672, 2002.

[WL01] WEI L., LEVOY M.: Texture synthesis over arbitrary manifold surfaces. In *SIGGRAPH 2001*, ACM Press, pp.355–360, 2001.

[XGS00] XU Y., GUO B., SHUM H.: *Chaos Mosaic: Fast and Memory Efficient Texture Synthesis*. Tech. Rep. MSR-TR-2000-32, Redmont, 2000.

[ZLW00] ZHU S., LIU X., WU Y.: Exploring texture ensembles by efficient markov chain monte carlo - toward a "trichromacy" theory of texture. *IEEE Trans. on Pattern Analysis and Machine Intelligence 22*, No. 6, pp.554–569, 2000.

A fast SAH-based construction of Octree

Xin Yang*, Duan-qing Xu, Lei Zhao

College of Computer Science, Zhejiang University

Hangzhou,China
*xinyang@zju.edu.cn

ABSTRACT

Recent GPU ray tracers can already achieve performance competitive to that of their CPU counterparts. Nevertheless, these systems can not yet fully exploit the capabilities of modern GPUs and can only handle medium-sized, static scenes.

We present an octree construction algorithm for the GPU that achieves real-time performance by heavily exploiting the hardware, which has been observed to give superior performance in ray tracing compared to other acceleration structures. We use streaming construction with the surface area heuristic (SAH) that significantly increase the coherence of memory accesses during construction of the octree.

Keywords: **Raytracing, Octree, GPU, SAH**.

I. INTRODUCTION

Ray tracing is a technique for rendering pictures from a three-dimensional model by following the paths of simulated light rays through the scene. One of the most serious problems of ray tracing is that it requires a relatively large amount of computation time. While CPU performance has increased dramatically over the last few years, it is still insufficient for many ray tracing applications. Commodity computer graphics chips are probably today's most powerful computational hardware for the dollar. As a result of continued demand for programmability, modern graphics processing units (GPUs) such as the NVIDIA GeForce 8 Series are designed as programmable processors employing a large number of processor cores [1].

Lately, ray tracing running on GPU have developed to an excellent substitute to CPU-based ray tracers [2, 3]. However, even though optimized for the GPU architecture, these implementations can still not utilize the full power of modern GPUs. GĂźnther et al.[4] point out that, two main problems need to be addressed for gaining maximum performance from the GPU, such as the NVIDIA GeForce 8. First, one needs to keep only a small state per thread to allow for enough active threads to run to keep the GPU busy. The ray tracer of Popov et al. required too many live registers which resulted in a poor GPU utilization of below 33% [3]. Secondly, one needs to assure the coherent execution of threads running in parallel, due to the very wide SIMD architecture of current GPUs (32–48 units execute the same instructions [5]). Execution divergence (i.e. incoherent branching) can limit performance of ray tracing to around 40% of the graphics board's theoretical potential [2].

Because of accustomed acceleration structure only with a relatively small number of nodes at the upper levels, which makes parallelizing over nodes inefficient and leaves the massive parallelism of GPUs underexploited. So, we propose a new spatial partitioning that allows construction of an improved octree (subtrees) by threads independently. This approach takes advantage of the parallelism present by heavily exploiting the hardware for the GPU. Specifically, Kun Zhou et al.[6] firstly implement their parallel kd-tree algorithms in BFS (breadth-first search) order to fully exploit the fine-grained parallelism of modern GPUs at all stages of kd-tree construction. Our algorithm also builds octree nodes

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

in this way, which has been observed to give superior performance in ray tracing compared to other acceleration structures [7], when built using the surface area heuristic (SAH) [8].

Section 2 presents previous work in this area, while Section 3 introduces more formal definitions of our octree construction. Section 4 discusses our system implementation, analyzes the performance of our implementation and compare it with current work. Finally, in Section 5, we offer some thoughts on future work.

II. PREVIOUS WORK

In this section we give a brief overview of prior work on acceleration structures for ray tracing dynamic scenes.

2.1 Ray Tracing On Parallel Architectures.

There has been a significant interest in studying ray tracing on parallel architectures. Ray tracing on GPUs has stimulated much interest recently. [9] implemented ray-triangle intersection on the GPU. [10] designed the first ray tracer that runs entirely on the GPU, employing a uniform grid for acceleration. [11] introduced two stackless kd-tree traversal algorithms, which outperform the uniform grid approach.[12] implemented a limited GPU ray tracer for dynamic geometry based on bounding-volume hierarchies and geometry images. None of the above GPU ray tracers outperforms a well-optimized CPU ray tracer. Recently, two techniques [2, 3] achieved better performance than CPU ray tracers. Both techniques use stackless kd-tree traversal and packet tracing. Unfortunately these two techniques work for static scenes only. For dynamic scenes, most existing methods are CPU-based (e.g., [13, 14]). Kun Zhou et al.[6] first implemented their parallel kd-tree algorithms in BFS(breadth-first search) order to fully exploit the fine-grained parallelism of modern GPUs at all stages of kd-tree construction, achieves real-time performance. Shevtsov et al. [15] presented a shared memory architecture with many CPU-like cores, including recent multi-core CPUs. The algorithm first partitions the space into several balanced sub-regions and then builds a sub-tree for each sub-region in parallel and in DFS (depth-first search) order. The algorithm cannot be mapped well to GPU architecture because modern GPUs require 103 ~ 104 threads for optimal performance [5], orders of magnitude greater than the possible thread number on multi-core CPUs (e.g., four threads tested in the paper).

2.2 Acceleration Structures

The relative performance of different acceleration structures has been widely studied. Havran [16] compares a large number of acceleration structures across a variety of scenes and determines that the kd-tree is the best general-purpose acceleration structure for CPU raytracers. It would seem natural, therefore, to try to use a kd-tree to accelerate GPU ray tracing. Construction of high quality KD-tree is bandwidth hungry and computationally expensive task. Attempts to reduce time spent on kd-tree construction were performed using hybrid data structure combining kd-tree with bounding volumes [17]. Similar combining of BVH and spatial partitioning for increasing overall performance was made in [18]. However these approaches still lack parallel implementation and optimized traversal like MLRT [19] and thus demonstrate modest overall performance.

Our algorithm relies on a modified work stealing approach to ensure an high performance width-first octree computation. We speed-up and optimize the construction of (SAH based) octrees. It ensures all processing units progress simultaneity to the bottom of the octree, enabling to stop the algorithm at anytime while ensuring a balanced Octree exploration (it avoids to waste processing power in a deep exploration of an octree branch, while an other process stays several depth level behind). We build our octree with all stages running in parallel with minimal synchronization overhead allowing to exploit as many threads as available to achieve fast octree re-build from scratch every frame. So our method is an excellent strategy for interactive ray tracing of dynamic scenes, which does not require any prior information about vertices motion.

III. CONSTRUCTION OF IMPROVED OCTREE

Using high-quality acceleration structures is essential for achieving interactive ray tracing performance. In this section, we describe how to build our improved Octrees for ray tracing.

3.1 Conventional Octree Construction

Typically an octree [20] is a hierarchical data structure showing how objects are distributed in the object space, which has been mainly used in image processing or solid modeling areas, it was first used for ray tracing by Glassner [21]. Conventional octree

construction divides a three-dimensional space for each axis using the spatial median, obtains eight subspaces, which can be represented by an octree. The root node of an octree represents the entire object space. If the entire space contains more objects than a given limit, the space is divided into eight sub-spaces represented by eight children nodes. A subspace thus created is defined as a voxel. These voxels are further divided into eight voxels, and this process is repeated until the voxels satisfy the given criteria. In general, the criteria used to determine whether or not the given octree should be divided further depend upon the number of objects intersecting with a voxel and the maximum depth of an octree allowed[22][23].

However, the octree contains cell boundaries that are static and their location is independent of the objects. This independence makes the more intersections. The kd-tree, on the other hand, places the boundaries around the objects, especially if the empty space is cut off, thus it can result in much higher intersection probability. Furthermore, conventional octree construction uses uniform voxels to get spatial partition, which leads to higher depth of the tree, and generates much empty nodes which waste much memory.

3.2 Improved Octree Construction

The kd-tree is the best general-purpose acceleration structure, which uses the SAH [8] estimates the ray tracing performance of a given acceleration structure. But modern GPU architecture contains multiple physical multi-processors and requires tens of thousands of threads to make the best use of these processors [5], while accustomed acceleration structure only with a relatively small number of nodes at the upper levels, which makes parallelizing over nodes inefficient and leaves the massive parallelism of GPUs underexploited. Therefore, we propose an improved octree to exploit the hardware to the largest possible degree.

Conventional SAH kd-tree evaluates the SAH costs for all splitting plane candidates, then pick the optimal candidate with the lowest cost and split the node into two child nodes. Unlike this method, we extensively work on all three dimensions at once during SAH evaluation, and build eight sub-nodes at one time. Furthermore, we record the optimal candidate with the lowest cost at X dimension, Y dimension and Z dimension. Then, the three candidates divide a node into eight sub-nodes, as shown in Figure 1.

For each potential partition we need to compute Eq. (1), hence we need to know the primitive counts and the surface areas of children. To compute these counts efficiently, Wald et al. [13, 24] proposed to sort the primitives. However, a much more efficient method was recently published, which avoids sorting and which additionally features memory friendly access patterns [25, 26]. For our octree builder, we adapt the binning method of [25], which was originally proposed for building kd-trees. We iterate over the primitives on all three dimensions at one time to bin

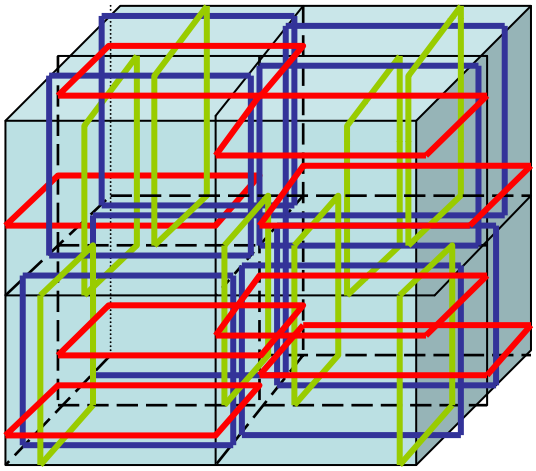


Figure 1: improved octree construction with SAH

them by means of their theory, and by doing so, to accumulate their count and extend in several bins. The gathered information in the bins is then used to reconstruct the primitive counts and the surface areas on both sides of each border between bins, and thus to compute the SAH cost function at each border plane. We can use SIMD operations to compute the SAH cost function on the three dimensions together that exploit the parallel performance. As Popov et al. [25], we minimize memory bandwidth by performing the binning in all three dimensions during the split of the parent node.

$$C_P = K_T + \frac{K_I}{SA(N)}[n_l SA(N_l) + n_r SA(N_r)], \qquad (1)$$

where n_l and n_r are the number of contained primitives in the respective child nodes. We take that partition that has minimal local cost C_P, or terminate if creating a leaf, which has cost $K_I \cdot n$, is cheaper, with $n = n_l + n_r$ being the number of primitives in the current node.

We give some details of our implementation concerning efficiency and robustness. Our octree is completely built on the GPU. We store an AABB and a counter in a bin. The primitives are represented by the centroid and the extent of their AABBs. For each primitive we compute the indices of the bins of all three dimensions from its centroid in SIMD. Then, the counters of all three bins are incremented, and their

AABBs are enlarged with the primitive's AABB using SIMD min/max operations. After all the calculations of a node are executed at three dimensions, we should record the counters and the surface areas, which will be used in the calculation of its children. The cost function needs to be sampled along all three dimensions.

The value of the cost function at a sampling position depends solely on the count of AABBs that intersect the left and right sub-volumes for the latter location. The algorithm collects this data in two phases. In the first phase, the algorithm sweeps the array of AABBs and stores at each sample the count of AABBs that end between this and the previous sample. The algorithm also stores the number of AABBs that start between the sample and the next one. In the second phase, the algorithm uses the collected information to incrementally reconstruct the AABBs counts at the sample locations using the formula

$$n_l^{i+1} = n_l^i + S^i,$$
$$n_r^{i+1} = n_r^i - E^i,$$

where $n^i_{l|r}$ describes the number of AABBs to the left/right of the sample i, and the number of AABBs that start and end between samples i and i+1 is denoted by Si and Ei, respectively. The algorithm evaluates the cost function at the sample locations and than distributes the AABBs to the left and right subtrees as described above.

Algorithm 1: **Improved Octree Construction**

Procedure BIN(AABB, sample)

D_{xyz} ← samples of min point of AABB

U_{xyz} ← samples of max point of AABB

For all dim ∈ {x,y,z} do

Increase sample.obj_S[U[$_{dim}$]]

Increase sample.obj_E[D[$_{dim}$]]

End for

End procedure

Function Found(head,dim)

Head[$_{dim}$].Left[0]←0

Head[$_{dim}$].Right[0]←NumObj

For each sample point at the dim

Head[$_{dim}$].Left[i]←Head[$_{dim}$].Left[i-1]+Head[$_{dim}$].obj_S[i]

Head[$_{dim}$].Right[i]←Head[$_{dim}$].Right[i-1]+Head[$_{dim}$].obj_E[i]

End for

Evaluates the cost function at the sample locations at each dim

Return the best found split

End function

For each input triangle t **in parallel**

Compute AABB for triangle t, add into AABBset

End for

Head←0

BIN(all aabb ∈AABBset,0)

For all dim ∈ {x,y,z} **do In parallel**

If NumObj<threshold **then**

Run conventional kd-tree routine for subtree

End if

Found(head,dim)

End for

Calculate the counters and the surface areas

Repeat

IV. IMPLEMENTATION AND RESULTS

In this section, we describe our implementation and the performance comparison. The described algorithm has been tested on an Intel Xeon 3.7GHz CPU with an NVIDIA GeForce 8800 ULTRA (768MB) graphics card.

4.1 Implementation

We implemented the above octree builder using NVIDIA's CUDA framework [5]. Previous GPU programming systems limit the size and complexity of GPU code due to their underlying graphics API based implementations. CUDA supports kernels with much larger code sizes with a new hardware interface and instruction caching. The GeForce 8800 allows for general addressing of memory via a unified processor model, which enables CUDA to perform unrestricted scatter-gather operations.

The GeForce 8800 consists of 16 streaming multiprocessors (SMs), each containing eight streaming processors (SPs), or processor cores, running at 1.35GHz. Each core executes a single thread's instruction in SIMD (single-instruction, multiple-data) fashion, with the instruction unit broadcasting the current instruction to the cores. Each core has one 32-bit, single-precision floating-point, multiply-add arithmetic unit that can also perform 32-bit integer arithmetic.

GĂźnther [4] point out that the number of bins is a crucial parameter controlling the construction speed and accuracy. The more bins there are, the more accurate is the sampling of the SAH cost function, but the more work has to be done during calculation of the SAH function from the binned data (the binning steps are independent from the number of the bins). Additionally, binning becomes inefficient if the number of bins is close to the number of to-be-binned primitives. Therefore the number of bins k per dimension should be adaptively chosen linearly depending on number of primitives n and bin-ratio r: k = n/r and clamp it to [kmin, kmax]. Just as what GĂźnther said on [4], we experimented with different parameter sets representing a trade-off between speed and accuracy. The default settings are kmax = 128, kmin = 8, and r = 6. The fast settings are kmax = 32, kmin = 4, and r = 16.

We need to specify the number of thread blocks and threads per block for the parallel primitives and the code fragments marked by in parallel. In our current implementation, we use 256 threads for each block. The block number is computed by dividing the total number of parallel threads by the number of threads per block. During octree construction, we store all data in linear device memory allocated via CUDA. For structures with many fields such as nodes and triangles, we use structure of arrays (SoA) instead of array of structures (AoS) for optimal GPU cache performance. As the level of the tree increases, the data that needs to be processed per thread in order to construct the next child node decreases, so the numbers of the threads in a block executed in parallel at one time can be increased significantly. Furthermore, because of the small sets of data, the amount of SAH calculation is decreased significantly, and shorten the time for dividing. Besides, our octree construction has lower levels, which makes ray tracing traversal faster. Considering that conventional kd-tree can be more efficient on small sets of data, so if the algorithm encounters a partition with size bellow some threshold, it switches to a conventional kd-tree construction at the point.

4.2 Performance Comparison

In this section we present the results of our experimental work. The quality of the trees is assessed in two ways. Firstly, we compute the construction time. Secondly, we evaluate the practical effect of tree quality on render time by using the constructed trees in a ray tracer. The results are presented as a table. To evaluate the performance of our construction algorithm, we compared it to the conventional construction algorithm by measuring the time needed to build a SAH kd-tree using a variety of scenes, ranging from simple to reasonably complex, namely BUNNY, FAIRYFOREST, and CONFERENCE. The scenes and the viewpoints for the tests can be seen on Figure 2. Table 1 summarizes the comparison results for several publicly available scenes as shown in Fig.2. As shown, our octree construction algorithm is 8 ~ 12 times faster for these scenes. Although our technique is capable of constructing high quality octrees in real-time, it has its limitations. For small scenes with less than 5K triangles, CUDA's API overhead becomes a major bottleneck. In this case, it is more efficient to switch to a complete CPU method.

Our octree construction algorithm also scales well with the number of GPU processors. The running time contains a scalable portion and a small non-scalable portion due to the overhead of CUDA API and driver. Theoretically, the running time is linear with respect to the reciprocal of the number of processors. As shown in Fig.3, we ran the algorithm on a GeForce 8800 ULTRA graphics card with 16, 32, 48, 64, 80, 96, 112, and 128 processors respectively.As shown in the table, our algorithm always offer better rendering performance. For dynamic scenes, our ray tracer can build an octree from scratch through our fast and efficient construction method without prior knowledge about geometry motion, which is a competitive alternative to a state-of-the-art ray tracer. Note that here we do not claim that our GPU ray tracer is faster than all CPU ray tracers. Indeed, implementing the fastest CPU ray tracer is like chasing a moving target because various optimizations could be used for higher performance and some optimizations are hardware dependent, and better performance can be achieved by adding more CPU cores.

Figure 2: The scenes used for testing, from left to right: 1)"BUNNY" 2)"CONFERENCE" 3)"FAIRYFOREST"

Scene and #triangles	Highly optimized kd-trees		Our routine	
	const r. time	FPS.	constr. time	FPS

 ISBN 978-80-86943-94-7

“BUNNY ”,69 K	0.62s	4.9	0.08s	9.3
“CONFERENCE ”,282K	1.41s	2.7	0.11s	5.27
“FAIRYFOREST ”,180K	1.15s	2.5	0.15s	4.17

Table 1: Construction time and FPS performance for 1024x1024 resolution， including shading with shadow. We include the time needed to read back the result from CUDA and draw them through OpenGL in our results.

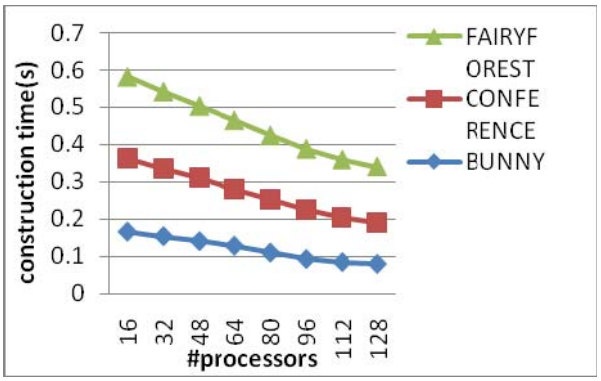


Figure 3: The tree construction time decreases quickly with the increase in the number of GPU processors

V.CONCLUSIONS AND FUTURE WORK

In this paper, we present an octree construction algorithm for the GPU that achieves real-time performance by heavily exploiting the hardware. This technique has four important features. Firstly, it builds octrees in real-time by exploiting the fine-grained parallelism on the GPU. Secondly, As the level of the tree increases, the data that needs to be processed per thread in order to construct the next child node decreases, so the count of SAH calculation is decreased significantly, and shorten the time for dividing. Thirdly, our octree construction has lower levels, which makes ray tracing traversal faster. Fourthly, because of using streaming construction, our octree consume less memory.

In future we plan investigation in the following directions. We plan to incorporate packets [13] into the GPU ray tracer for further performance enhancements. We also intend to amend the algorithm to add more secondary rays to render. Thirdly, we consider offloading some of the octree construction steps to a CPU. Vectorization using SIMD instructions will increase performance of the construction algorithm.

ACKNOWLEDGEMENTS

The first author would like to thank the anonymous reviewers for their insight and helpful comments， and Wen-qiao Zhu for his enthusiastic discussion during the early stage of this work. Ren C. has provided his help in implementation, and experimentation. This research work has been partially supported by National Key Technology R&D Program in the 11th Five year Plan of China(2007BAH11B05).

REFERENCES

[1] J. Owens. Streaming architectures and technology trends. GPU Gems 2, pages 457–470, March 2005.

[2] HORN D. R., SUGERMAN J., HOUSTON M., HANRAHAN P.: Interactive k-D Tree GPU Raytracing. In I3D ’07: Proceedings of the 2007 symposium on Interactive 3D graphics and games (2007), ACM Press, pp. 167–174.

[3]POPOV S., GĂźnther J., SEIDEL H.-P., SLUSALLEK P.: Stackless KD-Tree Traversal for High Performance GPU Ray Tracing. Computer Graphics Forum 26, 3 (Sept. 2007). (Proceedings of Eurographics)

[4] GĂźnther et al. Realtime Ray Tracing on GPU with BVH-based Packet Traversal the IEEE/Eurographics Symposium on Interactive Ray Tracing 2007

[5] NVIDIA: The CUDA Homepage.

http://developer.nvidia.com/cuda. 1, 2, 3

[6] Kun Zhou et al. Real-Time KD-Tree Construction on Graphics Hardware, SIGGRAPH Asia 2008

[7] V. Havran. Heuristic Ray Shooting Algorithms. PhD thesis, Faculty of Electrical Engineering, Czech Technical University in Prague, 2001.

[8] J. D. MacDonald and K. S. Booth. Heuristics for ray tracing using space subdivision. In Graphics Interface Proceedings 1989, pages 152–163.

[9] CARR, N. A., HALL, J. D., AND HART, J. C. 2002. The ray engine. In Proceedings of Graphics Hardware, 37–46.

[10] PURCELL, T. J., BUCK, I., MARK, W. R., AND HANRAHAN, P. 2002. Ray tracing on programmable graphics hardware. ACM Trans. Gr. 21, 3, 703–712.

[11] FOLEY, T., AND SUGERMAN, J. 2005. Kd-tree acceleration structures for a GPU raytracer. In Graphics Hardware'05.

