

**The 19th International Conference in Central Europe on Computer
Graphics, Visualization and Computer Vision**

in co-operation with

EUROGRAPHICS

W S C G ' 2011

Full Papers Proceedings

University of West Bohemia
Plzen
Czech Republic

January 31 – February 3, 2011

Co-Chairs

Gladimir Baranoski, University of Waterloo, Canada
Vaclav Skala, University of West Bohemia, Czech Republic

Edited by

Gladimir Baranoski, Vaclav Skala

WSCG'2011 Full Papers Proceedings

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

Managing Editor: Vaclav Skala

Published and printed by:
Vaclav Skala – Union Agency
Na Mazinách 9
CZ 322 00 Plzen
Czech Republic

Hardcopy: *ISBN 978-80-86943-83-1*

WSCG 2011

International Program Committee

Balcisoy, S. (Turkey)
Baranoski, G. (Canada)
Benes, B. (United States)
Benoit, C. (France)
Bilbao, J. (Spain)
Biri, V. (France)
Bittner, J. (Czech Republic)
Bouatouch, K. (France)
Buehler, K. (Austria)
Coquillart, S. (France)
Daniel, M. (France)
de Geus, K. (Brazil)
Debelov, V. (Russia)
Feito, F. (Spain)
Ferguson, S. (United Kingdom)
Flaquer, J. (Spain)
Gallo, G. (Italy)
Gavrilova, M. (Canada)
Gudukbay, U. (Turkey)
Gutierrez, D. (Spain)
Havemann, S. (Austria)
Havran, V. (Czech Republic)
Chmielewski, L. (Poland)
Chover, M. (Spain)
Jansen, F. (Netherlands)
Klosowski, J. (United States)
Lee, T. (Taiwan)
Max, N. (United States)
Molla Vaya, R. (Spain)
Murtagh, F. (Ireland)
Pasko, A. (United Kingdom)
Pedrini, H. (Brazil)
Platis, N. (Greece)
Puppo, E. (Italy)
Purgathofer, W. (Austria)
Rojas-Sola, J. (Spain)
Rokita, P. (Poland)
Rosenhahn, B. (Germany)
Rudomin, I. (Mexico)
Sakas, G. (Germany)
Segura, R. (Spain)
Schumann, H. (Germany)
Skala, V. (Czech Republic)
Slavik, P. (Czech Republic)
Sochor, J. (Czech Republic)
Stroud, I. (Switzerland)
Teschner, M. (Germany)
Theoharis, T. (Greece)
Tokuta, A. (United States)
Vergeest, J. (Netherlands)
Wu, S. (Brazil)
Wuethrich, C. (Germany)
Zara, J. (Czech Republic)
Zemcik, P. (Czech Republic)
Zitova, B. (Czech Republic)

WSCG 2011 was supported by



SILICON GRAPHICS s.r.o



Microsoft®

Microsoft, s.r.o. ČR



Faculty of Applied Sciences

Dept. of Computer Science &
Engineering

WSCG 2011

Board of Reviewers

Akleman, E. (United States)
Ariu, D. (Italy)
Assarsson, U. (Sweden)
Aveneau, L. (France)
Balcisoy, S. (Turkey)
Battiato, S. (Italy)
Benes, B. (United States)
Benoit, C. (France)
Biasotti, S. (Italy)
Bilbao, J. (Spain)
Biri, V. (France)
Bittner, J. (Czech Republic)
Bosch, C. (France)
Bouatouch, K. (France)
Boukaz, S. (France)
Bouville, C. (France)
Bruni, V. (Italy)
Buehler, K. (Austria)
Cakmak, H. (Germany)
Camahort, E. (Spain)
Capek, M. (Czech Republic)
CarmenJuan-Lizandra, M. (Spain)
Casciola, G. (Italy)
Coquillart, S. (France)
Correa, C. (United States)
Cosker, D. (United Kingdom)
Daniel, M. (France)
de Amicis, r. (Italy)
de Geus, K. (Brazil)
Debelov, V. (Russia)
Domonkos, B. (Hungary)
Drechsler, K. (Germany)
Duke, D. (United Kingdom)
Dupont, F. (France)
Durikovic, R. (Slovakia)
Eisemann, M. (Germany)
Erbacher, R. (United States)
Erleben, K. (Denmark)
Farrugia, J. (France)
Feito, F. (Spain)
Ferguson, S. (United Kingdom)
Fernandes, A. (Portugal)
Flaquer, J. (Spain)
Fontana, M. (Italy)
Fuenfzig, C. (France)
Gallo, G. (Italy)
Galo, M. (Brazil)
Garcia Hernandez, R. (Spain)
Garcia-Alonso, A. (Spain)
Gavrilova, M. (Canada)
Giannini, F. (Italy)
Gonzalez, P. (Spain)
Grau, S. (Spain)
Gudukbay, U. (Turkey)
Guggeri, F. (Italy)
Gutierrez, D. (Spain)
Habel, R. (Austria)
Hall, P. (United Kingdom)
Hansford, D. (United States)
Haro, A. (United States)
Hasler, N. (New Zealand)
Havemann, S. (Austria)
Havran, V. (Czech Republic)
Hernandez, B. (Mexico)
Herout, A. (Czech Republic)
Horain, P. (France)
House, D. (United States)
Chaine, R. (France)

Chaudhuri, D. (India)
Chmielewski, L. (Poland)
Chover, M. (Spain)
Iwasaki, K. (Japan)
Jansen, F. (Netherlands)
Jeschke, S. (Austria)
Jones, M. (United Kingdom)
Jones, M. (United States)
Juettler, B. (Austria)
Kheddar, A. (Japan)
Kim, H. (Korea)
Klosowski, J. (United States)
Kohout, J. (Czech Republic)
Kurillo, G. (United States)
Kyratzi, S. (Greece)
Lanquetin, S. (France)
Lay Herrera, T. (Germany)
Lee, T. (Taiwan)
Lee, S. (Korea)
Leitao, M. (Portugal)
Liu, D. (Taiwan)
Liu, S. (China)
Lutteroth, C. (New Zealand)
Madeiras Pereira, J. (Portugal)
Maierhofer, S. (Austria)
Manzke, M. (Ireland)
Marras, S. (Italy)
Maslov, O. (Russia)
Matey, L. (Spain)
Matkovic, K. (Austria)
Max, N. (United States)
Meng, W. (China)
Mestre, D. (France)
Michoud, B. (France)
Mokhtari, M. (Canada)
Molla Vaya, R. (Spain)
Montrucchio, B. (Italy)
Muehler, K. (Germany)
Murtagh, F. (Ireland)
Nishio, K. (Japan)
OliveiraJunior, P. (Brazil)
Oyarzun Laura, C. (Germany)
Pan, R. (China)
Papaioannou, G. (Greece)
Pasko, A. (United Kingdom)
Pasko, G. (Cyprus)
Patane, G. (Italy)
Patow, G. (Spain)
Pedrini, H. (Brazil)
Peters, J. (United States)
Pina, J. (Spain)
Platis, N. (Greece)
Puig, A. (Spain)
Puppo, E. (Italy)
Purgathofer, W. (Austria)
Reshetov, A. (United States)
Richardson, J. (United States)
Richir, S. (France)
Rojas-Sola, J. (Spain)
Rokita, P. (Poland)
Rosenhahn, B. (Germany)
Rudomin, I. (Mexico)
Sakas, G. (Germany)
Salvetti, O. (Italy)
Sanna, A. (Italy)
Segura, R. (Spain)
Sellent, A. (Germany)
Shesh, A. (United States)
Schultz, T. (United States)
Schumann, H. (Germany)
Sirakov, N. (United States)
Skala, V. (Czech Republic)
Slavik, P. (Czech Republic)
Sochor, J. (Czech Republic)
Sousa, A. (Portugal)
Srubar, S. (Czech Republic)
Stroud, I. (Switzerland)
Subsol, G. (France)
Sundstedt, V. (Sweden)
Tang, M. (China)
Tavares, J. (Portugal)
Teschner, M. (Germany)
Theoharis, T. (Greece)
Theussl, T. (Saudi Arabia)
Tokuta, A. (United States)
Tomori, Z. (Slovakia)
Torrens, F. (Spain)
Trapp, M. (Germany)

Umlauf, G. (Germany)
Vazques, P. ()
Vergeest, J. (Netherlands)
Vitulano, D. (Italy)
Vosinakis, S. (Greece)
Walczak, K. (Poland)
Weber, A. (Germany)
Wu, S. (Brazil)
Wuenske, B. (New Zealand)
Wuethrich, C. (Germany)

Yoshizawa, S. (Japan)
Yue, Y. (Japan)
Zara, J. (Czech Republic)
Zemcik, P. (Czech Republic)
Zhu, Y. (United States)
Zhu, J. (United States)
Zitova, B. (Czech Republic)

WSCG 2011

Full Papers Proceedings

Contents

- Braun,A., Mueller,S.: GPU-assisted 3D Pose Estimation under realistic illumination 1
- Costa,V., Pereira,J.M.: Compact Rectilinear Grids for the Ray Tracing of Irregular Scenes 9
- Lez,A., Zajic,A., Matkovic,K., Pobitzer,A., Mayer,M., Hauser,H.: Interactive Exploration and Analysis of Pathlines in Flow Data 17
- Juan,M.C., Furio,D., Alem,L., Ashworth,P., Cano,J.: ARGreenet and BasicGreenet: Two mobile games for learning how to recycle 25
- Lee,J.K., Tang,W.K.: Snake-based Technique for Automated Coronal Loop Segmentation 33
- Niizaka,T., Ohtake,Y., Michikawa,T., Suzuki,H.: Multi-material Volume Segmentation for Isosurfacing Using Overlapped Label Propagation 41
- Ryu,D.-S., Park,S.-Y., Cho,H.-G.: Photo Quality Assessment based on a Focusing Map Considering Shallow Depth of Field 49
- Choudhury,B., Raghathan,A., Chandran,S.: Image-based Animation 57
- Davidovic,T., Marsalek,L., Slusallek,P.: Performance Considerations When Using a Dedicated Ray Traversal Engine 65
- Mendes,D., Ferreira,A.: Virtual LEGO Modelling on Multi-Touch Surfaces 73
- Benes,P., Strnad,O., Sochor,J.: New path planning method for computation of constrained dynamic channels in proteins 81

GPU-assisted 3D Pose Estimation under realistic illumination

Anne Braun
Fraunhofer FIT

Department for Collaborative and Augmented
Environments
53754 Sankt Augustin, Germany
anne-kathrin.braun@fit.fraunhofer.de

Stefan Müller

University of Koblenz-Landau
Institute for Computational Visualistics
56070 Koblenz, Germany
stefanm@uni-koblenz.de

ABSTRACT

This paper describes an approach which combines computer vision methods with techniques from the area of computer graphics. This method which is called analysis-by-synthesis explicitly seeks for consideration of environmental information in order to improve the resulting estimation of the 3D camera pose. In this paper, two different kinds of pose estimation will be presented. The first approach uses intensity-based methods and the second one is a feature point-based approach. The described approaches are based on a GPU-assisted rendering considering the real world illumination. These real world lighting conditions are captured using a HDR sampling technique. The results of this GPU-assisted approach are that both methods, the intensity-based as well as the feature point-based method, achieve better results in terms of a more robust and stable 3D camera pose under consideration of the real environmental information.

Keywords

Markerless Tracking, Analysis by Synthesis, Deferred Shading, 3D Pose Estimation

1. INTRODUCTION

Common computer vision approaches for markerless pose estimation typically exploit features in the current video image, like edges or point features, without further knowledge of the physical process of image generation. Computer graphics on the other side considers the process of illumination and light material interaction and generates realistic looking synthetic images. The approach described in this paper combines both research areas in order to improve the creation and detection of features for robust and accurate tracking. In other words, the objective of the approach described in this paper is to estimate the camera pose under consideration of environmental information and object properties.

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.

This additional information will be used to support a computer vision-based technique for instance to match a virtual image against the real image. Since this approach explicitly seeks for consideration of environmental information the properties and conditions which will be covered are the 3D geometry of the object, the texture, and the environmental lighting condition which results in the surface color of the object. The approach will consider these properties in the way that the resulting tracking result will be improved. It doesn't require special infrastructure in the environment and it avoids drift. This method is called Analysis-by-synthesis. We developed and tested two different approaches: an intensity-based approach and a feature-based approach. Both methods will be described and compared.

This paper is structured as follows: in the next section we review some related work before we introduce the concepts of our approach. In section 5 we describe the realization of the several methods and continue in section 6 to present the results. We finally conclude and provide a glimpse into our future work.

2. RELATED WORK

The concept of intensity-based image registration is a widely used method in the area of medical imaging. In [Hip02] an intensity-based registration approach is described to match 3D images from a magnetic resonance angiography (MRA) to 2D x-ray angiography images. They realized their approach testing six different similarity measures, whereas the pattern intensity, gradient difference and gradient correlation performed consistently accurate and robust results. An intensity-based registration method to estimate the 3D camera pose is presented by Stricker [Stri01]. He computes a 2D transformation, which registers the current frame as a whole on a reference pattern. A Euclidian transformation between the live video image and one from a set of calibrated reference images is computed using the Fourier-Mellin Transform.

Feature-based approaches were much more explored in the past. Relevant work which made use of synthetic images can be found in [Lep03a] and [Rei06a]. Lepetit et al. describe a point-based approach, which uses 3D coordinates of registered keyframes from an underlying 3D model. These coordinates will be matched with the current camera image. Reitmayr et al. apply an edge-based approach on a textured 3D model. Another method using a textured model is presented by [Ros05a]. Their approach uses a textured surface mesh which is rendered in a virtual image. A modified block matching algorithm is applied to determine correspondences between patches of the surface mesh and points in an image. [Ble09a] presented a similar approach also based on a simplified textured CAD model of the environment. But in contrast she tracks feature points instead of edges and applies a sensor-fusion-based approach. Wuest et al. [Wue05a] use an OpenGL-based rendering for visibility testing. After each render step, the framebuffer and the depth buffer will be stored. A Canny edge-detector will be applied and on the framebuffer and the z-value of the depth buffer will be used to test the visibility.

In [Sch09a], an intensity-based and a feature-based approach is presented. But in contrast to the work described in this paper, their paper describes several matching and metric techniques instead of exploring the environmental influences.

3. ANALYSIS-BY-SYNTHESIS

The work described in this paper is based on the method which is called analysis-by-synthesis. Analysis-by-synthesis can be described as an optimized comparison of objects with a given model. A priori knowledge will be used to create this model. In case of 3D pose estimation, the a priori knowledge is the environmental information which will be used

in form of computer graphics-based knowledge. Therefore, a 3D model of the target scene, typically a CAD model, is used to predict the appearances of features in the camera images, usually by projecting the model from the predicted camera pose. The analysis-by-synthesis technique has many advantages. One advantage is that the graphic card can be used to generate a rendered image from the predicted pose efficiently. Using the rendered image as reference, the correct level of detail is guaranteed. A further advantage of the analysis-by-synthesis approach in contrast to the frame-to-frame tracking is that the light conditions and occlusions of features don't disturb the results. By creating a synthetic reference image for every frame, the disadvantages like drift will be avoided.

The aim is to improve the process of 3D pose estimation with all the information the computer graphics render process can provide. The assumption for our approach was that the degree of realism of the synthetic image will influence the results. This means, the more realistic the synthetic image looks like the more accurate is the resulting pose. Therefore, beside the 3D geometry, the a priori knowledge includes also the texture of the material and the realistic lighting condition of the environment respectively.

4. GPU-BASED TRACKING

The foundation of the analysis-by-synthesis approach provides the deferred shading technique. Deferred shading is a technique, where the parameters which are required by a shader, like position, material, normal etc., are rendered to buffers and the lighting calculation will be realized as 2D post-processing using the information stored in these buffers. This shading process heavily relies on multiple render targets (MRT) to perform in real-time. These

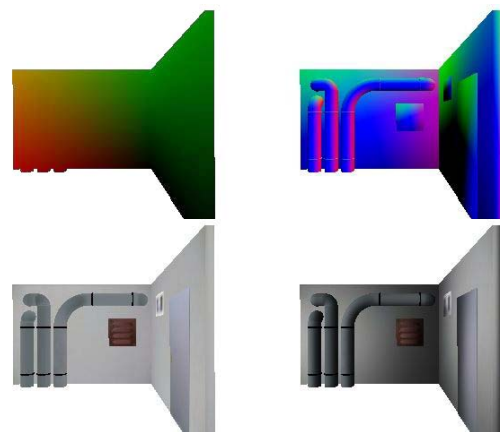


Figure 1. Multiple render targets (MRT) for Deferred Shading: vertex positions, normals, texture and final rendering

multiple render targets are called geometry buffer (G-buffer). An example of multiple render target and the final result is shown in figure 1. Initial research on deferred shading has been done by Deering et al. in 1988 [Dee88a] and Saito et al. [Sai90a] who introduced the geometric buffer (G-Buffer) in 1990.

As already mentioned, two different approaches will be described in this paper. The first approach is an intensity-based approach and the second is a feature-based method. Each of the approaches makes use of the G-buffer in a different way. The intensity-based approach applies the described deferred shading technique to create a synthetic image using the current object parameters. This identical virtual copy is used to detect the appropriate camera pose by a simple comparison of the real image and the virtual copy. Applying an optimization method, the parameters of the vector for the camera pose will converge to the solution. To determine the distance function, which is the difference between the real and synthetic image, the intensity-based approach will compare the images using the pixel intensities. The basic input data to the registration process are two images: the live video image and the rendered image. Registration is treated as an optimization problem with the goal of finding the spatial mapping that will bring the video image into alignment with the rendered image. A metric provides a measure of how well the video image is matched by the rendered image. This measure forms the quantitative criterion to be optimized by the optimizer over the search space defined by the parameters of the camera transform.

Like the intensity-based approach, the feature-based concept makes also use of the G-buffer. The features used for this approach can be either point-based, edge-based or a hybrid approach, which combines point and edge features. Running a feature detector on the video image, significant 2D point features will be obtained. These 2D point features together with the corresponding 3D point in the world will be used to calculate the 3D camera pose using a 2D-3D registration. According to the detected 2D features in the video image, the 3D information will be obtained from the G-buffer where the positions are stored at the corresponding position in the vertex buffer as color vector. Using the information of the render targets, 2D-3D correspondences of the features can be created and used for pose estimation. The analysis by synthesis approach will be applied by using the material buffer, vertex buffer and normal buffer to render a realistic lighting simulation to compare the features accordingly.

5. REALIZATION

The first step of the analysis by synthesis approach is the creation of the virtual image. This synthesis will be realized by acquiring the model data which are the geometry, the texture and the lighting information.

In this approach the geometry and texture were reconstructed manually using a laser scanner and the Photo Modeler software¹.

The lighting condition was reconstructed using a sampling method. The sampling technique doesn't require the entire image information. In contrast to environment mapping, it uses only a small amount of selected pixels. Although the number of light source will be reduced, the general amount of luminous flux will be guaranteed for several sampling densities. The sampling method is based on the K-mean clustering process. It requires a light-probe image in longitude -latitude format. Figure 2 shows an exposure series of nine images which was used to create a light probe.



Figure 2. Nine images captured with a fish-eye lens at different exposure levels

The output is a user defined number of light sources (s. figure 3). This kind of importance sampling [Kol03a] starts with one initial light source, which is randomly chosen. The pixel in the light probe image will be partitioned into sets according to the distance to the light source. Each of the k light sources is moved to the center of mass of its set. The pixels are repartitioned according to the new light source direction and the process will be repeated until convergence.

Using the gathered data, the virtual model of the video image will be rendered under consideration of the real environmental lighting.

The analysis-by-synthesis method can be described as an optimization problem. Figure 4 shows the general process of a registration approach based on analysis-by-synthesis. The first step is the acquisition

¹ <http://www.photomodeler.com/>

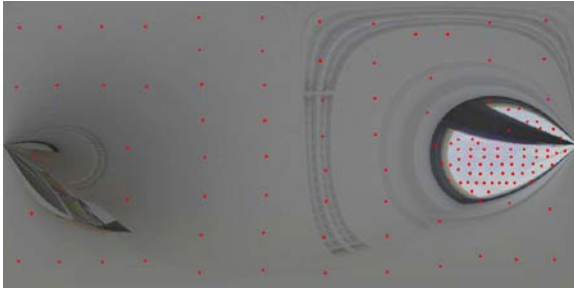


Figure 3. Light-probe image, generated of the images in figure 2, in longitude-latitude format, sampled with 128 light sources.

of image data before the similarity-features need to be decided. Similarity-features can be significant geometric primitives or pixel intensities. Furthermore, a similarity metric has to be selected. This metric describes the feature matches in the real and synthetic images. Finally, this metric will be optimized to get the best parameters. The kind of parameters depends on selected features and therefore on the different approach. Two different approaches will be distinguished in this work: Intensity-based and feature-based methods.

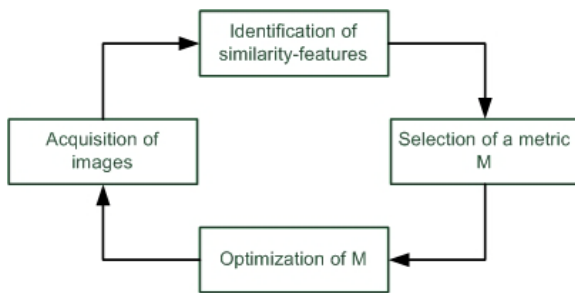


Figure 4. General process of pose estimation using an analysis-by-synthesis method

The deferred shading was realized with the OpenGL shading language GLSL. The rendering was divided in three phases: A geometry phase, a lighting phase and a post-processing phase. The G-buffer will be created in the geometry phase. The vertex shader of this phase receives the 3D model data which will be transformed to the view space and output it to the fragment shader. The fragment shader is responsible for filling the G-buffer's data. In the lighting phase, the according fragment shader uses the sampled light sources and light colors together with the data in the G-buffer to calculate the realistic illumination. The final solution texture map will be rendered to the main frame buffer in the post-processing phase.

The realization of the two computer vision-based approaches will be explained in the remainder of this section.

Feature-based approach

Applying the analysis-by-synthesis approach using features based on interest points, the technique will be used to predict the appearances of the natural interest points by rendering a 3D copy of the scene. A real-time rendered reference image has several advantages against a static feature map. By rendering the virtual image, the appearance of each feature can be adapted to illumination changes and therefore improve the registration of features in the two images which corresponds to each other. The feature point-based approach uses natural geometric primitives like points or corners which will be detected and tracked in the live image. These interest points are combined with a descriptor describing its appearance. Using this descriptor the features will be matched with a reference set of features or tracked in a new image. All feature point-based approaches have in common that they are invariant for rotation and translation. To be scale invariant, the features will be searched in different scale spaces.

For the work described in this paper, a SURF-based approach was used. The SURF (speed up robust features) feature method was developed by Bay et al. [Bay08a]. The SURF features are faster to detect and to match and the detector is more efficient as other feature detectors ([Bay08a] [Cheng07a]).

Starting from the initial camera pose, the features will be detected in both images. Using the G-buffer, the 3D position of the features in the rendered image can be reconstructed, since the coordinate is stored as RGB color in the vertex buffer. A detected feature in the video image will be matched using a similarity metric with all features in the rendered image. To measure the similarity the summed squared distance metric will be applied. This metric is based on the minimization of the differences of intensities in both images.

If the resulting error is below a certain threshold, the 2D feature in the video image and the 3D feature of the rendered image will set as 2D-3D correspondence.

$$M = \arg \min \sum_i \|m_i - f_i(M)\|^2$$

The pose estimation will be realized by minimizing the projection error between the 3D feature point M , projected with the projection function f_i and the 2D feature m_i in frame i .

Intensity-based approach

In contrast to the feature-based approach, the intensity-based method neither requires feature extraction nor is the search for correspondences necessary. The pixel values of two images

correspond in their coordinates and will be compared pair-wise. The realization of the analysis-by-synthesis approach using a direct intensity-based measurement was based on the Insight Segmentation and Registration Toolkit (ITK)². In ITK, registration is performed within a framework of pluggable components that can easily be interchanged. The framework provides several functions to calculate the similarity metric. In case of the camera pose estimation, the object in 3D space and a 2D image have to be matched. Since the ITK framework is actually for medical imaging, it doesn't provide the full functionality for these circumstances. The classes for the metric and transformation functions are therefore derivated from the original ITK classes and reimplemented according to the requirements of 3D pose estimation. The modification in the reimplemented metric classes affects especially the *GetValueAndDerivative()* function, since this function will be called in every optimization step and within that function, all the metric values and parameters will be returned to the optimizer.

There are seven parameters to be optimized: Four parameters for the rotation (axis plus angle) and three parameters for the translation. To calculate the metric values, these parameters will be modified. The resulting metric value will be compared to the former metric value and according to the difference, the derivation of the optimizer will be estimated. After the parameters have been changed, the image needs to be re-rendered with the modified camera pose and synthetic image of the registration process needs to be renewed according the current frame buffer. This process will be repeated until the metric value is minimal. An optimizer is required to explore the parameter space of the transformation and search for optimal values of the metric. As optimizer, the *itk::RegularStepGradientDescentOptimizer* class was selected. This optimizer belongs to the class of gradient descent methods. The size of the step lengths will be reduced depending of the direction of movement in the parametric space. For the intensity-based approach two similarity measurement methods were used: Sum of Squared Differences (SSD) and Normalized Cross-Correlation (NCC). The resulting values of the normalized SSD range between [0,1], where 0 is the highest similarity. The range of values for the NCC is between [-1, 1], where 1 is the highest similarity.

The normalized SSD and NCC metric are defined with:

$$SSD(I, M) = \frac{\sum_{x,y} (I(x, y) - M(x, y))^2}{\sqrt{\sum_{x,y} I(x, y)^2} \sqrt{\sum_{x,y} M(x, y)^2}}$$

² <http://www.itk.org/>

$$NCC(I, M) = \frac{\sum_{x,y} (I(x, y) - \bar{I}) \cdot (M(x, y) - \bar{M})}{\sqrt{\sum_{x,y} (I(x, y) - \bar{I})^2} \sqrt{\sum_{x,y} (M(x, y) - \bar{M})^2}}$$

where $I(x,y)$ is the value of the image pixel in the video image I and $M(x,y)$ is the pixel value in the image of the synthetic model M . \bar{M} and \bar{I} are the mean value of the video and the rendered image.

6. RESULTS

In order to evaluate the performance of the presented approaches, we estimated the pose of two different test objects and compared the results. The two tested scenarios were placed indoor. Figure 4 shows the two test objects as rendered image and as real image.

The experiments were performed under a controlled condition in an office environment. For the rendered image we sampled the environmental light probe with 128 light sources



Figure 4. The two test objects: in the video image (left) and rendered (right)

For the intensity-based approach we took a small wooden model of a basement room. This room consists of two sides with mostly homogenous colored walls, a door and a window, three pipes and a ventilation box. In the experiment the pose was estimated under three different conditions. The first condition was to match a rendered image with the same rendered image as video image. This setting was used as reference for the implemented approach. The second condition was the matching of the live video image with a rendered image. However, the virtual image was not rendered under realistic light conditions. The third condition tested the approach described in this paper. A real video image was matched with a synthetic image rendered using the real light conditions. For each of the three conditions,

the two metric approaches as described in the previous section were applied to measure the similarity.

Figure 5 shows the results for the two similarity metrics according the three test conditions. The diagrams show that the optimization converged already after four iterations. The initial step width of the optimizer for the four parameters of the rotation was 0.001 and the step width for the three parameters of the translation was 0.2. Regarding the diagrams, it's also obvious that the condition with the realistic rendered image provides better results than the synthetic reference image which was not rendered realistically. This can be seen on the high metric value of the NCC metric respectively the low metric value of the SSD metric. Comparing the SSD and the NCC metric, the NCC leads to a more precise solution.

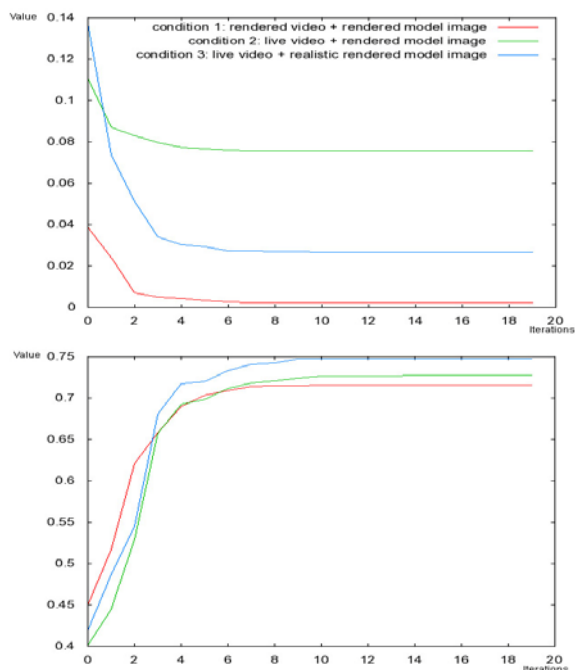


Figure 5. Results of the metric value for the two similarity metrics: The normalized NCC (bottom) and the normalized SSD (top). (according to the number of iterations for optimization).

Figure 7 shows the final results of the estimated poses under the three conditions. The first row shows the result of the first condition, where two identical rendered images were matched against each other. The resulting image and therefore the estimated pose is equal to the one of the input image. The second row presents the estimated pose of the second condition which registered a live video image with a rendered image without realistic illumination. The estimated pose was not precise and correct. Especially the z-direction of the translation was incorrect. The realistic rendering showed better

results, which can be seen in the third row. Compared to the condition 1, the approach converged with very good results. The translation is very precise, and the values of the rotation are only slightly different.

To sum up the results for the intensity-based approach, the analysis-by-synthesis approach using a realistic rendered image as synthetic model provides a better estimated pose as a synthetic image which was rendered without considering the realistic illumination. Light sources which are not precise or even incorrect lead to errors in pose estimation.

For the feature-based approach, we tested another object instead of the wooden model of the basement. Since this model consists of mainly homogenous surfaces, the feature point-based approach didn't succeed due to the lack of feature points in the environment. This markerless computer vision approach using feature points is therefore not suitable to estimate the pose in rooms which are poor of features. Instead of the model we have chosen a pattern with an earth texture on it for the second test to demonstrate and test the second approach.

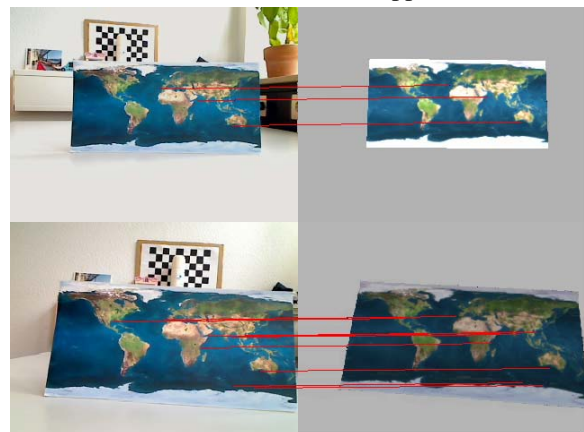


Figure 6. Results for the feature point-based approach: The scenario with the realistic lighting conditions provides more feature correspondences than the other scenario.

Testing this pattern-based scenario with the feature point-based approach, we also found out, that the more realistic the reference image is rendered the more robust and stable the pose will be estimated. The quality of the estimated pose depends on the number of correspondences, because possible outliers can be compensated by other correspondences. Furthermore, a large number of correspondence pairs reduce the risk of getting trapped in a local minimum.

We tested the earth pattern scenario under different conditions similar to the basement model scenario.

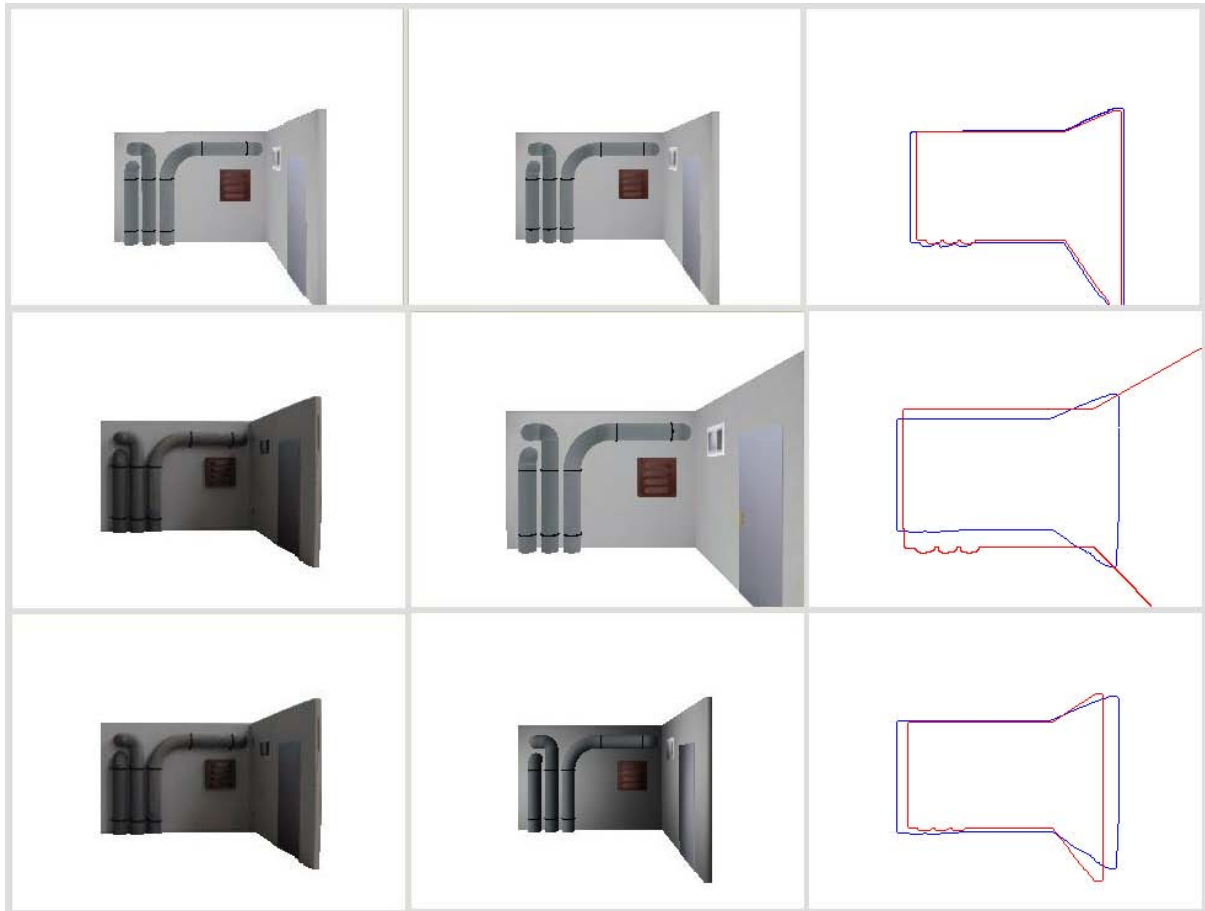


Figure 7. The final results of the estimated poses under three conditions: In the first row, two identical rendered images were matched against each other. The second row shows the estimated pose using a live video image with a rendered image without realistic illumination. The third row shows the rendered image rendered under realistic lighting conditions. The comparison in the third column shows the outlines of the input video image of the first column with blue lines and the synthetic reference image of the second column drawn with red lines.

We rendered a virtual copy of the pattern without considering the real lighting conditions for the first test. For the second test, we rendered the synthetic image using the real illumination. The result is shown in figure 6. The scenario with the realistic lighting conditions provides more feature correspondences than the other scenario.

We furthermore tested the amount of realism by comparing the number of correspondences. We therefore approach to realistic lighting conditions by increasing the number of light samples and respectively light sources. In figure 8 the results of the test is shown. An increasing number of light sources and therefore a more realistic appearance of the rendered image increase the number of feature correspondences.

The final result of the analysis-by-synthesis approach for the feature point-based method is that a more

robust and stable pose will be estimated under realistic rendered light conditions.

7. CONCLUSION

In this paper we presented two approaches for 3D pose estimation using the analysis-by-synthesis approach. We therefore rendered a synthetic image of the real scenario to apply a registration method to estimate the 3D camera pose. We realized an intensity-based approach and a feature point-based approach. Both methods achieved better results using a realistic rendered image under real lighting conditions. The real lighting conditions were reconstructed based on an importance sampling approach using a HDR image of the environment captured with a fisheye lens.

The described methods require environmental data like a 3D model, the lighting and the texture to

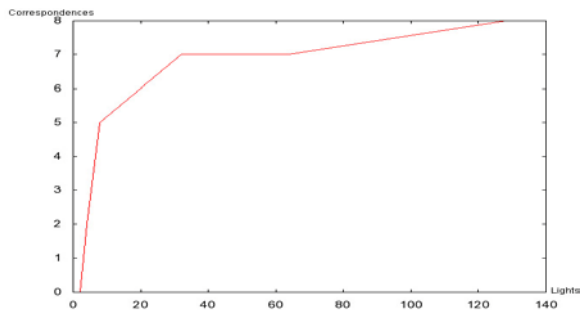


Figure 8. An increasing number of light sources and therefore a more realistic appearance of the rendered image increase the number of feature correspondences.

render a realistic virtual copy of the scenario. At first sight, this seems to be a disadvantage. But regarding the 3D model and the texture, many 3D models are already available for various situations and scenarios. Either it was reconstructed for public access for instance Google Earth® or the models exist anyway from previous design steps like in the construction industry. Furthermore, the reconstructed and sampled lighting condition can be used for improved rendering, for instance for photorealistic AR applications.

A significant contribution of the intensity-based approach is that the 3D camera pose can be estimated although the scenario doesn't provide many feature points. This advantage will overcome the problem of markerless computer vision based tracking in unconstrained environments without additional sensors. The advantage of the feature point-based approach is that errors and artifacts like drift will be avoided since this approach doesn't use a frame-to-frame tracking but a reference image.

8. FUTURE WORK

The future work will include several improvements. One task will be to replace the ITK based implementation of the intensity-based approach. Since this library is actually intended for medical registration, the modification for our approach results in slow computation times. A tailor-made solution for this problem will increase the frame rate. However, using this library enables us to try this approach and test first prototypes. Furthermore, the required data can be acquired automatically.

9. ACKNOWLEDGMENTS

We acknowledge the support from the European Commission for the research project CoSpaces under grant number FP7 IST-5-034245.

10. REFERENCES

[Bay08a] Bay Herbert, Ess Andreas, Tuytelaars Tinne, and Van Gool Luc. Speeded-up robust

features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, 2008.

[Ble09a] Bleser Gabriele, Stricker Didier. Advanced tracking through efficient image processing and visual-inertial sensor fusion. *Computer & Graphics*, Vol. 33, Pages 59-72, Elsevier, New York, 2009.

[Cheng07a] Cheng D., Xie Shane, and Himmerle E.. Comparison of local descriptors for image registration of geometrically-complex 3d scenes. pages 140–145, 2007.

[Dee88a] Deering Michael, Winner Stephanie, Bic Schediwy, Duffy Chris, and Neil Hunt. The triangle processor and normal vector shader: a vlsi system for high performance graphics. *SIGGRAPH '88: Proceedings of the 15th annual conference*, pages 21–30, New York, NY, USA, 1988. ACM.

[Hip02] Hipwell, J. H., Penney G. P., Cox T. C., Byrne J. V., and Hawkes D. J. 2D-3D Intensity Based Registration of DSA and MRA – A Comparison of Similarity Measures. *Lecture Notes in Computer Science*, 2489 (2002), pp. 501–508

[Kol03a] Kollig Thomas and Keller Alexander, editors. *Efficient Illumination by High Dynamic Range Images*, 2003.

[Lep03a] Lepetit V., Vacchetti L., Thalmann D., and Fua P.. *Fully Automated and Stable Registration for Augmented Reality Applications*, 2003.

[Rei06a] Reitmayr Gerhard and Drummond Tom W., editors. *Going out: Robust Modelbased Tracking for Outdoor Augmented Reality*. ISMAR, 2006.

[Ros05a] Rosenhahn B, Ho H, Klette R. Texture driven pose estimation. In: *Proceedings of the International Conference on Computer Graphics, Imaging and Visualization (CGIV'05)*, Beijing, China, July 2005. pp 271–277.

[Sai90a] Saito Takafumi and Takahashi Tokiichiro. *Comprehensible rendering of 3-d shapes*. volume 24, pages 197–206, New York, NY, USA, 1990. ACM.

[Sch09a] Schumann, M. Achilles, S., and Mueller, S. *Analysis by Synthesis Techniques for Markerless Tracking*. Gi VR-AR workshop, 2009.

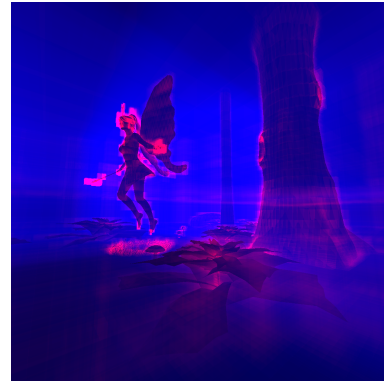
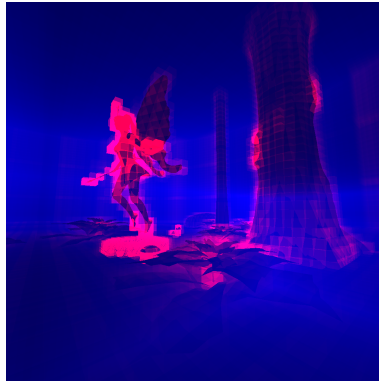
[Str01] Stricker, D. Tracking with Reference Images: A Real-Time and Markerless Tracking Solution for Out-Door Augmented Reality Applications. In *Proc. of VAST*, 2001

[Wue05a] Wuest Harald, Vial Florent, and Stricker Didier. *Adaptive Line Tracking with Multiple Hypotheses for Augmented Reality*. ISMAR, 2005

Compact Rectilinear Grids for the Ray Tracing of Irregular Scenes

Vasco Costa
INESC-ID/IST
Lisbon, Portugal
vasc@vimmi.inesc-id.pt

João Madeiras Pereira
INESC-ID/IST
Lisbon, Portugal
jap@vimmi.inesc-id.pt



Fairy Forest irregular scene rendering complexity

The regular grid spatial subdivision on the left provides a less well balanced triangle distribution per cell than the rectilinear grid structure on the right. Both spatial partitioning grids feature a similar resolution.

ABSTRACT

Regular grid spatial subdivision is frequently used for fast ray tracing of scanned models. Scanned models feature regular sized primitives, with a regular spatial distribution. Grids have worse performance, than other subdivision techniques, for irregular models without these characteristics. We propose a method to improve the performance of grids for rendering irregular scenes by allowing the individual placement of grid split planes: the rectilinear grid. We describe how to construct and traverse a rectilinear grid. To exploit cache memory in modern processors compression is used for the split plane and grid cell data. We demonstrate in a series of tests that the method has faster ray tracing rendering performance than a compressed regular grid of similar dimensions.

Keywords

Ray tracing, grids.

1. INTRODUCTION

Whitted ray tracing [Whi80] is a technique which is experiencing a renaissance in the graphics research community, since it is a simple, elegant algorithm which can accurately render not just local illumination, but also shadows, reflections, and refractions.

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 algorithm is also amenable for implementation into parallel architectures, because of its inherent per ray parallelism. In addition it can accurately render higher order object primitives such as spheres, cylinders, cones, or other quadrics. Not just triangles.

The recent improvements in graphics hardware and GPGPU programming languages have been empowering software developers everywhere to replace the rendering pipeline partially or even in its entirety. In order to be able to compete with rasterization, ray tracing must have good enough performance. It should be able to render the scenes which users expect to visualize today. It is the

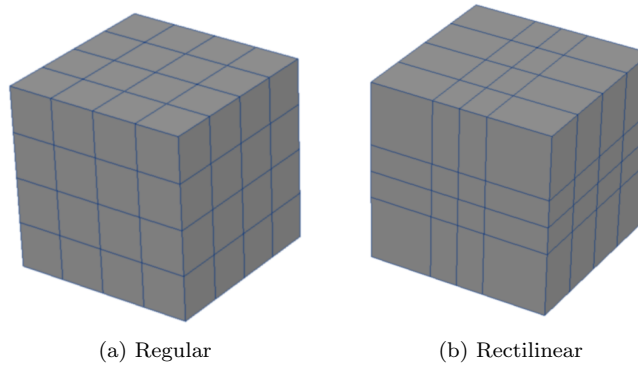


Figure 1. Grid spatial subdivision types

author’s opinion that is desirable to use ray tracing across the whole rendering pipeline as a replacement for rasterization. Barring the rendering performance reasons, which this work aims to address, using ray tracing across the whole pipeline would enable a more seamless experience for artists and application developers. One example is the application of shadows on a scene, where ray tracing does not suffer from the artifacts and difficult parametrization issues of current rasterization algorithms. The implementation issues of these algorithms are described by Kuehl et al. in [KBB07].

2. BACKGROUND

In order to have real-time ray tracing performance, for complex scenes, it is necessary to employ so called acceleration techniques. These techniques employ a divide-and-conquer strategy to solve the ray tracing problem. The most popular acceleration techniques are spatial subdivision techniques: bounding volume hierarchies (BVHs), kd-trees, and grids [WMG⁺07].

Grids are a 3D space subdivision method, introduced by Fujimoto in [FTI86], where space is subdivided into same sized cubically shaped cells. These cells also known in the literature as *voxels*. Grids have several interesting traits. They feature linear $O(N)$ construction time, constant $O(1)$ update time, and have fast traversal times for rendering. Best case traversal complexity is $O(1)$ and worse case traversal complexity is $O(\sqrt[3]{N})$. N is the number of objects in a scene.

The main issues with regular grids have been high memory consumption and poor adaptation to irregular scenes. Irregular scenes can feature different sized, irregularly distributed, geometry. One example of such troublesome geometry would be a highly detailed object inside a low detail box. This is known in the literature as the teapot in a

stadium problem. In contrast grids are very efficient at rendering scanned scenes.

Both of these issues are due to the way the splitting planes are positioned in a regular grid. All the cells must have the same size, so regular grids are a poor fit for irregular scenes. The heuristic used to compute the number of split planes for each axis of the 3D grid is usually some variation of:

$$\sqrt[3]{\rho \times N} \quad (1)$$

Where ρ is a user defined parameter which describes the density of the scene. Most implementations use a default ρ value of 4. This heuristic ensures the memory consumption is proportional to the number of primitives in the scene.

One approach to solve the issue is Jevans and Wyvill’s recursive grid [JW89]. In this approach grids are recursively applied to subdivide the scene in a shallow grid hierarchy. This technique places grids inside grids to enable variable subdivision cell sizes across the scene. This technique improves rendering performance substantially. The main issues with this approach are that it further increases memory consumption, makes it impossible to do $O(1)$ partial scene updates, and the method still has some issues adapting to irregular scenes compared to other techniques such as kd-trees and BVHs which employ surface area heuristics (SAH) to more accurately place the split planes [WH06, Wal07]. Much of the traversal time in the grids is still spent skipping empty cells with no geometry in them. Memory and time efficient methods to construct and traverse such recursive grids were described by Costa et al. [CPJ10].

Several researchers have tried in the past to improve regular grid heuristics with a limited degree of success. One approach, followed by Klimaszewski in [KS97], was to build a grid hierarchy overlaid on a previously constructed SAH bounding volume hierarchy. Cazals and Puech in

[CP97] analyzed the scene geometry by clustering geometry into groups and attempting to place the regions of regular geometry in the scene into separate grids. These techniques have not been very popular because of their implementation complexity, and a performance which is not globally better than that of the simpler to implement recursive grid. Havran did an interesting performance analysis in [HS99] of these grid spatial subdivision techniques. Ize provides in [ISP07] an in depth analysis of grid heuristics. There he describes how to build a good performing grid for several kinds of scenes. Ize also described an heuristic for recursive grids which maintains total space consumption proportional to the number of primitives in the scene. This recursive grid heuristic is included in the Manta interactive raytracer [BSP06].

3. RECTILINEAR GRIDS

We propose to relax the grid split plane positioning using a rectilinear grid (see Figure 1). This should enable a better balancing of the scene geometry among the grid cells resulting in faster ray tracing performance. Rectilinear grids have previously been used in the field of volume ray tracing [PPL⁺05]. In this work we describe how to efficiently construct and traverse a rectilinear grid for general ray tracing purposes.

Construction

First we compute the scene's bounding box. Next we finely sample the scene along each major axis x, y, z . The number of samples taken per axis is described by the following equation:

$$sample_i = 100 \times S_i \times \sqrt[3]{\frac{4 \times N}{V}} \quad i \in x, y, z \quad (2)$$

Where N is the number of primitives, V is the volume of the scene, and S_i contains the bounding box dimensions in that axis. Each sample contains a count of the number of primitives in that particular subvolume.