[12] CARR, N. A., HOBEROCK, J., CRANE, K., AND HART, J. C. 2006. Fast GPU ray tracing of dynamic meshes using geometry images. In Proceedings of Graphics Interface, 203–209.

[13] WALD I., HAVRAN V.: On building fast kd-trees for Ray Tracing, and on doing that in O(N log N). In Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing (Sept.2006), pp. 61–70. 4

[14] YOON, S.-E., CURTIS, S., AND MANOCHA, D. 2007. Ray tracing dynamic scenes using selective restructuring. In Eurographics Symposium on Rendering.

[15] SHEVTSOV, M., SOUPIKOV, A., AND KAPUSTIN, A. 2007. Highly parallel fast kd-tree construction for interactive ray tracing of dynamic scenes. In Eurographics'07, 395–404.

[16] HAVRAN V.: Heuristic Ray Shooting Algorithms. Ph.D. thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, November 2000. 1, 2

[17] HAVRAN V., HERZOG W., SEIDEL H.-P.: On the Fast Construction of Spatial Hierarchies for Ray Tracing. In Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing (2006), IEEE Computer Society, pp. 71–80.

[18] WÄCHTER C., KELLER A.: Instant ray tracing: The bounding interval hierarchy. In Proceedings of the Eurographics Symposium on Rendering (2006)

[19] RESHETOV A., SOUPIKOV A., HURLEY J.: Multi-level ray tracing algorithm. ACM Trans. Graph. 24, 3 (2005)

[20] Kyu-Young Whang, et al. Octree-R: An Adaptive Octree for Efficient Ray Tracing IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, VOL. 1, NO. 4, DECEMBER 1995

[21] AS. Glassner, "Space subdivision for fast ray tracing," IEEE Computer Graphics and Applications, vol. 4, no. 10, pp. 15-22, Oct. 1984.

[22] GOLDSMITH J., SALMON J.: Automatic Creation of Object Hierarchies for Ray Tracing. IEEE Computer Graphics and Applications 7, 5 (May 1987), 14–20. 3

[23] MACDONALD J. D., BOOTH K. S.: Heuristics for Ray Tracing using Space Subdivision. In Graphics Interface Proceedings 1989 (June 1989), A.K. Peters, Ltd, pp. 152–163. 1, 3

[24] WALD I., BOULOS S., SHIRLEY P.: Ray Tracing Deformable Scenes using Dynamic Bounding Volume Hierarchies. ACM Transactions on Graphics 26, 1 (Jan. 2007), 6. 1, 3, 4, 5, 6

[25] POPOV S., G¨U NTHER J., SEIDEL H.-P., SLUSALLEK P.: Experiences with Streaming Construction of SAH KD-Trees. In Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing (Sept. 2006), pp. 89–94. 1, 3, 4, 6

[26] HUNT W., STOLL G., MARK W.: Fast kd-tree Construction with an Adaptive Error-Bounded Heuristic. In Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing (Sept. 2006), pp. 81–88. 4, 6

Vision Aware Continuum Crowds

Wilfrid Lefer

University of Pau
64 013 Pau, France

lefer@univ-pau.fr

ABSTRACT

Crowd simulation has received increasing attention for two decades because potential applications of crowd simulators can be found in various societal domains. The continuum crowd model puts all information related to the decision-making process in a single equation, which can then be solved by a Fast Marching Method approach. In this paper we propose several improvements to the continuum crowd model: a new governing equation, a new collision avoidance method, and, our major contribution, we add vision capabilities to the characters, thus making them able to collect new information about their surrounding environment and to reconsider path planning according to up-to-date data.

Keywords

Computer animation, crowd simulation, continuous models.

1. INTRODUCTION

Crowd simulation has received increasing attention for two decades because potential applications of crowd simulators can be found in various societal domains: architecture, robotics, building evacuation, training systems, video games, etc. The problem has been addressed from various viewpoints, depending on the original background of people tackling it: architecture, physics, psychology, computer graphics, etc. Thus various issues require attention, although rarely all of them are addressed at the same time, ie in a single system: virtual human generation, character/rigid body animation, crowd simulation, virtual crowd rendering, and even interacting with virtual crowds. Rather than listing more or less exhaustive reference list, the reader may start with a recent dedicated book [Tha07]. A number of crowd simulators are now available, both commercial solutions and academic ones. Basically they can be divided into two categories: behavior versus rendering oriented. The first category of systems try to generate realistic behavior of crowds, generally for specific environments and simulations, such as building evacuation, the objective being to assess emergency procedures in realistic emergency conditions, that is with people in panic and so. The second category is mainly devoted to the games industry and systems must be able to simulate large crowds wit drastic frame rate constraints. Here rendering quality is more important than realistic behavior.

Several crowd models can be found in the literature, the most popular ones being rule-based systems, Reynolds's social forces [Rey87], and recently continuum crowd [Tre06]. More sophisticated models propose a layered architecture so as to consider different information abstractions and hence different behavior modeling levels, such as reactive versus cognitive [Bra03][Pel07]. Basically crowd simulation models can be divided as either discrete or continuous. In a discrete model, a number of parameters are evaluated at each time step and a decision is taken, for instance move forward or turn on the right or stay here. A small change of a single parameter can lead to a radically different decision. Typical such models are rule-based ones in which each possible action is predicated by a condition, all possible actions being tested in a given order until a valid one had been found. In a continuous model, a moving direction is computed as a – continuous - function of all the parameters. Thus a slight change of a parameter leads to a slight change of the moving direction. The continuum crowd method is probably the more elaborated such model [Tre06]. This paper extends this model by adding vision capabilities to the crowd.

The remaining of this paper is organized as follow. Section 2 recalls the main elements of the continuum crowd model, together with some improvements. Section 3 presents our solution to add vision skills to crowd characters. Some results and a brief discussion on the limits of our solution are presented in section 4 and section 5 is our conclusion.

2. CONTINUUM CROWD REVISITED

The basic issue any crowd simulation system has to address at each time step of a simulation is to

compute the next displacement of each character, that is the moving vector (direction and distance). Various data can be integrated in the decision-making process at this time but specific data are considered in any crowd simulator: physical environment, that is geometry (height, obstacles), and positions of other characters. The continuum crowd model considers 3 kinds of information: terrain slope, crowd density, and discomfort. All of them are continuous functions of $\Re^2$, i.e. the 2.5D simulation space. The discomfort field is a flexible way to design how comfortable is an individual at a given location. Thus it is possible to make characters that will more likely walk in pedestrian areas rather than on car drives. Obstacles are not explicitly defined but they can be easily described their impact on the discomfort field, typically by assigning high discomfort values to obstacles locations.

New equation

The principle of any path search method consists in minimizing a travelling cost C, which computation can involve various parameters:

$$\text{Min}[\int_P \mathbf{C} ds)] \qquad (1)$$

where P is the path and ds means that integration is taken with respect to path length.

This requires to be able to evaluate the cost function at any point for all possible directions, which yields to a function of two variables: location and direction of movement. The original continuum crowd cost function is as follow [Tre06]:

$$C(x,\theta) = (\alpha f + \beta + \gamma g) / f \qquad (2)$$

Where x and θ are the location and direction of movement, respectively, f is the speed at location x, g is the discomfort at position x+θ, and α, β, and γ are weights (values in the range [0,1] and verifying condition $\alpha + \beta + \gamma = 1$) for individual terms, which can be used to define the relative influence of each term on the final cost.

This equation raises some problems though, because it is not linear in the weights, which makes it difficult when an accurate control of the displacement times is required, for instance for the purpose of building evacuation certification. Another issue arises for very small values of f, for instance in congestion situations.

For this reason we will use a slightly different function:

$$C(x,\theta) = \alpha + \beta\,(1 - [(f(x,\theta) - f_{min}) / (f_{max} - f_{min})]) + \gamma\, g(x+\theta) \qquad (3)$$

This yields to expected results at limit conditions. For instance if we neglect speed (β=0) and discomfort (γ=0), the unit cost is 1, which yields to the shortest path. If distance (α=0) and discomfort are neglected, we minimize travelling time. F_{min} and F_{max} are the minimal and maximal speeds, which correspond to extreme situations, typically in congestion and free run at maximum negative slope, respectively. An interesting property of this function is that it generates values in the range [0,1].

Because equation 1 cannot be solved analytically, a discrete version is used, which allows us to compute an approximation of the optimal path. Thus the simulation space is discretized as a regular grid and each field (height, speed, discomfort, cost) is evaluated at each cell center for isotropic fields (height, discomfort) or using a MAC-style arrangement [Fed01] for anisotropic ones.

Collision avoidance versus minimum distance enforcement

The continuous continuum crowd model can prevent collisions, assuming density threshold is properly set. However, approximation solutions of equations 1 in discrete space lead to numerical errors, a typical consequence being that collisions actually occur. Treuille et al. propose a minimum distance enforcement mechanism to address this issue. It consists in explicitly checking all pairs of individuals in the vicinity of a character before moving it. The overhead involved by these tests can be important, especially in congestion situations.

We adopt a different strategy. Once theoretical displacements have been computed by the simulator, an advector is in charge of actually moving the individuals. Several specialization classes have been design, which all inherit from a generic advector class and implements different advecting strategies, the most basic one consisting in just advecting the crowd. But more sophisticated advectors have been developed, which allows us to evaluate different collision avoidance strategies.

One of them implements the following algorithm:

```
AdvectCrowd(list<character> L)
{
    While L is not empty Do
        Extract a character from the list randomly
        Vector v = theoretical displacement vector
        Position p = character position + v
        Integer #tries = 3
        While #tries > 0 And there is a collision Do
            v = v / 2
            p = p – v
            #tries = #tries – 1
        If #tries > 0 Then
            Move character to its new position
}
```

Changing the order of advection from a time step to the next is important because it contributes to keep crowd compactness and it avoids strong slow downs. This can arise for instance in a tight corridor in which overtaking is impossible. In such a situation, if it turns out that the individual at the head of the queue is advected in last position, all other characters will be blocked, their advections being impossible because there is no space in front of them. Note that except in such special situations, this kind of problem is properly addressed by the simulation phase, the high density in front of an individual will make it try to overtake by the right or by the left.

This algorithm offers a good trade-off between advection quality and computation complexity. Quality of advection can be measured as the distance between positions of the crowd using the theoretical advection, that is displacement vectors computed by the simulator, and actual positions, i.e. collision free positions.

3. VISION AWARENESS

Here we detail how individual characters of our continuum crowd model get vision capabilities and thus are able to react to changes in their visual environment.

Problem statement

In the original continuum crowd model [Tre06], every individual has a complete knowledge of the whole environment and is able to determine an optimal path toward his own group target. Obviously this model is not realistic in many common real world situations. A trivial case is a discovery process, in which someone is trying to find his/her way in a partially unknown environment. Another common situation is a dynamic environment, for instance a city in which a street becomes suddenly no longer available for some reason: people far from this place are typically not aware of this event and hence should not reconsider their path.

A realistic model should take individual knowledge into account, rather than collectivity knowledge. Moreover this knowledge should be subject to changes, as a function of the various events encountered by each individual, typically events occurring in his/her vision space. This implies for a simulator to include the following mechanisms:

- manage individual knowledge, that is each individ*ual has his own knowledge base,*
- handle knowledge evolution over time.

Individual knowledge representation

The original continuum crowd model does not consider the large variety of information that can be available in an environment, such as buildings, roads, trees, lakes, fire, hard-to-walk-in places such as dense forests or marsh, etc, but rather considers discomfort as a way to take any kind of information into account. A nice property is that equation 2 only involves three data fields: height, people density, and discomfort. Up to the end user to design any function to map the various information data to discomfort.

In order to have each individual reasoning with his/her own knowledge base, it is necessary to duplicate these three fields for every character. This is obviously memory consuming, especially for large crowds, which are indeed the main target of the original continuum crowd model. We decrease this memory overhead by storing only the cost field $C(x,\theta)$ instead (see equation 3).

Dynamic individual knowledge reconstruction

Now we describe how each individual knowledge base is updated at each time step of the simulation. At the beginning, people in a single group share a common knowledge base. Then at each time step, each individual updates his/her knowledge base according to his/her percepts of the environment. Percepts can be simply visual or they may include information gathered by any source, such as voice diffusion – a fire alert for instance – or mobile phone communications. For each kind of percept, a specific information acquisition mechanism, called percept processing unit, is set up, which affects some cells of the simulation grid. Terrain slope being likely not to change over time, only the crowd density and discomfort fields are affected by the percepts processing unit. Then $C(x,\theta)$ is updated only for the cells affected by the percepts. Last, the Fast Marching Method is run in order to produce a new potential field, which will give the moving direction for the avatar.

Vision percept computation

Even if various perception channels can be used to improve individual knowledge, we concentrate here on the visual acquisition of information. This implies to be able to determine which part of the scene is visible by a given character at any time. The data that can be perceived are stored in the simulation grid: people positions, moving obstacles such as cars, smoke, etc. Hence determining which information is directly visible by a character consists in determining the list of grid cells that are directly visible from the character's position. A trivial way to solve this problem consists in processing every obstacle in order to determine the shortest visible distance in each possible direction. For geometric obstacles, this can be typically implemented with a Z-buffer algorithm. But such an approach is not appropriate in the case of the our continuum crowd scene for the following reasons:

- there is no longer geometric obstacles but rather they have contributed to determine discomfort values (typically walls correspond to infinite discomfort values),
- we need the visibility region as a list of cells and thus we would have to compute this list from the Z-buffer values, which would had a computation overhead,
- the process could be time consuming for a large number of obstacles, while only a few of them is generally visible from a given location.

Instead we formulate the visibility condition as the solution of two separate Eikonal equations [Set99a]:

$$|\nabla T_{\text{no-obstacles}}| = 1 \qquad |\nabla T_{\text{obstacles}}| = f(x,y)$$

with $f(x,y)=\infty$ where $g(x,y)=\infty$ and $f(x,y)=1$ anywhere else.

Remember that the discomfort field $g(x,y)$ is everywhere positive and equal to ∞ for cells that fall inside an obstacle, typically a wall. Solving this equations system consists in solving each equation separately and then comparing results cell by cell. For a given cell, two strictly equal results means that the cell center is visible from the character position, otherwise the cell center is not visible. To understand this, remember that solving the Eikonal equation amounts to computing the optimal path from the source to the target. Visual rays use the straight line, which means that solution of the left equation corresponds to the straight line. Solution to the right equation computes the optimal path by treating $f(x,y)$ as a cost function, the time necessary to travel through a given cell being inversely proportional to the cost associated to that cell. Hence the only case for which both equations lead to the same value if when all cells on the straight line have the minimum cost of 1.

The implementation of this algorithm is not trivial, so we skip it here, the reader will find all details on the Fast Marching Methods implementation in [Set99b]. Now we suppose that there exists a function called *ComputeVisibilityList(c0 :<cell>)*, which computes the visibility cell list for a given character position c0.

What we need actually to update each character's knowledge base is the list of cells whose information has changed since the last time they have been processed by this character. So we withdraw from the visibility list the cells that were already in the visibility list at the previous time step and whose attached values have not changed at the current time step.

The algorithm for updating individual knowledge bases is as follows:

```
<list of cells> DetermineListOfCellsToUpdate(c0 :
<cell>)
{
    Vt : visibility list at time step t
    It(c) : information stored in cell c at time step t
    c : <cell>
    Vtemp, Vchange : <lists of cells>
    Vtemp = Vt
    While Vtemp is not empty Do
        c << Vtemp
        If It(c)≠ It+θ(c) Then
            Vchange << c
    Vt+θ = ComputeVisibilityList(c0)
    Vθ = Vt+θ - Vt + Vchange
    Return Vθ
}
```

Once cells which cost changed at the last time step have been identified, we update the potential field by running the Fast Marching Method on the updated cost field C.

4. RESULTS

Figure 1 shows a simulation of two avatars moving in a building. Since we are just concerned by their trajectories, a basic 2D rendering is used here. The blue avatar implements the original continuum crowd algorithm whereas the magenta one is managed by our vision aware method. Both characters start from the same location at the same time (frame #0). Suddenly (frame #34), a fire arises at a place hidden from the characters' positions. The fire is modeled by increasing discomfort at the corresponding location. More precisely a discomfort footprint with a Gaussian shape is added to the discomfort field so as to not only prevent people from crossing the fire but also to simulate the heat effect, which should ideally make people passing at a reasonable distance from the fire spot. We can observe (frame #38) that, as soon as the fire starts, the blue character changes his trajectory and finds a more appropriate path, though not viewing the fire, as if he would be immediately aware of this event. The magenta character, who has visual capabilities, continues on the same path (frame #88). As soon as he is able to watch the fire (frame #100), he changes his path accordingly, which in this case yields to going back in the corridor (frame #108) and follow an alternative path (frames #180 and #280).

Our simulator can also provide a realistic 3D rendering with 3D characters, which are animated with the Cal3D library [Cal3D].

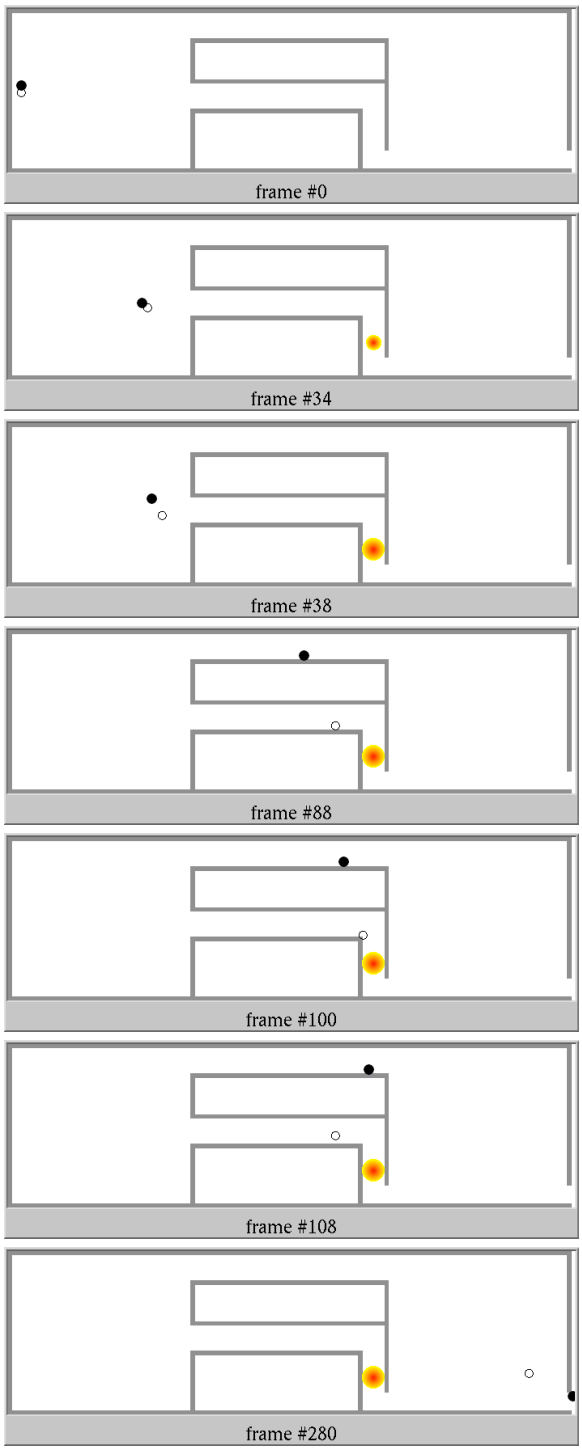


Figure 1: Original continuum crowd model (blue) versus vision aware model (magenta).

Limits

One of the nice properties of the original continuum crowd formulation is its scalability with the number of avatars. The memory and computational complexity is linear with the number of avatar groups, rather than the number of avatars. In large crowds of people having common objectives and common discomfort fields, computation is particularly efficient. Introducing variety in the discomfort as a function of information gathered along each individual path leads to having as many copies of the discomfort field as the number of avatars. This leads to both memory and computational overheads. But people starting at about the same location toward a common goal leads to a flocking phenomenon [Rey87]. As a consequence, they tend to observe the same phenomena at the same time. Such set of people might be grouped together as sub-groups, so as not to duplicate their discomfort fields. An error metrics could be defined in order to evaluate the correlation degree of individual knowledge within each sub-group, together with conditions for sub-group cutting.

Simulating heat effects of the fire is one of the properties that a Gaussian footprint can provide but of course this is not a reliable approach to model heat transfer: after a certain amount of time, heat should be felt at a distance that is not directly related to visibility. But another nice property is to avoid a geometric description of the fire, which would probably lead to people passing exactly at the limit of the fire zone, thus describing unrealistic perfectly circular trajectories.

5. CONCLUSION

Continuum crowds is an elegant model for crowd simulation, providing a mathematical and hence robust environment for various crowd behavior modeling. Our current work consists in evaluating the potential of this model for various crowd behaviors, which are typically modeled through a discrete formulation. Among them, vision awareness is an important issue since all entities that may be candidates for simulation do have vision capabilities: pedestrians, cars with their drivers, animals, even robots. Our solution is successful in enabling individuals to perceive their visual environment but adds an important computation overhead. Further investigations are required to find ways to decrease this cost, grouping individuals with similar viewpoints being a possible solution.

A further step in the direction of realistic crowd modeling needs to consider individual diversity in the way this information is processed in each brain and to model the underneath cognitive mechanisms. Coupling a cognitive reasoning module with a continuum crowd model in a hybrid architecture is one of our challenges now. One of the main issues is the continuous versus discrete models, cognitive models, such as those found in the Multi-Agent Systems literature, being generally discrete.

6. REFERENCES

[Bra03] Braun, A., Raupp Musse, S., and L.O.B. Bodmann. Modeling Individual Behaviors in Crowd Simulation. Computer Animation and Social Agents, New Brunswick (USA), 2003, pages 143-148.

[Fed01] Fedkiv, R., Stam, J., and H. Jensen, H. Visual Simulation of Smoke. In ACM Computer Graphics (SIGGRAPH'01 proceedings), pages 15–22.

[Pel07] Pelechano, N., , Allbeck, J., and N. Badler. Controlling Individual Agents in High-Density Crowd Simulation. ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'07), August 3-4, San Diego (USA), 2007.

[Rey87] Reynolds, C. W. Flocks, Herds, and Schools: A Distributed Behavioral Model. Computer Graphics 21(4) (SIGGRAPH'87 Conference Proceedings), pages 25-34.

[Set99a] Sethian, J.A., Level Set Methods and Fast Marching Methods, Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Material Science, 2nd edition, 1999, Cambridge University Press.

[Set99b] Sethian, J.A., Fast Marching Methods, SIAM Review 41, July 1999.

[Tha07] Thalmann, D., and S. Raupp Musse. Crowd Simulation. Springer, 2007.

[Tre06] Treuille, A., Cooper, S, and Z. Popovic. Continuum Crowds. ACM Transactions on Graphics 25(3) (SIGGRAPH'06 Conference Proceedings).

[Cal3D] http://home.gna.org/cal3d/

Video-Based Bicycle Detection in Underground Scenarios

Vítězslav Beran, Adam Herout, Ivo Řezníček

Brno University of Technology
Faculty of Information Technology
Department of Computer Graphics and Multimedia
Božetěchova 2, 612 66 Brno, CZ

beranv, herout, ireznice@fit.vutbr.cz

ABSTRACT

Automatic surveillance systems are an important emerging application of object detection algorithms in video. The nature of such systems implies several requirements on the used algorithms. Also, searching for less usual objects (in contrast to frontal human faces, car masks, etc.) is required, such as detection of bicycles. It appears that detection of such objects cannot be solved by just applying a standard statistical or other general detector, but by constructing a specialized detector composed of several standard image processing and object-detection techniques combined together ad hoc. A detector of bicycles in video data from standard low-resolution CCTV surveillance system is presented in this contribution.

Bicycle detection approach covered by this paper aims to cope with highly-noisy low-resolution data, to use simple image-processing methods and to work in real time. Although the method itself does not constitute a generally usable object detector, it covers several interesting aspects which can be re-used in tasks similar to the given one. Low-level features extracted from the video used for wheel-candidate classification are described in detail. The system is applied and evaluated on real data and the results are discussed.

Keywords

Video surveillance, object detection, bicycle detection

1. INTRODUCTION

Contemporary surveillance systems monitoring public places provide massive amounts of data. There exist several common scenarios such as cross-roads, highways, parks and public transport places. Each of the scenarios presents a different type of issues in video-processing point of view.

The presented work aims to cope with the data from underground scenarios, especially from the subway stations, and to detect bicycles.

The task of searching for bicycles in general positions does not seem to be solved in the past. Bicycle detection is used in traffic lights control, but the means of detection are typically induction loops placed in the road surface. Detection based on image processing has also been scarcely explored by some researches (e.g. [Zhi03]), but the algorithms rather detect any object in the road and do not attempt to distinguish the bicycles of a special category.

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

General object detection algorithms can also deal with bicycle class detection and classification [PASCAL05].

Another research of searching bicycles in video sequences from colour CCTV cameras is bicycle theft detection. We can imagine a bicycle parking place and a CCTV camera which captures this place. This approach is based on identifying the bicycle in the video sequence and detecting the person who dropped the bicycle somewhere in parking place. System stores features of person who dropped the bike and associates it with the current bike. If someone tries to pick up the same bike later, system compares him with a person who dropped this bike and is able to generate alarm. This approach is more thoroughly described in [Dan07].

Besides the theft detection, the bicycle detection improves the control of underground safety. Most of the underground stations forbid to enter with bicycle and automatic bicycle detection decrease the work load of security guards.

The following section shows the structure of the system and introduces the main logic blocks. The

features used to classify bicycle candidates are described in section 3 in detail. The data used for method evaluation are more specified in section 4 together with experiment results. The last section concludes the achievements and proposes possibility for future work.

2. SYSTEM STRUCTURE

The designed system processes the video stream frame-by-frame and provides the position information and probability of possible bicycles.

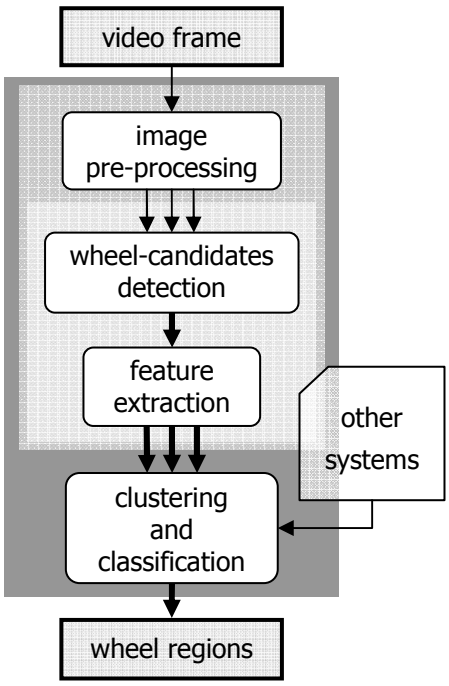


Figure 1. Schema of bicycle's wheel detection procedure.

Because the bicycle detection system works in parallel to other systems processing the same video stream, the design also incorporates some information from other systems. Although it is not crucial for the method itself, it is depicted in the system structure also.

The method pre-processes each frame several times with different parameters. That gives a set of slightly different images. Those images are then one-by-one searched for wheel candidates for which the features are extracted. This repetitive process returns a number of candidates that are finally clustered and classified. The final classification is improved by information from the other systems. In the case of bicycle detection in subway scenario, the information about active regions of interest is used to validate the classification.

Image Pre-Processing

Out method for bicycle detection basically looks for wheel-like shapes. Analysis of the actual data from the scenario (subway access corridors, platforms) showed that it would be impossible to look for the whole shape of a bicycle, because the bicycle can often be partially covered by a human figure. On the other hand, the shape of a wheel of reasonable dimensions does not appear in the videos just except for the case of bicycles. The analysis of the data also allows us to use a prior knowledge about the supposed colour of the wheels. Most of bicycles have black wheels, so the pre-processing step is designed to emphasize the dark grey patches of a particular tone.

Figure 2. Data examples (video frames and zoomed positive and negative samples).

Firstly, the input video frame is morphologically eroded and smoothed. The smoothing step helps to reduce the influence of harsh compression noise. The algorithm then generates images for detection of wheel candidates by computing the distance from dark-grey-colour model with different grey intensities. The dark-grey-colour model is implemented using a thresholding step, morphology closing and finally computing the distance from given grey value. This process is summarized in the following steps:

1. preprocessing: erosion, smoothing
2. colour model image created
3. several reference intensity images with different thresholds

Several image layers are generated by thersholding the color model image (see Figure 3). The wheel candidates are then detected on each layer.

Figure 3. Candidates from different layers.

The information about the candidate layer is used in final clustering. The more candidates are detected on the same location and different consecutive layers the higher candidate probability is.

Detection of Wheel-Candidates

The image with emphasized and thresholded dark patches is used to detect wheel candidates based on contours of clustered areas of constant colour [Son99] which are suspected to be insides of the bicycle wheels and are further examined to either confirm or reject this hypothesis. A huge amount of initially detected contours is reduced using filtering by allowed size range (dimensions in image axes and contour area) or width/height ratio. The next step is to find out, whether the contour has the shape of a wheel. For that task the ellipse-fitting algorithm [Dan07] is used. Having the contour described by an affine transform matrix, further filtering can take place. The error, the difference between the original contour and the fitted ellipse, from the ellipse-fitting algorithm serves as one of the candidate features (see section 3 below).

Clustering and Classification

The method takes into account several measurements. Besides the features described in the next section, also the candidate stability is used to increase its robust classification.

The stability feature is evaluated from the several sets of wheel-candidates obtained from appropriate image. The candidates are clustered using equivalency computation [Cor04] based on the Euclidean distance between samples' positions and sizes.

The final candidates are then represented by the clusters where the feature values, poses and sizes are computed as the mean of all cluster samples. The final candidates are classified by summing their weighted feature values and by comparing the final feature value to the main threshold. The main threshold is one of the tuning parameters of the method. The feature weights were set experimentally.

3. WHEEL FEATURES

Designing features for presented method, we experimented with standard ones, e.g. template matching and also with some we made up for the task purpose itself, e.g. ellipse profiling. Besides those, which are presented in more details in following sub-sections, we computed several others, such as the contour curvatures, length, contour length-area ratio, etc. Due to the low resolution of the data and the wheel sizes, those features have no use for classification.

Multi-Scale Template Matching

When detecting wheels using template matching, there arises a problem of different wheel widths. We solve the problem by utilizing several wheel templates with different circle widths (see example templates on Figure 4).

Figure 4. Set of templates covered different wheel width.

The first step before the matching takes place is to normalize the candidate patch. We use the ellipse parameters to compute affine transformation matrix. The normalized patch is then resampled to several different scales. The template matching is then executed for all templates and all scaled patches. Because the templates and also the scaled patches are normalized, we then get the maximum of all matches as the wheel feature.

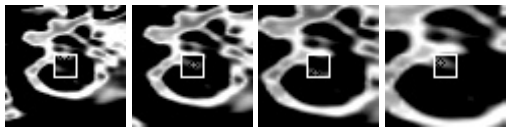


Figure 5. Rescaled sample with the results of matching showed in the rectangular area.

The detector robustness is improved by looking for the best match not only over the scale-space, but also in spatial domain. The white rectangles in the middle of subfigures on Figure 5 show the position change range. The pixel intensities in the rectangles correspond to matching results for particular template position and size.

Ellipse Profiling

The ellipse-profiling is the principal method we designed and develop. The method gives the most robust and distinctive features out of all others used by our system.

The main idea of the ellipse profiling method is to analyze the image intensity profiles defined by ellipses with different parameters. The image intensity values represent the probability given by dark-grey-colour model computed in pre-processing stage. For each wheel candidate, the algorithm generates several ellipses with increasing size. Then the image intensity values corresponding to ellipse profiles are summed up.

Each candidate region is described by affine transformation matrix. The transformation is used to

compute the elliptic profile and read image pixel values bellow the profile. The equation of the elliptic profile is:

$$ep(\mathbf{A}, r) = \int_0^{2\pi} I\left(\mathbf{A}\begin{bmatrix} r\cos t \\ r\sin t \end{bmatrix}\right) dt$$

where **A** is a affine transformation matrix describing the candidate region (position, rotation and scale), r is ellipse profile size and I is the color probability image.

We then search for maximal difference between consecutive summed values. The maximal difference represents the biggest change of ellipse profiles. Such change can occur, when one ellipse minimally overlaps the analyzed candidate image patch and the consecutive ellipse overlaps it maximally.

$$r_{\max} = \max_r ep(\mathbf{A}, r)'$$

The maximal difference found is the next candidate feature:

$$ep(\mathbf{A}, r_{\max})'$$

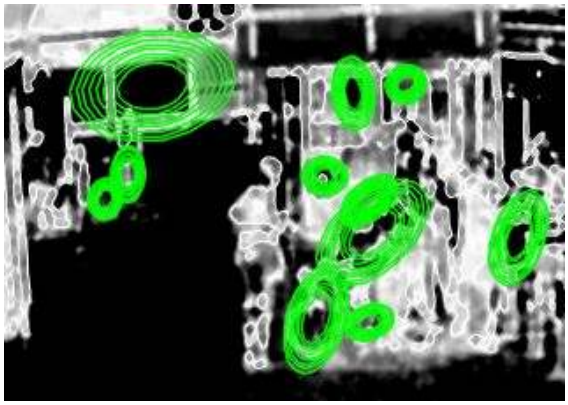


Figure 6. Elliptic profiles on several wheel candidates.

We improve the robustness of the method by evaluating more features by changing the ellipse positions within some small range. The candidate feature is then the best one.

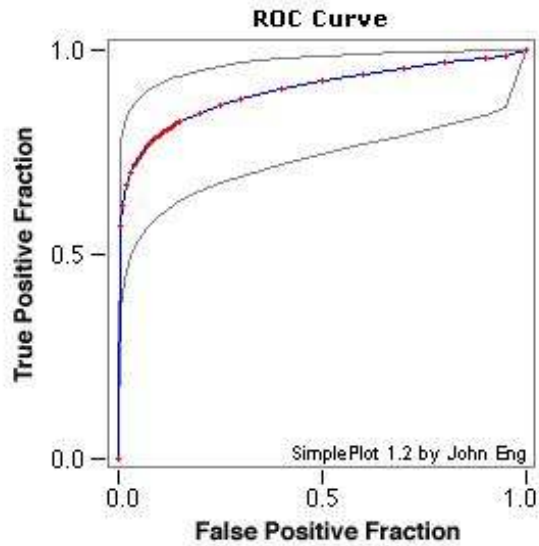
4. EVALUATION

The presented method is designed for video data from underground scenarios such as subway stations. Such data are noisy due to the high compression and the resolution is low. On the other hand, the lighting conditions are absolutely stable and the setting of the scene (where people can appear, stable objects in the scene etc) is reliably predefined. The precision of the method is highly sensitive to its parameters setting, which, in the constant conditions, is not a serious handicap. Also the method itself merely detects bicycle wheels. This limiting factor does not appear to be too restrictive: every appearance of anything similar to a wheel in the sample data (dozens of hours) really was a part of a bicycle.

The detector performance was evaluated on images from video of Roma undergrounds. The image resolution 720x576 using format 24bits/pixel, RGB.

Since the system is based on the wheel detector, the detector's ability and properties are crucial for the overall performance. For the wheel detector evaluation, about 60 still images containing bicycles were taken and manually annotated that gives about 120 wheels – positive samples.

The ROC curve below was generated by processing the wheel detector on each image and the wheel candidates were compared to the ground truth data. The ROC curve describes the trade-off between true positive and false positive detection rates. The gray curves are 95% confidence interval of the fitted ROC curve. The ROC curve was generated by [ROC].



7. ROC curve of the wheel detector.

In real applications, other supporting information is taken into account. Also, when the detector is used in the system for bicycle detection in the subway scenario, the moving object information, active regions of interest knowledge and temporal information from several consecutive frames improve the overall ability of the system.

Our system is designed to work on CCTV data, so instead of classifying each frame whether there is a bicycle or not, the whole video shot is classified. The Figure 8. shows the application screenshot in real environment.

The bicycle detector is sensitive to wheel-like objects that do not necessary need to be a bike. Such false positives are baby carriages, logos on clothes or shiny

bald spots. Such candidates are not properly distinguished by the system.

The presented system was integrated and evaluated in the surveillance system called CareTaker. The system is fully functioning and running in Roma and Torino underground.

5. CONCLUSION

We have presented a method for bicycle detection for underground scenarios. This method is optimized and targeted to surveillance systems with CCTV cameras, i.e. on low resolution low quality systems with requirement of real-time or close-to-real-time operation, but accepting compromises in reliability of results. Besides use of the common approaches, we designed a new method for similarity measure between cycle in raster image and ellipse. We designed a system to run experiments and evaluate our approach and performed preliminary evaluation of the algorithm.

8. Screenshot of bicycle detection application running in real environment.

Current work is on designing a generalized algorithm to cope with any shape that could be described by parametric curve and evaluating its robustness and stability.

6. ACKNOWLEDGEMENTS

This work has been supported by the CareTaker, IST EU project number 027231 and the "Centre of Computer Graphics" (CPG-LC06008), Czech Ministry of Education, Youth, and Sports grant.

7. REFERENCES

[Dan07] Damen, D., Hogg, D. Associating People Dropping off and Picking up Objects. In British Machine Vision Conference, 2007.

[Son99] Šonka, M., Hlaváč, V. and Boyle, R.Image Processing, Analysis and Machine Vision. PWS, ISBN 0-534-95393-X, 1999.

[Fit99] Fitzgibbon, A. W., Pilu, M. and Fisher, R. B. Direct least-squares fitting of ellipses. In IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(5), 476—480., 1999.

[Cor04] Cormen, H. T. 2001. Introduction to Algorithm. MIT Press, ISBN 0262032937.

[Zhi03] Zhijun, Q, et al. 2003, The study of the detection of pedestrian and bicycle using image processing. In Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE Volume 1, Issue , 12-15 Oct. 2003 Page(s): 340 - 345 vol.1

[PASCAL05] M. Everingham, A. Zisserman, C. Williams, L. Van Gool, M. Allan, C. Bishop, O. Chapelle, N. Dalal, T. Deselaers, G. Dorko, S. Duffner, J. Eichhorn, J. Farquhar, M. Fritz, C. Garcia, T. Griffiths, F. Jurie, D. Keysers, M. Koskela, J. Laaksonen, D. Larlus, B. Leibe, H. Meng, H. Ney, B. Schiele, C. Schmid, E. Seemann, J. Shawe-Taylor, A. Storkey, S. Szedmak, B. Triggs, I. Ulusoy, V. Viitaniemi, and J. Zhang. In *Selected Proceedings of the First PASCAL Challenges Workshop*, LNAI, Springer-Verlag, 2006

[ROC] Eng John. ROC-Analysis, Web-based Calculator for ROC Curves, Johns Hopkins University, Baltimore, Maryland, USA. http://www.rad.jhmi.edu/jeng/javarad/roc/JROCFITi.html

Conversion of biquadratic rational Bézier surfaces into patches of particular Dupin cyclides: the torus and the double sphere

Lionel Garnier, Bertrand Belbis, Sebti Foufou
LE2I, FRE CNRS 2309
UFR Sciences, Université de Bourgogne, BP 47870,
21078 Dijon Cedex, France
<lgarnier, bbelbis, sfoufou>@u-bourgogne.fr

ABSTRACT

Toruses and double spheres are particular cases of Dupin cyclides. In this paper, we study the conversion of rational biquadratic Bézier surfaces into Dupin cyclide patches. We give the conditions that the Bézier surface should satisfy to be convertible, and present a new conversion algorithm to construct the torus or double sphere patch corresponding to a given Bézier surface, some conversion examples are illustrated and commented.
Keywords: *Torus and Dupin cyclides surfaces, rational biquadratic Bézier surfaces.*

1 Introduction

Rational biquadratic Bézier surfaces are tensor product parametric surfaces widely used in the first generation of computer graphics applications and geometric modelling systems. Good introductions to these surfaces may be found in [PT89, For68, DP98, HL93].

Dupin cyclide surfaces represent a family of ringed surfaces, i.e., surfaces generated by a circle of variable radius sweeping through space [Pra90, Deg94]. It is possible to formulate them either as algebraic or parametric surfaces. In recent decades, the interest of several authors in these surfaces relates to their potential value in the development of CAGD tools [Pra95, DMP93, Gar07]. Also, cyclide intersections and the use of cyclides as blending surfaces have been investigated [BP98, She98].

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

Copyright UNION Agency - Science Press

The primary aim of this paper is to present an algorithm to convert a rational biquadratic Bézier surface into a particular Dupin cyclide patch. This conversion allows the obtention of parameters of the implicit equation of Dupin cyclide corresponding to the converted surface. Section 2 recalls the definition of Bézier curves and surfaces, and Dupin cyclides. Section 3 shows the construction of the control points and the computation of the weights of a Bézier surface which can be represented by a Dupin cyclide patch. Section 4 details the conversion algorithm. Section 5 presents our conclusions and suggests directions for future work.

2 Background

2.1 Rational Bézier curves and surfaces

Rational quadratic Bézier curves are second degree parametric curves defined by:

$$\overrightarrow{OM(t)} = \frac{\sum_{i=0}^{2} w_i B_i(t) \overrightarrow{OP_i}}{\sum_{i=0}^{2} w_i B_i(t)},\ t \in [0,1] \qquad (1)$$

where $B_i(t)$ are quadratic Bernstein polynomials defined as:

$$B_0(t) = (1-t)^2\,, B_1(t) = 2t\,(1-t) \text{ and } B_2(t) = t^2$$

and for $i \in \{0,1,2\}$, w_i are weights associated with the control points P_i. For a standard rational quadratic Bézier curve, w_0 and w_2 are equal to 1, while w_1 can be used to control the type of conic defined by the curve [Far93, Far99, Gar07].

Rational biquadratic Bézier surfaces are defined by control points $(P_{ij})_{0\leq i,j\leq 2}$ and weights $(w_{ij})_{0\leq i,j\leq 2}$ as:

$$\overrightarrow{OM(u,v)} = \frac{\sum_{i=0}^{2}\sum_{j=0}^{2} w_{ij} B_i(u) B_j(v) \overrightarrow{OP_{ij}}}{\sum_{i=0}^{2}\sum_{j=0}^{2} w_{ij} B_i(u) B_j(v)} \quad (2)$$

More details on Bézier surfaces can be found in [Far93, Far99, Gar07]. In the remaining of this paper, we only consider rational Bézier curves and surfaces of degree two to which we refer, for short, by Bézier curves and Bézier surfaces.

2.2 Dupin cyclides

Non-degenerate Dupin cyclides, figure 1(a), have been definied by P. Dupin [Dup22]. A. R. Forsyth [For12] and G. Darboux [For12, Dar17] have given two equivalent implicit equations:

$$\left(x^2+y^2+z^2-\mu^2+b^2\right)^2 = 4\left(ax-c\mu\right)^2 + 4b^2y^2 \quad (3)$$

$$\left(x^2+y^2+z^2-\mu^2-b^2\right)^2 = 4\left(cx-a\mu\right)^2 - 4b^2z^2 \quad (4)$$

in an orthonormal basis $(O, \overrightarrow{\imath_0}, \overrightarrow{\jmath_0}, \overrightarrow{k_0})$ where O is called Dupin cyclide center. Parameters a, b and c are related by $c^2 = a^2 - b^2$. The parameter a is always greater than or equal to c. Parameters a, c and μ determine the type of the cyclide. When $c < \mu \leq a$ it is a ring cyclide, when $0 < \mu \leq c$ it is a horned cyclide, and when $\mu > a$ it is a spindle cyclide.

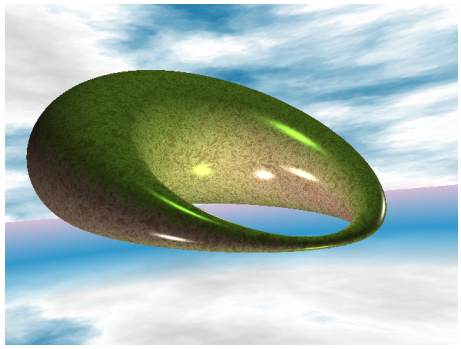
A Dupin cyclide admits two planes of symmetry $\mathcal{P}_y : y = 0$ and $\mathcal{P}_z : z = 0$ which define two couples of circles, called principal circles, figures 2(a) and 2(b). From the knowledge of a couple of principal circles and the Dupin cyclide type, it is easy to calculate Dupin cyclide parameters [Gar07].

If $c = 0$ and $a \neq 0$ then the Dupin cyclide is a torus, figure 1(b), and then :

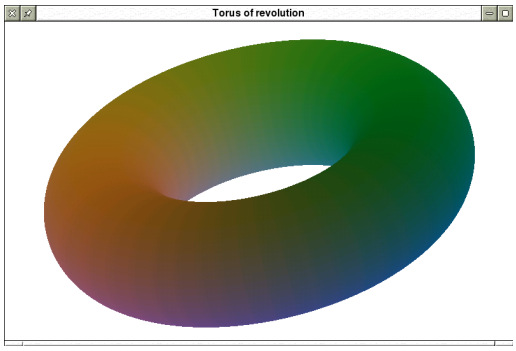
- principal circles of the Dupin cyclide in $\mathcal{P}_y$ have the same radius, they represent a torus meridian;
- principal circles of the Dupin cyclide in $\mathcal{P}_z$ become concentric circles.