Then we compute the sum of these primitive counts for each axis:

$$sum_i = \sum_{j=0}^{sample_i} count_j \quad i \in x, y, z \quad (3)$$

The scene is partitioned into $ncells_x \times ncells_y \times ncells_z$ voxels. The split planes for each cell are placed in order to divide the scene into regions with a similar number of primitive counts per axis where:

$$ncells_i = S_i \times \sqrt[3]{\frac{4 \times N}{V}} \quad i \in x, y, z \quad (4)$$

Algorithm 1 : Rectilinear grid traversal

```

function NEAREST-AXIS(int)
  if intx < inty then
    if intx < intz then
      return x
    else
      return z
    end if
  if inty < intz then
    return y
  else
    return z
  end if
end if
end function

function TRAVERSE(ori, dir, planes, cells, ncells)
  test if the ray hits the bounding box of the scene
  find the nearest ray/box intersection point p
  id ← FIND-FIRST-CELL(p, planes)
  for all i ∈ x, y, z do
    if diri < 0 then
      stepi, incri, stopi ← 0, -1, -1
    else
      stepi, incri, stopi ← 1, +1, ncellsi
    end if
    inti ← (planesi[idi + stepi] - orii) / diri
  end for
  t ← +∞
  loop
    if cells[idx, idy, idz] ≠ ∅ then
      traverse cell
      find the nearest ray/object intersection t
    end if
    repeat ▷ skip empty cells
      i ← NEAREST-AXIS(int)
      if inti ≥ t then
        return t
      end if
      idi ← idi + incri
      if idi = stopi then
        return t
      end if
      inti ← (planesi[idi + stepi] - orii) / diri
    until cells[idx, idy, idz] ≠ ∅
  end loop
end function

```

This rectilinear grid has a similar number of cells compared to a regular grid because we use the same heuristic to compute the number of split points. Contrary to a regular grid, the split planes are not equidistant; but divide regions with similar amounts of geometry. Non empty cells in a rectilinear grid will thus contain a smaller amount of primitives on average than a regular grid of the same dimensions. This will be explored in greater detail in Section 4. This construction method has a computation time complexity of $O(N)$. Space complexity is also $O(N)$.

Traversal

We employ a generalization of the voxel traversal method described by Cleary and Wyvill [CW88].

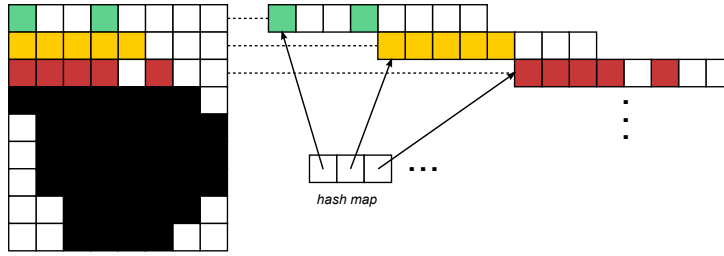


Figure 2. Row displacement compression

This method is described in Algorithm 1. Since we have variable positions for the split planes, it is not so worthwhile to precompute many of the traversal variables. If we cached these values we would spend many computing cycles performing slow memory loads just to save a couple of fast arithmetic operations. Profiling results showed that one of the main bottlenecks of this rectilinear grid traversal method consists in loading the split plane data from cache. Another large bottleneck is the branch mispredictions which occur while searching the next non empty cell. This rectilinear grid traversal method spends more time doing cell traversals than a regular grid traversal method because of these loads and the extra arithmetic operations. However, since each cell contains less primitives, less time is spent performing ray/primitive intersections. The end result is a net increase in rendering performance for many scenes.

Finding the First Cell

We have the split plane data for each axis in the rectilinear grid. The most expedient way to im-

Algorithm 2 : Finding the first cell in a rectilinear grid in $O(1)$ time with a stored compressed bit array

```

function UNPACK(axis, i)
    diff ← 0
    for all j such that  $0 \leq j < msb[axis]$  do
        diff ← diff × 2 + storage[axis][i × msb[axis] + j]
    end for
    return diff
end function

function LOOKUP(axis, i)
    predict ←  $i \times n_{cells}_{axis} / n_{samples}_{axis}$ 
    diff ← UNPACK(axis, i)
    return  $min_{axis} + predict + diff$ 
end function

function FIND-FIRST-CELL(p, planes)
    for all axis ∈ x, y, z do
        i ← CLAMP( $\frac{p_{axis}}{sample_{size}_{axis}}$ , 0,  $n_{samples}_{axis} - 1$ )
        idaxis ← LOOKUP(i)
    end for
    return id
end function

```

plement an algorithm to find a point in this grid, without by using any extra memory, is by doing a binary search in the ordered list of the split points. This approach has $O(\log N)$ time complexity. This is clearly worse performing than the method employed for finding the first point in a regular grid which has $O(1)$ time complexity.

We speeded up this part of the algorithm by employing a lookup table which maps the quantized fine regular grid coordinates used in the sampling step during construction to the actual rectilinear grid coordinates. This lookup table, if uncompressed, would not easily fit in the caches compromising the rendering performance of the algorithm.

To fit the lookup table into the processor cache we compressed the data using arithmetic encoding (see Algorithm 2). The unpacking function, which loads a bit array from memory into an integer register, can be further optimized using assembly instructions. The prediction function we employed assumes most split planes will be regularly separated at constant intervals. If the distance between all the split planes is the same, which is common for scanned scenes, we have a perfect prediction. This means for such scenes no additional memory would be required to store the table. This prediction function also behaves well for other more irregular scenes. This can be seen in the results for the Fairy Forest scene. Time complexity with the lookup table is $O(1)$.

Cell Compression

We minimized the memory required to store the grid by employing the row displacement compression algorithm [LD08] described by Lagae and Dutré. This algorithm works by compressing empty grid cells via row hashing (see Figure 2). Frequently grid acceleration structures feature 90% or more empty cells. This compression scheme reduces the memory required to store a grid up to 20 : 1.

The main issue with this scheme is that it makes it harder to perform partial grid updates. How-

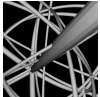
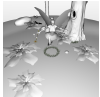


				
	AEK-24-cell	Fairy Forest	Buddha	Thai Statue
scene				
num triangles	122.88 K	174.11 K	1.09 M	10.00 M
memory	2.14 MB	4.22 MB	18.67 MB	171.66 MB
regular grid				
grid res	79x78x79	140x36x140	121x295x121	302x508x261
% empty cells	93.58%	79.12%	94.86%	98.44%
avg objects / n-empt cell	14.06	6.70	13.02	29.25
avg cells / object	3.58	5.67	2.66	1.83
mem cells	249.39 KB	800.85 KB	1.88 MB	8.97 MB
mem object lists	1.68 MB	3.76 MB	11.03 MB	69.78 MB
memory	1.92 MB	4.55 MB	12.91 MB	78.75 MB
build time	0.03 s	0.06 s	0.29 s	2.38 s
render time	2.54 s	4.95 s	0.70 s	1.63 s
rectilinear grid				
sample res	7944,7829,7903	13968,3570,13968	12130,29537,12144	30187,50824,26072
grid res	79x78x79	139x36x139	121x294x121	300x506x259
% empty cells	91.02%	78.40%	92.82%	96.52%
avg objects / n-empt cell	1.13	1.59	0.77	0.61
avg cells / object	4.49	6.37	3.06	2.41
mem cells	319.09 KB	929.48 KB	2.42 MB	13.64 MB
mem object lists	2.11 MB	4.23 MB	12.69 MB	92.08 MB
mem planes	960 B	1.24 KB	2.11 KB	4.18 KB
memory	2.42 MB	5.14 MB	15.12 MB	105.72 MB
build time	0.09 s	0.14 s	0.85 s	7.35 s
render time	2.23 s	2.76 s	0.71 s	1.38 s

Table 1. Performance results for several scenes. All scenes were rendered at 1024×1024 resolution with one ray sample per pixel. Only one thread was employed.

ever it is quick to rebuild such a grid, most of the steps performed in the construction algorithm can be parallelized, interactive frame rates can be achieved for many scenes.

4. RESULTS

A prototype implementation was written in order to test the viability of this algorithm for ray tracing complex scenes. The implementation was coded in ANSI C++ with use of the Boost libraries. This implementation does not use intrinsics or assembly instructions. The implementation was run on an Intel Core 2 CPU at 3.0 GHz with 2 GB of RAM under the Linux operating system.

All test scenes were rendered at 1024×1024 resolution with one ray sample per pixel and diffuse shading. Only one rendering thread was employed. Ray tracing performance scales linearly with the number of processor cores in the system.

Triangles are stored in memory using indexed vertex arrays. Ray/triangle intersection is done using

the Möller-Trumbore algorithm [MT05].

The performance comparison baseline is the compressed regular grid algorithm described in [LD08]. Our rectilinear grid algorithm uses a similar grid resolution as can be seen at Table 1. Both of these algorithms were implemented by us on our ray tracing system.

We selected four scenes for testing purposes: AEK-24-cell, Fairy Forest, Buddha, Thai Statue. These scenes are representative for many kinds of applications. AEK-24-cell contains a scene with data similar to that used for scientific visualization, Fairy Forest is an irregularly distributed scene similar to what we could find in a game application, the Buddha and Thai Statue are scanned scenes with heavy geometry.

Several things can be noted by examining the test results. As expected our rectilinear grid implementation has much improved render time performance (79% faster) for the irregular Fairy Forest scene. It also provides a performance speedup

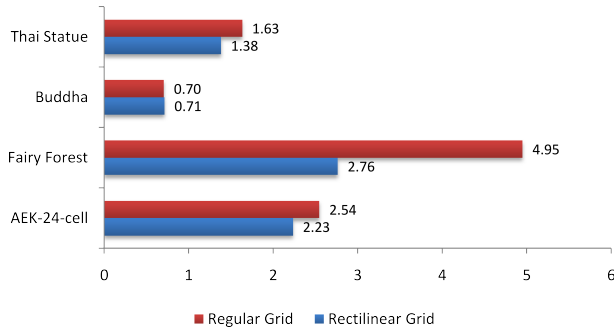


Figure 3. Render time in seconds. Lower values are better.

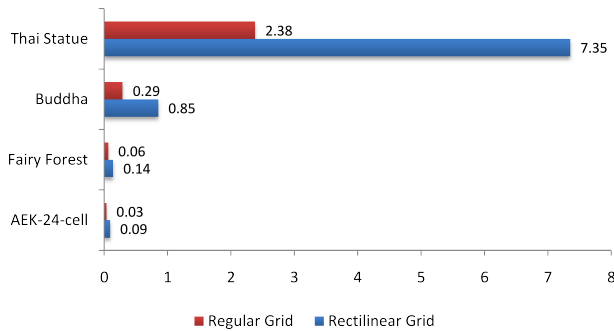


Figure 4. Build time in seconds. Lower values are better.

for the AEK-24-cell (14%) and Thai Statue (18%) scenes. In contrast the rectilinear grid has slightly worse rendering performance (-1%) for the Buddha scene.

We decided to examine these results in more depth. The rectilinear grids contain less triangles in each cell than a regular grid would. This was expectable due to the way we compute the split plane positions. However each triangle in the rectilinear grid overlaps more cells. This may make it worthwhile to employ mailboxing in our rendering system. Our implementation does not have this feature.

The split plane data easily fits into the L1 cache minimizing the amount of memory fetches required. Both grid implementations feature a 3D bitmap. Each bit in the bitmap states if a grid cell is empty or not. This allows us to reduce the amount of memory bandwidth required to traverse empty cells.

The build times for the rectilinear grid (see Figure 4) are 2-3x slower. This is mostly due to the sampling step during preprocessing. The sampling is done at a 100x higher resolution, per axis, than that of the grid itself.

Increasing the sampling resolution further does not improve the rendering performance for the

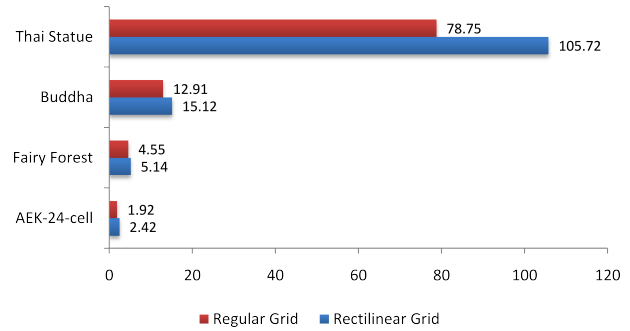


Figure 5. Memory consumption in megabytes. Lower values are better.

tested scenes. It would be interesting to further investigate different heuristics for selecting the amount of samples. In particular it would be possible to have quite different heuristics for selecting the sample size, and the grid size, rather than the simple multiple we use now.

Memory consumption (see Figure 5) is also higher for the rectilinear grid. This is due to the greater number of occupied cells, and large cell triangle lists.

Figures 6, 7, 8, 9 display the number of traversal steps required to render a given test scene. Pixels with a lighter tone of blue have more cell traversals. Pixels with a lighter tone of red have more triangle intersections. In this way it is easier to visualize the pros and cons of each acceleration structure.

5. CONCLUSIONS

We have proposed and implemented a rectilinear grid ray tracing algorithm. This algorithm has up to 79% better performance compared to a regular grid on the tested scenes using a similar grid resolution. It is especially well suited to render irregular scenes, which have typically been an issue with grid ray tracing.

Now that we can individually control the split plane positions, it should be easier to devise more sophisticated surface area grid heuristics based on previous work on BVHs and kd-trees.

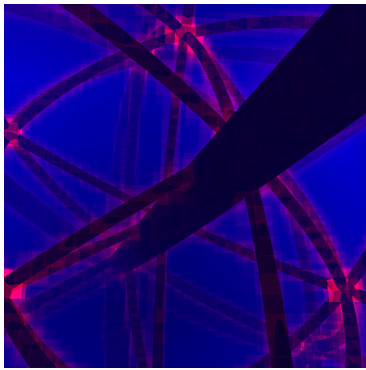
It should also be interesting to apply the scene sampling information generated during the construction stage to further construct a hierarchical data structure to speed up empty cell traversal. There are many other techniques to speed up empty cell traversal such as macro-regions [Dev89] and proximity clouds [CS94] which could prove useful for this purpose.

6. ACKNOWLEDGEMENTS

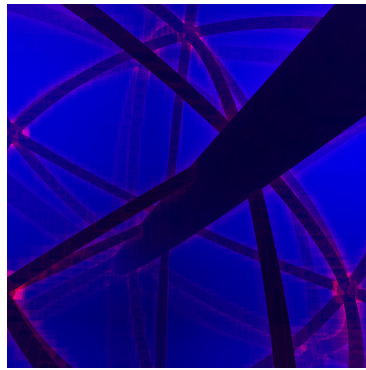
We would like to thank the Utah 3D Animation Repository for the AEK-24-cell and Fairy Forest scenes, the Stanford 3D Scanning Repository for the Buddha and Thai Statue scenes.

7. REFERENCES

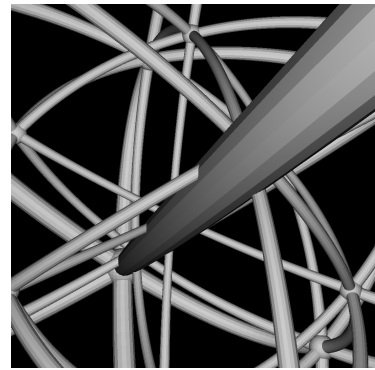
- [BSP06] J. Bigler, A. Stephens, and S.G. Parker. Design for parallel interactive ray tracing systems. In *Proceedings of the IEEE Symposium on Interactive Ray Tracing*, pages 187–195, 2006.
- [CP97] F. Cazals and C. Puech. Bucket-like space partitioning data structures with applications to ray-tracing. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 11–20. ACM New York, NY, USA, 1997.
- [CPJ10] V. Costa, J. Pereira, and J. Jorge. Multi-Level Hashed Grids for Ray Tracing. In *WSCG'2010*, 2010.
- [CS94] D. Cohen and Z. Sheffer. Proximity clouds - an acceleration technique for 3d grid traversal. *The Visual Computer*, 11(1):27–38, 1994.
- [CW88] J.G. Cleary and G. Wyvill. Analysis of an algorithm for fast ray tracing using uniform space subdivision. *The Visual Computer*, 4(2):65–83, 1988.
- [Dev89] O. Devillers. The macro-regions: an efficient space subdivision structure for ray tracing. In *Eurographics*, volume 89, pages 27–38, 1989.
- [FTI86] A. Fujimoto, T. Tanaka, and K. Iwata. Arts: Accelerated ray-tracing system. *Computer Graphics and Applications, IEEE*, 6(4):16–26, 1986.
- [HS99] V. Havran and F. Sixta. Comparison of hierarchical grids. *Ray Tracing News*, 12(1):1–4, 1999.
- [ISP07] T. Ize, P. Shirley, and S. Parker. Grid creation strategies for efficient ray tracing. In *Interactive Ray Tracing, 2007. RT'07. IEEE Symposium on*, pages 27–32, 2007.
- [JW89] David Jevans and Brian Wyvill. Adaptive voxel subdivision for ray tracing. In *Graphics Interface '89*, pages 164–172, June 1989.
- [KBB07] B. Kuehl, K.J. Blom, and S. Beckhaus. Generation of Shadows in Scene Graph based VR. In *WSCG'2007*, 2007.
- [KS97] K. Klimaszewski and TW Sederberg. Faster ray tracing using adaptive grids. *IEEE Computer Graphics and Applications*, 17(1):42–51, 1997.
- [LD08] Ares Lagae and Philip Dutré. Compact, fast and robust grids for ray tracing. *Computer Graphics Forum (Proceedings of the 19th Eurographics Symposium on Rendering)*, 27(8), 2008.
- [MT05] T. Möller and B. Trumbore. Fast, minimum storage ray/triangle intersection. In *International Conference on Computer Graphics and Interactive Techniques*. ACM Press New York, NY, USA, 2005.
- [PPL⁺05] S. Parker, M. Parker, Y. Livnat, P.P. Sloan, C. Hansen, and P. Shirley. Interactive ray tracing for volume visualization. In *ACM SIGGRAPH 2005 Courses*, page 15. ACM, 2005.
- [Wal07] I. Wald. On fast construction of sah-based bounding volume hierarchies. In *Interactive Ray Tracing, 2007. RT'07. IEEE Symposium on*, pages 33–40, 2007.
- [WH06] I. Wald and V. Havran. On building fast kd-trees for ray tracing, and on doing that in $O(N \log N)$. In *IEEE Symposium on Interactive Ray Tracing 2006*, pages 61–69, 2006.
- [Whi80] T. Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, 1980.
- [WMG⁺07] I. Wald, W.R. Mark, J. Gunther, S. Boulos, T. Ize, W. Hunt, S.G. Parker, and P. Shirley. State of the art in ray tracing animated scenes. *Eurographics 2007 State of the Art Reports*, 2007.



(a) regular grid



(b) rectilinear grid

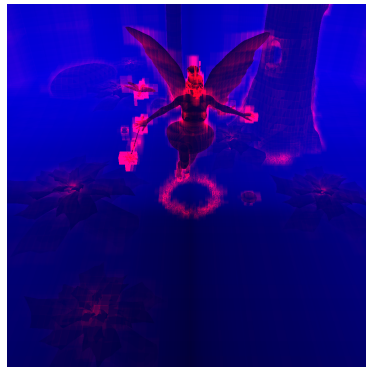


(c) rendered image

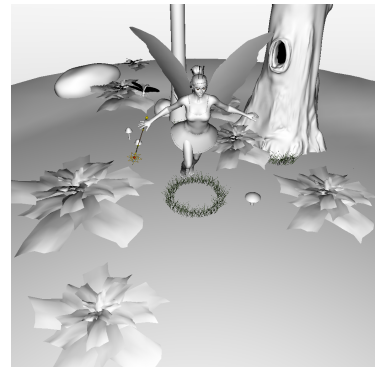
Figure 6. AEK-24-cell (122.88 ktri)



(a) regular grid

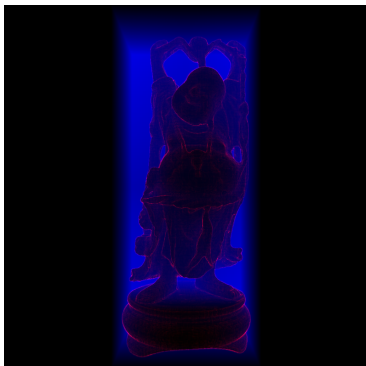


(b) rectilinear grid

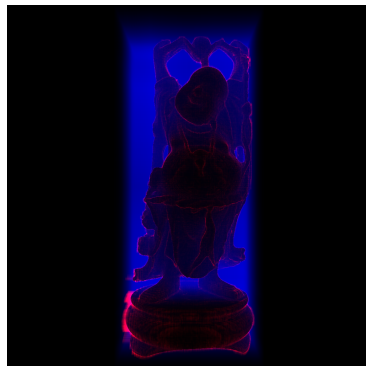


(c) rendered image

Figure 7. Fairy Forest (174.11 ktri)



(a) regular grid

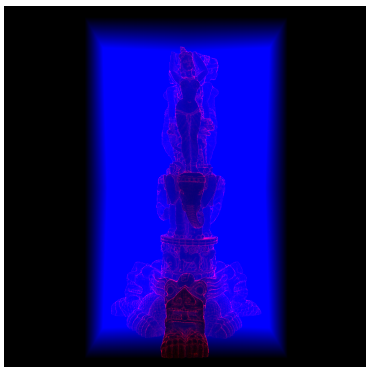


(b) rectilinear grid

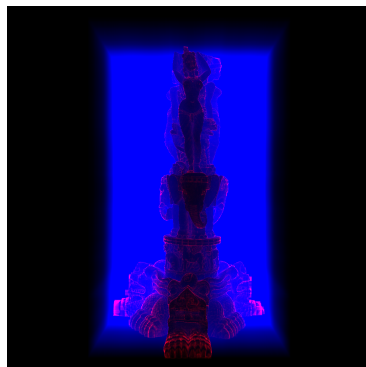


(c) rendered image

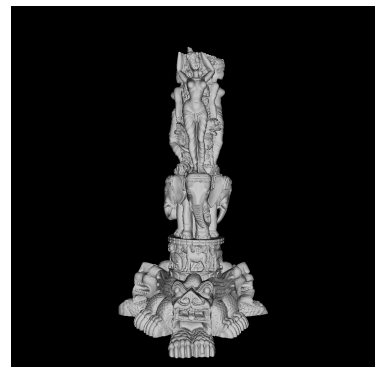
Figure 8. Buddha (1.09 Mtri)



(a) regular grid



(b) rectilinear grid



(c) rendered image

Figure 9. Thai Statue (10.00 Mtri)

Interactive Exploration and Analysis of Pathlines in Flow Data

Alan Lež

VRVis Research Center
in Vienna, Austria
lez@vrvvis.at

Andreas Zajic

VRVis Research Center
in Vienna, Austria
zajic@vrvvis.at

Krešimir Matković

VRVis Research Center
in Vienna, Austria
matkovic@vrvvis.at

Armin Pobitzer

University of Bergen, Norway
armin.pobitzer@uib.no

Michael Mayer

AVL List GmbH, Graz, Austria
michael.mayer@avl.com

Helwig Hauser

University of Bergen, Norway
Helwig.Hauser@UiB.no

ABSTRACT

The rapid development of large-scale scientific computing nowadays allows to inherently respect the unsteady character of natural phenomena in computational flow simulation. With this new trend to more regularly consider time-dependent flow scenarios, an according new need for advanced exploration and analysis solutions emerges. In this paper, we now present three new concepts in pathline analysis which further improve the abilities of analysis: a multi-step analysis which helps to save time and space needed for computation, direct pathline brushing, and the use of pre-configured view arrangements. We have found that a clever combination of these three concepts with already existing methods creates a very powerful tool for pathline analysis. A solution that follows the concept of coordinated multiple views (CMV) with iterative composite brushing enables a quick information drill-down. We illustrate the usefulness of this approach in the context of an example from the automotive industry.

Keywords: Interactive Visual Analysis, Pathlines, Flow Visualization, Exhaust Manifold

1 INTRODUCTION

Advances in hardware and simulation technology make it possible to simulate and visualize flows in time-dependent vector fields. In contrast to a steady state flow, in this case the unsteady vector field also changes over time itself. The analysis of time-dependent flows is far from trivial and still a largely unsolved problem in visualization [15]. There are many ways to visualize flows, and in the case of unsteady flow some very intuitive methods are based on pathlines. Pathlines describe paths of massless particles over time in the flow and thus the analysis of their behavior is tightly related to the analysis of the dynamic behavior of the underlying flow fields.

Interactive visual analysis (IVA) [19, 9] is useful to analyze large and complex data. As a set of pathlines represents such a case, we illustrate how IVA can help domain experts in engineering to analyze simulated time-dependent scenarios.

Our approach starts with the computation of pathlines and a set of pathline attributes. These attributes can be either scalar values or functions of time (time series). We have focused on attributes describing the (local or global) behavior of the pathlines. We focus on classical

and well-established attributes from vector algebra (for the unsteady flow field) or differential geometry (for the pathlines). The result of this step is a multivariate dataset collecting all computed pathlines and features and the attributes stored along them. The pathlines in the dataset are seeded at different points in time as the seeding time plays an important role in pathline characteristics.