If $a = c = b$ then a Dupin cyclide is a double sphere and principal circles in both planes are identical.

The planes containing circles of curvature of a Dupin cyclide form two pencils of planes, figure 3, and define two



(a)



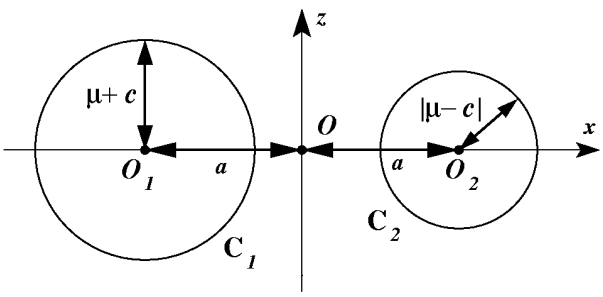
(b)

Figure 1: A ring Dupin cyclide (a) and a ring torus (b).

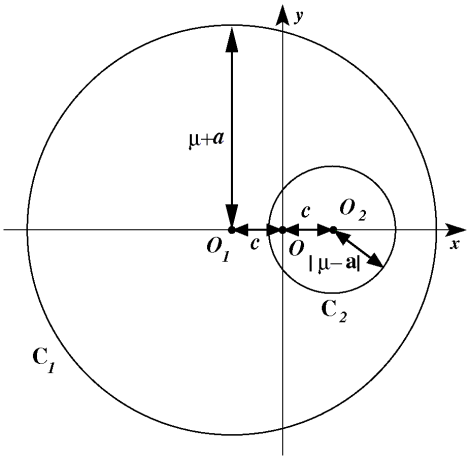
straight lines Δ_θ as the intersection of the planes of the first pencil and Δ_ψ as the intersection of the planes of the other pencil. If the Dupin cyclide is a torus, the line Δ_ψ belongs to the infinity plane (the planes containing circles of curvature are parallel).

Figure 4 shows lines Δ_θ and Δ_ψ with the principal circles of the ring Dupin cyclide in plane $\mathcal{P}_y$ ($\mathtt{C}_1^\theta$, $\mathtt{C}_2^\theta$) and in plane $\mathcal{P}_z$ ($\mathtt{C}_1^\psi$, $\mathtt{C}_2^\psi$). Δ_0 is the common perpendicular to Δ_θ and Δ_ψ. More details about properties of Dupin cyclide can be found in [Pra90, She98, AD96, DMP93].

Several authors have proposed algorithms to convert a Dupin cyclide patch into a Bézier surface [Pra90, Ued95, AD96, FGP05, Gar07] and vice-versa [Gar07, GFN06]. Table 1 gives the four most important properties of control points of a Bézier surface obtained by the conversion of a Dupin cyclide patch.

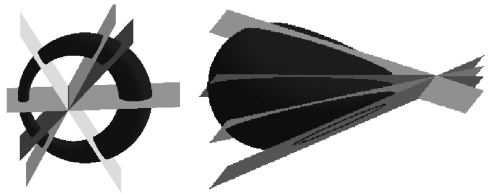


(a)



(b)

Figure 2: Principal circles of ring Dupin cyclides. (a) : in plane $\mathcal{P}_y : y = 0$. (b) : in plane $\mathcal{P}_z : z = 0$.



(a) (b)

Figure 3: Two pencils of planes generated by Dupin cyclide curvature circles defining Δ_θ (a) and Δ_ψ (b).

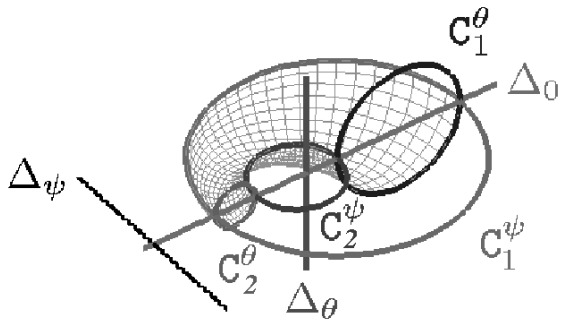


Figure 4: Straight lines Δ_θ and Δ_ψ obtained as intersections of two pencil planes.

(PG1)	P_{00}, P_{02} , P_{22} et P_{20} are cocyclical
(PG2)	$P_{00}P_{01} = P_{01}P_{02}$ $\quad P_{02}P_{12} = P_{12}P_{22}$ $P_{22}P_{21} = P_{21}P_{20}$ $\quad P_{00}P_{10} = P_{10}P_{20}$
(PG3)	$\overrightarrow{P_{00}P_{10}} \perp \overrightarrow{P_{00}P_{01}}$ $\quad \overrightarrow{P_{02}P_{01}} \perp \overrightarrow{P_{02}P_{12}}$ $\overrightarrow{P_{22}P_{12}} \perp \overrightarrow{P_{22}P_{21}}$ $\quad \overrightarrow{P_{20}P_{21}} \perp \overrightarrow{P_{20}P_{10}}$
(PG4)	$\overrightarrow{P_{00}P_{11}} \bullet \left(\overrightarrow{P_{00}P_{10}} \times \overrightarrow{P_{00}P_{01}}\right) = 0$ $\overrightarrow{P_{02}P_{11}} \bullet \left(\overrightarrow{P_{02}P_{01}} \times \overrightarrow{P_{02}P_{12}}\right) = 0$ $\overrightarrow{P_{22}P_{11}} \bullet \left(\overrightarrow{P_{22}P_{12}} \times \overrightarrow{P_{22}P_{21}}\right) = 0$ $\overrightarrow{P_{20}P_{11}} \bullet \left(\overrightarrow{P_{20}P_{21}} \times \overrightarrow{P_{20}P_{10}}\right) = 0$

Table 1: Geometrical properties of a Bézier surface obtained by conversion of a Dupin cyclide.

In table 1, property (PG4) can be presented as:

$$\begin{aligned} P_{11} \quad &\in \quad Aff\{P_{00}; P_{01}; P_{10}\} \cap Aff\{P_{02}; P_{12}; P_{01}\} \\ &\cap \quad Aff\{P_{20}; P_{21}; P_{10}\} \cap Aff\{P_{22}; P_{21}; P_{12}\} \end{aligned} \tag{5}$$

where $Aff\{A; B; C\}$ designate the affine space generated by points A, B and C.

3 Construction of the Bézier surface

In this section, we construct a Bézier surface convertible to a Dupin cyclide patch and so properties of table 1 must

 ISBN 978-80-86943-94-7

be satisfied [Gar07, GFN06]. The construction of a Bézier surface convertible to a torus patch or a double sphere patch is also considered.

3.1 Dupin cyclide case

As Dupin cyclide curvature lines are circles, the border lines of the Bézier surface to be convertible must be circular arcs. To ensure the convertibility of the Bézier surface, the following three conditions, on weight computation, has to be satisfied:
(i) we have $w_{00} = w_{02} = w_{20} = 1$ and value of w_{22} is calculated using Ueda's method [Ued95];
(ii) as border lines of a Bézier surface are Bézier curves reprenting circular arcs, it is easy to determine the weights w_{10}, w_{01},w_{21} and w_{12} [Gar07];
(iii) the computation of weight w_{11} is more complex and can be done using theorem 1 where $\mathsf{b}ar\left\{(A_i,\alpha_i)_{i\in I}\right\}$ indicates the barycentre of collection $(A_i,\alpha_i)_{i\in I}$ of level-headed points:

Theorem 1 *Barycentric middle curve*
Let us consider a Bézier surface defined by control points $(P_{ij})_{0\le i,j\le 2}$ *and weights* $(w_{ij})_{0\le i,j\le 2}$.
Let $G_i^u = \mathsf{b}ar\left\{(P_{i0};w_{i0}),(P_{i1};2w_{i1}),(P_{i2};w_{i2})\right\}$ *and* $\alpha_i^u = w_{i0}+2w_{i1}+w_{i2}$ *where* $i\in[\![0;2]\!]$.
Let $G_i^v = \mathsf{b}ar\left\{(P_{0i};w_{0i}),(P_{1i};2w_{1i}),(P_{2i};w_{2i})\right\}$ *and* $\alpha_i^v = w_{0i}+2w_{1i}+w_{2i}$ *where* $i\in[\![0;2]\!]$.
If $\sum_{i=0}^2 \alpha_i^u \neq 0$, *the barycentric middle curve* $u \mapsto M\left(u,\frac{1}{2}\right)$
is a Bézier curve with control points $(G_i^u;\alpha_i^u)_{i\in[\![0;2]\!]}$.
If $\sum_{i=0}^2 \alpha_i^v \neq 0$, *the barycentric middle curve* $v \mapsto M\left(\frac{1}{2},v\right)$
is a Bézier curve with control points $(G_i^v;\alpha_i^v)_{i\in[\![0;2]\!]}$.

Proof:

$$\overrightarrow{OM\left(u,\tfrac{1}{2}\right)} = \frac{\sum_{i=0}^2\sum_{j=0}^2 w_{ij}B_i(u)B_j\left(\frac{1}{2}\right)\overrightarrow{OP_{ij}}}{\sum_{i=0}^2\sum_{j=0}^2 w_{ij}B_i(u)B_j\left(\frac{1}{2}\right)}$$

$$= \frac{\sum_{i=0}^2 B_i(u)\left(w_{i0}B_0\left(\frac{1}{2}\right)\overrightarrow{OP_{i0}}+w_{i1}B_1\left(\frac{1}{2}\right)\overrightarrow{OP_{i1}}+w_{i2}B_2\left(\frac{1}{2}\right)\overrightarrow{OP_{i2}}\right)}{\sum_{i=0}^2 B_i(u)\left(w_{i0}B_0\left(\frac{1}{2}\right)+w_{i1}B_1\left(\frac{1}{2}\right)+w_{i2}B_2\left(\frac{1}{2}\right)\right)}$$

$$= \frac{\sum_{i=0}^2 B_i(u)\left(w_{i0}\overrightarrow{OP_{i0}}+2w_{i1}\overrightarrow{OP_{i1}}+w_{i2}\overrightarrow{OP_{i2}}\right)}{\sum_{i=0}^2 B_i(u)(w_{i0}+2w_{i1}+w_{i2})}$$

$$= \frac{1}{\sum_{i=0}^2 B_i(u)\alpha_i}\sum_{i=0}^2 B_i(u)\,\alpha_i\overrightarrow{OG_i^u}$$

where $G_i^u = \mathsf{b}ar\left\{(P_{i0};w_{i0}),(P_{i1};2w_{i1}),(P_{i2};w_{i2})\right\}$
with $\alpha_i = w_{i0}+2w_{i1}+w_{i2}$.
The second proof is similar.
■

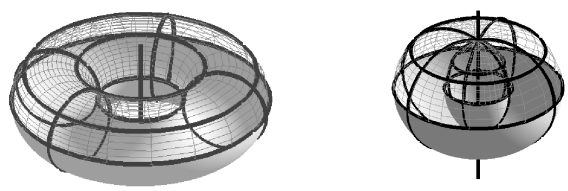
To determine the weigth w_{11}, we impose that the point G_1^u (resp. G_1^v) belongs to the perpendicular bissector plane of $[G_0^uG_2^u]$ (resp. $[G_0^vG_2^v]$).

The following section considers the conversion of a Bézier surface into a patch of a torus or a patch of a double sphere which are particular cases of Dupin cyclides. The general cases (conversion of a Bézier surface into a regular Dupin cyclide) has been consider earlier [Gar07, GFN06]. To summerize, a convertible Bézier surface must satisfy control points properties of table 1 and the three weight conditions given above.

3.2 Torus and double sphere case

We consider that two opposite edges (circular arcs) of the Bézier surface to be converted are in two parallel planes. In this case, the line Δ_ψ belongs to the infinity space and the conversion result will be a patch of a torus ($c=0$, $a\neq 0$) or a patch of a double sphere ($a=c=0$). To distinguish between these two cases, we have to consider the remaining edges of the Bézier surface: if the circles containing the edges have the same diameter line, it is a double sphere patch, otherwise, it is a torus patch. The type of torus can be determined throught the following three tests:

- if the circles are disjoint, the result is a ring torus patch;
- if the circles are secant in two points, the result is a spindle torus patch and the two points of intersection define Δ_θ which will be used as a frame axis in the conversion algorithm, figure 5(b);
- if the circles are tangent, the result is a horn torus patch.



(a) (b)

Figure 5: Torus as Dupin cyclide. (a) : ring torus. (b) : spindle torus.

We note here that the frame $(O,\overrightarrow{\imath_0},\overrightarrow{\jmath_0},\overrightarrow{k_0})$ in which the resulting patch is defined is not the same as the one of the initial Bézier surface. Vectors $\overrightarrow{\imath_0}$ and $\overrightarrow{\jmath_0}$ are perpendicular and belong to the vector plane attached to the affine planes containing the parrallel circles. The third vector of the frame is $\overrightarrow{k_0} = \overrightarrow{\imath_0}\times\overrightarrow{\jmath_0}$.

4 The conversion algorithm

Let γ be a standard Bézier curve definied by level-headed control points $(P_0;1)$, $(P_1;w_1)$ and $(P_2;1)$ such that:

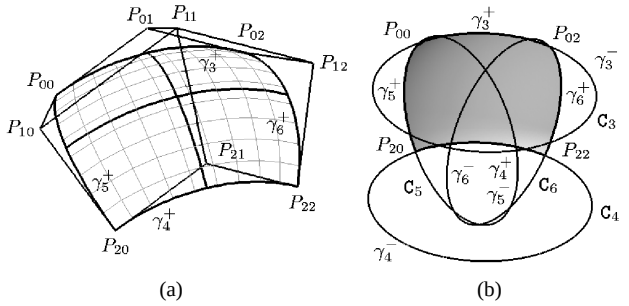
- $w_1 = f_w\left(P_0;P_1;P_2\right)$ where $f_w : {\mathcal{E}_3}^3 \to \mathbb{R}$ [Gar07];

 ISBN 978-80-86943-94-7

- γ is a circular arc.

Algorithm 1 details the steps required to convert a Bézier surface into a torus patch or a double sphere patch. Figure 6(a) shows the original Bézier surface, its control polyhedron, its two barycentric middle curves (theorem 1) as well as Bézier curves γ_3^+, γ_4^+, γ_5^+ and γ_6^+ which determine the edge of the Bézier surface.

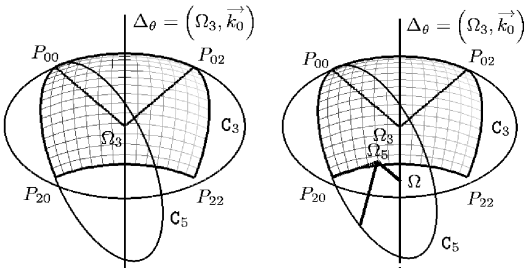
First step of algorithm 1 is the determination of circles delimiting the Bézier surface. Each circle is reprented by an union of two Bézier curves having extremal weights equal to 1, figure 6(b). e.g., circle C_3 is the union of Bézier curves having control points P_{00}, P_{01} and P_{02}, and opposite median weights $f_w\left(P_{00};P_{01};P_{02}\right)$ and $-f_w\left(P_{00};P_{01};P_{02}\right)$ [Gar07].



(a) (b)

Figure 6: The conversion algorithm.(a) : the Bézier surface. (b) : circular Bézier surface edges.

Figure 7(a) shows the third step of the algorithm. Straight line Δ_θ is perpendicular to planes generated by parallel circles. It passes through the center of one of these circles (C_3 in the figure). Figure 7(b) shows the fourth step of the algorithm. Plane $\mathcal{P}_z$ is the plane passing through Ω with $\overrightarrow{k_0}$ as an orthogonal vector, where Ω is the perpendicular projection of center Ω_5 of circle C_5 onto the straight line $(\Omega_3, \overrightarrow{k_0})$.



(a) (b)

Figure 7: The conversion algorithm. (a) : determination of Δ_θ. (b) : determination of new frame origin Ω and plane $\mathcal{P}_z$.

Figure 8(a) permits, using Dupin cyclide plane of symmetry $\mathcal{P}_z$, the construction of points A et B belonging to

Algorithme 1 : Conversion of a Bézier surface into a torus patch or a double sphere.

Input data:
Let S be a Bézier surface defined by level-headed control points $(P_{ij};w_{ij})_{0\leq i,j\leq 2}$ such as:

$$Aff\left(P_{00};P_{01};P_{02}\right) //_{\neq} Aff\left(P_{20};P_{21};P_{22}\right) \qquad (6)$$

Begin

1. Bézier surface edges are represented by standard Bézier curves:

Name	Control points	Intermediate weight
γ_3^+	$(P_{00},P_{01};P_{02})$	$f_w\left(P_{00};P_{01};P_{02}\right)$
γ_3^-	$(P_{00},P_{01};P_{02})$	$-f_w\left(P_{00};P_{01};P_{02}\right)$
γ_4^+	$(P_{20};P_{21};P_{22})$	$f_w\left(P_{20};P_{21};P_{22}\right)$
γ_4^-	$(P_{20};P_{21};P_{22})$	$-f_w\left(P_{20};P_{21};P_{22}\right)$
γ_5^+	$(P_{00};P_{10};P_{20})$	$f_w\left(P_{00};P_{10};P_{20}\right)$
γ_5^-	$(P_{00};P_{10};P_{20})$	$-f_w\left(P_{00};P_{10};P_{20}\right)$
γ_6^+	$(P_{02};P_{12};P_{22})$	$f_w\left(P_{02};P_{12};P_{22}\right)$
γ_6^-	$(P_{02};P_{12};P_{22})$	$-f_w\left(P_{02};P_{12};P_{22}\right)$

   Given circles $\mathsf{C}_3 = \gamma_3^+ \cup \gamma_3^-$, $\mathsf{C}_4 = \gamma_4^+ \cup \gamma_4^-$, $\mathsf{C}_5 = \gamma_5^+ \cup \gamma_5^-$ and $\mathsf{C}_6 = \gamma_6^+ \cup \gamma_6^-$.
   Condition (6) implies $\mathsf{C}_3 \,//\, \mathsf{C}_4$, figure 6(b).

2. New reference frame determination: $\overrightarrow{\imath_0}$ and $\overrightarrow{\jmath_0}$ are two unit orthogonal vectors generating vector plan $Vect\left(P_{00};P_{01};P_{02}\right)$ and $\overrightarrow{k_0}$ is determined by :

$$\overrightarrow{k_0} = \overrightarrow{\imath_0} \times \overrightarrow{\jmath_0}$$

3. Altitude axis is $\Delta_\theta = (\Omega_3, \overrightarrow{k_0})$ where Ω_3 is the center of circle C_3, figure 7(a).

4. Let Ω_5 be the center of circle C_5. The origin of new reference frame is Ω, the orthogonal projection of Ω_5 onto $(\Omega_3, \overrightarrow{k_0})$. $\mathcal{P}_z$ is the plane passing through Ω with $\overrightarrow{k_0}$ as the orthogonal vector, figure 7(b).

5. Determination of points A and B such that: $\{A;B\} = \mathsf{C}_5 \cap \mathcal{P}_z$ and $\Omega B \leq \Omega A$, figure 8(a).

6. In $\mathcal{P}_z$, principal circles C_1 and C_2 are determined by center Ω and radius $\rho_1 = \Omega A$ and $\rho_2 = \Omega B$ respectively, figure 8(b).

7. Dupin cyclide parameters computation.
   If $\mathsf{C}_1 = \mathsf{C}_2$, we obtain a double sphere with $a = c = 0$ and $\mu = \rho_1$.
   If $\sharp\left(\mathsf{C}_6 \cap \mathsf{C}_5\right) = 2$, we obtain a spindle torus with $c = 0$, $a = \frac{\rho_1-\rho_2}{2}$ and $\mu = \frac{\rho_1+\rho_2}{2}$, else we obtain a ring torus or a horned torus with $c = 0$, $a = \frac{\rho_1+\rho_2}{2}$ and $\mu = \frac{\rho_1-\rho_2}{2}$, figure 9(a).

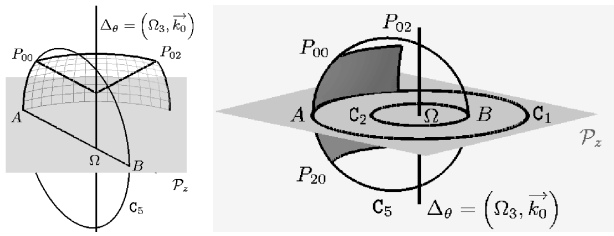
8. Determination of values θ_0, θ_1, ψ_0 and ψ_1 delimiting the obtained patch [Gar07], figure 9(b).

End

Output: A torus patch or a double sphere patch representing the input Bézier surface.

 ISBN 978-80-86943-94-7

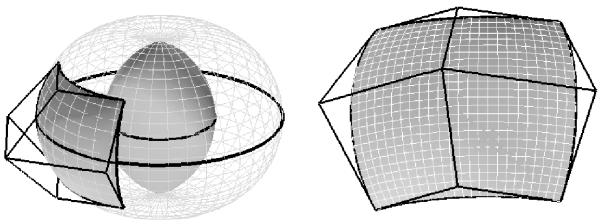
principal circles. Condition $\Omega B < \Omega A$ allows to identify immediately the great and the small torus principal circles. Condition $\Omega B = \Omega A$ implies that Dupin cyclide is a double sphere. Obviously, the center of these circles is the point Ω. Figure 8(b) shows two principal circles in $\mathcal{P}_z$.



(a) (b)

Figure 8: The conversion algorithm. (a) : plane $\mathcal{P}_z$. (b) : torus principal circles belonging to $\mathcal{P}_z$.

Figure 9(a) shows the torus, two principal circles, the Bézier surface with its control polyhedron. Obtained parameter values are $c = 0$, $a \simeq 1,63$ and $\mu \simeq 4,32$. Figure 9(b) shows the Bézier surface, its control polyhedron and the resulting torus patch which is delimited by curvature lines situated at: $\theta_0 \simeq 2,526112925$, $\theta_1 \simeq 3,757072362$, $\psi_0 \simeq 2,427868285$ and $\psi_1 \simeq 3,85531702$.

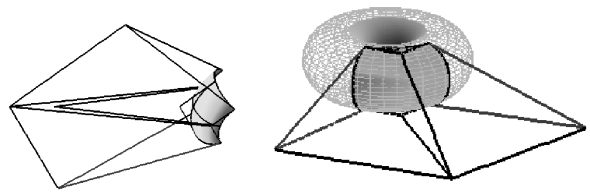


(a) (b)

Figure 9: The conversion algorithm. (a) : the Bézier surface with the control polyhedron and the spindle torus. (b) : the spindle torus patch with the control polyhedron of the initial Bézier surface.

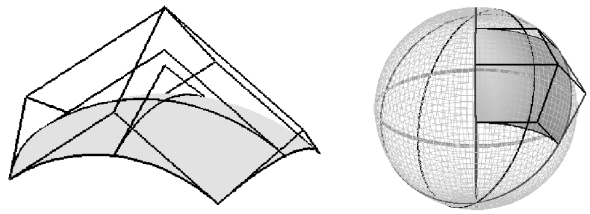
Figure 10(a) shows the Bézier surface with its control polyhedron, barycentric middle curves with their control polygons. Figure 10(b) shows the Bézier surface with its control polyhedron and the ring torus.

Figure 11 shows the conversion of a Bézier surface into a double sphere patch. Figure 11(a) shows the Bézier surface with its control polyhedron and the barycentric middle curves with their control polygons. Figure 11(b) shows the resulting double sphere patch with the control polyhedron of the initial Bézier surface.



(a) (b)

Figure 10: Conversion of a Bézier surface into a patch of a ring torus. (a) : the Bézier surface with the control polyhedron. (b) : the resulting patch of ring torus togother with the control polyhedron of the initial Bézier surface.



(a) (b)

Figure 11: Conversion of a Bézier surface into a double sphere patch. (a) : the Bézier surface with the control polyhedron. (b) : the resulting patch of the double sphere together with the control polyhedron of the initial Bézier surface.

5 Conclusion

In this paper, we have presented an algorithm which permits the conversion of a rational biquadratic Bézier surface into a torus patch or a double sphere patch which are particular Dupin cyclide patches. So, the rational biquadratic Bézier surface is fully represented by an implicit equation of degree 4. Moreover, if Dupin cyclide is a double sphere, it is possible to use an equation of degree 2 (the equation of the sphere).

An interesting extension of this work is to find the sufficiant conditions to construct rational biquadratic Bézier surfaces fully convertible into Dupin cyclide patches. The study of conversion of rational biquadratic Bézier surfaces into supercyclide patches will also be considered.

References

[AD96] G. Albrecht and W. Degen. Construction of Bézier rectangles and triangles on the symetric Dupin horn cyclide by means of inversion. *Computer Aided Geometric Design*, 14(4):349–375, 1996.

[BP98] W. Boehm and M. Paluszny. General cyclides as joining pipes. *Computer Aided Geometric Design*, 15:699–710, 1998.

[Dar17] G. Darboux. *Principes de géométrie analytique*. Gauthier-Villars, 1917.

[Deg94] W. L. F. Degen. Generalized Cyclides for Use in CAGD. In A. Bowyer, editor, *The Mathematics of Surfaces IV*, pages 349–363, Oxford, 1994. Clarendon Press.

[DMP93] D. Dutta, R. R. Martin, and M. J. Pratt. Cyclides in surface and solid modeling. *IEEE Computer Graphics and Applications*, 13(1):53–59, January 1993.

[DP98] G. Demengel and J. P. Pouget. *Mathématiques des Courbes et des Surfaces. Modèles de Bézier, des B-Splines et des NURBS*. Ellipse, 1998.

[Dup22] C. P. Dupin. *Application de Géométrie et de Méchanique à la Marine, aux Ponts et Chaussées, etc.* Bachelier, Paris, 1822.

[Far93] G. Farin. *Curves And Surfaces*. Academic Press, 3rd edition, 1993.

[Far99] G. Farin. *NURBS from Projective Geometry to Pratical Use*. A K Peters, Ltd, 2 edition, 1999. ISBN 1-56881-084-9.

[FGP05] S. Foufou, L. Garnier, and M. Pratt. Conversion of Dupin Cyclide Patches into Rational Biquadratic Bézier Form. In R. Martin, H. Bez, and M. Sabin, editors, *Proceedings of the 11th Conference on the Mathematics of Surfaces*, pages 201–218. Springer-Verlag Berlin Heidelberg, September 2005. ISBN: 3-540-28225.

[For12] A. R. Forsyth. *Lecture on Differential Geometry of Curves and Surfaces*. Cambridge University Press, 1912.

[For68] A. Forest. *Curves and Surfaces for Computer-Aided Design*. PhD thesis, University of Cambridge, 1968.

[Gar07] L. Garnier. *Mathématiques pour la modélisation géométrique, la représentation 3D et la synthèse d'images*. Ellipses, 2007.

[GFN06] L. Garnier, S. Foufou, and M. Neveu. Conversion d'un carreau de Bézier rationnel biquadratique en un carreau de cyclide de Dupin quartique. *TSI*, 25(6):709–734, 2006. numéro spécial AFIG'04, Hermès.

[HL93] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. A.K.Peters, Wellesley, MA., 1993.

[Pra90] M. J. Pratt. Cyclides in computer aided geometric design. *Computer Aided Geometric Design*, 7(1-4):221–242, 1990.

[Pra95] M. J. Pratt. Cyclides in computer aided geometric design II. *Computer Aided Geometric Design*, 12(2):131–152, 1995.

[PT89] L. Piegl and W. Tilles. A managerie of rational b-spline circles. *IEEE Computer Graphics and Applications*, 9(5):46–56, 1989.

[She98] C. K. Shene. Blending two cones with Dupin cyclides. *Computer Aided Geometric Design*, 15(7):643–673, 1998.

[Ued95] K. Ueda. Normalized Cyclide Bézier Patches. In *Mathematical Methods for Curves and Surfaces*, pages 507–516, Nashville, USA, 1995. Vanderbilt University Press.

Efficient Semiautomatic Segmentation of Plant Biological Objects Based on Live-wire Methods

Wolfram Schoor
Fraunhofer Institute for Factory Operation and Automation (IFF)
Sandtorstrasse 22,
39106 Magdeburg, Germany
wolfram.schoor@iff.fraunhofer.de

Thomas Seidl
Fraunhofer Institute for Factory Operation and Automation (IFF)
Sandtorstrasse 22,
39106 Magdeburg, Germany
thomas.seidl@iff.fraunhofer.de

Felix Bollenbeck
Fraunhofer Institute for Factory Operation and Automation (IFF)
Sandtorstrasse 22,
39106 Magdeburg, Germany
felix.bollenbeck@iff.fraunhofer.de

Udo Seiffert
Fraunhofer Institute for Factory Operation and Automation (IFF)
Sandtorstrasse 22,
39106 Magdeburg, Germany
udo.seiffert@iff.fraunhofer.de

Bernhard Preim
Otto von Guericke University,
Universitätsplatz 2,
39106 Magdeburg, Germany
preim@isg.cs.uni-magdeburg.de

Rüdiger Mecke
Fraunhofer Institute for Factory Operation and Automation (IFF)
Sandtorstrasse 22,
39106 Magdeburg, Germany
ruediger.mecke@iff.fraunhofer.de

ABSTRACT

This paper presents a novel method for efficient semiautomatic multi-label segmentation of plant biological image data. The approach extends live-wire methods in order to facilitate exact user-steered segmentations for atlas generation. By integrating a segmentation-specific user interaction model into the live-wire formulation i) more exact segmentations, ii) increased computational efficiency, iii) without loss of generality are achieved. The concept of mutual influence of image feature based path costs and user input uncertainty are consistently combined. By incorporating user behavior into cost based delineation a more intuitive user interface is obtained also yielding in a more accurate segmentation. We introduce path-based methodologies, specific user interaction models and propose the combination of both of them. The purposefulness of the method is shown in an application comprising segmentation of histological section data supporting the generation of 3-D atlases.

Keywords: computer vision, user interaction, live-wire.

1 INTRODUCTION

Image segmentation is the basic principle behind various image processing tasks. The main goal is to obtain a higher level of information than a purely numerical representation of an image. The pixels are therefore combined into semantically equivalent groups [Tön05]. Segmentation consists of two steps, recognizing relevant objects in an image and delineating exact spatial dimensions and their features [FUM00]. Different algorithms that can automatically segment plant biological image data have been introduced for this non-trivial task [BS08] or [DBS+08]. These algorithms are generally based on existing reference data. Such model data is not available to generate atlases of unexplored plant biological histological images and must therefore either be segmented manually or semiautomatically. A set of semiautomatically segmented image slices (approximately 5% of a serial section image stack) can replace the purely manual segmentation of plant biological image data as presented in [GDB+07]. Image data consists of previously sectioned plant biological objects, i.e. caryopses of the Hordeum Vulgare, digitized with a color CCD camera with a resolution of 1600 x 1200 pixels under a light microscope. To facilitate their processing, the digitized images were converted to an 8 bit gray scale representation without any significant loss of relevant information (see [SBH+08]). Figure 1 is an image for such a segmentation task. The detail of the segmented region bounded in blue reveals obvious differences in the contour's characteristics (red and green box).

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

Plzen, Czech Republic.
Copyright UNION Agency – Science Press

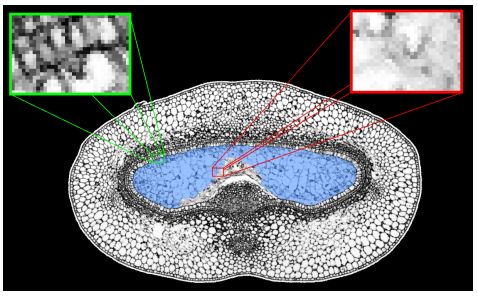


Figure 1: Slice of *Hordeum Vulgare* with different tissue characteristics (red and green box).

2 RELATED WORK

A graph-based semiautomatic interactive segmentation method, namely *live-wire*, was introduced by MORTENSEN et al. [MMBU92], [MB95], [BM97] and is related to *geometric active contours* presented by KASS et al. [KWT87]. The key feature of live-wire methods is their exact extraction of a boundary segment between a start point P_s and an end point P_e indicated by the position of the input device. The task of segmenting an entire region is broken down into optimal delineation of several parts comprising the entire contour. To do so, a cost map is calculated based on each pixel edge's properties and interpretation. A graph is introduced reflecting the cost map, which allows a path search for optimal path computation by the minimal accumulated path costs (cf. [HNR68] A*-algorithm). A set of weighted parameters derived from the pixel's neighborhood is employed to compute the path between the start and the end point. In an ideal case, a rough sketch of the contour suffices to extract the segment boundaries. Figure 2 illustrates the principle of the live-wire method. For clarity's sake, only a 4-cell neighborhood was selected for the visualization.

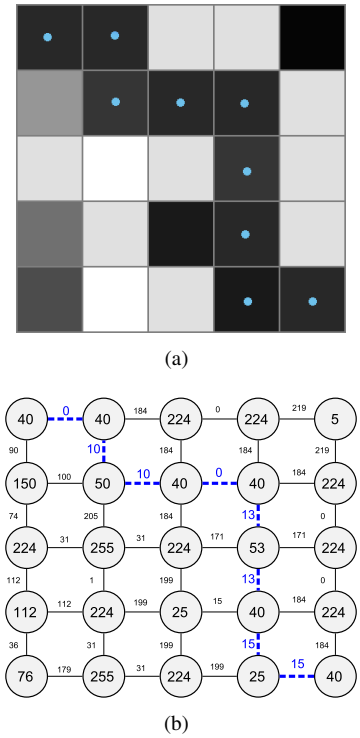


Figure 2: (a) portion of a gray scale image, (b) shows respectively the graph, resulting from intensity differences. The intensity values appear in the nodes, related cost components between the nodes. Blue points indicate the optimal connection along the lowest total cost components (dashed line). The path follows the dark pixels from the upper left to the lower right corner.

3 METHODS

A core element of the live-wire method is the calculation of the aforementioned costs related to the edges in the graph. Depending on the image data, different cost factors can be applied for good segmentation results. The following listed cost factors constitute a combination of different cost factors, which are suitable for the used image data. The cost factors are derived from [MMBU92], [MB95] and are also based on own formulations.

The impact of different cost factors is weighted as an overall cost function. Let $\boldsymbol{p}$ and $\boldsymbol{q}$ be two adjacent pixels in an image slice S. The costs for a connection from $\boldsymbol{p}$ to $\boldsymbol{q}$ can be determined by a cost function $C(\boldsymbol{p},\boldsymbol{q})$, rendered thusly:

$$C(\boldsymbol{p},\boldsymbol{q}) = \sum_{n=0}^{N} \omega_n C_n(\boldsymbol{p},\boldsymbol{q}), \tag{1}$$

where N denotes the count of the different cost factors, ω_n the respective weighting factors and $C_n(\boldsymbol{p},\boldsymbol{q})$ the value of the n-th cost component between $\boldsymbol{p}$ and $\boldsymbol{q}$. The cost factors C_n can be differentiated as *static* and *dynamic* factors:

Static factors are calculated before segmentation and are independent of user interaction. Static cost factors indicate the cost to directly connect one pixel to its m neighbors in the graph-based path search. Filter operations are applied to determine the static cost factors (see Table 1):

Cost Factor	Cost Identifier (C_n)
Laplacian	C_L
Laplacian of Gaussian	C_{LoG}
Gradient magnitude	C_{GM}
Roberts Cross	C_{RX}
Difference of Intensity Values	$C_{\Delta I}$
Brightness Feature	C_{BF}

Table 1: Components of local cost functions between adjacent pixels.

An individual local cost factor is calculated following Equations 2-10:

$$C_L(p) = 1 - (L_{kernel}(x,y) \cdot S) \tag{2}$$

$$L_{kernel} = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} \tag{3}$$

were each summand in L_{kernel} stands for the second partial derivation in x respectively in y direction.

$$C_{LoG}(p) = 1 - (LoG_{kernel}(x,y) \cdot S) \tag{4}$$

$$LoG_{kernel} = -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right) e^{-((x^2+y^2)/2\sigma^2)} \tag{5}$$

 ISBN 978-80-86943-94-7

$$C_{RX}(p) = 1 - (RX_{kernel}(x,y) \cdot S) \tag{6}$$

$$RX_{kernel} = \frac{\partial p}{\partial x} + \frac{\partial p}{\partial y} \tag{7}$$

$$C_{GM}(q) = 1 - \frac{1}{max(G)} \sqrt{\left(\frac{dS(x,y)}{dx}\right)^2 + \left(\frac{dS(x,y)}{dy}\right)^2}, \tag{8}$$

$max(G)$ represents the largest gradient magnitude in the image. The $C_{\Delta I}(p)$ cost can be calculated as follows:

$$C_{\Delta I}(p) = \frac{|I(p) - I(q)|}{2^B - 1}, \tag{9}$$

where $I(p)$ and $I(q)$ denote the intensity value of their respective pixel positions and B the color depth of the current image. The brightness cost function C_{BF} can be expressed as:

$$C_{BF}(p) = I/max(I), \tag{10}$$

where low intensity values I correspond to low costs C_{BF}.
The *dynamic costs* cannot be computed in advance of a path search. They are computed during segmentation and reliant on user input. Table 2 lists the dynamic costs.

Cost Component	Cost Identifier (C_n)
L_0 distance to direct path	C_{L0}
L_1 distance to end point	$C_{\Delta P}$

Table 2: Dynamic components of the cost function.

Equation 11 calculates the orthonormal vector $\vec{v}$ to the straight line between P_s and P_e.

$$\vec{v} = \frac{\begin{pmatrix} P_{e_y} - P_{s_y} \\ -(P_{e_x} - P_{s_x}) \end{pmatrix}}{\sqrt{(P_{e_x} - P_{s_x})^2 + (P_{e_y} - P_{s_y})^2}} \tag{11}$$

Equation 12 determines the vector $\vec{r}$ from the start point P_s to the present location q.

$$\vec{r} = \begin{pmatrix} P_{q_x - s_x} \\ P_{q_y - s_y} \end{pmatrix} \tag{12}$$

Projecting $\vec{r}$ onto the normalized vector $\vec{v}$ yields the L_0 distance from point q to the straight line spanned by P_s and P_e and called $d_{L_0}(\boldsymbol{q})$ (see Equation 13).

$$d_{L_0}(\boldsymbol{q}) = \vec{v} \cdot \vec{r} = v_x\, r_x + v_y\, r_y \tag{13}$$

The costs increase as the pixel's distance to the ideal connection path increases (see Equation 14).

$$C_{L0}(P_s, P_e, q) = g(d_{L_0}(\boldsymbol{q})) \tag{14}$$

The function $g(x)$ may express this influence in a variety of ways, e.g. as a linear dependency or square root. Equation 15 provides an appropriate method:

$$g(x) = a\, x^2 \tag{15}$$

This quadratic function penalizes small deviations from the ideal line with small cost values, whereas larger distances become comparatively expensive. The increasing coefficient a of 0.25 damps too high cost values for distant input device positions.

The cost function $C_{\Delta P}$ represents the costs of a pixel q emerging from the distance to the end point P_e in relation to the neighboring pixel's p distance to P_e (see Equation 17). The term $d_{\Delta P}(i)$ with $i \in \{p,\ q\}$ denotes the L_1 distance (Manhattan distance) to P_e and is calculated with Equation 16.

$$d_{\Delta P}(i) = (P_{e_x} - i_x) + (P_{e_y} - i_y) \tag{16}$$

$$C_{\Delta P} = \begin{cases} 0.0, & \text{if } d_{\Delta P}(p) - d_{\Delta P}(q) \geq 1 \\ 0.5, & \text{if } d_{\Delta P}(p) - d_{\Delta P}(q) = 0 \\ 1.0, & \text{if } d_{\Delta P}(p) - d_{\Delta P}(q) \leq -1 \end{cases} \tag{17}$$

The basic live-wire methods may be extended in a variety of ways, e.g. by extending the search space to the third dimension [SPP00], [SPP01], [SUG+06] or by extracting the segmentation parameters for segments or segment parts [FUS+98] or [EKS92].
A detailed discussion of these extension is out of the scope of this paper. For a detailed analysis of live-wire's implementation and interaction methods for image segmentation, the authors refer to [BM97] or [OS01].

4 EXTENDING THE LIVE-WIRE METHOD

In [SBH+08], SCHOOR et al. introduced interaction-based support of segmentation, i.e. *speed-dependent automatic zooming* (SDAZ) based on IGARASHI [IH00] and *pseudohaptic feedback* (see LÉCUYER [LBE04]), to improve the segmentation process in terms of segmentation speed and accuracy of results. A user study demonstrated that this also works in practical use [SBH+08].
Different application-dependent algorithms and heuristics accelerate the run- and reaction-time of the application presented here.

4.1 Restricting the Search-Space

Apart from the large quantity of image data, the main disadvantage is that only a fraction of the data is relevant to the determination of the correct path. A

combination of new and established strategies computes paths faster and supports real-time interaction.

Decreasing the Global Search-Space:
In this very important step, the information space must be narrowed before calculating the path, even before generating the cost map. This can be done by simply thresholding the image and ignoring undesired intensity values in image parts or pixels.
Due to the nature of the image data approximately 50% of the pixel amount can so be masked in advance of all further computational steps. Unimportant areas (black) can be masked as in Figure 3.

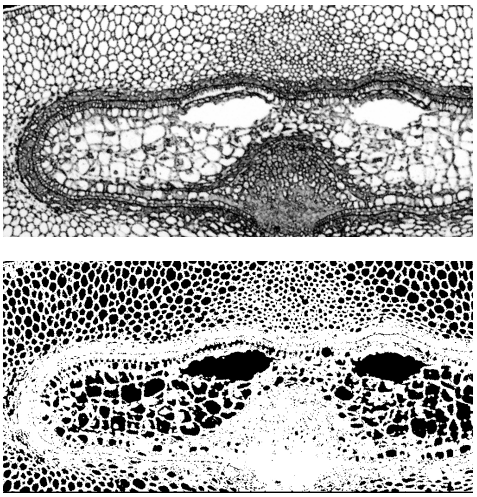


Figure 3: Original (top) and masked (bottom) image slice.

On the Fly Path Restriction:
The segmentation of object boundaries can be difficult, especially if the boundaries are discontinuous and very close to each other (see Figure 1). Manually or automatically setting a user-defined control point in the system restricts the search space to a rectangular path search between the two consecutive points and a predefined pixel distance to the direct path connection.
Unlike [SPP00] where the target search space can broadly be masked before segmentation starts, this method restricts the path on the fly between the last segment and the new current point. Figure 4 presents the principle. The yellow border encloses the search space with the present distance parameter.
This restriction is useful because a pixel's corresponding node is not expanded in the graph if its distance to the ideal line exceeds the defined value. This limits the depth of a path search and allows it to be terminated faster. The expected speed increase will be small in regions with strong straight edges but will prevent the bleeding of the contour in homogeneous regions with weak edges. Selecting a small distance value accelerates a path search (very small information search space) but attracts the contour line to the ideal line between the two considered points and can therefore differ from the required values.

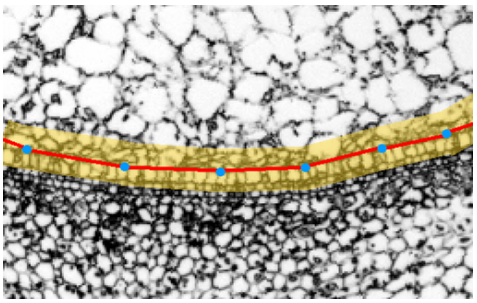


Figure 4: A border (with predefined pixel distance to the ideal line between two consecutive points) is used to restrict the information search space for the path search.

Speed Dependent Path Restriction:
The speed of user interaction during segmentation is a characteristic feature of segmentation (see [SBH+08]) that can be used to determine whether homogeneous regions or curvy contours must be segmented. This allows determining the distance value automatically. If user interaction during segmentation is slow, the distance value should be decreased.

Masking of Regions
Recalling LUCCHESE [LM01], that the segmentation is a subdivision of the image into n disjunct sections, a sequentially performed segmentation of two adjacent regions can cause the formation of "isles", due to filter operations, user inaccuracies or other reasons. To avoid this and speed up the segmentation process a masking of already segmented regions ensures an ideal segmentation of the image (one pixel only belongs to one region). If a user navigates into one masked region, the pixel adjacent to that regions's boundary is selected as the current point for path extraction. Figure 5 presents an example of a new path segment (blue) adapting to the region masked in red.

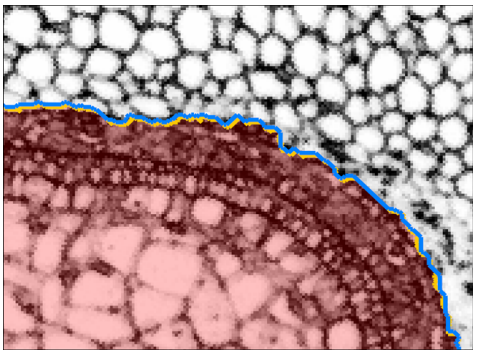


Figure 5: The masked region (transparent red) cannot be segmented again.