Our IVA approach utilizes coordinated multiple views (CMV) with linking and brushing. Besides usual 2D views, used to visualize derived attributes, we added a 3D view which depicts the volume geometry and the pathlines themselves. To save space and time, as required by the computation and analysis, we introduce a multi-step analysis, where in one step a lower resolution (bigger cell size) dataset is investigated. In the next step a higher resolution (smaller cells) dataset is visualized and only spatio-temporal areas which were identified in the previous step as interesting are then analyzed further in more detail. This refinement can be repeated several times, but often two steps are already enough.

We also introduce the use of projections, as a way to do direct pathline brushing (a data visualization technique that identifies and highlights data subsets), and pre-configured view arrangements, to help the user with the screen space organization. All of this, combined into one tool, contributes new opportunities to the analysis of pathlines in flow data.

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.

We demonstrate the usefulness of our approach in the context of an exhaust manifold analysis. An exhaust manifold collects the exhaust gases from multiple cylinders into one pipe from an internal combustion engine. The goal of the exhaust manifold design is to decrease flow resistance (back pressure) and to increase the volumetric efficiency of the engine, resulting in a gain in power output or a reduction of fuel consumption. As a team of visualization researchers and a domain expert, we analyzed one such design together. The main goal was to check if the proposed design meets initial design requirements, and, if not, which directions for improvement could be suggested. We illustrate how the clever selection of pathline attributes can support domain experts in the analysis of time-dependent flows through the interactive visual analysis of pathlines and their attributes.

2 RELATED WORK

The idea to segment a flow domain into areas of certain flow properties has been used for 3D steady flow fields for a variety of features, including topological features [12, 26], or vortex regions [14]. Salzbrunn and Scheuermann [17] provide a general framework of this in the context of topological features. Laramée et al. [10] and Salzbrunn et al. [16] give an overview on flow visualization techniques focusing on feature extraction approaches. These techniques are used on 3D time-dependent fields by observing the feature regions over time, observing either topological features [23, 4], or vortex features, see Theisel et al. [22]. Although these approaches provide insight into the flow behavior at arbitrary time steps, the analysis of the dynamic behavior based on pathlines makes specialized approaches necessary. Wiebel and Scheuermann [28] visualize a number of carefully selected pathlines to get static representations of the dynamic flow. Theisel et al. [24] consider a segmentation of the flow domain based on local properties of the pathlines.

Combining several feature detectors in order to investigate combinations of them using IVA has been suggested by Bürger et al. [2]. Shi et al. [18] have a very similar approach to ours, and we build on that adding more attributes, the multi-step approach, (enabling us to work with much bigger datasets interactively), the use of projections as a method to enable direct pathline brushing, and pre-configured view arrangements for better screen space organization.

Texture-based visualization approaches to capture pathline characteristics are given by Weiskopf et al. [27]. The idea of connecting information visualization and scientific visualization approaches is considered to be one of the ‘hot topics’ in visualization [8].

Due to space limitations, we refer to Pobitzer et al. [15] for a more detailed discussion of related work.

3 EXPLORING PATHLINES

Advances in simulation and hardware technology make it now possible to simulate time-dependent flow fields, i.e., flow fields where the underlying vector field changes over time. The analysis of steady state flows (where the underlying vector field does not change) is a relatively mature research field with many good visualization and feature extraction methods. However, the extension to the unsteady state is not straightforward and is an active area of research [15]. The amount and complexity of data in an unsteady state case often grows to such an extent that the fully automatic methods are not sufficient any more. In order to efficiently cope with such a large and complex datasets, IVA approaches (such as presented by Konyha et al. [8]) balance human advantages (perception, experience, imagination) with those of automatic methods. The interactive visual analysis helps the user to be sure that nothing is missed and gives the user a better understanding of the flow by visualizing computed results and allowing interaction with them. It can also be used as a support tool to train novice engineers.

When analyzing the flow, experts are usually interested in problematic areas in the flow, which result, for example, in vortices and swirling. Such a behavior can be easily identified using pathlines and the attributes computed from them.

The main idea is to compute a set of pathlines, as well as a set of attributes for each pathline, and to use a coordinated multiple views system to analyze the data. The data in this case follows a more complex data model than the usual data model used in interactive visual analysis. As we want a pathline to be an entity, we build records in our dataset around pathlines. This means that one record contains pathline coordinates, various scalar attributes (such as, e.g., length, distance between start and end point, etc.), and various time series attributes (such as, e.g., velocity as a function of position along the pathline, curvature, etc.). Scientific data often follow such a data model (see Konyha et al. [9]), and IVA has been proven to be useful for analyzing such data. An IVA tool that supports such data has to be able to depict a family of curves, i.e., a certain time series attribute given for all pathlines, for example all velocity functions. Konyha et al. [9] introduced the curve view and the accompanying line brush which can be used to quickly drill down by simply selecting (brushing) or de-selecting curves that cross a line drawn by the user.

A state of the art CMV system which can show several views simultaneously and that also supports linking and brushing (selecting records in one view and highlighting corresponding items from the same records in all other views) is a prerequisite for such an approach. The pathlines data scenario, due to its complexity and

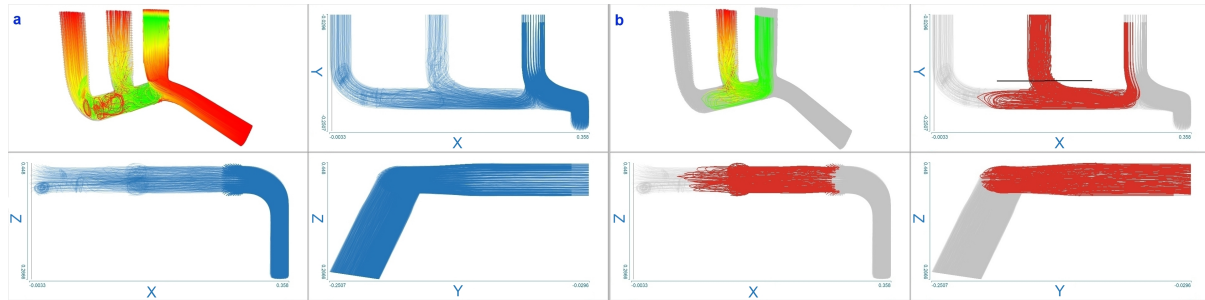


Figure 1: An example of using projection views in the pre-configured view arrangement, which the user can get with one click, to explore the pathlines (about 10000 pathlines in this case). The pre-configured view arrangement consists of a 3D view (top-left) and three projection views (top-right, bottom-left, and bottom-right views). Time is color mapped [13] in the 3D view going linearly from green, for the smallest, to red, for the biggest value, in this case. Other standard color mappings (e.g., rainbow color map, grayscale color map, cyan to mauve color map, etc.) are also available. **a:** 3D view and projection views. **b:** All pathlines passing through the middle tube are brushed in the top-right projection view.

size, requires an especially careful and innovative design to provide an effective exploration tool.

Three key aspects of improvement over the existing methods were identified: multi-step analysis, direct pathline brushing, and the usage of pre-configured view arrangements. The complete set of all pathlines would be really huge in most cases and it would often contain a lot of irrelevant data. There are areas in the flow where a domain expert can predict flow with a great confidence, and where no interesting features appear, so it is not meaningful to compute a huge number of pathlines in such areas. Our multi-step approach will help us to reduce the data size without losing important information. Furthermore, we need to depict pathlines themselves (in addition to their attributes) and be able to interact with them directly. The curve view shows families of planar curves, but the pathlines are curves in 3D space, so brushing them directly on the 2D screen is difficult. As a solution we present direct pathlines brushing (detailed further below). And finally, in order to allow the user to focus on the actual analysis, we have seen that pre-configured view arrangements are crucial. A clever combination of those three aspects supports an efficient time dependent flow analysis by means of IVA wrt. pathlines and their attributes.

In the next three subsections, each of the identified aspects is described in detail, and in the last subsection available pathline attributes are shortly covered.

3.1 Multi-Step Analysis

An important observation, when thinking about pathlines, is that, depending on the volume size and the resolution (cell size), the pathlines dataset can be very large. This can make IVA very slow. In order to efficiently cope with such large datasets a multi-step analysis approach is proposed. The main idea is to analyze a coarse, lower resolution dataset, and identify possible spatio-temporal (position in the volume and time steps)

areas of interest. In case of pathline analysis, the areas of interest will, for example, be the areas where pathlines pass through vortices. Those identified areas of interest are then analyzed in more detail using a higher resolution dataset. This process can be repeated as many times as necessary, but our experience says that often two steps are enough, with the exception of very big or very complex datasets. The identified areas can be further refined at any time and in any of the lower resolution datasets from the previous steps.

The multi-step analysis saves storage space and time needed for the computation and analysis depending on the resolution used and size of the areas of interest. To demonstrate it we tried to compute all pathlines (seeded at the areas of interest as identified by the domain expert) and 16 attributes (5 scalar and 11 time series attributes), on a high resolution (about 2 million cells) dataset for 25 time steps. As assumed, the progress was very slow. It took several hours to compute only about 20% of the pathlines and their attributes, and the produced dataset was over 20GB in size. It is obvious that, from IVA perspective, this is not the right way, so the computation was stopped. Such a large dataset contains many uninteresting pathlines and it unnecessarily uses computational and storage resources. The handling of such a large dataset can also be a problem. Our multi-step approach proved to be much more efficient. Using a lower resolution (36524 cells) dataset to compute pathlines (seeded in the same areas as before) and the same 16 attributes for 25 time steps resulted in a dataset of about 500MB in only a few minutes. After several areas of interest were identified interactively, in the second step a higher resolution (2 million cells) dataset was used to compute pathlines and the 16 attributes seeded only in those areas. This computation again took several minutes, only, and the result was less than 1GB in size. A more detailed IVA of these areas was then possible. The described process resulted in

datasets consisting of at most 10000 pathlines in this case.

There are some questions when using this multi-step approach about how to be sure we will not miss anything in the low resolution dataset, where to seed, etc. A careful decision making of the domain expert is required. With the knowledge of the dataset (how complex it is, how big the velocities are, etc.) a decision wrt. the resolution (minimizing the risk of missing important features in the flow) can be made without too much effort and in short time.

3.2 Direct Pathlines Brushing

Using attributes and composite iterative brushing, the user can find pathlines of unwanted (or wanted) behavior very easily, and drill down even from tens of thousands of pathlines to just one in just a few iterations (in many cases it can be done even in one). Still in some cases it would be very useful to be able to brush the pathlines directly. Immediately something like the already mentioned line brush from the curve view comes to mind. But since we are dealing with pathlines in 3D on the 2D screen this would be very hard to use. So the idea of doing orthogonal projections (onto X-Y, X-Z and Y-Z planes) and then using the line brush in the projection views was born. An example of how this looks can be seen in Figure 1. This way, the user can brush pathlines directly using line brushes the same way he/she would do it in the curve view for time series, and to be able to do it precisely, also, zooming is available. Projections also help in getting a better perspective of the position of some point in the 3D volume when looking at it on 2D screen, which in combination with 3D view's translation, rotation and zooming also gives user a better understanding of pathlines' behavior in 3D space.

3.3 Pre-configured View Arrangements

When doing interactive visual analysis very often multiple screen setups are used. The user can have a lot of views open and that can result in an unorganized screen space, where it is hard to focus on the actual analysis. To reduce this problem we propose using pre-configured view arrangements, which help user in working efficiently and focus on what he/she is actually searching for, and not on view arrangement and screen space organization. Identifying standard tasks and providing pre-configured view arrangements for solving those tasks should be used as often as possible. Also the user is given an option to create and edit pre-configured view arrangements. The type of views and their arrangement is chosen by the user and saved. New instance of the pre-configured view arrangements is then available with just one click on the appropriate toolbar button. To enable easier access to projections, pre-

Pathline Attributes		
Geometry Based	Scalar	Length
		Relative start-end distance
		Average particle velocity
		Seeding cell
		Seeding time
	Time Series	Curvature
		Torsion
		Winding angle
		Particle velocity
		Pathlines' coordinates
Vortical Structures	Field Based	Vorticity magnitude [20]
		Swirling strength [25]
		λ_2 [6]
		Hunt's Q [5]
	Core Detectors	Helicity method [11]
		Eigenvector method [21]
		Eigenvector method for unsteady flow [3]

Table 1: Table showing available pathline attributes and vortical structures. All vortical structures are saved as a time series along the pathlines.

configured view arrangement, which can also be seen in Figure 1, was created.

3.4 Pathline Attributes

We distinguish between two groups of attributes. Attributes computed from the pathlines' geometry (e.g., their curvature) and attributes computed from flow field characteristics along the pathline. For both groups we can have scalar attributes (e.g., the pathline length) or time series attributes (e.g., torsion along the pathline). For some of the time series attributes additional scalar aggregates, such as maximum or arithmetic mean values, are computed, depending on analysis questions. Table 1 shows attributes that we propose, and which proved to be very useful when used in different combinations. Of course any other attribute can be added depending on the task faced. Most of them are well known and for the description of some more complex ones the reader is referred to Jiang et al. [7].

All of the methods described in the last four subsections complement each other and, when combined intelligently, they form a powerful tool for pathline analysis. In the next section, a few examples are given of how this tool can be utilized and what can be found.

4 APPLICATION EXAMPLE

The dataset we studied in this application case is the result of a flow simulation of an exhaust manifold. The simulation was done using the AVL FIRE tool [1] and the result was a dataset consisting of 36524 cells for the first step (low resolution) and about 2 million cells for the second step of the analysis. This exhaust manifold is used in a six cylinder car, but we have only analyzed one side (exits from three cylinders) since the sides

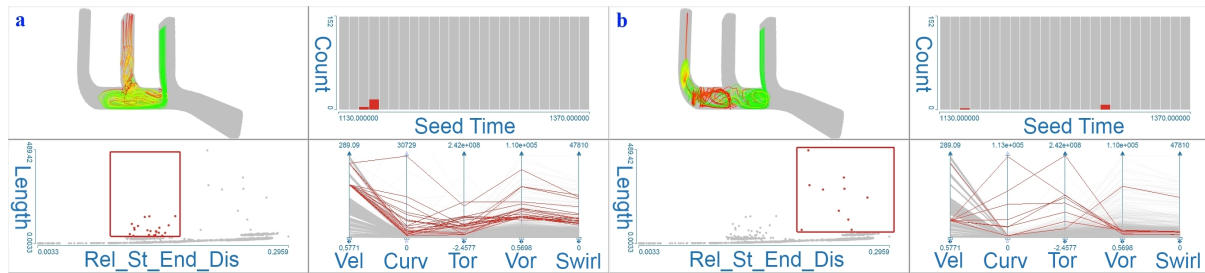


Figure 2: An example of data exploration using scalar attributes in histograms, scatterplots and parallel coordinates. A 3D view with color mapping enabled is used to depict pathlines. **a**: pathlines with medium relative start-end distances and high length values are brushed. In the 3D view we can see that those are the pathlines finishing somewhere in the left pipe. **b**: pathlines with high relative start-end distances and high length values are brushed and in the 3D view we can see that those are the pathlines finishing somewhere in the middle pipe.

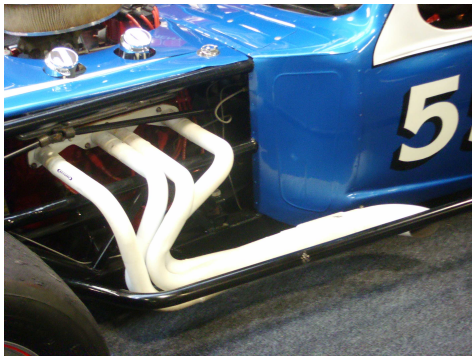


Figure 3: An example of a performance-car manifold. (Image courtesy of <http://www.zircotec.com>.)

are symmetrical. From the simulation dataset pathlines and their selected attributes are computed. The dataset proved to be very challenging because of its pulsing nature, where only one pipe (on one side) is active at specific time points and for only a short period of time. Also, as velocity values of the flow field are very high we had to pay special attention to how to compute pathlines, as they would otherwise leave the, in our case, relatively small volume very early. To cope with that, we decided to use cellular integration instead of the one with a constant time step. Cellular integration iterates cell by cell and builds a pathline, thus making sure that all the vectors of the flow field along the pathline way are taken into account. This frees the user of the sometimes difficult decision of which integration time step to use, especially in cases of very high velocities. The resulting datasets consisted of up to 10000 pathlines and their attributes (5 scalar and 11 time series attributes).

4.1 Exhaust Manifold

Every internal combustion engine has an exhaust manifold. It is generally a simple unit that collects engine exhaust gases from multiple cylinders and delivers them to the exhaust pipe. Each cylinder has its own exhaust head-pipe, and they usually converge into one tube called a collector.

When an engine starts its exhaust stroke, the piston moves up the cylinder bore, decreasing the total chamber volume, which increases the pressure in the cylinder, and when the cylinder's valve opens, the high pressure exhaust gas exits into the exhaust manifold, creating an exhaust pulse comprising three main parts. They are: high-pressure head, medium-pressure body, and the low-pressure tail component. The momentum of the high and medium pressure components reduces the pressure in the combustion chamber to a lower than atmospheric level. This relatively low pressure helps to extract all the combustion products from the cylinder and induct the intake charge during the overlap period when both intake and exhaust valves are partially open. The effect is known as scavenging. Length, cross-sectional area, and shape of the exhaust ports and pipeworks influence the degree of scavenging effect, and the engine speed range over which scavenging occurs.

Selecting the length and diameter of the primary tubes must be done very carefully depending on what we want to accomplish (more power, lower fuel consumption, ...).

4.2 Identifying Areas of Interest

To get a better overview over the time series attributes some first order statistics (mean, max, ...) were computed. These newly computed scalars were depicted using scatterplots, parallel coordinates, and histograms. Also we had the 3D view active showing pathlines with color mapping [13] (mapping of the colors to the values) enabled, so we could see if the features found in other views as interesting are the result of the interesting behavior of the pathlines. For color mapping we use time, particle velocity, or any other time series attribute of the ones listed in Table 1. The flow field features stored as time series along the pathline proved to be very useful, since they give us an insight into what made pathlines behave as they do.

One such example can be seen in Figure 2. In the scatterplots (lower-left views), depicting relative start-

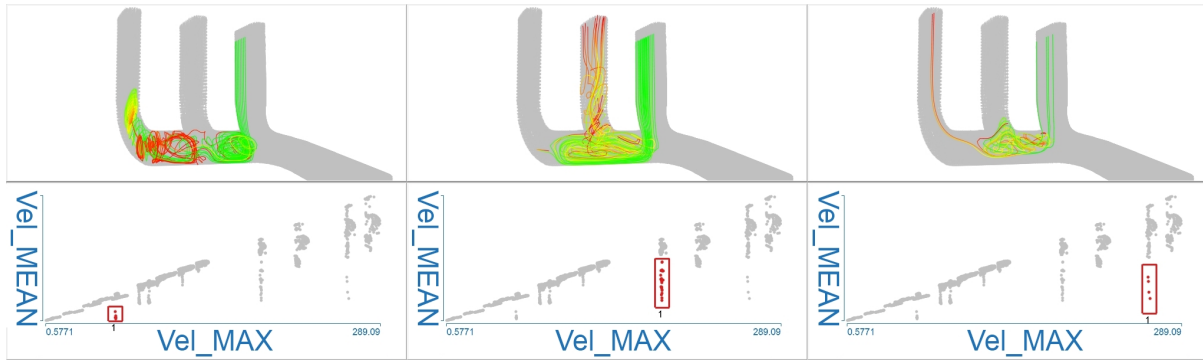


Figure 4: One more example of using scalar attributes to identify areas of interest. Attributes depicted in scatter plots are particle velocity mean and maximum. We can clearly see how brushing different clusters in the scatter plot views (bottom views) results in pathlines of different behavior being selected (top views). Also similarities within the groups are very obvious.

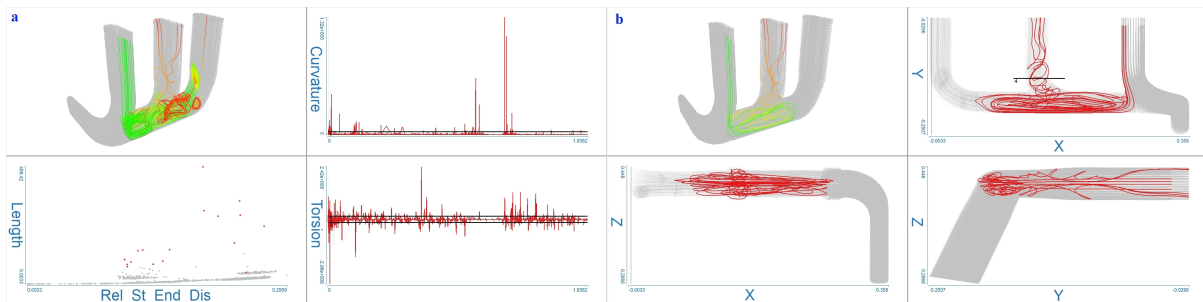


Figure 5: **a**: Finding “problematic” pathlines using composite brushing in two curve views depicting curvature and torsion. High curvature and torsion, as a stable indicator of pathlines forming a vortex somewhere along its way, are brushed. This way almost the same pathlines are selected as in previous two cases, but with less control (there are no clusters, i.e., groups are not as easy to identify). **b**: using pre-configured view arrangement for projections to brush the pathlines going to the middle tube. We add one more brush to the composite brush from a, thus getting better control over what is selected.

end distances and length attributes, clusters are clearly recognizable, so we investigated what we get when we brush some of them. The most interesting proved to be clusters with relatively high length values. We have selected pathlines with high length, and medium (Figure 2a) and high (Figure 2b) distance between start and end. There are other pathlines that traveled the same distance (gray points in the scatterplots below the brush), but they have shorter length. The higher length of the brushed pathlines is the result of vortices along their way, and this is confirmed when looking at the parallel coordinates view with relatively high maximums of torsion, curvature, vorticity magnitude, and swirling strength values. As, in the case of an ideal exhaust manifold, gases would travel as fast as possible to the exhaust, this is certainly an unwanted behavior.

Continuing the exploration, we investigate other attribute combinations which support the analyst in getting insight into the flow. Having multiple ways to identify possible problems gives us a better chance to identify all of them. Since different attribute combinations (both scalar and time series) are used in these approaches, it also gives us a better understanding of

the attributes. We can find correlations between them, and thus get better insight into the dataset.

Another such example, using scalar attributes, but in a different combination, can be seen in Figure 4. We use average and maximum particle velocity attributes. As can be seen, using a different combination does result in a different clustering. In this case, clusters are even more easy to notice and brushing them reveals groups of pathlines, with the behavior of the pathlines being very similar within a group, which was not always the case in the previous approach, where these three groups were mixed into two. So it seems that we have found a better way (that gives us better control) for finding pathlines of interest.

Using time series attributes we get a similar result, as can be seen in Figure 5a. Two curve views are used to depict time series attributes, which are in this case curvature and torsion. Again almost the same pathlines are selected, but in this case we do not have as good control over what is selected as in previous cases. To gain better control we can use the projection views with their direct pathline brushing ability, as can be seen in Figure 5b.

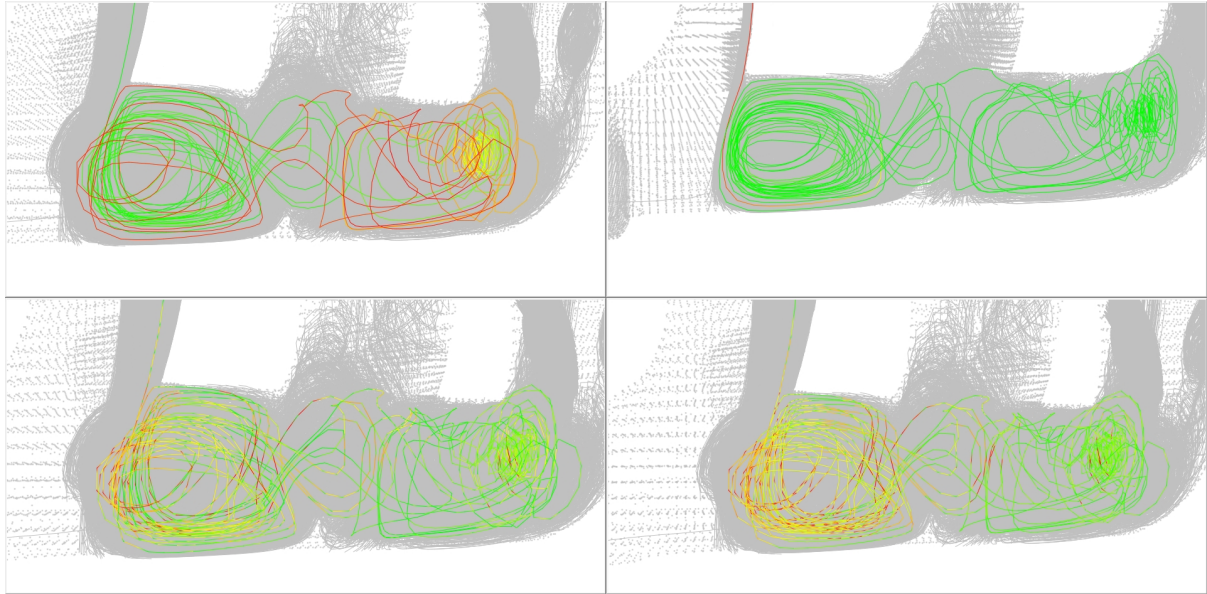


Figure 6: Analysis of the particle velocity along the pathline. We use one pathline with four attributes color mapped: time (top-left), particle velocity (top-right), swirling strength (bottom-left), and vorticity magnitude (bottom-right). Color mapping is done from green to red, green representing minimum value and red representing maximum value.