Tiling of the Image:
The mass of histological image data make it impossible

to guarantee that the different subsequent image based filter operations are calculated efficiently, especially when other time-critical calculations must be solved synchronously. One solution is to use graphics hardware as numerical processor as outlined in [RVC07]. The drawback of this solution is the need for appropriate graphics hardware in the meaning of limited texture size. Another solution is to break down an image segmentation problem into smaller elements, e.g. by using overlapping tiles as demonstrated in CRISP et al. [CPR03]. The overlap of the tiles depends on the maximum filter size used. An example is presented in Figure 6.
The graph segment corresponding to a tile is generated when:

1. The input device enters a new tile
2. The path search arrives a new tile
3. Enough time for further tile computation is left (concerning the real time interaction criterion)

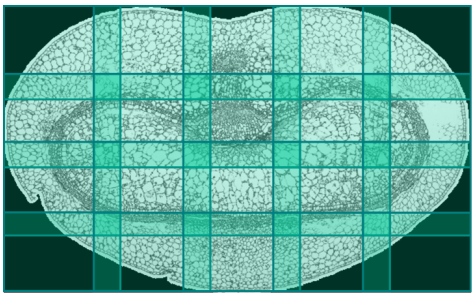


Figure 6: Tiling a large image slice with overlap.

4.2 Improving User Interaction

The system's real time response speed during segmentation can be assumed. More important is a simple interaction mechanism that supports users during the time consuming task of segmentation.

Automatic End Point Shifting:
The contour of the region between the last point and the input device's current position is calculated during semiautomatic segmentation. Control points and input device positions are entirely bound to user interaction. The connection between these points can be calculated very efficiently. Control point position and end point position must therefore be placed exactly. Manually setting control points can be time consuming and entails additional user effort. A possible end point is set orthogonally to the line between the last control position and the input device's current position (see Figure 7).

Automatic and Manual Control Points:
The complexity of path calculation depends directly

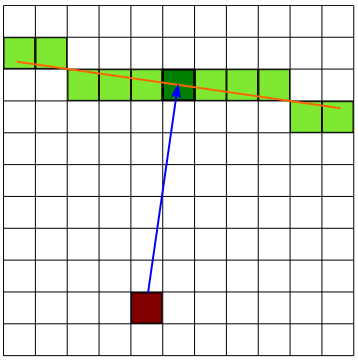


Figure 7: Scheme to calculate potential end points.

on the path length. Are start and end points for the path calculation further away from each other, more pixel have to be taken into account for the correct path determination. A divide and conquer strategy is employed to accelerate path determination. The user roughly sketches the contour to segment. If the input device position exceeds a predefined distance Δ_d to the start position or a predefined time interval Δ_t, a control point is automatically added to the position. The path determined becomes fixed.
The path determination for the whole path will be calculated in the following only between the last control point and the end point respectively the next automatically or manually inserted control point.
Figure 8 presents a segment part (red dashed line) following a segment boundary. Green points are control points manually inserted in difficult positions in the image (e.g. intersecting lines, line breaks, sharp changes of direction, etc.). Purple points are cooled control points that have remained unchanged for a long time (see [BM96] for cooled control points) or have a distinct number of following control points and therefore, cannot be deleted by incidence (for example by doing a segmentation loop). Blue points are recently inserted control points. These points can be deleted by entering a point surrounding region with a predefined radius. Purple points can then warm up again, i.e. turn blue again.

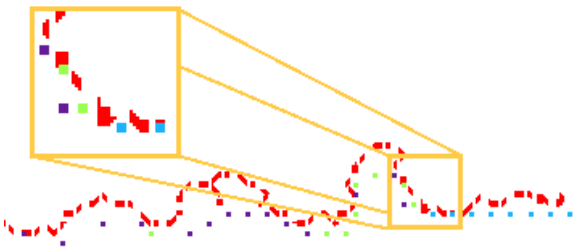


Figure 8: Temporary segment part with manually inserted control points (green), automatically inserted break points (blue) and locked break points (purple), also shown in closer detail.

 ISBN 978-80-86943-94-7

Disabling the Path Search:
Simply deactivating a path search may be advantageous in some cases, e.g. if the contour line changes radically over time. The parameters initially used to extract the contour may result in erroneous segmentation results. Thus, the segmentation can be partially done manually by contemporaneously pressing a predefined key.

5 RESULTS

The algorithms presented were tested on histological plant biological data sets. The results from the segmentation of a representative number of image slices can be used for the method of automatic segmentation proposed by BOLLENBECK et al. [BS08].

5.1 Experience

In Figure 9 a complete segmentation of a histological slice is shown with respective tissues.

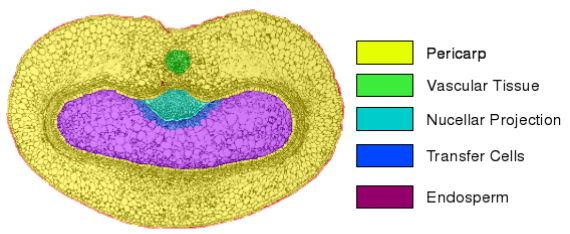


Figure 9: Labeled histologic image slice with legend of tissues.

Experimental studies evaluated the weights of the particular cost factors. The cost factors are mapped in the normalized interval [0, 1]. This enables a uniform weighting due to the weighting factors. Table 3 presents the different tissues of a grain caryopsis and their weighting factors. The weighting factors are bounded by the interval [0, 100]. These experimental evaluated values can be used for similar segmentation tasks of histological plant biological image data as initial values for the used cost factors. This can lead to a faster setting of the cost factors for well suited segmentation results.
Figure 10 presents a part of a reconstructed model and the respective sample segmentation slices.

5.2 User Tests

The developed live-wire method was used in practical tests for the segmentation of biological plant data. Users were asked to segment the *endosperm* of the barley grain, which is shown in Figure 9 (violet). This segment was chosen because its border is defined by different features, including strong clear edges but also

Tissue	C_L	C_{LoG}	C_{GM}	C_{BF}	C_{RX}	$C_{\Delta I}$	C_{L0}	$C_{\Delta P}$
Exterior	30	20	0	20	15	25	5	5
Vascular bundle	10	20	5	15	20	10	10	10
Pericarp	30	20	0	20	15	25	5	5
Transfer cells	20	15	10	20	15	20	5	5
Blowhole	25	25	5	10	10	20	5	5
Nucellar projection	30	25	0	20	10	10	5	5
Endosperm	30	25	0	20	15	10	10	5

Table 3: Weights of the different cost factors for each tissue.

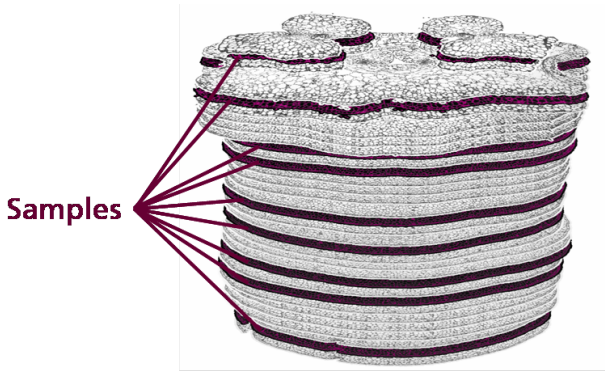


Figure 10: Reconstructed upper model part of a grain model using sample segmentations

weak unclear or even interrupted edges.
The users were also asked to segment the chosen part completely manually without system assistance. Eight out of ten users achieved faster results with the here proposed semiautomatic method. Partially the live-wire method was twice as fast as the manual segmentation. The two users, who needed more time declared that they did not use the option to switch to manually segmentation during the process.
This leads to time-consuming segmentation attempts by repeating wrong segmentations in some parts of the image with weak edges or large gaps between corresponding edges.

Accuracy:
According to MORTENSEN AND BARRETT [MB95] the accuracies of the users segmentations were compared with a segmentation, which was manually done by an expert (gold standard). The resulting accuracy graphs show a very high degree of similarity over all segmen-

tation tools which were tested. The commercial software solutions achieve a better correlation than the here provided software in the range of very small deviations to the gold standard (especially Photoshop). If the Hausdorff-distance is greater than 2 pixels the presented approach performs best.
With respect to physical diameters of a tissue boundary, which can be for example up to 4 pixels in size in the image data, Hausdorff-distances of 3 or 4 pixels are still within an acceptable accuracy range.

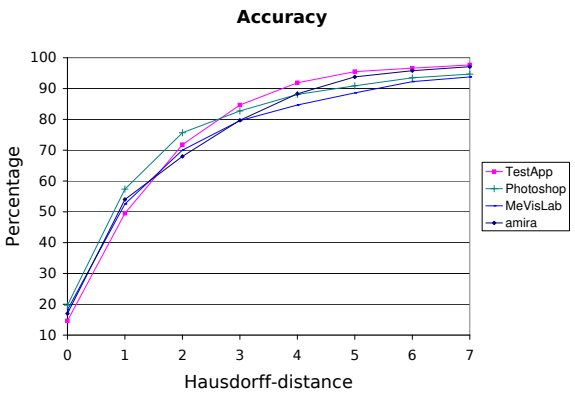


Figure 11: Accuracy results in comparison with other segmentation tools.

Reproducibility:
By comparing the results of different segmentations with each other the developed method had a high reproducibility in intra-user and inter-user comparison (see Figure 12 a) and Figure 12 b)). That means that different segmentations of one user were almost the same but also the segmentations of different user had a very high similarity.

Usability:
Asking the users for a subjective rating of the presented method, nine out of ten users confirmed a good usability of the presented method, which is easy and fast to learn. Especially the automatic placement and the different states of the control-points were appreciated by most users.

6 CONCLUSIONS

A variety of enhancements improve semiautomatic segmentation tasks on digitized plant biological serial sections. A user test demonstrated that the system is suitably usable. The objective improvements in comparison to commercial segmentation software must be further evaluated. The presented system has been proven to be as accurate as *amira* `www.amira.com`, *MeVisLab* `www.mevislab.de` and *Photoshop*'s `www.adobe.com` semiautomatic segmentation capabilities for selected image data.
The system was especially designed for large image

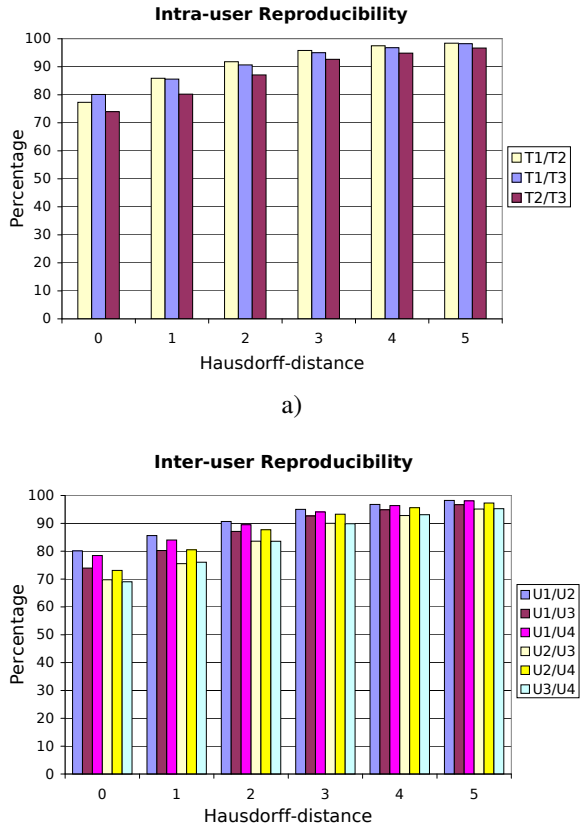


Figure 12: The reproducibility is shown on the basis of one histological image slice segmentation. The matching results of three segmentations $T1 - T3$ of one user are shown in a) and the matching of segmentations of four different users $U1 - U4$ in b). The X-Axis stands for the difference of the compared segmentation results in pixel (Hausdorff-distance). The ordinate indicates the amount of pixel which fulfill this criterion.

slices. In contrast, standard medical applications (e.g. CT/NMRi) use typically less than a quarter of the here required image size for segmentation tasks.
The combination of mutual influence of image feature based path costs and user input uncertainty lead to faster and more accurate segmentations than manual segmentations. The incorporation of user behavior into a cost based delineation was perceived as intuitive by almost all users.
Furthermore, path-based methodologies and specific user interaction models were introduced and the usefulness of a combination of both of them were shown. The presented semiautomatic segmentation is capable of fast and accurate segmentations of plant biological sample data. Therefore, the results can be used as input for the automatic segmentation method proposed by BOLLENBECK et al. [BS08].
The presented paper constitutes a contribution to support the generation of plant biological 3-D atlases.

7 FUTURE DIRECTIONS

Further investigations will point toward the evaluation of the presented strategies' influence to the segmentation results. A user study must be done to validate the tendency of the performed user tests.
The results have to be further verified for different segmentation scenarios especially in the context of practical use.

8 ACKNOWLEDGEMENT

The Federal Ministry of Education and Research (BMBF) is supporting this research with the grants 0313821B and 0313821A. The authors would like to thank their colleagues from the IPK Gatersleben and their student assistants for their support and assistance.

9 REFERENCES

[BM96] W. A. Barrett and E. N. Mortensen. Fast, Accurate, and Reproducible Live-wire Boundary Extraction. In *VBC '96: Proceedings of the 4th International Conference on Visualization in Biomedical Computing*, pages 183–192, London, UK, 1996. Springer-Verlag.

[BM97] W. A. Barrett and E. N. Mortensen. Interactive Live-wire Boundary Extraction. *Medical Image Analysis*, 1:331–341, 1997.

[BS08] F. Bollenbeck and U. Seiffert. Fast Registration-based Automatic Segmentation of Serial Section Images for High-resolution 3-D Plant Seed Modeling. In *ISBI*, pages 352–355. IEEE, 2008.

[CPR03] D. J. Crisp, P. Perry, and N. J. Redding. Fast Segmentation of Large Images. In *ACSC '03: Proceedings of the 26th Australasian computer science conference*, pages 87–93, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[DBS+08] V. J. Dercksen, C. Brüß, D. Stalling, S. Gubatz, U. Seiffert, and H.-C. Hege. Towards Automatic Generation of 3D Models of Biological Objects Based on Serial Sections. In L. Linsen, H. Hagen, and B. Hamann, editors, *Visualization in Medicine and Life Sciences*, pages 3–25. Springer-Verlag Berlin Heidelberg, 2008.

[EKS92] P. J. Elliott, J. M. Knapman, and W. Schlegel. Interactive Image Segmentation for Radiation Treatment Planning. *IBM Syst. J.*, 31(4):620–634, 1992.

[FUM00] A. X. Falcão, J. K. Udupa, and F. K. Miyazawa. An Ultra-fast User-steered Image Segmentation Paradigm: Live-wire-on-the-fly. *IEEE Trans. on Medical Imaging*, 19(1):55–62, January 2000.

[FUS+98] A. X. Falcão, J. K. Udupa, S. Samarasekera, S. Sharma, B. E. Hirsch, and R. A. Lotufo. User-steered Image Segmentation Paradigms: Live-wire and Live-lane. *Graphical Models and Image Processing*, 60(4):233–260, July 1998.

[GDB+07] S. Gubatz, V. J. Dercksen, C. Brüß, W. Weschke, and U. Wobus. Analysis of Barley (Hordeum Vulgare) Grain Development Using Three-dimensional Digital Models. *The Plant Journal*, 52(4):779–790, 2007.

[HNR68] P. E. Hart, N. J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[IH00] T. Igarashi and K. Hinckley. Speed-dependent Automatic Zooming for Browsing Large Documents. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 139–148, New York, NY, USA, 2000. ACM Press.

[KWT87] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *Proceedings of the First International Conference on Computer Vision*, pages 259–268, June 1987.

[LBE04] A. Lécuyer, J.-M. Burkhardt, and L. Etienne. Feeling Bumps and Holes Without a Haptic Interface: The Perception of Pseudo-haptic Textures. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 239–246, New York, NY, USA, 2004. ACM Press.

[LM01] L. Lucchese and S. K. Mitra. Color Image Segmentation: A State-of-the-art Survey. In *Proceedings of the Indian National Science Academy*, pages 207–221, 2001.

[MB95] E. N. Mortensen and W. A. Barrett. Intelligent Scissors for Image Composition. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 191–198, New York, NY, USA, 1995. ACM.

[MMBU92] E. Mortensen, B. Morse, W. Barrett, and J. Udupa. Adaptive Boundary Detection Using "Live-wire" Two-Dimensional Dynamic Programming. *IEEE Proceedings Computers in Cardiology*, pages 635–638, October 1992.

[OS01] S. D. Olabarriaga and A. W. M. Smeulders. Interaction in the Segmentation of Medical Images: A Survey. *Medical Image Analysis*, 5(2):127 – 142, 2001.

[RVC07] Y. Roodt, W. Visser, and W. A. Clarke. Image Processing on the GPU: Implementing the Canny Edge Detection Algorithm. In *Proceedings of the Eighteenth International Symposium of the Pattern Recognition Association of South Africa (PRASA 2007)*, pages 45–50, 2007.

[SBH+08] W. Schoor, F. Bollenbeck, M. Hofmann, R. Mecke, U. Seiffert, and B. Preim. Automatic Zoom and Pseudo Haptics to Support Semiautomatic Segmentation Tasks. In V. Skala, editor, *16th WSCG 2008*, WSCG'2008 Full Papers Proceedings, pages 81–88. WSCG, University of West Bohemia, February 2008. Full Paper.

[SPP00] A. Schenk, G. P. M. Prause, and H.-O. Peitgen. Efficient Semiautomatic Segmentation of 3D Objects in Medical Images. In *MICCAI '00: Proceedings of the Third International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 186–195, London, UK, 2000. Springer-Verlag.

[SPP01] A. Schenk, G. Prause, and H.-O. Peitgen. Local Cost Computation for Efficient Segmentation of 3D Objects with Live Wire. In M. Sonka and K. M. Hanson, editors, *Medical Imaging 2001: Image Processing*, number 4322 in Proceedings of SPIE, pages 1357–1364, 2001.

[SUG+06] A. Souza, J. K. Udupa, G. Grevera, Y. Sun, D. Odhner, N. Suri, and M. D. Schnall. Iterative Live Wire and Live Snake: New User-steered 3D Image Segmentation Paradigms. In *SPIE Medical Imaging*, volume 6144, pages 1159–1165, 2006.

[Tön05] K. D. Tönnies. *Grundlagen der Bildverarbeitung*. Pearson Studium, München, May 2005. ISBN-10: 3827371554.

Fiberbundle-Based Visualization of a Stirred-Tank Flow

Werner Benger
Feng Jijao
Center for Computation & Technology
at Louisiana State University
239 Johnston Hall
USA, LA 70803, Baton Rouge
werner@cct.lsu.edu
fjiao@cct.lsu.edu

Marcel Ritter
Center for Computation & Technology at LSU
and
Department of Computer Science
University of Innsbruck
Technikerstrasse 21a
Austria, A-6020 Innsbruck
marcel@cct.lsu.edu

Sumanta Acharya
Somnath Roy
Louisiana State University
Department of Mechanical Engineering
USA, LA 70803, Baton Rouge
acharya@me.lsu.edu
sroy13@lsu.edu

ABSTRACT

We describe a novel approach to treat data from a complex numerical simulation in a unified environment using a generic data model for scientific visualization. The model is constructed out of building blocks in a hierarchical scheme of seven levels, out of which only three are exposed to the end-user. This generic scheme allows for a wide variety of input formats, and results in powerful capabilities to connect data. We review the theory of this data model, implementation aspects in our visualization environment, and its application to computational fluid dynamic simulation of flow in an impeller-stirred tank. The computational data are given as a velocity vector field and a scalar pressure field on a mesh consisting of 2088 blocks in curvilinear coordinates.

Keywords

Computational fluid dynamics, data model, stream lines, scientific visualization.

1 INTRODUCTION

Computational fluid dynamics (CFD) has advanced significantly in the last few years and can now provide high fidelity temporally and spatially resolved numerical data. This data is based on meshes that range from a few million cells to tens of million cells, and for several hundred thousand time steps, with data files that are of the order of terabytes. A key challenge therefore is the ability to easily and cost-effectively mine this data for key features of the flow field and to display these spatially evolving features in the space-time domain of interest. In this work, we present an integrated interdisciplinary effort that takes utilizes of a generic approach to handle scientific data sets leading to new visualization capabilities in a natural way.

The CFD dataset is obtained from a large eddy simulation (LES) of flow inside a stirred tank reactor (STR), such as depicted in Figure 1. The STRs are widely used as mixing devices in chemical industry. The STR that we investigated here is a cylindrical tank with a hemispherical bottom, vertical baffles mounted along the cylindrical walls, and rotating impellers consisting rectangular blades with 45° pitch angle mounted on a shaft passing through its center, see Figure 2.

Figure 1: Stir Tank
(Courtesy Dow Chemicals)

As the impeller rotates, its blades pump the fluid axially downward towards the bottom of the tank. The fluid-jet then hits the hemispherical bottom wall and sets in a circulating motion of fluid within the tank promoting mixing between the top and the bottom of the tank.

The calculations are done on a multi-block curvilinear mesh as shown in Figure 3. Calculations are done on nearly three million cells using a multi-block body fitted finite volume flow solver. Since the tank contains a set of impeller blades that are rotating and also contains stationary

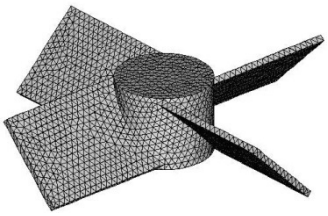


Figure 2: Impeller geometry

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

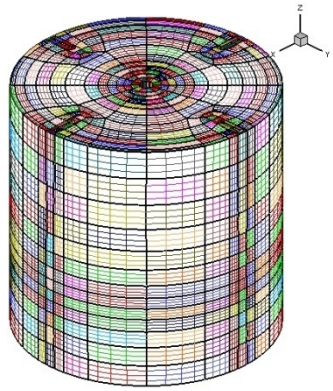


Figure 3: Boundaries of the 2088 curvilinear blocks of the computational mesh

features such as baffles on the outer walls, special algorithms (Immersed Boundary Methods) are used to accommodate moving interfaces. The method intrinsically uses a fixed curvilinear mesh fixed to the stationary features of the geometry and tracks or flags the location of the nodes adjacent to the moving surfaces or boundaries. At these flagged nodes, forcing terms/interpolations are performed to satisfy the boundary conditions at the moving surfaces. This approach has been extensively tested in an earlier communication [TRHA07] where computational solution for a similar stirred tank flow has been validated with experimental observations.

The visualizations presented here show the axial pumping as well as the circulation zones suggesting the mechanisms of fluid mixing inside the tank. As the impeller blade moves, the fluid jet stream coming out of the top and side wall of the impeller blade interact with each other to form a vortical motion following which fluid elements move in azimuthal direction [RA07]. These vortex structures are identified from the existences of pressure minima.

While traditional commercial visualization packages such as Tecplot (www.tecplot.com) and Ensight (www.ensight.com) are commonly used for visualization, they have inherent limitations. It is therefore important to be able to develop sophisticated visualization capabilities that allow a variety of features and controls not commonly available with the commercial software. The VISH visualization environment [BRH07] employed here has been designed to allow a step in this direction.

2 DATA MODEL CONCEPT

"*A common denominator for scientific data already exists, we just have to use them.*" was the paradigm under which Butler & Pendley [BP89] proposed the mathematics of fiber bundle as a foundation for a data model that is able to describe all cases needed for scientific visualization in a common way, since the mathematics behind all the different implementation is an universal language of science already. Yet a common data model has hardly been used, with the IBM Data Explore one of the few exceptions. The idea of using a fiber bundle as inspiration for a common data model has been revived by Benger [Ben04] to handle the complexity of data stemming from general relativity. These concepts have been built upon and further refined by Heinzl [HEI07] to develop generic software components. In these approaches, concepts have been based on the mathematics of topology and differential geometry to describe data sets.

2.1 Mathematics of Fiber Bundles

A fiber bundle is in the mathematical sense is set of a total space E and a base space B with a projection map f such that the union of fibers of a neighborhood U of each point of B is homeomorphic to $U \times F$ with F the so called fiber space and the projection of $U \times F$ is U again. It is also said that the space F "fibers" over the base space B. If the total space E can be written globally as $E=B \times F$, then E is called a "trivial bundle". The paradigm is that numerical data sets that are usually needed for scientific visualization can be formulated as trivial bundles. In practice it means that we may distinguish data sets by their properties in the base space and the fiber space. At each point of the – discretized – base space we have some data in the fiber space attached.

The structure of the *base space* is described as a CW-complex, which categorizes the topological structure of an n-dimensional base space by a sequence of k-dimensional *skeletons*, with the dimensionality of the skeletons ranging from zero to n. These skeletons carry certain properties of the data set: the 0-skeleton describing vertices, the 1-skeleton the edges, 2-skeleton the faces, etc., of a triangulation of some mesh. For structured grids the topological properties are given implicitly.

The structure of the *fiber space* is (usually) not discrete and given by the properties of the geometrical object residing there, such as a scalar, vector, co-vector, tensor. Same as the base space, the fiber space has a specific dimensionality, though the dimensionality of the base space and fiber space is independent. If the fiber space has vector space properties, then the fiber bundle is called a vector bundle. The most simple *vector bundle* is the *tangential bundle* of a manifold, which consists of the manifold as base space and the space of tangential vectors at each point. With n the dimensionality of the manifold, the dimension of the tangential bundle is $2n$.

Basically a fiber bundle is a set of points with neighborhood information attached to each of them. An n-dimensional array is a very simple case of a fiber bundle (with neighborhood information given implicitly).

2.2 Benefits of Fiber Bundles

The concept of a fiber bundle leads to a natural distinction of data describing the base space and data describing the fiber space. This distinction is not common use in computer graphics, where topological properties (base space) are frequently intermixed with geometrical properties (coordinate representations). Operations in the fiber space can however be formulated independently from the base space, which leads to a more reusable design of software components. Coordinate information, formally part of the base space, can as well be considered as fiber, leading to further generalization. The data sets describing a fiber are ideally stored as contiguous arrays in memory or disk, which allows for optimized array and vector operations. Such a storage layout turns out to be particularly useful for communicating data with the GPU using vertex buffer objects: fibers are basically vertex arrays in the notation of computer graphics.

2.3 The 7-Level Hierarchy

In the data model implementation of [Ben04] data are formulated in a graph of maximally seven levels, each level representing a certain property of the entire data set. These levels, constituting a "Bundle", are:

1. Slice
2. Grid
3. Skeleton
4. Representation
5. Field
6. (Fragment)
7. (Compound Elements)

Actual data arrays are stored only below the "Field" level. An actual data set is described by which data sets exist in which level. The actual meaning of each level is described elsewhere [Ben04], [BRH07], [Ben08]. Only two of these hierarchy levels are exposed to the end-user, these are the "*Grid*" and "*Field*" levels.

2.4 Bundles, Grids and Fields

An entire dataset, including all time steps or any information given on a parameter space in general, is denoted as a *Bundle*, following the mathematical term of a fiber bundle. The objective is to formulate all data that is used for scientific visualization within this *Bundle*. A *Grid* is subset of data within the Bundle that refers to a specific geometrical entity, for instance a mesh carrying data such as a triangular surface, a data cube, a set of data blocks from a parallel computation, etc. A *Field* is the collection of data sets given as numbers on a specific topological component of a *Grid*, for instance floating point values describing pressure or temperature.

The names of *Grids* and *Fields* are arbitrary and specified by the user. All other levels of the data model have pre-defined meanings and values which are used internally to describe the properties of the *Bundle* as construction blocks. The usage of these construction blocks constitutes a certain language to describe a wide range of data sets. For instance, a *Slice* is identified by a single floating point number representing time (generalization to arbitrary-dimensional parameter spaces is possible); a *Skeleton* is identified by its dimensionality, index depth (relationship to the vertices of a Grid) and refinement level; a *Representation* is identified via some reference object, which may be some coordinate system or another *Skeleton*. The meaning of such identifiers is only used internally, but transparent to the user. The lowest levels of *Fragments* and *Compound* describe the internal memory layout of a *Field* data set and are optional. They are described in detail in [Ben08].

2.5 Formulating Data

The existence of data sets in the hierarchy of a *Bundle* defines a data set. We will give some examples here on the data sets that have been involved in our work with the stir tank computational fluid dynamics data.

2.5.1 Isosurface

An isosurface is the explicit polygonal representation of the points where a possibly time-dependent scalar field given on a volumetric manifold has constant value. It consists of the following properties:

1. A sequence of *Grids*, one for each time step
2. Coordinates for each vertex, which is a field on the Skeleton of index depth 0 ("Vertices") on each grid, as represented in Cartesian coordinates.
3. Connectivity information for each triangle, which is a field on the Skeleton of dimension 2 and index depth 1 (faces) on each Grid, as represented in the Vertices.
4. Normal vectors define a vector field (precisely: a bi-vector or co-vector field) given on the Vertices Skeleton of the Grid.
5. Optional data fields on the vertices, such as another 3D field mapped on the surface.

An Isosurface is a sequence of Grid objects with two Skeletons defined on it, with the Skeleton for the Vertices carrying two or more fields.

2.5.2 Line Set

A set of lines, such as the result of the computation of stream lines of a vector field, is given by

1. A sequence of Grids, one for each time step, for the set of lines valid at each time step
2. Coordinates for each point along the lines, a field on the Skeleton of index depth 0
3. Connectivity information for the points building up a line, which is a field on the Skeleton of dimension 1 and index depth 1 (edges).
4. Tangential vectors define a vector field given on the Vertices Skeleton of the Grid
5. Optional data fields on the vertices may exist, as retrieved from a 3D field mapped on the surface.

A set of lines is very similar to an isosurface in this data model, though a different Skeleton on the Grid objects is employed.

2.5.3 Multiblock Data

The numerical data as provided by the computational fluid simulation are given as a collection of 2088 three-dimensional arrays describing coordinate location, pressure and fluid velocity for each grid point. These fields are all fibers on the vertices in the fiber bundle data model, where we also treat the coordinates as a field over the vertices. As the topological structure is regular the edges and faces are given implicitly. The decomposition of the data into blocks is represented as the internal memory layout structure of the fields, thus as field fragments visible in the 6th level of the data hierarchy. The topological structure of the blocks is thereby transparent to the user, but algorithms operate on collections of arrays instead of contiguous arrays, which is relatively straightforward extension to existing algorithms such as the computation of isosurfaces. To store information that is specific to each block, we consider each block as a collection of volume cells. Volume cells are topologically 3-Skeletons in a triangulation, and of index depth 1 in the fiber bundle data model. Collections of such elements are thus of index depth 2. Fibers on this Skeleton are thus a natural place to store e.g. the geometrical bounding box information for a block, or the min/max data range of a scalar field (which speeds up repeated isosurface computations) The layout of a multiblock data set thus consists of:

1. A sequence of Grid objects, one for each time step
2. A coordinate field on the Vertices Skeleton, which is shared among all time steps if the geometry remains constant over time
3. A scalar field on the Vertices Skeleton
4. A vector field on the Vertices Skeleton
5. A Skeleton of dimension 3 and index depth 2 describing a collection of volume elements, which are the respective blocks.
6. Optional scalar fields on the (3,2) Skeleton with min/max information of a scalar field per block, or coordinate fields for bounding box.

2.6 Data Operations

Given the certain components constituting the data model, we may formulate abstract operations among such components. These operate on abstract high-level objects without requirement to know the internal structure of the objects, though their concrete implementation will have to deal with them.

2.6.1 Isosurface Computation

The computation of an isosurface is an operation that takes a Field as input and yields a Grid object, and will be called for each instance of a time sequence. The operation is parameterized by some isolevel value. Certain conditions must be fulfilled by the input field for this operation to succeed, such as being a scalar-valued field residing on a regular grid. However, more advanced implementations rather than the standard marching cubes may well be formulated through the same interface, such as direct computation of magnitude isolevels of vector fields, or isosurface computation on tetrahedral grid. The high-level operation

```
Grid = ComputeIsosurface(Field, float);
```

remains the same, and no changes in the user interface are required. For instance, more advanced operations could be invoked via some runtime plugin mechanism, transparent to the user, as it is supported by VISH.

2.6.2 Grid Evaluation

Given Field data on one Grid instance, like a 3D volume, they may be evaluated on another Grid, like a surface or a line set. Such is formulated as an EvalGrid operation which requires specification of a destination grid and a source field (implicitly given on another Grid object):

```
Field = EvalGrid(Grid Dest, Field Source);
```

Certain constraints may apply to the input Grid, since the evaluation of a field is not possible on an arbitrary Grid. For instance, data given on a surface cannot be uniquely extrapolated into an entire 3D volume. However, a wide range of operations can be specified through this common API.

3 STREAMLINE STRATEGIES

Visualizing vector fields is a common need in CFD for investigating velocity fields. [LaH05] describes several tools for investigating CFD data. [PWS06] shows the combination of different geometry based and texture based techniques in a CFD application. More applications of texture based vector field visualization can be found in [LEG].

With more complicated and large data sets it is increasingly important to have a variety of tools for feature extraction and data exploration at hand. Thus, developing a flexible framework seems the right way to meet the increasing requirements effectively. We use streamlines in our approach because it is a well known standard technique and they can be described well within the context of fiber bundles.

The challenge here was to deal with the multi-block curvilinear data structure (as shown in Figure 3) and to verify the applicability of the *Fiber Bundle* data model. Using this model, our streamline visualization implementation modules separate into four groups: *Vector Field Data*, *Seed Point Data*, *Streamline Computation* and *Line Rendering*. The software modules are connected in a *directed graph* communicating data using *Grids* and *Fields*. This allows to exchange modules by other modules (high reusability) and to combine modules in different ways (high flexibility).The *Streamline Computation Module* takes a vector *Field* and an arbitrary *Grid* defining the seed points as inputs and outputs a *Line Set* as *Grid*.

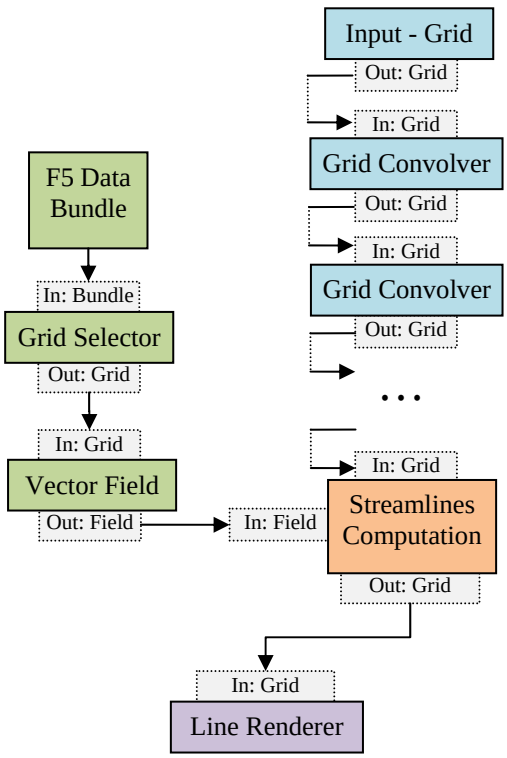


Figure 4: Streamline modules and dataflow

The computation of the streamline involves finding the location of a given *world position* in the dataset by identifying its block, cell and local curvilinear cell coordinates. These local coordinates are then used for linear interpolation of the vector field. The interpolated vector is used to advance to the next streamline point of the streamline, another world position.

Firstly, a kD-Tree is employed to find blocks, which might contain the *world position*, secondly a look up data structure called *UniGridMapper* that maps a uniform grid cell to curvilinear cells that might contain the world position is used and finally a Taylor approximation and Newton iteration, as described in [STA98], retrieves the local curvilinear cell coordinates.

Besides storing the calculated stream lines as *Line Sets* in the output *Grid* additional *Fields* are stored that carry the information necessary for interpolation. Block IDs and local cell coordinates are stored. This information can then be used by other independent modules to evaluate other data *Fields* on the streamline *Grid*, e.g. to evaluate a pressure *Field*. The final *Line Rendering* module employs *illuminated stream lines*, similar to those described in [SZH97]

3.1 Defining Seed Point for Streamlines

To visualize the characteristics of a certain vector field by streamlines it is important to find good starting points (seed points) for the streamline integration.

3.1.1 Grid Convolver

An operation called *Grid Convolver* allows the user to create sophisticated seed point geometries by 'convoluting' vertices of an input *Grid* with a set of parameters into an output *Grid*, similar to the mathematical convolution operation. Possible *Grid* convolution operations are *Point*, *Line*, *Rectangle*, *Circle*, *Ellipsoid* and *Uniform Grid*.

Figure 5 demonstrates a setup for creating seed points involving three *Grid Convolvers*. The first *Grid Convolver(a)* gets one point as input *Grid*. It 'convolutes' this input on a vertical *Line* with a subdivision of three points. This is then outputted into the second *Grid Convolver(b)* which 'convolutes' the three-point-*Line* on its horizontal two-point-*Line*. This results in the output *Grid* seen in (c). A final *Grid Convolver(d)* now 'convolutes' on its *Circle* geometry resulting in the final seed points in the output *Grid* of (d), shown in (e).

Since the module can be connected to any other modules that output *Grid* objects it is possible to create many different seed geometries. The module thus is highly reusable and flexible. Figure 4 illustrates a typical dataflow involving *Field* and

 ISBN 978-80-86943-94-7

Grid objects. Figure 6, 7, and 8 demonstrate how this module was used to investigate the stir tank velocity field with different settings of position and rotation and different number of connected *Grid Convolvers.*

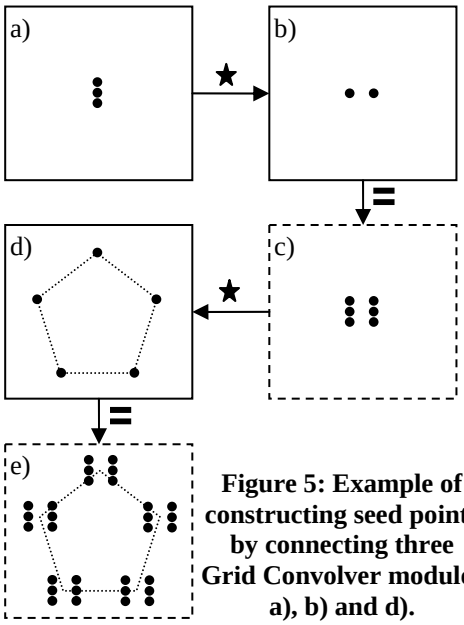


Figure 5: Example of constructing seed points by connecting three Grid Convolver modules a), b) and d).

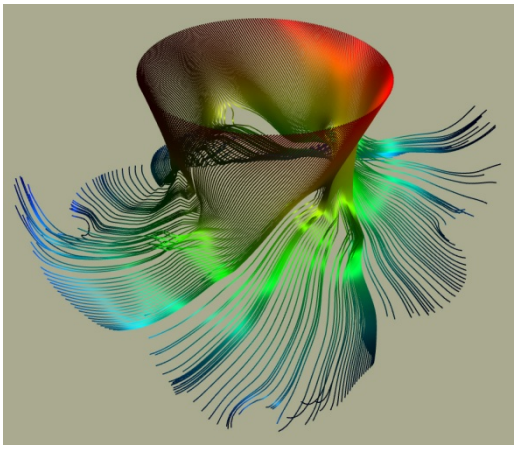


Figure 6: Streamlines with seed points on a circle.

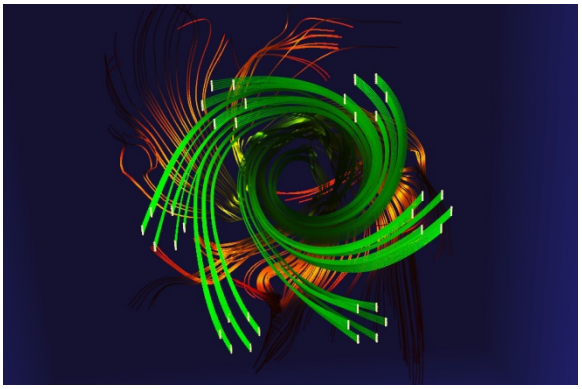


Figure 7: Streamlines emitted from seed points on a circle of circles of lines by using three Grid Convolvers.

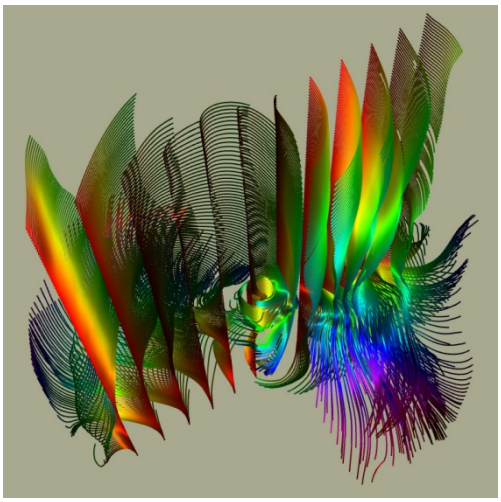


Figure 8: Streamlines emitted on an array of lines.

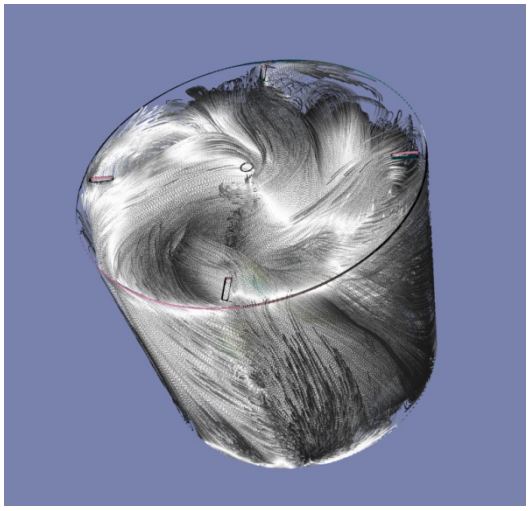


Figure 9: Space-filling streamlines in the STR

Besides interactively specifying seed points it is also possible to utilize other seeding methods such as those found in AMIRA [SWH05], where a threshold on a scalar field can specify seed points for a vector field, or similar to Weinkauf's method of computing the curvature measures of a vector field to find critical regions as indicators where streamlines were most interesting [WTH02]. Even a full coverage of the entire volume may provide worthwhile information (Figure 9).

3.1.2 Seed Points by Iso Surfaces

The usage of a *Grid* input at the *Streamline Computer* allows usage of arbitrary compatible *Grid* objects for defining seed points. For example, the *Grid* of an iso surface (the vertices of the triangular surface) computed on the pressure scalar field can be used as input, as shown in Figure 9 and Figure 10. With such streamline seeding the vector field close to the surface can be explored, similar to a texture based technique applied to the surface such as [LSH04]. This is a unique feature of VISH that can potentially be exploited to better understand the flow physics.

4 PERFORMANCE RESULTS

Computation of 100 streamlines in the given multiblock dataset of 2088 curvilinear blocks required about 7 seconds using an Euler integration scheme for 100 steps on a Intel Xeon CPU, 2.5GHz. Tecplot required 35 seconds using a comparable setup. We could not compare with Amira, since this data type is not supported there. The computation time of the isosurface crucially depends on the chosen level, and is below 1/30th second for most values, but may require up to 2 seconds in particular cases. Computing streamlines from the isosurface vertex requires about 5-10 seconds for the setup as shown in Figure 11, but will linearly depend on the chosen length. Future improvements will utilize higher order integration schemes and parallelization, for which we expect to be able to reduce computation times under one second. The frame rates for rendering itself once the streamlines are computed are under 1/30th of a second in all cases except Figure 9, where we got about 10 frames per second on an NVidia Quadro FX 5600 graphics board.

5 DISCUSSION

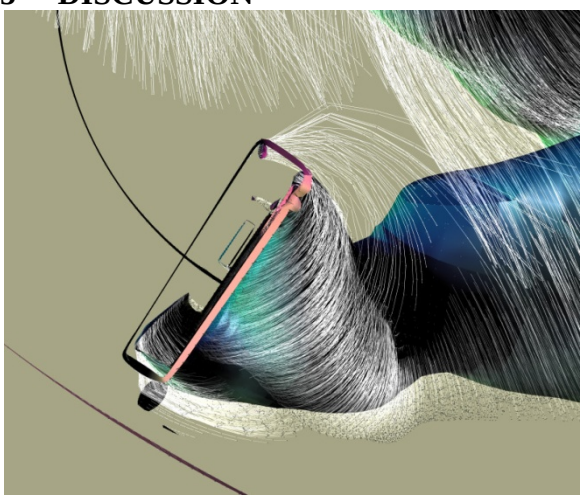


Figure 10: Detail of the emission of streamlines from the pressure isosurface.

Figure 10 shows velocity vectors adjacent to the impeller blades. It can be clearly seen that as the impeller rotates in the clockwise direction, the pitched blade surfaces force the fluid in its surface-normal direction imposing both radially outward and downward velocities on the flow along with an azimuthal velocity component due to its rotation. The pressure isosurface depicts the boundary layer formed over the impeller blade surface and its evolution downstream. Circular depressions (marked as P in Figure 11) can be observed in the pressure isosurface. These depressions point to local pressure minima suggesting formations of vortical structures in the direction opposite to the impeller rotation. These vortices (commonly termed as trailing-edge vortices) convect fluid in the azimuthal direction and play a key role in mixing within the tank.