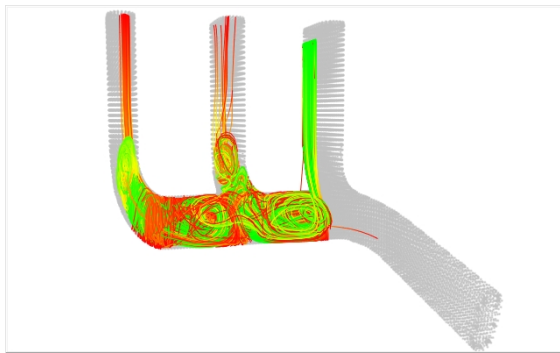


Figure 7: Result of the higher resolution simulation for the seed cells of pathlines brushed in Figure 2b. Immediately it is obvious that, unlike in lower resolution case, here we have some pathlines ending in the middle pipe.

What can be seen in all cases is that none of the brushed pathlines goes out through the expected pipe, but instead they finish in the left or middle pipe, which is undesirable. Also, almost all of these pathlines are seeded or pass near the edges of the pipes. The according histogram views show us that this happens only in few time steps, and for a small amount of cases, but still this could be an indication of a problem in the exhaust manifold. To gain a better understanding why is this happening we use the finer (higher resolution) dataset for those time steps and seeding points and then do a more detailed analysis to see what can be done, if anything, to reduce them even more.

4.3 Detailed Analysis

For the detailed analysis based on the high-resolution dataset, pathlines are computed only in the spatio-temporal areas of interest identified in the previous step. What becomes immediately obvious, when looking at Figure 7, showing a 3D view depicting pathlines computed in the higher resolution dataset, but only for the seed cells of pathlines brushed in Figure 2b, is that, unlike in lower resolution case, now we have pathlines ending in the middle pipe also. This shows how complex the flow is and that very small changes can drastically change the result.

Next we focus on one interesting pathline from the pathlines shown in Figure 7. Using color mapping with time series attributes and also different types of views with attributes we try to discover as much as we can about the behavior of that pathline, i.e., the behavior of the particle in the flow. Figure 6, which shows four 3D views depicting the same pathline, but with different attributes color mapped, shows an example of such an analysis. Our focus here was on particle velocity (color mapped in the top-right view in the figure). We can see that the particle velocity dropped very rapidly once it started changing direction, and the reason that it did change direction is because it entered areas of high swirling strength and vorticity magnitude which can be seen in the two bottom views.

With this we have shown an example of how our tool can be used in the analysis of the exhaust manifold dataset. The domain expert found it very interesting, and thinks it has a great potential, especially when used

in combination with other tools, and we will continue our collaboration.

5 CONCLUSION AND FUTURE WORK

In this paper we have shown how pathlines in combination with a carefully designed tool can give a user new abilities in analyzing the flow field. Pathlines and their attributes are used to help identify possible problems in the exhaust manifold which can cause loss of engine power and increase of fuel consumption.

We show how computing power limitations can be avoided by first using lower resolution to identify spatio-temporal areas of interest and then using higher resolution for detailed analysis of those areas. Furthermore, direct pathline brushing by using projections is introduced. Also, usage of pre-configured view arrangements, to help with screen space organization, is proposed.

The domain expert gave us positive feedback and we plan to continue our work on this subject in the future, specially focusing on the selection of the attributes available and combining our tool's results with other tools. We think that our tool is a step forward and can lead to further developments in this area.

6 ACKNOWLEDGMENTS

The project SemSeg acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number 226042.

REFERENCES

- [1] Avl fire. <https://www.avl.com/fire>, 2010.
- [2] R. Bürger, P. Muigg, H. Doleisch, and H. Hauser. Interactive cross-detector analysis of vortical flow data, 2007.
- [3] R. Fuchs, R. Peikert, H. Hauser, F. Sadlo, and P. Muigg. Parallel vectors criteria for unsteady flow vortices. 14(3):615–626, 2008.
- [4] C. Garth, X. Tricoche, and G. Scheuermann. Tracking of vector field singularities in unstructured 3d time-dependent datasets. In *VIS '04: Proceedings of the conference on Visualization '04*, pages 329–336, Washington, DC, USA, 2004. IEEE Computer Society.
- [5] J. C. R. Hunt, A. A. Wray, and P. Moin. Eddies, stream and convergence zones in turbulent flows. In *2. Proceedings of the 1988 Summer Program*, pages 193–208, 1988.
- [6] J. Jeong and F. Hussain. On the identification of a vortex. *Journal of Fluid Mechanics*, 285:69–84, 1995.
- [7] M. Jiang, R. Machiraju, and D. Thompson. Detection and visualization of vortices. In *The Visualization Handbook*, pages 295–309. Academic Press, 2005.
- [8] C. Johnson. Top scientific visualization research problems. *Computer Graphics and Applications, IEEE*, 24(4):13–17, July–August 2004.
- [9] Z. Konyha, K. Matković, D. Gračanin, M. Jelović, and H. Hauser. Interactive visual analysis of families of function graphs. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1373–1385, 2006.
- [10] R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post, and D. Weiskopf. The state of the art in flow visualization: Dense and texture-based techniques. *Computer Graphics Forum*, 23(2):203–221, 2004.
- [11] Y. Levy, D. Degani, and A. Seginer. Graphical visualization of vortical flows by means of helicity. *AIAA Journal*, 28:1347–1352, August 1990.
- [12] K. Mahrous, J. Bennett, G. Scheuermann, B. Hamann, and K. I. Joy. Topological segmentation in three-dimensional vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):198–205, 2004.
- [13] K. Moreland. Diverging color maps for scientific visualization. In *ISVC (2)*, pages 92–103, 2009.
- [14] R. Peikert and M. Roth. The “parallel vectors” operator: a vector field visualization primitive. In *VIS '99: Proceedings of the conference on Visualization '99*, pages 263–270, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.
- [15] A. Pobitzer, R. Peikert, R. Fuchs, B. Schindler, A. Kuhn, H. Theisel, K. Matkovic, and H. Hauser. On the way towards topology-based visualization of unsteady flow - the state of the art. In *EuroGraphics 2010 State of the Art Reports (STARs)*, 2010.
- [16] T. Salzbrunn, C. Garth, G. Scheuermann, and J. Meyer. Path-line predicates and unsteady flow structures. *Vis. Comput.*, 24(12):1039–1051, 2008.
- [17] T. Salzbrunn and G. Scheuermann. Streamline predicates. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1601–1612, 2006.
- [18] K. Shi, H. Theisel, H. Hauser, T. Weinkauff, K. Matkovic, H.-C. Hege, and H.-P. Seidel. Path line attributes - an information visualization approach to analyzing the dynamic behavior of 3D time-dependent flow fields. In Hans-Christian Hege, Konrad Polthier, and Gerek Scheuermann, editors, *Topology-Based Methods in Visualization II*, Mathematics and Visualization, pages 75–88, Grimma, Germany, 2009. Springer.
- [19] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proc. of the 1996 IEEE Symp. on Visual Languages*, page 336, 1996.
- [20] R.C. Strawn, D. Kenwright, and J. Ahmad. Computer visualization of vortex wake systems. In *American Helicopter Society 54th Annual Forum*, 1998.
- [21] D. Sujudi and R. Haimes. Identification of swirling flow in 3D vector fields. Technical Report 95-1715, AIAA, 1995.
- [22] H. Theisel, J. Sahner, T. Weinkauff, H.-C. Hege, and H.-P. Seidel. Extraction of parallel vector surfaces in 3d time-dependent fields and application to vortex core line tracking. In *IN PROC. IEEE VISUALIZATION 2005*, pages 631–638, 2005.
- [23] H. Theisel and H.-P. Seidel. Feature flow fields. In *VISSYM '03: Proceedings of the symposium on Data visualisation 2003*, pages 141–148, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.
- [24] H. Theisel, T. Weinkauff, H.-C. Hege, and H.-P. Seidel. Topological methods for 2d time-dependent vector fields based on stream lines and path lines. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):383–394, 2005.
- [25] D. S. Thompson and C. H. Berdahl, editors. *Eduction of swirling structure using the velocity gradient tensor*, June 1991.
- [26] T. Weinkauff, H. Theisel, H.-C. Hege, and H.-P. Seidel. Boundary switch connectors for topological visualization of complex 3d vector fields. In *In Proc. VisSym 04*, pages 183–192, 2004.
- [27] D. Weiskopf, F. Schramm, G. Erlebacher, and T. Ertl. Particle and texture based spatiotemporal visualization of time-dependent vector fields. In *IEEE Visualization*, page 81, 2005.
- [28] A. Wiebel and G. Scheuermann. Eyelet particle tracing - steady visualization of unsteady flow. In *IEEE VISUALIZATION 2005*, pages 607–614. IEEE Computer Society, 2005.

ARGreenet and BasicGreenet: Two mobile games for learning how to recycle

M. Carmen Juan; David Furió
Instituto ai2
Universitat Politècnica de
València
Camino de Vera, s/n. 46022
Valencia, Spain

Leila Alem; Peta Ashworth
CSIRO ICT Centre
PO Box 76
Epping NSW 1710

Juan Cano
Escola d'Estiu
Universitat Politècnica de
València
Camino de Vera, s/n. 46022
Valencia, Spain

ABSTRACT

In this paper, a new Augmented Reality (AR) mobile phone game 'ARGreenet' is presented. The game aims to raise individuals' awareness of the importance of recycling and teaching participants how to do it. In this research, the 'ARGreenet' is compared with a similar 'Basic' mobile phone game for recycling. Thirty eight children aged from 8 to 13 years of age participated in this study. To quantify aspects of the utility and effectiveness of the games, the children answered questionnaires both before and after using each game. Aspects examined included the level of engagement and fun of each game, the ease of use and perceived value of each game, and the perceived learning about recycling. We report a positive change in intended behavior with both games. The results suggest that playing both games is likely to have a positive influence in changing participants' recycling behaviour. These preliminary results also suggest that the mobile phone is potentially a good platform for not only learning about recycling but also influencing people to change their behaviour. A majority of the participants expressed a preference for ARGreenet game. They perceived it as easy to use and more engaging and fun than the BasicGreenet game.

Keywords

Augmented Reality, mobile phones, recycling, behaviour change, learning.

1. INTRODUCTION

It is now widely accepted that anthropogenic actions are a major cause of the rising CO₂ levels in the earth's atmosphere (IPCC, Fourth Assessment Report. Intergovernmental Panel on Climate Change, 2007). Linked to this are human consumption patterns that generate enormous volumes of waste, particularly in developed countries. The waste problem has been recognised by world leaders for some time. Significant recognition was given to the problem as part of the Agenda 21 for sustainable development – an action plan devised at the Earth Summit held in Rio de Janeiro (1992). In 2005, the European Landfill Directive sets targets to minimise waste to landfill through increased levels of recycling and recovery and the EU's Sixth Environment Action Programme identifies waste prevention and

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.

management as one of its top four priorities. As a result all member states of the European Union are implementing a number of waste management systems.

This paper reports on an Augmented Reality (AR) mobile phone game which aims to educate the user on how to recycle their waste effectively. AR refers to the introduction of virtual content into the real world. The AR game (ARGreenet) is presented alongside a basic game (BasicGreenet) that shared the same purpose. The aim of the research was to test the hypothesis that the ARGreenet would have greater influence on a number of variables than the BasicGreenet.

2. PREVIOUS WORK

Augmented Reality

AR systems running on PCs have been designed for their application in many fields including: medicine; military; robotic; maintenance and repair applications; learning; entertainment or edutainment; [Azu97] [Azu01]. However, with the advent of portable computers and notebooks, mobile AR became possible and later, different applications for PDA's and mobile phones were developed for their

application in several fields such as learning (e.g. [Liu07] [Wan09]), edutainment (e.g. [Wag07a]), etc.

Related to learning, systems for learning subjects as different as English [Liu07] or heritage temples [Wan09] have been presented. Li et al. [Liu07] developed a handheld AR system for learning English called HELLO based on 2D barcodes. A four-week pilot study was conducted and the results indicated that 2D barcodes and AR were useful for English learning. Wang et al. [Wan09] tested three user interface prototypes for learning heritage temples. Their study showed that users preferred animated and interactive virtual elements with sound effects, and that the superimposed information should not cover more than 30% of the screen.

Related to edutainment (term that points out the connections and the positive correlations between the educational field and the entertainment one), an example could be the Virtuoso Arts History Game [Wag07a]. It is a collaborative educational game for up to four players. The players' objective is to sort a collection of artworks. A virtual animated character called Mr. Virtuoso can provide help for players that are stuck. Another example could be Alien Contact! [Osh09] that was the first game developed in the Handheld Augmented Reality Project (HARP), <http://isites.harvard.edu/harp>. In Alien Contact!, participants use GPS-enabled handheld computing units. Alien Contact! is based on a scenario where aliens have crash landed. Students work in teams, and learn math and literacy skills.

Finally, with the appearance of the iPhone, different AR applications have been presented for this device. Several of them can be downloaded from the Apple Store.

Enhancing recycling behaviours

As this paper focuses on bringing about behaviour change it is useful to examine the theoretical constructs which helped to inform the research design of the AR waste management game. The Theory of Reasoned Action is a useful theoretical construct for designing processes to enhance recycling behaviours because it is strongly dependent on the concept of behavioural intention – the commitment to a certain action or behaviour [Ajz80]. The theory asserts that behaviour is a deliberate act based on the beliefs of the individual and the norms imposed by society [Ton04]. Therefore, when an individual is positively predisposed toward a particular behaviour, and when they perceive support for that behaviour from people around them, then they will form a positive behavioural intention towards that behaviour. Behavioural intention, in turn, leads to actual performance of the relevant behaviour [Ajz80].

In the context of recycling behaviour, over the past decade there has been an increasing expectation for individuals to recycle their household rubbish. That is, a subjective norm exists that recycling is a reasonable thing to do, but this is unlikely on its own to produce recycling behaviour. However, if individuals also hold a positive attitude towards recycling they are more likely to actually perform the behaviour [Gar08].

The Theory of Reasoned Action was later modified to Ajzen's Theory of Planned Behaviour [Ajz91]. This theory incorporates the person's belief about how easy or difficult it is to perform a specific behaviour, based on their abilities, opportunities and resources [Gar08]. Because recycling requires enormous individual effort it is helpful to understand which characteristics will help them to make the decision to recycle more often [Bol95]. Within this research we examined participants' knowledge and attitudes towards recycling and the environment to identify if this impacted on their ability to perform the recycling behaviours required in the ARGreenet and BasicGreenet.

3. DESCRIPTION OF THE GAMES

ARGreenet

The objective of ARGreenet is that participants learn how to recycle effectively. ARGreenet uses markers (a white square with a black border containing symbols or letters). The player has to pick up objects that appear over the objects' marker and place them in the correct recycling bin. Only one object appears over the objects' marker at a time (e.g. Figure 1), but this object will vary at different stages of the game. The recycling bins appear over four different markers, with the following letters in their interior: A, B, C and G (e.g. Figure 2). The markers are independent and are placed over a table by the person in charge of the experiment. The markers can be placed on the floor or in any desired place. There are three different levels within the game. In the first level only two recycling bins and 2 objects randomly selected among 6 possibilities for each type of rubbish that appears. That is, over the objects' marker only two different types of rubbish appear and only two recycling bin markers are used, A and B. The two possible recycling bins appear over these markers (one recycling bin over A marker and the second recycling bin over B marker). In the second and third level more recycling bins and more objects appear, specifically, 3 and 4 recycling bins and 4 and 6 objects, respectively. That is, in the second level three recycling bin markers are used, A, B and C, and in the third level the four recycling bin markers are used. When the player correctly places the rubbish they are rewarded by the game showing two hands

applauding over the recycling bin. If the participant wrongly places the object, the game shows a red cross over the recycling bin.

The game applies the usual rules for games. A player gains or loses points for correctly or incorrectly recycling or leaving the rubbish outside the recycling bins. If the object has been correctly placed then the player gains points while on the contrary, the player loses points if incorrect. If the player is unsure about the correct recycling bin for a type of rubbish, he has the possibility to place the rubbish outside all recycling bins. In this case, the game subtracts points, but less than incorrectly placing the rubbish. The game goes to the next level when the player has achieved a fixed number of points for each level.

The game also has an allocated time for each level, if the player finishes before the allocated time, he gains 5 points for each second left. The game also includes a number of questions in each level that are randomly selected in each run. These questions are also related to recycling. The questions offer three possible answers of which only one is correct. The player has to choose among these three options. Again, the player gains points if he answers correctly or loses points if he answers incorrectly. The questions are stored in an XML file which facilitates the inclusion and removal of questions. The game records the top ten players' names which are stored in a file and can be consulted as an option of the game. The game also includes a help option where all the rules of the game are explained. Figures 1 and 2 show two images of the game. In Figure 1 is possible to see rubbish (a cardboard box) over the object marker. Figure 2 shows a step of the game where the player has placed rubbish over the correct recycling bin.

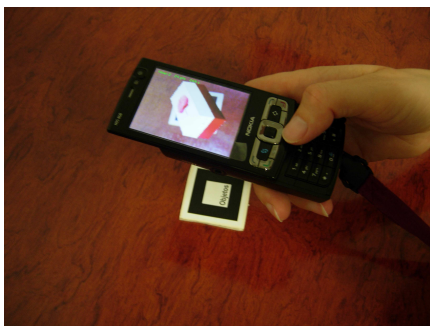


Figure 1. ARGreenet. Player is picking up a residue

For hardware the only required device is a mobile phone. This research used the Nokia N95 with 8GB. The most outstanding features of this mobile phone for AR are: Large 2.8" QVGA (240 x 320 pixels); Carl Zeiss Optics camera with 5 Megapixels; VGA video capture of up to 30 frames per second.

The software was distinguished, firstly, between the required development environment and the additional software for programming for the selected mobile phone in C++; and secondly, the library for the AR facilities. This research developed a system using Microsoft Visual Studio 2005. But for running an application simulating its running in the selected mobile phone, an emulator of the mobile phone is required. For including this type of tools the S60 Platform SDK, 3rd Edition was used. For programming in C++ for Symbian OS it is also required the installation of Carbide.vs 3.0.1.



Figure 2. ARGreenet. Player has correctly placed a residue

For AR facilities the ARToolKit 2.65 [Kat99] was ported onto a mobile phone running on Symbian OS and Series60. In 2003, Wagner & Schmalstieg [Wag03] ported ARToolKit to Windows CE. In 2005, Henrysson et al. [Hen05] ported ARToolKit to the Symbian platform. Later, in 2007, Wagner & Schmalstieg [Wag07b] presented the ARToolKitPlus library for its use on mobile devices (e.g. PDA's). Studierstube Tracker [Sch07] was a posterior version of ARToolKitPlus. Another framework presented by Wagner and colleagues was Studierstube ES [Wag09]. For developing our AR library, we studied two possibilities. The first one was to port the well-known ARToolKit to mobile phones and later to incorporate additional functionalities. This portability has already been achieved successfully [Hen05] [Wag07b]. Therefore, we were sure it was possible. These were the only two previous experiences when we started our work and Henrysson et al.'s library was not freely available. The second one was to use the ARToolKitPlus and incorporate to it additional functionalities. The result of both developments should be similar. We decided to choose the first possibility because of our earlier experiences in modifying ARToolKit, for having more knowledge about the code and for possible improvements.

BasicGreenet

The objective of the BasicGreenet is the same as the ARGreenet. The recycling bins appear on the lower part of the screen and rubbish goes down from the top

of the screen. There are three different levels within the game as in ARGreenet. In the first level only two recycling bins appear in the lower part of the screen, and 2 objects randomly selected among 6 possibilities for each type of rubbish go down from the top of the screen. In the second and third level more recycling bins and more objects appear, specifically, 3 and 4 recycling bins and 4 and 6 objects, respectively. The player has to correctly place rubbish into the correct recycling bin by pressing the left/right keys on the mobile phone. If the player wants the rubbish to go down quickly, he can press the down key. If the player is not sure about the type of rubbish and the correct recycling bin to place it, he can place the rubbish outside any of the recycling bins. In this case, the game does not decrease points. The rules of the game and the points that the player gains or loses based on their different actions, are similar to ARGreenet described above. However, in this case the two animations for placing correctly/wrongly rubbish are not used. Instead of this, in the top left of the screen appears the level; in the top centre of the screen appears the consumed time in seconds of the current level; and in the top right appears the score achieved in the current level. Several question about recycling appear after the player has achieved an already - established score. If the player correctly places rubbish or answers correctly a recycling question, the score increases, if not, the score decreases. Figure 3 shows an image of this game in which it is possible to see the third level (recycling an apple core).

As hardware, the only required device is the same mobile phone that was used in the ARGreenet, the Nokia N95 with 8GB. In relation to the software NetBeans IDE 6.0.1 was used as the development environment. The language used for the development was Java, J2ME. The plug in Java ME Wireless Toolkit for CLDC was incorporated into the development environment to provide the required classes for loading/writing files in mobile devices.



Figure 3. BasicGreenet. Third level

4. EXPERIMENTAL DESIGN AND MEASURES

The research experiment involved 38 children engaged in playing both the ARGreenet and BasicGreenet. All participants experienced both games but in a different order, with one group of participants experiencing the ARGreenet first, while the second group experienced the BasicGreenet first. Each group had 19 participants.

The research involved the children firstly completing an entry questionnaire (Table 1). This questionnaire includes questions about mobile phone experience, gaming experience, knowledge of recycling, beliefs about the environment/attitudes, behaviours, and intended behavior/motivation to change. In order to familiarize students with the elements that appear in either game the children would then spend time examining a page where the type of recycling boxes with their corresponding rubbish are shown. Once this was done the children participated in their first game, either ARGreenet or BasicGreenet, and completed a post questionnaire (Table 2). The students then repeated the process by familiarizing themselves with the elements of the second game, using the game and completing another post questionnaire. After playing the ARGreenet, the children were asked to complete two questions around presence which were “I had a sense of being in the room where there are rubbish and recycling boxes” and “There were times during the experience when I thought that objects and images were in the room, over the table or over my hand”. After playing both games, the children were asked to complete a final questionnaire (Table 3).

Quantitative data was collected using questionnaires. Because the target age group was young (< 15years) the questionnaire was kept short. All questions were measured on a 7 point Likert scale where in most cases 1 = none and 7 = a great deal. In the case where the meaning of 1-7 was different, the meaning is referred to in the related question.

In addition to basic demographic data including age and gender, there were a number of questions to investigate individuals’ experiences with mobile phones and the phone being used in the trial, followed by a question about the students’ levels of experience with gaming devices. Informed by the theories of reasoned action [Ajz80] and planned behaviour [Ajz91], further questions were asked to ascertain levels of participant knowledge of recycling, their attitudes towards recycling and the environment, current recycling behaviours and their perceived willingness to change the behaviours. The post game and final questionnaires are presented in Tables 2-3, with different aspects identified by different white/grey background colours.

Quest. ID	Questions	Mean(SD)
E1	Mobile phone experience How much experience do you have using mobile phones	3.87(1.42)
E2	Please indicate your level of expertise with the Nokia N95 phone	1.05(0.23)
E3	Gaming experience How much experience do you have in playing games on a PC or mobile phone?	4.74(1.74)
E4	Knowledge of recycling How much do you know about what can be recycled and how to recycle?	4.97(0.91)
E5	How much do you know about the effect of recycling on your environmental footprint?	4.00(1.59)
E6	Please indicate your level of expertise in what recycling is and why recycling is important: (1-Novice, 7-Expert)	4.68(1.21)
E7	Beliefs about the Environment/Attitudes People should be recycling more in order to reduce their environmental footprint	6.47(0.92)
E8	Behaviors I recycle my garbage and separate the cans, the bottles, newspapers etc.	5.18(1.18)
E9	Intended behaviour/Motivation to change I am willing to taking new actions to improve my recycling behaviour. (1-would not accept, 7-would accept)	5.92(1.32)

Table 1. Entry questionnaire

Quest. ID	Questions
P1	Engagement and fun I enjoyed playing this game.
P2	This game was fun
P3	Easy to use Please indicate if the game has been easy to play (1-not easy, 7-very easy)
P4	Perceived value I think playing this game could help me better recycle
P5	I would be willing to play this game again because it has some value to me
P6	Attitudes How strongly do you agree with the following statement? People should be recycling more in order to reduce their environmental footprint. (1-strongly disagree, 7-strongly agree)
P7	Intended behaviour/Motivation to change I am willing to taking new actions to improve my recycling behaviour. (1-would not accept, 7-would accept)
P8	Intention to change As a result of playing this game I will talk to my friends and family members about recycling.
P9	As a result of playing this game I will think more about recycling and its effect on the environment.
P10	As a result of playing this game I will make changes to my current behavior

Table 2. Post questionnaire

Quest. ID	Questions
F1	Perceived learning about recycling Please indicate the number that most closely describes how much you think you have learned as a result of playing these games How did you learn about what can be recycled and how to recycle? (1-nothing, 7-very much)
F2	Please indicate your level of expertise about the rubbish you can recycle as a result of playing these games (1-Novice, 7-Expert)
F3	Preference Which game did you like the most? Normal game: AR game:
F4	Why?. Any comment that you like to add
F5	Any comment that you like to add

Table 3. Final questionnaire

Question	ARGreenet Post-test	BasicGreenet Post-test	t	p
P1	6.40(0.89)	6.29(0.80)	0.702	0.487
P2	6.18(1.01)	6.05(0.96)	0.777	0.442
P3	6.26(0.89)	5.79(1.19)	2.303**	0.027**
P4	6.53(0.80)	6.42(0.79)	1	0.324
P5	6.24(1.17)	6.00(1.19)	1.326	0.193
P6	6.82(0.46)	6.92(0.36)	-1.434	0.160

Question	ARGreenet Post-test	BasicGreenet Post-test	t	p
P7	6.45(0.83)	6.45(0.76)	0	1
P8	5.76(1.24)	5.87(1.02)	-1	0.324
P9	6.00(1.04)	6.18(0.83)	-1.641	0.109
P10	6.05(1.06)	6.13(1.12)	-0.723	0.474

Table 4. Means (SD) of the ARGreenet and the BasicGreenet, and paired t-test of post-test scores. d.f. 37, ‘*’ indicates significant differences**

5. RESULTS

The sample was comprised of thirty eight participants with a mean age of 10.76(1.49) years. Within the sample group there were more males (63.2%) than females represented.

From the entry questionnaire, means (SD) are in Table 1. From the these scores, we can deduce that participants reported to have relatively little mobile phone experience (E1), and considered themselves to be novices with the Nokia N95 (E2). Participants reported to have some experience with gaming (E3). Overall participants initially stated they had a moderately high knowledge of recycling (E4, E5 and E6). Furthermore most reported positive beliefs toward recycling (E7). The participants also reported a willingness to do more (E9), although the majority reported that they were already strong recyclers (E8).

Paired t-tests were applied to the scores given to all questions of the post questionnaire filled out after playing each game. These analyses are shown in Table 4. None of the statistical paired t-tests applied to the results showed significant differences between the two games except for question P3. The significance level was set to 0.05 in all tests. From this data, we can deduce that the two games were very positively accepted by the players. The combined mean among all the ten questions for the two games is 6.24(0.31). Moreover, the games had a very similar influence on responses with the actual difference between the mean responses being very small. From the findings the following, trends can be inferred:

- The BasicGreenet had a marginally, more positive influence on responses to the belief question (P6), and questions regarding intentions to change behaviours (P8-P10).
- The ARGreenet had a marginally more positive influence on responses to the engagement and fun questions (P1-P2), ease of use (P3), and the perceived value questions (P4-P5).
- Each game had a very similar influence on intended behaviour/motivation to change (P7).

In order to determine whether using either of the games first has any effect on the scores for the second game, the sample was divided into two groups: the participants who used the ARGreenet first; and the participants who used the BasicGreenet first. One-

way ANOVA analyses were applied to the scores for all questions (20 in total). Only three of the statistical ANOVA tests applied to the results showed significant differences between the two games. From this data, we can deduce that the order of playing did not significantly affect the scores for the second game.

To confirm if participants changed their attitudes as result of playing the games the scores for question E7 and the related questions answered after playing both games (P6), were compared using paired t-tests. The results showed significant differences (for ARGreenet, $t(37)=-2.589$, $p=0.014$; for BasicGreenet, $t(37)=-2.903$, $p=0.006$; confirming that participants’ attitudes have been influenced by the games. We also checked if participants’ intentions to change behaviours were altered after playing the games. For this the scores for question E9 and the related questions answered after playing both games (P7), were compared using paired t-tests. Again, the results showed significant differences for ARGreenet, $t(37)=-2.603$, $p=0.013$; and for BasicGreenet, $t(37)=-2.477$, $p=0.018$. Therefore, participants’ intentions to change behavior appear to have been influenced by the games.

For the relationship between the intended behavior before (E9) and after (P7) playing the games Pearson’s correlation was used. The null hypothesis is that the correlation coefficient comes from a population in which the correlation is 0. In order to determine if the correlation is significant, we checked whether the correlation coefficient is within the sample distribution specified by the null hypothesis with different probabilities. The Pearson Correlations significance levels are shown in Table 5. Using the results from the game that was first used by each of the two groups of children (i.e. when ARGreenet is first used or when BasicGreenet is first used) we can deduce that ARGreenet presents a more significant correlation. These results confirm that a positive change in the intended behaviour has been brought about by using the games, especially the ARGreenet. Considering Gardner & Ashworth’s [Gar08] advice that “if individuals also hold a positive attitude towards recycling they are more likely to actually perform the behaviour” and the results of the games’ influence on players’ attitudes and intended

behaviours, it appears they will most likely recycle better as a result of playing the games.

Game used	Order of use	Significance level
ARGreenet	First used	0.621(0.005)
BasicGreenet	First used	0.566(0.02)
ARGreenet	Used second	0.318(0.2)
BasicGreenet	Used second	0.361(0.2)

Table 5. Pearson Correlations significance levels

Checking if participants' perception for learning about recycling has been influenced by playing the games, the scores for question E6 and the question answered after playing both games (F2), were compared using paired t-tests. The results showed significant differences for all data, $t(37)=-5.011$, $p<0.001$; when ARGreenet is first used, $t(18)=-3.082$, $p=0.006$; and when BasicGreenet is first used, $t(18)=-4.135$, $p=0.001$). We also compared, using paired t-tests, the scores for question E4 and the question answered after playing both games (F1). The results showed significant differences for all data, $t(37)=-6.047$, $p<0.001$; when ARGreenet is first used, $t(18)=-4.324$, $p<0.001$; and when BasicGreenet is first used, $t(18)=-4.135$, $p=0.001$). Therefore, participants' perception for learning about recycling has been influenced by the games. Moreover, the mean (SD) of F1 scores were equal or more than 6 for all data, 6.05(0.84); when ARGreenet is first used, 6.00(0.67); and when BasicGreenet is first used, 6.11(0.99) and so it is also possible to deduce that the players' feel that they have learnt about what can be recycled and how to recycle.

In our study, two questions relating to the sense of presence were included in the questionnaire asked at the end of playing the ARGreenet only. This questionnaire was based on the Slater et al. [Sl94] questionnaire. The first presence question was "I had a sense of being there in a room where there are rubbish and recycling boxes". Participants could answer from 1 = not at all to 7 = very much. The second questions was "There were times during the experiences when I thought that draws and images were in the room, over the table or over my hand" where 1 = at no time and 7 = almost all the time. The presence score or SUS Count is taken as the number of answers that have a score of 6 or 7. In our study, the SUS Count was 0.974(0.753). The SUS Mean across the two questions was 5(1.484) so although the presence scores were quite high, but they did not reach 6.

With regard to preferences, children answered to question F3. Most participants (69.4%) preferred the ARGreenet. When the BasicGreenet is first played, 82.4% of participants preferred the ARGreenet,

whereas 57.9% of participants preferred the ARGreenet when the ARGreenet is first played.

Several explanations why the children gave their preference for the ARGreenet were: It was fun to have things over my hand that really they were not there; The AR game was more original; The AR game was more amusing; The AR game was more real; It is different to typical games.

However, there were some children who liked the BasicGreenet better who gave the following responses: I like playing remaining seated; I prefer to use the mobile with my hand rather than moving around the room; I prefer to use the keys of the mobile to play; Because in the Basic game the objects appear and you do not have to look for them.

A few children added some final comments in response to the question "Any comments that you like to add". These included: I want to know how the AR game works in order to explain to my parents; I propose to commercialize both games.

An observation remarked by the person in charge of the experiment was: "With the AR game, several children played with the markers placing them in different places (over their t-shirt, their cap, etc.)".

6. CONCLUSION

The results from our research show that the two games, ARGreenet and BasicGreenet have been very positively accepted by players with an overall mean of 6.24 (on a scale 1-7). The results did not show statistical significant differences between the two games. However, 69.4% of the participants preferred the ARGreenet game, they perceived it as easy to use and more engaging and fun than BasicGreenet. From our point of view, if a game is easy and fun to play, children will play it, and - consequently - the overall impact of the game on their behaviour will be much higher than in case of a game which is more difficult and less fun to play. From the results, there is not any statistical evidence that ARGreenet is perceived to be different from BasicGreenet, the majority of participants preferred the ARGreenet game and five of the ten analysed questions showed that ARGreenet offered greater means than the BasicGreenet game. While one question offered the same mean for both games. This implies that these preliminary results corroborate the hypothesis that the ARGreenet would have greater influence on a number of variables than the BasicGreenet. Based on the sense of presence questions our results suggest that participants experienced a moderately high sense of presence using ARGreenet.

From the analyses is also possible to infer that the games did influence the knowledge of participants, their attitudes and had a positive influence on their

intentions to change behaviours. Based on the advice of Gardner & Ashworth [Gar08] that “if individuals also hold a positive attitude towards recycling they are more likely to actually perform the behaviour”, these results suggest that playing the games is likely to have some influence to change participants’ behaviour. Future research could be conducted at a later stage to confirm if players’ actual recycling behaviour has been positively affected.

All these conclusions suggest that the mobile phone is potentially a good platform not only for learning about recycling but also persuading people to change their behaviour, and that AR mobile phone applications is probably likely to be more positively received, particularly from a fun point of view. However, more experiments should be carried out in order to determine, first, if AR gaming is preferred to simple mobile games for edutainment in general; second, if educational games help children change their attitude towards recycling; third, if games are preferable to other forms of media, e.g. TV, etc.

The games and the trial can be improved in several ways. The games could incorporate more rubbish types apart from those already included, and more questions relating to these types of residues. The trial could also be improved by controlling the way in which the games are played. It would be useful to conduct a trial where all players start with the BasicGreenet and then graduate to the ARGreenet and then conduct another trial where only the ARGreenet game is used and compare the responses to questionnaires. A more extensive final questionnaire could be used to enable improved comparison between the two games. The trial could also be improved, especially, in the influence of learning of both games, including the related question after using both games. In order to evaluate the acquired knowledge of players, a final examination could also be included. It would also be possible to use another learning practice in which the knowledge would be presented in-game and let children learn through their engagement, e.g. an adventure game.

7. ACKNOWLEDGMENTS

We would like to thank:

- This work was partially funded by:
 - the GreenHunt project (PCI2006-A7-0676).
 - the APRENDRA project (TIN2009-14319-C02).
- The Summer School of the Technical University of Valencia for its collaboration.
- Carlos Fuster for his assistance in developing the BasicGreenet game.
- Simone Carr-Cornish for her assistance in analysing the data of the experiment.

8. REFERENCES

- [Ajz80] Ajzen, I., Fishbein, M. Understanding attitudes and predicting social behaviour. Englewood Cliffs, NJ: Prentice Hall, 1980.
- [Ajz91] Ajzen, I. Theory of planned behaviour. *Journal of Organisational Behaviour Human Dimensions Journal*. 50, pp. 179-211, 1991.
- [Azu97] Azuma, R.T. A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*, 6 (4): 355 – 385, 1997.
- [Azu01] Azuma, R.T., Bailiot, Y., Behringer, R., Feiner, S., Julier, S., MacIntyre, B. Recent advances in augmented reality, *IEEE Computer Graphics and Applications*, 21: 34-37, 2001.
- [Bol95] Boldero, J. The prediction of household recycling of newspapers: the role of attitudes, intentions and situational factors. *Journal of Applied Social Psychology*. V25(5): 440-462, 1995.
- [Gar08] Gardner, J., Ashworth, P. Chapter 12 Towards the intelligent grid: A review of the literature. *Urban Energy Transitions*. P. Droege. London, Elsevier, 2008
- [Hen05] Henrysson, A., Billinghurst, M., Ollila, M. Face to Face Collaborative AR on Mobile Phones. *International Symposium on Augmented and Mixed Reality*, pp. 80-89, 2005
- [Kat99] Kato, H., Billinghurst, M., Marker tracking and HMD calibration for a video-based augmented reality, In *Proceedings of 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR’99)*, pp. 85-94, 1999.
- [Liu07] Liu, T., Tan, T., C, Y., 2D barcode and Augmented Reality supported English learning system, *Computer and Information Science (ICIS’07)*, pp. 5-10, 2007.
- [Osh09] O’Shea, P., Dede, C., Mitchell, R., Johnston, C. Lessons learned about designing augmented realities, *International Journal of Gaming and Computer-Mediated Simulations*. Vol. 1. N. 1, pp. 1-15, 2009
- [Sch07] Schmalstieg, D., Wagner, D. Experiences with Handheld Augmented Reality, *The Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality*. pp. 1-13, 2007
- [Sla94] Slater, M., Usoh, M., Steed, A. Depth of presence in virtual environments. *Presence: Teleoperators and Virtual Environments*, 3: 130-144, 1994.
- [Ton04] Tonglet, M., Phillips, P.S., Read, A. D. Using the Theory of Planned Behaviour to investigate the determinants of recycling behaviour: a case study from Brixworth, UK, *Resources, Conservation and Recycling* 41: 191-214, 2004.
- [Wag03] Wagner, D., Schmalstieg, D. First Steps Towards Handheld Augmented Reality. *The 7th International Conference on Wearable Computers*, pp. 127-135, 2003
- [Wag07a] Wagner, D. Handheld Augmented Reality, Dissertation thesis, Graz University, Austria, 2007
- [Wag07b] Wagner, D., Schmalstieg, D. ARToolKitPlus for Pose Tracking on mobile devices, *12th Computer Vision Winter Workshop*. pp. 139-146, 2007
- [Wag09] Wagner, D., Schmalstieg, D. Making Augmented Reality Practical on Mobile Phones, Part 1. *IEEE Computer Graphics and Applications*, 29(3), 12-15, 2009
- [Wan09] Wang, K, Chen, L., Chu, P., Cheng, Y. A study on the design of Augmented Reality user interfaces for mobile learning systems in heritage temples, *Virtual and Mixed Reality, LNCS*, vol. 5622/2009, pp. 282-290, 2009.

Snake-based Technique for Automated Coronal Loop Segmentation

Jong Kwan Lee
Department of Computer Science
Bowling Green State University
leej@bgsu.edu

Woon Khang Tang
Department of Computer Science
Bowling Green State University
woonkhang@gmail.com

ABSTRACT

A new approach to automatically segment the solar coronal loop structures from intensity images of the Sun's corona is described. The approach is based on the active contour models (snakes) and exploits the Gaussian-like shape of the coronal loop cross-sectional intensity profile to refine the snake's position. The approach utilizes the principal component analysis to automatically initialize the snake's position. It then uses a greedy minimization method to attract the snake toward the center of coronal loop structures in each image. Its effectiveness is evaluated through experiments on synthetic and real images.

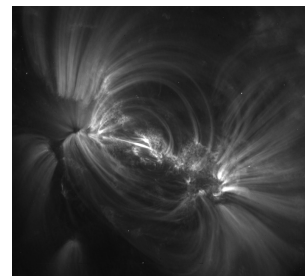
Keywords: Active contours, Solar coronal loops segmentation, Image processing

1 INTRODUCTION

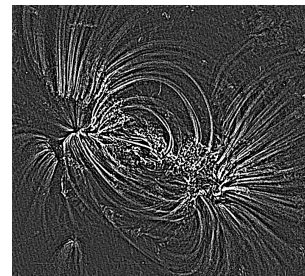
Automated feature segmentation techniques have been used and studied widely in many applications (e.g., iris recognition [Teo05], fingerprint recognition [Pan01, Mag09], face recognition [Wis97, Kuk04], remote sensing [Cao09], etc.). Automated segmentation techniques can aid understanding of the feature characteristics. In this paper, we consider the problem of automatically segmenting the coronal loop structures from solar imagery.

The corona is the outermost layer of the Sun's atmosphere. Many solar scientists study the corona to gain insight of the solar activities (e.g., solar storms) that impact our geo-space environment. One common way to study the corona is via observation of the loop structures in the corona. These loop structures are the traces of the solar magnetic field that ultimately drives the Sun's dynamic activities [Lee06]. The typical way solar scientists observe the coronal loop structures is by considering them in solar imagery collected by a satellite. One preferred source of coronal images is NASA's TRACE satellite [Sch99]. (Details of TRACE image acquisition process are also explained in [Sch99] and omitted here.) A sample TRACE coronal image is shown in Figure 1 (a). In the figure, the bright arching structures are the coronal loops.

Automatic segmentation of the coronal loop structures is challenging as the structures have complex



(a)



(b)

Figure 1: (a) TRACE coronal image and (b) Pre-processed image

shapes and very blurry boundaries. Moreover, they tend to appear very close to each other or even overlap each other in the image. In addition, image noises and other non-loop features (e.g., sponge-like wide white spots) are present on the coronal images.

In this paper, we present a new method, based on active contour models (i.e., also known as snakes) [Kas87], to fully-automatically segment the coronal loop structures from the TRACE images. The method uses the principal component analysis-based loop direction measures to initialize the snake position. It also exploits the shape of the coronal loop's

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.

cross-sectional intensity profile to refine a greedy algorithm-based snake.

The paper is organized as follows. Section 2 discusses related work. In Section 3, the new snake-based approach for segmenting solar coronal loop structure is described. The experimental results of applying the new approach to synthetic and real coronal images are presented in Section 4. Section 5 concludes this paper.

2 RELATED WORK

In this section, the snake that our approach is based on and the prior efforts on automated coronal loop segmentation are discussed.

Kass et al. [Kas87] have introduced the snake as an energy minimizing contour that is represented parametrically as $v_i(s) = (x_i(s), y_i(s))$, where (x_i, y_i) is the i^{th} point position along the contour. An energy function is associated with the snake. Snake finds an image feature of interest (e.g., lines or edges) by minimizing its energy function. Snake's energy function (E_{snake}) is defined as the sum of internal energy term (E_{int}) and external energy term (E_{ext}) of each snake point and is defined as

$$E_{snake} = \int_0^1 E_{int}(v(s)) + E_{ext}(v(s)) ds, \quad (1)$$

where E_{ext} is the sum of image energy (E_{img}) and external constraint energy (E_{con}). E_{int} can be defined as

$$E_{int} = \frac{\alpha(s)|\mathbf{v}_s(s)|^2 + \beta(s)|\mathbf{v}_{ss}(s)|^2}{2}, \quad (2)$$

where $\alpha(s)$ is the weight of the first-order term (i.e., $\mathbf{v}_s(s)$) and $\beta(s)$ is the weight of the second-order term (i.e., $\mathbf{v}_{ss}(s)$). $\mathbf{v}_s(s)$ and $\mathbf{v}_{ss}(s)$ terms prevent gaps and sharp corners. E_{img} can be expressed as a weighted combination of the line energy (E_{line}), the edge energy (E_{edge}), and the termination energy (E_{term}). E_{line} can force the snake to move toward high intensity points. E_{edge} can force the snake to move toward the image points with large image gradient. E_{term} can force the snake to move toward the end points of lines or edges. E_{con} can be defined as $E_{con} = -k(\mathbf{x}_1 - \mathbf{x}_2)^2$, where k is the spring constant, \mathbf{x}_1 is a snake point, and \mathbf{x}_2 is an image point.

Kass et al. [Kas87] have minimized the snake energy function iteratively using Euler equations expressed in a sparse matrix form. However, using their minimization method, some snake points tend to congregate in certain image points and the process can be unstable. Amini et al. [Ami90] have proposed a dynamic programming-based minimization process that can avoid some of Kass et al. [Kas87]'s problems. However, the dynamic programming-based snake approach can be slow. Williams and Shah [Wil92] have proposed a greedy algorithm for the snake's energy minimization which is

more than an order of magnitude faster than the dynamic programming-based approach. In addition, it tends to be more stable. Here, we note that our new snake-based approach also uses the greedy algorithm-based approach.

Next, the prior works on automated coronal loop segmentation are discussed.

Oriented Connectivity Method (OCM) by Lee et al. [Lee06] is the first automated coronal loop segmentation method. OCM is a constructive edge linkage method that utilizes a simplified solar magnetic field estimate to guide the loop segmentation process. In particular, the magnetic field estimate is used to progressively link neighboring loop points together that have orientations that are consistent with the magnetic field's orientation.

Lee et al. [Lee06b] have also introduced another automated coronal loop segmentation method called the *Dynamic Aperture-based Method* (DAM). DAM segments coronal loops by exploiting the Gaussian-like shape of loop cross-sectional intensity profiles [Car03]. Specifically, DAM links the image points that have similar Gaussian shape parameters (as determined by a ruled Gaussian surface fitting on image points) and have similar loop orientation (as determined by the principal component analysis (PCA) [Dud00]).

Smith [Smi05] has adapted the *Unbiased Detection Method* (UDM) by Steger [Ste98] for coronal loop segmentation. UDM segments curvi-linear structures with different lateral contrast by taking account the geometry of the structure surroundings. It utilizes the second derivatives of image intensities in the direction perpendicular to the structure to find the centroid position of a curvi-linear structure.

Sellah and Nasraoui [Sel08] have utilized a type of randomized Hough Transform called *Incremental Random Hough Transform* (I-RHT) to detect coronal loop structures. (Randomized Hough Transform (RHT) [Kul90] is a fast Hough Transform approach that alleviates some of the expensive computational requirements of the standard Hough Transform (HT) [Hou62].) I-RHT detects a coronal loop structure by using a stream clustering algorithm to continuously update and extract the maximum bin of the HT accumulator in an incremental fashion. However, their approach was limited to detection of only one elliptical coronal loop structure.

Recently, Aschwanden [Asc10] has introduced a coronal loop tracing technique called *Oriented Coronal Curved Loop Tracing* (OCCULT), which is based on his earlier method [Asc08]. OCCULT utilizes the local loop directivity and the curvature radius constraints in coronal loop tracing. In particular, the curvature radius constraint enables better loop tracing by providing estimates of loop direction range based on previously-traced loop segment.

Other coronal loop segmentation works include the solar loop mining system by Durak et al. [Dur09, Dur10]. Their system includes a block-by-block loop detection processing to retrieve images with coronal loops from large number image datasets. Another recent coronal loop segmentation method is McAteer et al.'s *2D Wavelet-Transform Modulus Maxima Method* [McA10]. Their method uses the derivative of a 2D Wavelet-based smoothing function as an edge detector in segmenting coronal loop structures.

3 NEW APPROACH

Next, the new snake-based approach for automated segmentation of coronal loop structure is introduced.

The approach introduced in this paper exploits the Gaussian-shape of the coronal loop's cross-sectional intensity profile in its snake minimization process. In particular, the new energy terms based on this intensity profile property are considered to enable the snake to more-accurately lock on the coronal loop structures. We will call this new snake the *Gaussian-snake* (*G-snake*) in this paper. In our approach, a G-snake is positioned on a part of a coronal loop structure initially and "grows" along the coronal loop structure until the entire loop structure is segmented. Then, this process is repeated until all the coronal loop structures are considered.

Next, the details of the new approach including the pre-processing steps to "clean" the image and the initialization and minimization of the G-snake are discussed.

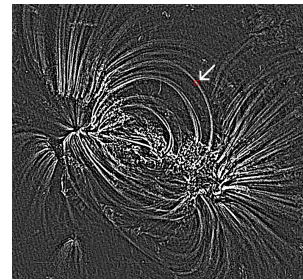
3.1 Pre-processing Steps

In our approach, a series of pre-processing steps are applied to the coronal image to remove image noise (e.g., impulse noise) and to enhance the contrast between the coronal loops and the background; thus, these steps provide images that are well-suited for the G-snake to segment the coronal loops accurately.

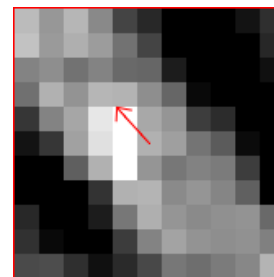
To remove impulse noise, 3×3 median filtering and 3×3 low-pass filtering are applied. Then, to sharpen the loop structures, a 3×3 unsharp masking is applied. (Unsharp masking subtracts a blurred version of an image from the image itself to reduce image intensity by a local background defined by the smoothing process [Gon02].) We note that the pre-processing steps used here were empirically determined and are similar to the ones used in [Lee06, Lee06b, Asc10]. Figure 1 (b) shows the pre-processed image of the coronal image shown in Figure 1 (a). As shown in the figure, the pre-processed image contains less noise and the coronal loop structures are much more distinguishable.