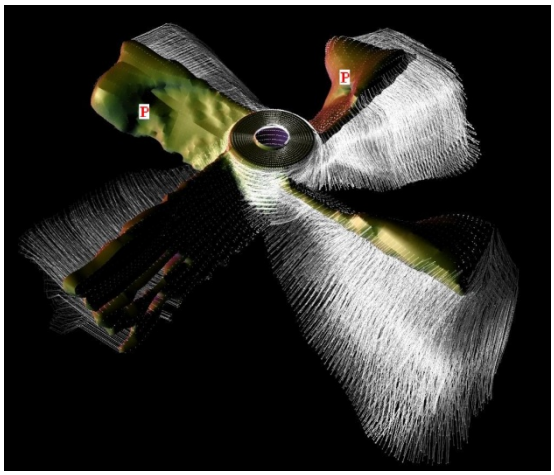


Figure 11: Velocity micro-streamlines and pressure isosurfaces over the impeller blade

Figure 12 shows the streamlines coming out of an impeller blade. The downward pumping from the impeller is clearly observable. Also, the streamlines are observed to bend near the hemispherical bottom of the tank and establish a circulating motion. The colors along the streamlines indicate the residence time of the fluid particles. The suction of the upper fluid is identified by the green color, the yellow colored impeller jet stream has both radial and axial component as the streamlines show almost a 45° bend. Later (orange color) these lines hit the bottom of the tank and bend upwards (red coloration) and the fluid is again convected to the upper part of the tank. This circulation is how the fluid mixing operation takes place in the stirred tank.

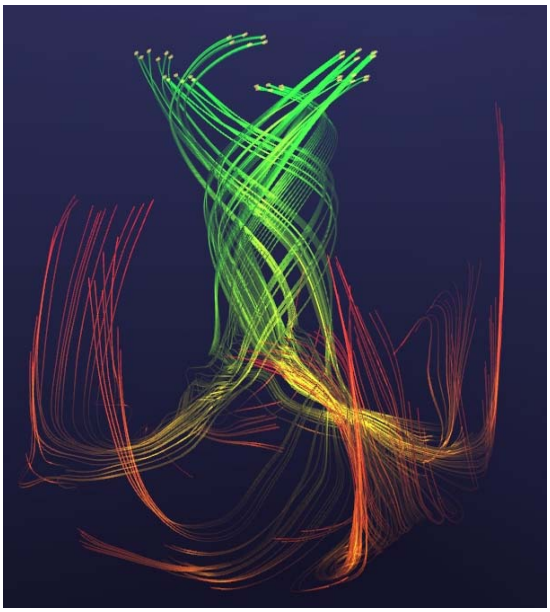


Figure 12: Streamlines over the tank volume

6 SUMMARY

We have described and applied the concept of "Grid objects" as elementary tools within a highly modular visualization environment to provide powerful seeding mechanisms for streamline computation to visualize flow in a stirred tank. Operations such as "Grid convolution" and seeding by pressure isosurfaces are natural consequences of utilizing the described data model. It is argued here that this approach offers specific capabilities that several other visualization platforms are unable to provide.

7 ACKNOWLEDGMENTS

This research employed resources of the Center for Computation & Technology at LSU, 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). We thank Georg Ritter for his industrious work on the VISH infrastructure.

8 REFERENCES

[Ben04] Benger, W. **Visualization of General Relativistic Tensor Fields via a Fiber** , PhD - thesis FU Berlin, 2004

[BRH07] W. Benger, G. Ritter and R. Heinzl, **The Concepts of VISH**, Proc. 4th High End Visualization Workshop Obergurgl, p. 26-39, 2007, Lehmann's Media

[Ben08] Benger, W. **Colliding galaxies, rotating neutron stars and merging black holes - visualizing high dimensional data sets on arbitrary meshes**, *New J. Phys. 10 (2008) 125004*, http://stacks.iop.org/1367-2630/10/125004.

[BP89] O. Butler, D. M. & Pendley, M. H. (1989). **A visualization model based on the mathematics of fiber bundles**. *Comp. in Physics*, 3(5), 45-51.

[HEI07] R. Heinzl: "*Concepts for Scientific Computing*"; PhD Thesis; Institut für Mikroelektronik, TU Wien, 2007;

[LaH05] Robert S. Laramee & Helwig Hauser, **Interactive 3D Flow Visualization Based on Textures and Geometric Primitives**, in NAFEMS World Congress Conference Proceedings, The International Association for the Engineering Analysis Community, May 17-20, 2005, St. Juliens Bay, Malta

[LEG] R. S. Laramee, G. Erlebacher, C. Garth, T. Schafhitzel, H. Theisel, X. Tricoche, T. Weinkauf, D. Weiskopf, **Applications of Texture-Based Flow Visualization**, Engineering Applications of Computational Fluid Mechanics, in publication

[LSH04] R. S. Laramee, J. Schneider & H. Hauser: **Texture-Based Flow Visualization on Isosurfaces from Computational Fluid Dynamics**, Proceedings of the 6th Joint IEEE TCVG - EUROGRAPHICS Symposium on Visualization (VisSym 2004), May 19-21, 2004, Konstanz, Germany

[MPS05] O. Mallo, R. Peikert, C. Sigg, F. Sadlo, **Illuminated Lines Revisited**, *Proceedings of IEEE Vis 2005, pp. 19-26 (Minneapolis, MN, USA, October 23-28, 2005)*

[PWS06] C. Petz, T. Weinkauf, H. Streckwall, F. Salvatore, B.R. Noack, H.-C. Hege, **Vortex Structures at a Rotating Ship Propeller**, Presented at the 24th Annual Gallery of Fluid Motion exhibit, held at the 59th Annual Meeting of the American Physical Society, Division of Fluid Dynamics, Tampa Bay, November 2006

[RA07] S. Roy, and S. Acharya, "Study on Flow and Turbulence Inside a Stirred Tank and Investigation on the Effects of Macroinstability on Trailing Vortex Structures", *ASME International Mechanical Engineering Congress and Exposition*, Seattle, November 2007

[STA98] D. Stalling. "Fast texture-based algorithms for vector field visualization." Dissertation, FU Berlin, Preprint SC-98-58, Konrad-Zuse-Zentrum Berlin (ZIB), December 1998.

[SWH05] D. Stalling and M. Westerhoff and H.-C. Hege, "Amira - an object oriented system for visual data analysis", Visualization Handbook, Christopher R. Johnson and Charles D. Hansen, Academic Press, 2005

[SZH97] D.Stalling, M.Zockler, and H.-C. Hege. **Fast display of illuminated field lines**. In *IEEE Transactions on Visualization and Computer Graphics*, vol.3, pages 118-128, 1997.

[SZH97] D.Stalling, M.Zockler, and H.-C. Hege. **Fast display of illuminated field lines**. In *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, pages 118-128, 1997.

[TRHA07] M. Tyagi, S. Roy, S. Acharya, and A-D Harvey III, "Simulation of laminar and turbulent impeller stirred tanks using immersed boundary method and large eddy simulation technique in multi-block curvilinear geometries", *Chemical Engineering Sciences*, volume 63, issue 5, pages 1351-1363, 2007

[WTH02] T. Weinkauf, H. Theisel, **Curvature Measures of 3D Vector Fields and their Applications**, Journal of WSCG 10(2), WSCG 2002, Plzen, Czech Republic, February 4 - 8, 2002

3D Modeling of Free-Form Object (Interpolation, Visualization and Volume Estimation)

Aleksandrs Sisojevs
Riga Technical University
Meza str. 1/3-304
Latvia (LV), Riga
alexiv@inbox.lv

Katrina Krechetova
Riga Technical University
Meza str. 1/3-304
Latvia (LV), Riga
katrina.krecetova@rtu.lv

Aleksandrs Glazs
Riga Technical University
Meza str. 1/3-304
Latvia (LV), Riga
glaz@egle.cs.rtu.lv

ABSTRACT
This work describes an efficient method of 3D object modeling. The method is applied for solving biomedical engineering tasks. The method is based on interpolation model creation. The model is built using Coons surface basis. For further implementation of the model the Coons surface basis is transformed to Bezier surface model. This work also describes a method of volume estimation of the reconstructed 3D model. Example of practical implementation is solving various biomedical tasks.

Keywords
Volume Estimation, Interpolation, Visualization, 3D Model.

1. INTRODUCTION
The problem of free – form objects 3D visualization and their further analysis is a very interesting task of computer graphics. Also, the solving of this task is very actual in practical implementation for biomedical engineering. This work describes a new method of free-form objects reconstruction, with a medical object used as an example. The object is visualized and its volume is estimated (a very actual task for physicians). The known methods of object volume estimation do not take into consideration the 3D structure of the object (this leads to miscalculations in volume estimation), meaning that the method of objects volume estimation using reconstructed 3D model is a more precise approach.

The proposed methods and algorithms can be used in various spheres (scientific, medical). To show practical use of the proposed methods, medical images are used in this work.

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

2. PROPOSED METHODS
Segmentation
The goal of image segmentation is to categorize the pixels (or voxels) of an image into a certain number of regions, which are homogenous with respect to some characteristics (for example: intensity, color, texture, etc.). In order to estimate the volume of the object and to visualize it, image segmentation should be done first.

Because practical implementation of our method is biomedical engineering tasks, a method of segmentation was developed, that was specifically designed for medical image segmentation, taking into account the special features of medical images.

Standard medical imaging software provides manual image segmentation, which, although is most accurate if done by expert physician is also very time-consuming. On the other hand, automatic image segmentation is often doubted by the physician, because a computer could never replace a doctor in diagnostics. Therefore, a simple and quick semi-automatic segmentation algorithm is proposed [1], with the possibility to also manually adjust the region of interest.

The proposed method combines the advantages of grey-level thresholding, region based and edge based segmentation. Segmentation algorithms are combined with developed image enhancement algorithm that is based on image histogram

enhancement and existing segmentation result post-processing algorithm.

The possibility of manual adjustment of ROI (Region of Interest) borders minimizes the risk of imprecise segmentation that automatic segmentation could provide.

The proposed method was tested on medical images of a brain acquired with use of computer tomography. Figure 1 shows the results of this method.

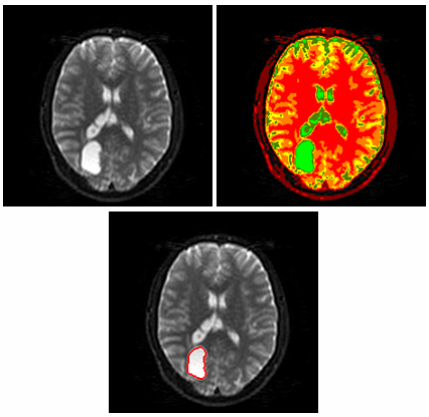


Fig. 1 Segmentation results. Top left: initial image. Top right: segmentation map. Bottom: extracted pathology zone (no manual adjustment)

The algorithm automatically removes the background from the medical image and segments the data, providing a segmentation map of the image. From this point, the physician can select any region of interest and then manually adjust borders of the region until the result is satisfactory.

Mathematical model for rendering 3D objects

The input data for developing the mathematical model of a 3D object is an array of points, obtained from segmented regions. The task of image visualization in this case can be reduced to the task of creating an array of patches of the modeled surface. We use patches that are described with Coons surface to build the resulting surface. Coons surface in matrix form can be expressed as follows:

$$Q_{Coons}(u,v) = [U]\cdot[N_C]\cdot[P_{Coons}]\cdot[N_C]^T\cdot[V], \qquad (1)$$

where $[U]$, $[V]$ – matrix of parameters,

$[N_C]$ – matrix of basis function coefficients,

$[P_{Coons}]$ – array of control points.

In this work, it is proposed to use bi-cubic Coons surfaces for modeling the resulting surface. In this case the array of points for one separately taken patch is calculated as follows:

$$[P_{Coons}] = \begin{bmatrix} P(i,j) & P(i,j+1) & P_v(i,j) & P_v(i,j+1) \\ P(i+1,j) & P(i+1,j+1) & P_v(i+1,j) & P_v(i+1,j+1) \\ P_u(i,j) & P_u(i,j+1) & P_{uv}(i,j) & P_{uv}(i,j+1) \\ P_u(i+1,j) & P_u(i+1,j+1) & P_{uv}(i+1,j) & P_{uv}(i+1,j+1) \end{bmatrix} \qquad (2)$$

where $P(i,j)$ – control points in the corners of the patch (the points from the initial array of data),

$P_u(i,j)$, $P_v(i,j)$ – tangent vectors in parametric directions u and v,

$P_{uv}(i,j)$ – twist vectors.

Tangent vectors are calculated as follows:

$$P_u(i,j) = \frac{P(i,j+1) - P(i,j-1)}{2}$$
$$P_v(i,j) = \frac{P(i+1,j) - P(i-1,j)}{2}, \qquad (3)$$

Twist vector is calculated by tangent vector multiplication:

$$P_{uv}(i,j) = P_u(i,j) \times P_v(i,j), \qquad (4)$$

The twist vector can be used as a zero-vector in further patch calculation. The direction of twist vector is used for polygonal visualization of the model.

Transformation of the mathematical model for use with standard computer graphics library

To visualize the 3D object by standard means of computer graphics it is necessary to transform Coons surfaces to Bezier surfaces. Bezier surface in matrix form can be expressed as follows:

$$Q_{Bezier}(u,v) = [U]\cdot[N_B]\cdot[P_{Bezier}]\cdot[N_B]^T\cdot[V], \qquad (5)$$

where $[U]$, $[V]$ – matrix of parameters,

$[N_B]$ – matrix of basis function coefficients,

$[P_{Bezier}]$ – array of control points.

To transform the Coons surface to Bezier surface we use the relation between $Q_{Bezier}(u,v)$ and $Q_{Coons}(u,v)$. As described by Rogers et. al. [2], the relation between Coons surface and Bezier surface can be expressed by equating the two surface equations (1) and (5). In matrix form this can be described as follows:

$$[U]\cdot[N_B]\cdot[P_{Bezier}]\cdot[N_B]^T\cdot[V] = [U]\cdot[N_C]\cdot[P_{Coons}]\cdot[N_C]^T\cdot[V], \qquad (6)$$

where $[U]$, $[V]$ – matrix of parameters,

$[N_B]$, $[N_C]$ – matrix of coefficients,

$[P_{Bezier}]$, $[P_{Coons}]$ – matrix of control points.

The array of control points for equivalent Bezier surface is calculated using the following formula:

$$[P_{Bezier}] = [N_B]^{-1} \cdot [N_C] \cdot [P_{Coons}] \cdot [N_C]^T \left[[N_B]^T\right]^{-1}, \qquad (7)$$

In case of bi-cubic surfaces this formula assumes the following form:

$$[P_{Bezier}] = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1/3 & 1 \\ 0 & 1/3 & 2/3 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \qquad (8)$$

$$\cdot \begin{bmatrix} P(i,j) & P(i,j+1) & P_v(i,j) & P_v(i,j+1) \\ P(i+1,j) & P(i+1,j+1) & P_v(i+1,j) & P_v(i+1,j+1) \\ P_u(i,j) & P_u(i,j+1) & P_{uv}(i,j) & P_{uv}(i,j+1) \\ P_u(i+1,j) & P_u(i+1,j+1) & P_{uv}(i+1,j) & P_{uv}(i+1,j+1) \end{bmatrix} \cdot$$

$$\cdot \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1/3 & 1 \\ 0 & 1/3 & 2/3 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

In the end, the mathematical model of the 3D object is an array of Bezier patches. This allows using standard methods of computer graphic visualization. In this work, we use standard computer graphics library – OpenGL [3] and rendering by ray tracing. The method of visualization by ray tracing is described in [4]. Figure 2 shows the results of visualization using standard graphic library OpenGL and ray tracing.

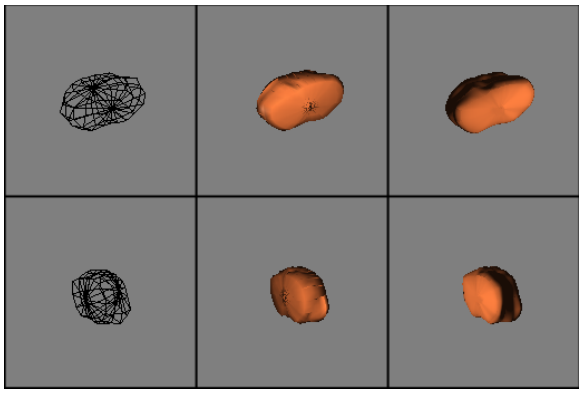


Fig. 2 Left column: mesh of the model. Middle column: models obtained using standard OpenGL library. Right column: models obtained by ray tracing.

Volume Estimation

Volume estimation algorithms were also tested on medical images (pathology zone extraction and its volume estimation). There are various approaches to volume estimation in medical images. The most common methods are the Trapezoidal estimation and Cavalieri's method, both described in [5]. These methods use the 2D data obtained from segmenting medical images. Unfortunately, this could lead to miscalculations in volume estimation, because such methods have no information about the form of the pathological region. This means there is no information about the pathology zone between the medical slices. Therefore, in this work it is proposed to use the reconstructed 3D model of the pathology zone.

The input data for the volume estimation algorithm is an array of Bezier patches (mesh), which were obtained using segmented data. The patches are aligned so that the z axis is in the center of the medical object. Based on this, the volume estimation formula is described as a sum of patch volumes:

$$V = \sum_{i=1}^{P} V_i \ , \qquad (9)$$

where V_i – patch volume,

P – number of patches.

To calculate the volume of each patch a surface integral is used:

$$V = \frac{1}{2} \iint_S \left|\vec{r}\right| \cdot \cos(\vec{r}, \vec{p}) \cdot ds, \qquad (10)$$

where $\vec{r}$ – vector from the patch to Z axis,

$\vec{p}$ – a perpendicular vector to patch.

To successfully implement the surface integral in a program, first it is transformed to a double integral. Because the surface of the model is given in parametric form, after the transformation, the integral (10) assumes the following form:

$$V = \frac{1}{2} \int_0^1 \int_0^1 \left|\vec{r}\right| \cdot \frac{r_x p_x + r_y p_y + r_z p_z}{\left|\vec{r}\right|\left|\vec{p}\right|} \cdot \sqrt{A^2 + B^2 + C^2}\, du dv, \qquad (11)$$

where

r_x, r_y, r_z – are vector's $\vec{r}$ coordinates

p_x, p_y, p_z – are vector's $\vec{p}$ coordinates

$$A = \begin{vmatrix} \frac{\partial y}{\partial u} & \frac{\partial z}{\partial u} \\ \frac{\partial y}{\partial v} & \frac{\partial z}{\partial v} \end{vmatrix} \quad B = \begin{vmatrix} \frac{\partial z}{\partial u} & \frac{\partial x}{\partial u} \\ \frac{\partial z}{\partial v} & \frac{\partial x}{\partial v} \end{vmatrix} \quad C = \begin{vmatrix} \frac{\partial x}{\partial u} & \frac{\partial y}{\partial u} \\ \frac{\partial x}{\partial v} & \frac{\partial y}{\partial v} \end{vmatrix} \qquad (12)$$

 ISBN 978-80-86943-94-7

Since

$$p = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ \frac{dx}{du} & \frac{dy}{du} & \frac{dz}{du} \\ \frac{dx}{dv} & \frac{dy}{dv} & \frac{dz}{dv} \end{vmatrix}, \quad (13)$$

the double integral (11) can be transformed to the form

$$V = \frac{1}{2}\int_0^1\int_0^1 x(u,v)\cdot A(u,v) + y(u,v)\cdot B(u,v)dudv, \quad (14)$$

where *x(u, v), y(u, v)* – surface point coordinates,

To calculate this integral in a program, graphic integration is used. The integral (14) assumes the following form:

$$V \approx \frac{1}{2nm}\cdot\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} x(u_i,v_j)\cdot A(u_i,v_j) + y(u_i,v_j)\cdot B(u_i,v_j), \quad (15)$$

where *n,m* – are the number of steps in parametric directions *u* and *v*.

Since the proposed method uses the 3D model of the pathological region, it has information about the form of the pathology zone, meaning the method can estimate the region even between the slices. This is more correct than the usual methods that use information only from 2D medical images.

The experiments were conducted on an ideal model object (cylinder and frustum combination) and a medical object.

Table 1 shows results on pathology zone's volume estimation, obtained by the proposed method and trapezoidal method.

	Model object	Difference from proposed, %	Medical object	Difference from proposed, %
Proposed method, mm^3	905982	-	3475	-
Trapezoidal method, mm^3	904216	0,19%	3508	3,1%
Cavalieri method, mm^3	904216	0,19%	3539	4,0%

Table 1 Results

As can be seen from table 1, the results on 2D trapezoidal method and the proposed method are similar when calculating the volume of a model object, but are also quite different when the object is irregular. Additional research has to be conducted, to test whether the reconstructed 3D model corresponds ideally to the real object.

3. CONCLUSIONS

Several methods are proposed for segmentation, visualization and volume estimation of the 3D objects.

The proposed semi-automatic medical image segmentation algorithm combines the advantages of grey-level thresholding, region based and edge based segmentation. It is fast and efficient, while allows manual adjustment to the region of interest, if necessary.

3D object visualization is based on standard graphic library OpenGL. This reduces application's requirements for computer resources and allows real-time rendering of the modeled object. The alternative method of visualization is based on ray tracing. In this case the surface of the modeled object is visualized with more detail. But the time of rendering does not allow using this method in real time.

The method of volume estimation uses 3D object model as initial data. It is more correct than the known methods that are based on 2D image data, because these methods have no information about the object between slices. Since the proposed method has information about the form of the medical object, it can correctly estimate the volume even between slices, where there is no information about the object. Although additional research has to be conducted to test whether the reconstructed model correctly describes the real object.

4. REFERENCES

[Kre07a] Krechetova K, Glaz A Development of a new segmentation method for medical images. Biomed. Eng. conference proc., Kaunas, Lithuania, 2007, pp 133-136

[Rog90a] Rogers D, Adams J Mathematical Elements for Computer Graphic, 2nd Ed., McGraw-Hill, Boston, MA, 1990.

[Hea04a] Hearn D, Baker M Computer Graphics with OpenGL, 3rd Ed., Prentice Hall, 2004.

[Sis08a] Sisojevs A, Glazs A An new approach of visualization of free-form surfaces by a ray tracing method, IEEE MELECON Proceedings, 2008, pp 872-875

[Smi06a] Smitha S, Revathy K, Kesavadas C Segmentation and Volume Estimation of Brain tissues from MR Images. IMECS, 2006, pp 543-547