3.2 G-snake Initialization

The first step of the G-snake approach is to position a G-snake on a coronal loop structure automatically. (Here,



(a)



(b)

Figure 2: Directionality of a loop point (center point of the box) determined by PCA

we note that automated initialization of snake position is a very challenging task for any type of snake-based approaches.)

In our approach, a set of initial snake points are positioned on the image region where the points in the region have very similar angular directions (since the nearby points on the same coronal loop have similar angular directions). In this paper, we denote the angular direction of a point as the *directionality* of the point and defined as the angular direction of high-intensity point variation in a small image region around the point. The directionality is measured using the principal component analysis (PCA) as the arctangent of the maximum eigenvector's components. (Here, we note that we empirically determined that 11×11 image region is suitable image size to compute the directionality of a point.) Figure 2 shows an example of directionality of a loop point determined using PCA. The directionality of the loop point (i.e., the center point of the red box shown in Figure 2 (a)) determined using PCA is marked as a red arrow in Figure 2 (b). As shown in the figure, PCA determines the loop direction accurately. (We note that this directionality measure has previously been used in [Lee06b].)

Using the directionality measures, we search for $N_1 \times N_2$ rectangular image regions where the directionalities of the points in the region are very similar. (We used $N_1 = 10$ and $N_2 = 3$ for the G-snake.) Specifically, we place a rectangular box on the image where the longer side of the box is parallel to the x-axis of the image. Then, we rotate the rectangular box with

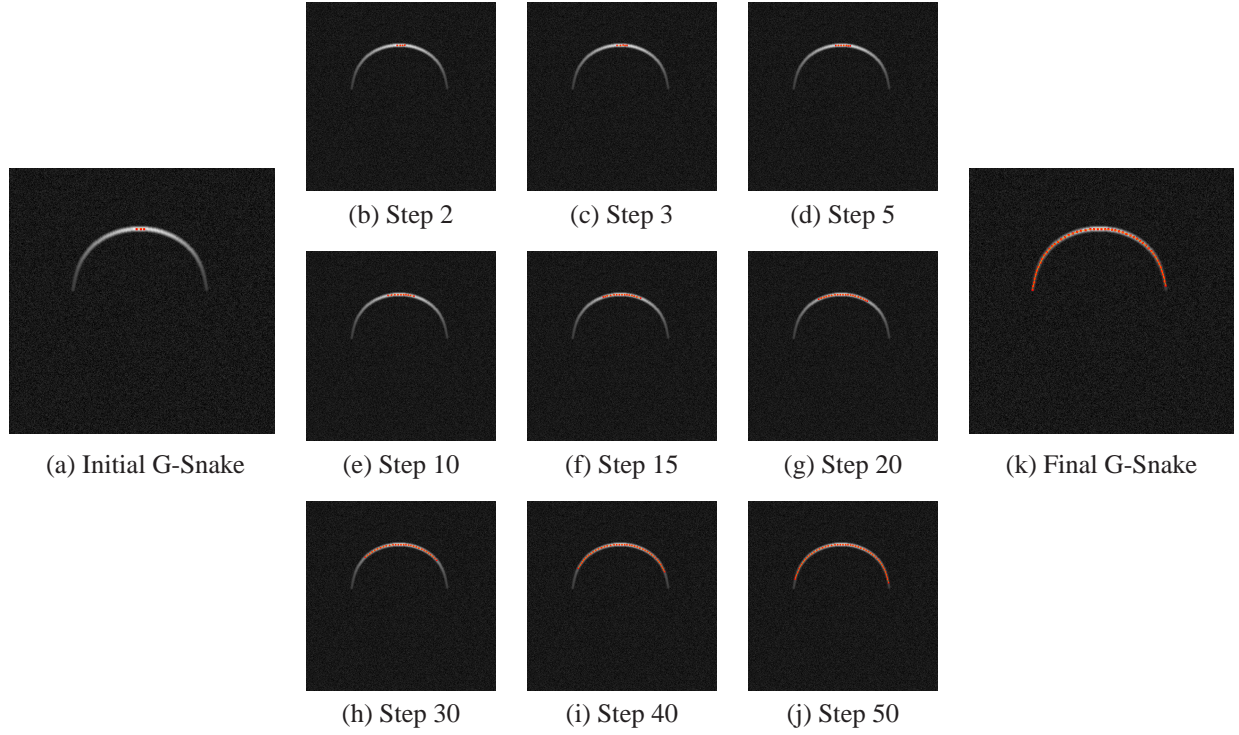


Figure 3: Illustration of G-Snake minimization

respect to the region center point's directionality and determine the divergence of the directionalities for the points in the rotated rectangular box. If the directionality divergence of the rectangular box is smaller than a threshold value, three equally-spaced G-snake points are positioned along the center of the rectangular image region. We empirically determined that three equally-spaced points are a good set of initial G-snake points as they are positioned closely such that they are always on the same coronal loop structure. An example of G-snake initialization is shown in Figure 3 (a). In the figure, three G-snake points (shown in red) are positioned on a synthetic loop structure successfully.

3.3 G-snake Energy Minimization

For each initial G-snake points, the G-snake energy function is minimized to segment the coronal loop structure.

Our G-snake uses the greedy-based algorithm in its minimization process. In the greedy-based algorithm, a G-snake is defined as a set of N points and each point is moved to a new position of the local minimum (which leads to the global minimization).

The energy function used in the G-snake are

$$E_{G\text{-snake}} = \sum_{i=1}^N (a_i E_{cont} + b_i E_{curv} + c_i E_{grow} + d_i E_{img} + e_i E_{gauss} + f_i E_{direc}), \quad (3)$$

where a_i , b_i , c_i , d_i , e_i , and f_i are the energy weights ($0 \leq a_i, b_i, c_i, d_i, e_i, f_i \leq 1$) and E_{cont} , E_{curv} , E_{grow} , E_{img} ,

E_{gauss} , and E_{direc} are the continuity energy, the curvature energy, the growth energy, the image energy, the Gaussian energy, and the directionality energy, respectively. (For the segmentation results shown in Section 4, an equal weight of 1 is used for all the energies.)

E_{cont} forces the distance between adjacent points to be maintained at an equal distance. E_{cont} is defined as

$$E_{cont} = (\bar{d} - \|v_i - v_{i-1}\|)^2, \quad (4)$$

where \bar{d} is the average Euclidean distance between all adjacent G-snake points.

E_{curv} forces the snake to be smooth and avoid oscillations. E_{curv} is defined as

$$E_{curv} = \|(v_{i+1} - v_i) - (v_i - v_{i-1})\|^2. \quad (5)$$

E_{grow} is a new energy that forces the G-snake to grow along a coronal loop. To prevent "over-growing", E_{grow} is considered only for the first and last G-snake points (i.e., $c_i = 0$ for all $i \neq 1$ and N). E_{grow} is defined as

$$E_{grow} = -\|v_i - v_{i-1}\|^2. \quad (6)$$

E_{img} forces the G-snake points to move toward points with high intensity values. E_{img} is defined as

$$E_{img} = -I(x, y), \quad (7)$$

where $I(x, y)$ is the intensity value at a point (x, y) .

While E_{img} attracts the G-snake points to points on the coronal loop structures (as the coronal loop appear

very bright in the image), it can also move the G-snake toward noise pixels with high intensity. To prevent this effect, new energies, E_{gauss} and E_{direc} , are used in G-snake.

E_{gauss} forces the G-snake to move toward the points whose intensity profile along the direction that is perpendicular to the loop direction is well-fitted by a Gaussian curve. (As mentioned in Section 2, the cross-section intensity profile of a coronal loop exhibits a Gaussian-shape.) E_{gauss} is defined as

$$E_{gauss} = -G(A, B), \quad (8)$$

where A is the height (i.e., peak intensity) and B is the center (i.e., horizontal position of peak) of the Gaussian that has been fitted using the intensity profile along the direction that is perpendicular to the loop direction. G is defined as

$$G(A, B) = \begin{cases} 0, & \text{if poor Gaussian fit} \\ |B - C_p|, & \text{otherwise,} \end{cases} \quad (9)$$

where C_p is the center of the profile. We define a *poor Gaussian fit* case as a case when A is negative or too large or when the sum of squared Gaussian fitting error is too large. We have used the Levenberg-Marquardt algorithm [Lev44, Mar63] for the Gaussian curve fitting. (We note that we empirically determined that using 10 image points produces reasonable fitting results for the TRACE images).

E_{direc} forces the G-snake to move toward the points whose directionality is consistent with nearby loop points' directionalities. E_{direc} constraints the snake points to stay on one loop structure—it prevents a snake point to move toward other nearby loops. E_{direc} is defined as

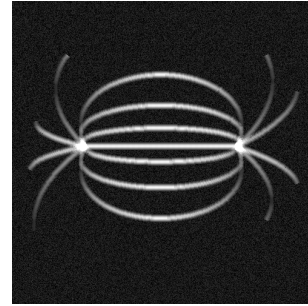
$$E_{direc} = \sum_{i=1}^{N_a} \sum_{j=1}^{N_b} |d_{ij} - \mu|, \quad (10)$$

where d_{ij} denotes the directionality of a point at (i, j) and μ denotes the mean directionality in a $N_a \times N_b$ image region.

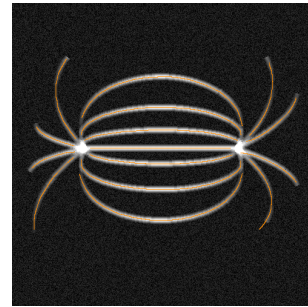
$E_{G\text{-snake}}$ is minimized iteratively until the G-snake becomes *stable*. We define a *stable* G-snake as a G-snake whose the number of points moving to new position is zero or the number of times points oscillating the same points is more than a pre-defined threshold value.

Here, we note that using only three initial G-snake points is often not enough to segment a (long) coronal loop structure. Thus, in our approach, an intermediate point is added to each G-snake when the distance between two adjacent G-snake points is greater than a pre-defined threshold (e.g., 10 in pixels).

Figure 3 shows an example of G-snake minimization process for a synthetic loop. As shown in the figure, the G-snake grew toward the ends of the loop and segmented the loop successfully. (Here, we note that the snake without the new energy terms can not segment the



(a)



(b)

Figure 4: (a) Synthetic image and (b) Segmented loops

loop correctly; the new energy terms introduced here enable segmentation of loop structures that exhibit the Gaussian-shaped cross-sectional intensity profile.)

4 EXPERIMENTAL RESULTS

We have applied our new method to synthetic and real coronal images. In this paper, we report segmentation results for ten synthetic and five real images. We have used a similar scheme used in [Lee06] to generate the synthetic coronal images. Specifically, a simple magnetic field model is used to create “loop lines” on 1024×1024 images and these loop lines simulated to follow Gaussian-shaped cross-sectional intensity profiles. In addition, normal-distributed random image noise is added to simulate real coronal image noise. The

Table 1: GPE measures on synthetic images

Datasets	Max.	Min.	Mean	Std. dev.
Syn. 1	18.44	0.00	0.32	1.05
Syn. 2	11.18	0.00	0.30	0.76
Syn. 3	7.07	0.00	0.57	1.01
Syn. 4	3.00	0.00	0.27	0.46
Syn. 5	6.74	0.00	0.35	0.59
Syn. 6	24.17	0.00	0.45	1.00
Syn. 7	12.81	0.00	0.33	0.60
Syn. 8	2.24	0.00	0.28	0.46
Syn. 9	4.00	0.00	0.29	0.48
Syn. 10	2.83	0.00	0.34	0.49
Average	9.24	0.00	0.35	0.69

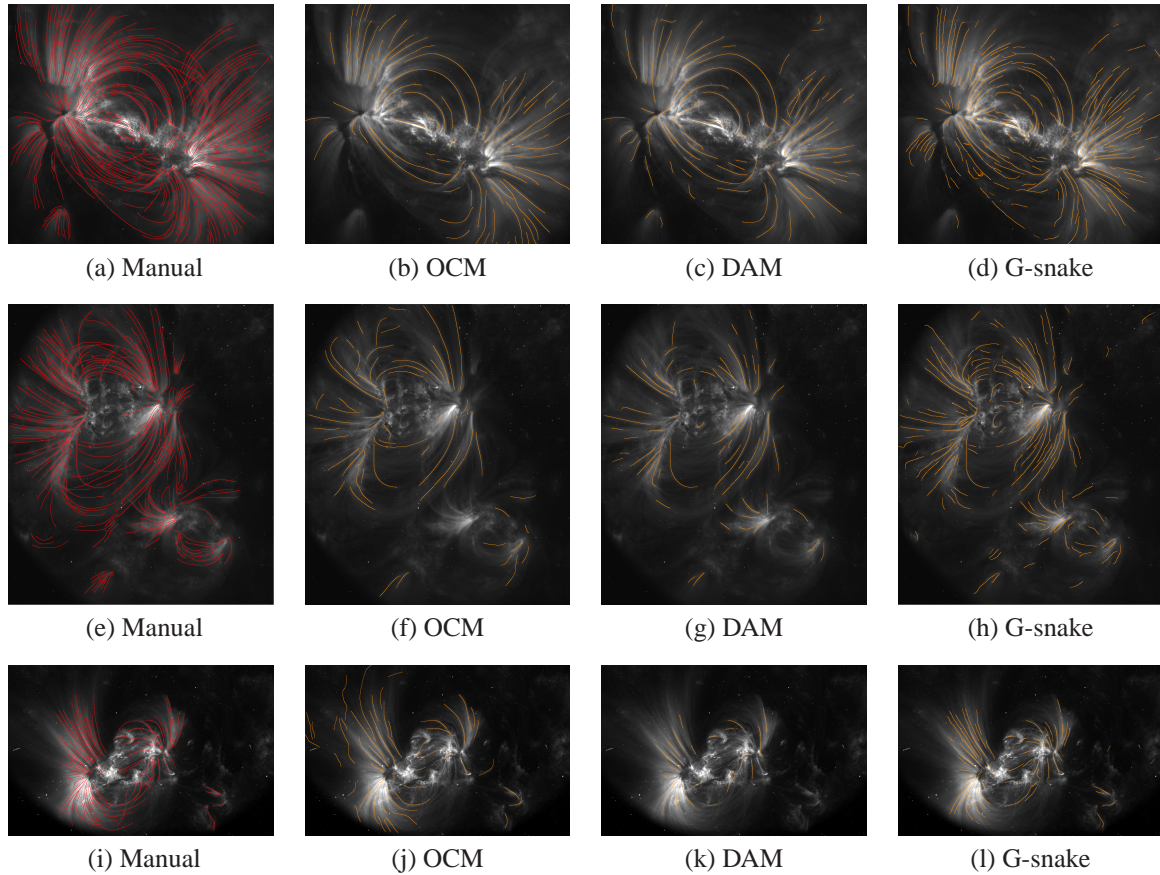


Figure 5: Coronal loop segmentation on real images (OCM and DAM results are borrowed from [Lee06, Lee06b])

size of the tested real images was 1024×1024 . (Here, we note that different set of G-snake's parameters might be needed for non-TRACE coronal images.) To evaluate the effectiveness of G-snake, we have measured the four metrics (i.e., maximum, minimum, mean, and standard deviation) of the global positional error (GPE). GPE is defined as the global estimation of the shortest Euclidean distance difference between the real loops and the segmented loops [Lee06].

Figure 4 shows the G-snake segmentation result for one synthetic image. As shown in the figure, G-snake well-segmented the loop structures.

Table 1 shows the four metrics of GPE measures for the ten synthetic images we tested. For these synthetic images, the average of the mean GPE was only 0.35 (in pixels).

For the real coronal image testings, we have used the manual segmentation as the gold standard (since the actual loop positions were unknown). In addition, we have compared the G-snake's results with the OCM and DAM results. Figure 5 shows the segmentation results on three real coronal images. In the figure, the manually-segmented loops are overlaid with red curves (in the sub-figures (a), (e), (i)) and the automatically-segmented loops are overlaid with orange curves. Sub-

figures (b), (f), and (j) show the OCM results, Sub-figures (c), (g), and (k) show the DAM results, and Sub-figures (d), (h), and (l) show the G-snake results. As shown in the figure, the G-snake seems to produce the best coronal loop segmentation results; the G-snake segmented many of the loops that were not segmented by OCM and DAM.

Table 2 shows the four metrics of GPE measures for OCM, DAM, and G-snake on real coronal images. We have chosen these five images for our testing as they have been used by others. As shown in the table, G-snake produced the minimum mean GPE measures for all images. The overall average of the mean GPE for G-snake was less than 1 pixel. (In some cases, the maximum and standard deviation GPE measures of G-snake were slightly higher than that of DAM.)

5 CONCLUSION

In this paper, the G-snake for automated coronal loop segmentation was described. The approach is the first method that utilizes a snake in coronal loop segmentation. It uses new energies (i.e., E_{grow} , E_{gauss} , and E_{direc}) in the snake's minimization process to enable accurate coronal loop segmentation. Through evaluation of the technique, we have shown that the G-snake can provide

Table 2: GPE measures for OCM, DAM, and G-snake on real coronal images

Datasets	Max.	Min.	Mean	Std. dev.
Real 1	34.79	0.00	2.19	3.21
Real 2	17.03	0.00	1.93	1.78
Real 3	47.41	0.00	2.82	4.71
Real 4	37.54	0.00	3.00	4.60
Real 5	57.01	0.00	3.11	5.94
Average	38.76	0.00	2.61	4.05

(a) OCM

Datasets	Max.	Min.	Mean	Std. dev.
Real 1	19.00	0.00	1.73	1.79
Real 2	11.18	0.00	1.67	1.39
Real 3	12.00	0.00	1.41	1.60
Real 4	36.35	0.00	2.36	3.64
Real 5	16.97	0.00	1.56	1.80
Average	19.10	0.00	1.75	2.04

(b) DAM

Datasets	Max.	Min.	Mean	Std. dev.
Real 1	24.19	0.00	0.85	1.39
Real 2	28.79	0.00	1.10	1.76
Real 3	13.60	0.00	0.78	1.05
Real 4	20.62	0.00	1.03	1.83
Real 5	27.20	0.00	1.07	1.75
Average	22.88	0.00	0.97	1.56

(c) G-Snake

consistent and accurate segmentation results of coronal loop structures.

In the future, we plan to perform further effectiveness comparisons with other coronal loop segmentation results (e.g., [Asc10]). In addition, we plan to adopt the G-snake-based coronal loop segmentation to other solar studies, such as solar magnetic field parameter recovery [Lee09]. We also hope to apply the G-snake to other scientifically-interesting structures that follows similar characteristics (i.e., G-snake may be adopted to segment other image structures that follow a different cross-sectional intensity profiles).

ACKNOWLEDGMENT

We acknowledge that the results of OCM and DAM were borrowed from [Lee06, Lee06b]. We also thank the reviewers for their valuable comments which improved our paper.

REFERENCES

[Ami90] Amini, A.A., Weymouth, T.E., and Jain, R.C., Using Dynamic Programming for Solving Vari-

ational Problems in Vision, Pattern Recognition, Vol. 12 (9), pp. 855–867, 1990.

[Asc08] Aschwanden, M.J., Lee, J.K., Gary, G.A., Smith, M., and Inhester, B., Comparison of Five Numerical Codes for Automated Tracing of Coronal Loops, *Solar Physics*, Vol. 248, pp. 359–377, 2008.

[Asc10] Aschwanden, M.J., A Code for Automated Tracing of Coronal Loops Approaching Visual Perception, *Solar Physics*, Vol. 262 (2), pp. 399–423, 2010.

[Cao09] Cao, C., Newman, T.S., and Germany, G.A., New Shape-based Auroral Oval Segmentation Driven by LLS-RHT, *Pattern Recognition*, Vol. 42 (5), pp. 607–618, 2009.

[Car03] Carcedo, L., Brown, D., Hood, A., Neukirch, T., and Wiegelmann, T., A Quantitative Method to Optimise Magnetic Field Line Fitting of Observed Coronal Loops, *Solar Physics*, Vol. 218, pp. 29–40, 2003.

[Dud00] Duda, R.O., Hart, P.E., and Strok, D.G., *Pattern Classification*, 2nd Edition, Wiley, New York, 2000.

[Dur09] Durak, N., Nasraoui, O., and Schmelz J., Coronal Loop Detection from Solar Images, *Pattern Recognition*, Vol. 42 (11), pp. 2481–2491, 2009.

[Dur10] Durak, N., Nasraoui, O., and Schmelz J., Automated Coronal-Loop Detection based on Contour Extraction and Contour Classification from the SOHO/EIT Images, *Solar Physics*, Vol. 264, pp. 383–402, 2010.

[Gon02] Gonzalez, R.C. and Woods, R.E., *Digital Image Processing*, 2nd Edition, Prentice Hall, Upper Saddle River, New Jersey, 2002.

[Hou62] Hough, P.V.C., Method and Means for Recognizing Complex Patterns, U.S. Patent, 3,069,654, 1962.

[Kas87] Kass, M., Witkin, A., and Terzopoulos, D., Snakes: Active Contour Models, *International Journal of Computer Vision*, Vol. 1 (4), pp. 321–331, 1987.

[Kuk04] Kukharev, G. and Nowosielski, A., Visitor Identification- Elaborating Real Time Face Recognition System, The 12th Int’l Conference in Central Europe on Computer Graphics, Visualization, and Computer Vision (WSCG 2004), Plzen, Czech Republic, pp. 157–164, 2004.

[Kul90] Kultanen, P., Xu, L., and Oja, E., Randomized Hough Transform (RHT), Proceedings, 10th International Conference on Pattern Recognition, Atlantic City, pp. 631–635, 1990.

[Lee09] Lee, J.K. and Gary, G.A., Recovery of

- 3D Solar Magnetic Field Model Parameter using Image Structure Matching, Proceedings, 4th International Conference on Computer Vision/Computer Graphics Collaboration Techniques, MIRAGE 2009, France, pp. 172-181, 2009.
- [Lee06] Lee, J.K., Newman, T.S., and Gary, G.A., Oriented Connectivity-based Method for Segmenting Solar Loops, *Pattern Recognition*, Vol. 39 (2), pp. 246–259, 2006.
- [Lee06b] Lee, J.K., Newman, T.S., and Gary, G.A., Dynamic Aperture-based Solar Loop Segmentation, Proceedings, 7th IEEE Southwest Symposium on Image Analysis and Interpretation, Denver, pp. 91–94, 2006.
- [Lev44] Levenberg, K., A Method for the Solution of Certain Non-linear Problems in Least Squares, *Quarterly of Applied Mathematics*, Vol. 2, pp. 164–168, 1944.
- [Mag09] Magalhaes, F., Oliveira, H.P., and Campilho, A.C., A New Method for the Detection of Singular Points in Fingerprint Images, Proceedings, IEEE Workshop on Applications of Computer Vision (WACV 2009), Snowbird, UT, pp. 1–6, 2009.
- [Mar63] Marquardt, D.W., An Algorithm for Least Squares Estimation of Nonlinear Parameters, *Journal of the Society for Industrial and Applied Mathematics*, Vol. 11 (2), pp. 431–441, 1963.
- [McA10] McAteer, R.T.J., Kestener, P., Arneodo, A., and Khalil, A., Automated Detection of Coronal Loops using a Wavelet Transform Modulus Maxima Method, *Solar Physics*, Vol. 262 (2), pp. 387–397, 2010.
- [Pan01] Pankanti, S., Prabhakar, S., and Jain, A.K., On the Individuality Fingerprints, Proceedings, 2001 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2001), Kauai, HI, pp. I-805–I-812, 2001.
- [Sch99] Schrijver, C.J., Title, A.M., Berger, T.E., Fletcher, L., Hurlburt, N.E., Nightingale, R.W., Shine, R.A., Tarbell, T.D., Wolfson, J., Golub, L., Bookbinder, J.A., DeLuca, E.E., McMullen, R.A., Warren, H.P., Kankelborg, C.C., Handy, B.N., and Pontieu, B.D., A New View of the Solar Outer Atmosphere by the Transition Region and Coronal Explorer, *Solar Physics*, Vol. 187, pp. 261–302, 1999.
- [Sel08] Sellah, S. and Nasraoui, O., An Incremental Hough Transform for Detecting Ellipses in Image Data Streams, Proceedings, IEEE International Conference on Tools with Artificial Intelligence, Dayton, Ohio, pp. 45–48, 2008.
- [Smi05] Smith, M., 2005, <http://twiki.mssl.ucl.ac.uk/twiki/bin/view/SDO/LoopRecognition>, accessed on May 16, 2010.
- [Ste98] Steger, C., An Unbiased Detector of Curvilinear Structures, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 20 (2), pp. 113–125, 1998.
- [Teo05] Teo, C.C. and Ewe, H.T., An Efficient One-Dimensional Fractal Analysis for Iris Recognition, The 13th Int’l Conference in Central Europe on Computer Graphics, Visualization, and Computer Vision (WSCG 2005), Plzen, Czech Republic, pp. 157–160, 2005.
- [Wil92] Williams, D.J. and Shah, M., A Fast Algorithm for Active Contours and Curvature Estimation, *Computer Vision, Graphics and Image Processing*, Vol. 55 (1), pp. 14–26, 1992.
- [Wis97] Wiskott, L. and Fellous, J.M. and Kruger, N. and Malsburg, C.V.D., Face Recognition by Elastic Bunch Graph Matching, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 19 (7), pp. 775–779, 1997.

Multi-material Volume Segmentation for Isosurfacing Using Overlapped Label Propagation

Takuma Niizaka Yutaka Ohtake Takashi Michikawa Hiromasa Suzuki

The University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo, 153-8904, JAPAN
{ niizaka | yu-ohtake | michi | suzuki }@den.rcast.u-tokyo.ac.jp

ABSTRACT

This paper proposes a simple and efficient method for segmenting multi-material volumes. Unlike standard image segmentation techniques, this approach aims to extract accurate isosurfaces representing the boundary surfaces between different materials. The segmented volumes obtained in this way meet the requirements of the topological correctness and geometrical accuracy for isosurfacing. The approach involves three steps: first, voxels far from the boundary surfaces are labeled; then, the labels are propagated with overlapping to ascertain which materials meet near the boundary surfaces; finally, an appropriate material is selected from among the multiple labels assigned to a voxel with adaptive thresholding. Since the method consists of iterative local operations, it is easy to parallelize for fast computation. The efficiency of the technique is demonstrated here using CT scanned objects for complex shapes.

Keywords:

Volume segmentation, multi-material, adaptive thresholding, isosurface extraction.

1 INTRODUCTION

3D images (referred to here as *volumes*) consisting of scalar fields sampled on a regular grid are obtained as the output of CT/MRI scanning. Due to technological advances in scanning, volume data is widely used for many applications including medical imaging, computer graphics and industrial applications. In this research, we considered the problem of accurately extracting the boundary surfaces of CT scanned objects. This issue was mainly focused on industrial applications in which the required level of accuracy is too high for the voxel size.

Image segmentation methods are generally used to extract the boundary surfaces. So far, many segmentation approaches have been developed [1, 19, 4, 5, 6] (see also references therein). However, these techniques produce only the boundary surfaces with a level of voxel-size accuracy that is too rough for use in industrial applications. As a result, standard segmentation techniques are not suitable for solving our problem.

Another conventional method for extracting boundary surfaces is isosurface extraction [12, 3, 10]. In

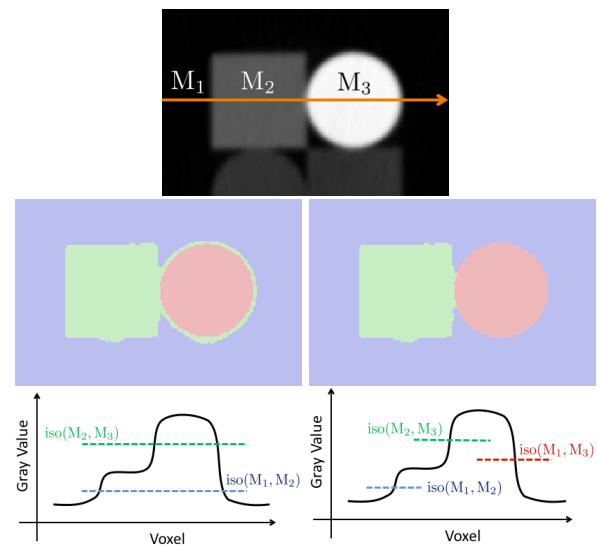


Figure 1: Segmentation results of a multi-material 2D image. The input CT image (top) is segmented (middle) with setting threshold values (bottom). Left: two iterations of thresholding with globally constants. Right: adaptive thresholding proposed in this paper.

isosurfacing, a threshold value is specified as an iso-value, and a polygon mesh approximating the isosurface is then generated. Since the positions of the mesh vertices are computed with continuous interpolation of scalar values at grid points, sub-voxel accuracy can be achieved. This simple method is successfully used when the scanned object consists of a single material

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.

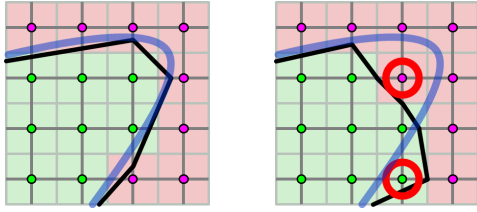


Figure 2: Geometrical accuracy of segmentation results for isosurfacing. Left: in the case the segmentation does not contradict the iso-contour (blue curve), a good polygonal approximation (black polyline) is obtained by a grid-based polygonization. Right: if the segmentation contradicts the iso-contour at the voxels indicated by the red circles, a poor approximation is obtained.

to segment the volume into its background (air) and the object itself.

Let us consider a case in which the scanned object consists of multi-materials (see Fig. 1 for an example). In such a case, one might consider simply performing two iterations of isosurfacing with setting globally constant isovalues, but this solution leads to a misclassification around material M_3 (shown in white). This problem is caused by blurring effects common in CT images, which means there are intermediate values corresponding to material M_2 between materials M_1 (black) and M_3 (white).

To extract accurate boundary surfaces from a multi-material volume, we purpose to develop volume segmentation that satisfies the following two criteria:

- Topological correctness
- Geometrical accuracy for isosurfacing

In order to satisfy the second criterion, the boundary surfaces of the segmented volume should not contradict the isosurfaces, as shown in Fig. 2. Further, the isosurfaces are defined by isovalues depending on the materials meeting at the boundaries. For example, the boundary of material M_3 in Fig.1 is defined by two isovalues $\text{iso}(M_1, M_3)$ and $\text{iso}(M_2, M_3)$ which are adaptively selected according to the materials meeting at the boundary.

In this paper, we describe a simple method for segmenting a volume using label propagation. The important point here is that the propagation generates overlapped labeling, which means multiple labels are assigned to each voxel. Since the multiple labels tell us which materials may be present at the voxel, the isovalues to be used for adaptive thresholding can be calculated from these labels.

Fig. 3 demonstrates the effectiveness of our method. The isosurfaces with incorrect topology are obtained by two iterations of isosurfacing with globally constant isovalues. In contrast, we can extract the topologically correct isosurface via our volume segmentation.

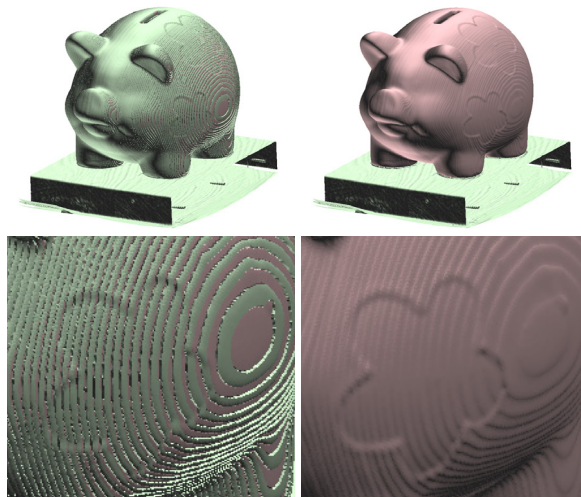


Figure 3: Polygon meshes approximating the isosurfaces of a multi-material volume using two iterations of isosurfacing with globally constant isovalues (left), and isosurfacing with adaptively selected isovalues (right).

This paper is organized as follows. We mention the related work to our method in Section 2. Section 3 describes the multi-material segmentation algorithm, and Section 4 outlines the results, details related limitations and future work.

2 RELATED WORK

Active contours

To extract edges on a noisy image, the active contour model [9] is often used. Using a closed curve as an active contour, the input image is segmented into the foreground and the background. Since polyline or spline is used for the evolution of a boundary curve, we can obtain the boundary curve with sub-pixel accuracy. It is possible to extract the boundary surface in a volume by extending polylines to polygon meshes [20]. Level-set formulations for active contours [15, 13] are also useful to extract topologically complicated boundaries. For extracting the boundaries of multi-material objects, we can use multiphase level-set functions [21].

The active contour models are effective to extract smooth boundaries in noisy volumes, for example, medical imaging. In industrial applications focused in this paper, input CT volumes are more clear and less noisy because they are measured with more radiation exposure of X-rays. Basically, it is possible to adapt the active contours to our problem, but it is not an appropriate choice for solving the problem in terms of its computational cost and accuracy. The surface evolution with minimizing an objective function is a time-consuming task, and a smoothness term in the objective function prevents the surfaces from fitting the accurate isosurfaces. Our method is specialized for fast extraction of the accurate boundaries of multi-material objects in a clear CT volume.

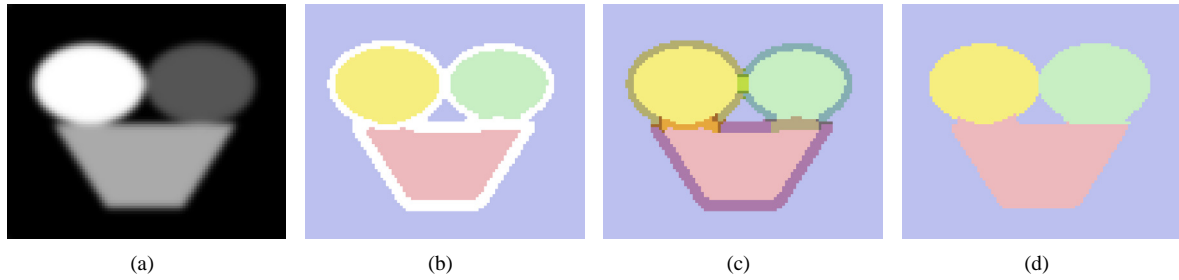


Figure 4: Algorithm overview. (a) A cross-section of an artificial volume. (b) Labeling of the voxels far from the boundary. (c) Overlapped labeling with the label propagation. (d) Labels decided by the adaptive thresholding.

Polygonization of multi-material objects

Volume segmentation is regarded as a pre-processing of polygonizing multi-material objects, that means polygonizers require the segmented volume (identified materials) as input data in order to find the boundaries. So far, several types of polygonizers have been proposed: extensions of Marching Cubes [22, 7], dual of Marching Cubes [8, 14] and an accurate method for voxels equipped with volume of fractions [2].

Volume segmentation for isosurfacing

Most closely related works are the volume segmentation algorithms proposed in [17, 18]. These segmentation algorithms are also purposed for accurate isosurfacing. In [17], binary partitioning using Graph-cut [4] are repeated after setting the seed regions obtained with region-growing [1]. In [18], a voxel-based active contour model is introduced. Both of the methods give us highly accurate and topologically correct segmentation results.

Comparing with the previous methods, our method can be parallelized easily because label propagation is performed as a local iterative operation. It is more efficient than the graph-cut or active contour based algorithms while offering a similar quality of results.

3 MULTI-MATERIAL SEGMENTATION ALGORITHM

Here we introduce an efficient algorithm for multi-material segmentation of volumes. We denote the input volume as V , which is a set of voxels $\{v_j\}$. A *gray value* denoted by g_j is assigned to voxel v_j . Given number of the materials n , the output is a segmented volume S , which is a set of *material IDs* $\{s_j\}$ each representing the material ID $s_j \in \{1, \dots, n\}$ of voxel v_j .

As shown in Fig. 4, the algorithm consists of several steps, which are briefly described below.

1. Voxels far from the boundary surfaces are labeled with thresholding of the gray values and their gradients. (Fig. 4 (b))

2. Labels near the boundary surfaces are propagated with overlapping to ascertain which materials meet at the boundaries. (Fig. 4 (c))
3. The appropriate label is selected as the material ID among the multiple labels assigned to a voxel with adaptive thresholding. (Fig. 4 (d))

The rest of this section explains the details of the above steps.

3.1 Labeling apparent regions

With thresholding, we first label the regions in which a material is apparently present. This section first explains how the threshold values are computed semi-automatically, and then describes the method of labeling voxels according to the gray values of the volume.

Semi-automatic computation of threshold values

We compute threshold values using k -means clustering on the gray values [11]. The computation is started by setting the initial threshold values $\{T_i\}$ ($i = 1, \dots, n-1$) given by the user. The threshold T_i is given as the user's guess for the isovalue $\text{iso}(M_i, M_{i+1})$. We also assume that the values are enumerated in the ascending order as $T_1 < T_2 < \dots < T_{n-1}$.

The following steps are then iterated to optimize the threshold values.

1. Define sets of voxels $\{G_i\}$ ($i = 1, 2, \dots, n$) as

$$G_i = \begin{cases} \{v_j | g_j \leq T_1\} & \text{if } i = 1 \\ \{v_j | T_{i-1} < g_j \leq T_i\} & \text{if } 1 < i < n \\ \{v_j | T_{n-1} < g_j\} & \text{if } i = n. \end{cases}$$

2. For each G_i , calculate the average value m_i and the standard deviation σ_i of the gray values $\{g_j\}$.
3. Update the threshold as $T_i = (m_i + m_{i+1})/2$.
4. Go back to Step 1 if the new thresholds are far from the old ones.

According to our experiments, the above iterative process quickly converges within ten iterations. We use a

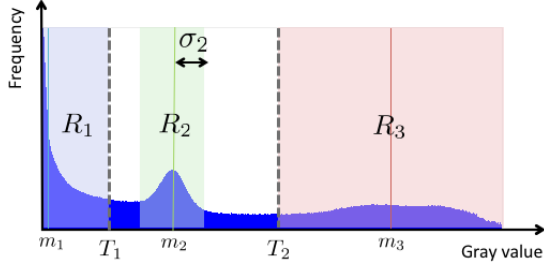


Figure 5: Setting threshold ranges $\{R_i\}$ for three materials on the histogram.

histogram of the gray values to accelerate the computation of the average value and the standard deviation via integrating the gray values in G_i .

We then define a set of ranges $\{R_i\}$ referred to as the *threshold range*, as shown in Fig. 5.

$$R_i = \begin{cases} [g_{\min}, T_1) & \text{if } i = 1 \\ [\max(T_{i-1}, m_i - \sigma_i), \min(T_i, m_i + \sigma_i)) & \text{if } 1 < i < n \\ [T_{n-1}, g_{\max}] & \text{if } i = n, \end{cases}$$

where g_{\min} and g_{\max} are the minimum and maximum of the whole gray values, respectively.

The average value m_i corresponds to the representative values of the material M_i . Using the values $\{m_i\}$, we compute the isovalues which consists of a full symmetric matrix. The (α, β) -th element of the matrix corresponds to the isovalue $\text{iso}(M_\alpha, M_\beta)$. More details are described in Section 3.3.

For volumes obtained with well-analyzed CT scanning systems, the representative values of known materials are sometimes given. In such a case, the ranges $\{R_i\}$ are computed with fixing the average values $\{m_i\}$ as the given representative values.

Labeling with the threshold range

We decide on the label for voxel v_j , whose gray value g_j is in the range of R_i and whose gradient magnitude is low. Let L_i be the set of voxels classified as the material M_i .

$$L_i = \{v_j \mid g_j \in R_i \text{ and } \|\nabla g_j\| < \varepsilon\},$$

where the gradient ∇g_j is computed using 3D Sobel filter, and ε is a user-specified parameter.

Once the voxels in L_i are decided, we eliminate isolated voxels from L_i . The elimination rule is formulated as

$$L_i \leftarrow L_i \setminus \{v_j \mid N_j \cap L_i = \emptyset\},$$

where N_j represents 6-connected neighbors of voxel v_j .

We also define *doubtful voxels* D as

$$D = V \setminus \bigcup L_i.$$

Fig. 6 shows an example of labeling with the threshold range.

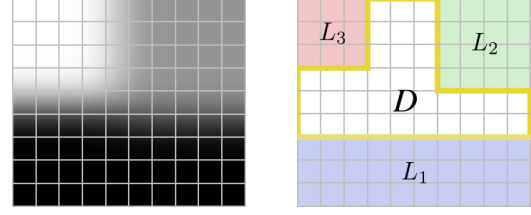


Figure 6: Labeling apparent regions. Left: gray values $\{g_j\}$. Right: labeling with the threshold ranges $\{R_i\}$.

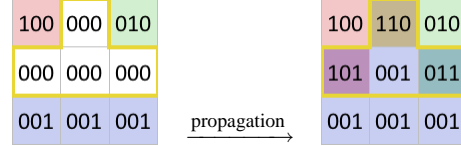


Figure 7: Label propagation with bitwise OR operation.

3.2 Label propagation

To fill out doubtful voxels D , the steps of label propagation are performed by setting apparently labeled voxels as seeds. The proposed method involves two propagations. The first aims to fill voxels without overlapping, and the second generates multiply labeled voxels while allowing overlapping of labeled regions.

Before describing the details of these propagations, we introduce the data structure for representation of multiply labeled voxels.

Data structure of labels

To represent sets of labeled voxels $\{L_i\}$ efficiently, we assign an n -bit sequence f_j to voxel v_j . The i -th bit denoted by $f_j^{(i)} \in \{0, 1\}$ is defined by

$$f_j^{(i)} = \begin{cases} 1 & \text{if } v_j \in L_i \\ 0 & \text{otherwise.} \end{cases}$$

For convenience, we denote the number of “1” in f_j by $|f_j| = \sum_{i=1}^n f_j^{(i)}$.

Label propagation without overlapping

We propagate label over doubtful voxels D by performing bitwise OR operation denoted by \vee on the neighboring labels, as shown in Fig. 7.

$$f_j \leftarrow \begin{cases} \bigvee_{v_k \in N_j} f_k & \text{if } |f_j| = 0 \\ f_j & \text{otherwise} \end{cases} \quad (1)$$

The above updating rule is simultaneously applied to all voxels in D , and is iterated until there are no voxels with $|f_j| = 0$. The two images of Fig. 8 show an intermediate step and the converged result.

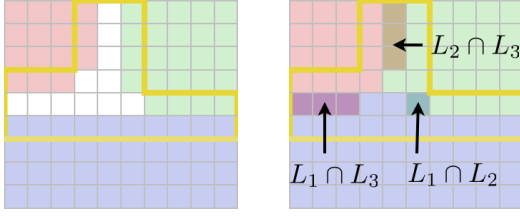


Figure 8: Label propagation without overlapping by equation (1). Left: the labels after performing a propagation step for the right image in Fig.6. Right: the converged labels.

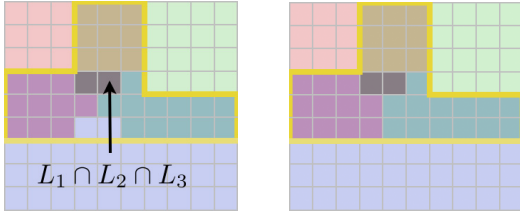


Figure 9: Label propagation with overlapping by equation (2). Left: the labels after performing a propagation step for the right image in Fig.8. Right: the converged labels.

Label propagation with overlapping

The second propagation is similar to the first one, and is as shown below.

$$f_j \leftarrow \begin{cases} \bigvee_{v_k \in N_j} f_k & \text{if } |f_j| = 1 \\ f_j & \text{otherwise} \end{cases} \quad (2)$$

The above propagation does not change the labels at the regions in D each of which is surrounded by a single label. Except in such regions, the iteration will converge when voxels in D are given at least two labels. The two images of Fig. 9 show an intermediate step and the converged result.

3.3 Adaptive thresholding

The final result of volume segmentation is obtained by selecting one label at multiply labeled voxels. The selected label at voxel v_j is set as a material ID s_j in the segmented volume S . The left image of Fig. 10 shows the segmentation result of the input shown in Fig.6.

Our segmentation is aimed at extracting the boundary surface as an isosurface. Accordingly, the isovalue $\text{iso}(M_\alpha, M_\beta)$ should be between the two gray values at the neighboring voxels segmented as materials M_α and M_β . As proposed in [17], the isovalue is calculated by

$$\text{iso}(M_\alpha, M_\beta) = (m_\alpha + m_\beta)/2.$$

Thresholding doubly labeled voxels

Before considering the thresholding of multiply labeled voxels in general, we explain a simple case involving the thresholding of a doubly labeled voxel v_j with $|f_j| =$

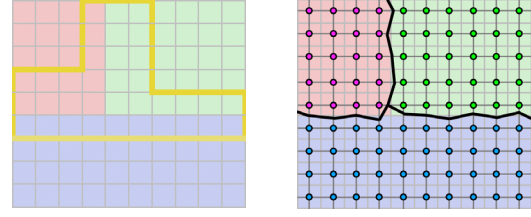


Figure 10: Adaptive thresholding. Left: the selected labels on the right image of Fig.9 with adaptive thresholding. Right: the extracted iso-contours using the segmentation result.

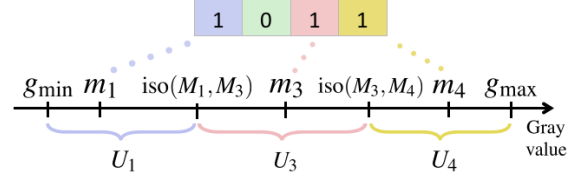


Figure 11: Ranges $\{U_k\}$.

2. Let i_1, i_2 ($i_1 < i_2$) be two material indices such as $f_j^{(i_1)} = 1, f_j^{(i_2)} = 1$, and all other bits cleaned.

In this case, voxel v_j is classified by comparing its gray value g_j and the isovalue $\text{iso}(M_{i_1}, M_{i_2})$. The material ID s_j is decided by the following rule.

$$s_j = \begin{cases} i_1 & \text{if } g_j < \text{iso}(M_{i_1}, M_{i_2}) \\ i_2 & \text{if } g_j \geq \text{iso}(M_{i_1}, M_{i_2}) \end{cases}$$

Thresholding multiply labeled voxels

In the general case of $|f_j| = K$, we divide the whole range of the gray values $[g_{\min}, g_{\max}]$ into K sub-ranges. Then, the range containing the gray value g_j is found to decide the material ID s_j .

We first prepare the sequence $\{i_1, \dots, i_K\}$ satisfying $f_j^{(i_k)} = 1$ ($k = 1, \dots, K$). The following ranges $\{U_{i_k}\}$ are then created, as shown in Fig. 11.

$$U_{i_k} = \begin{cases} [g_{\min}, \text{iso}(M_{i_1}, M_{i_2})] & \text{if } k = 1 \\ [\text{iso}(M_{i_{k-1}}, M_{i_k}), \text{iso}(M_{i_k}, M_{i_{k+1}})] & \text{if } 1 < k < K \\ [\text{iso}(M_{i_{K-1}}, M_{i_K}), g_{\max}] & \text{if } i = K \end{cases}$$

Finally, the material ID s_j is set to the index satisfying the following relationship.

$$g_j \in U_{s_j}$$

See the appendix for more details on the implementation of the above procedure.

Isosurface extraction

Given the input volume V and its segmentation S , an isosurface is polygonized with a grid-based algorithm, for example Marching Cubes [12]. As shown in the right image of Fig. 10, we generate mesh vertices which

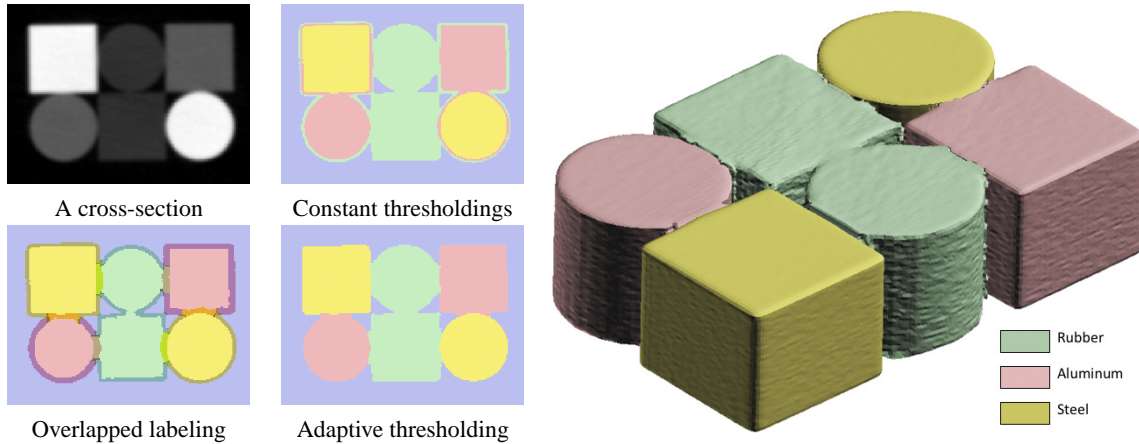


Figure 12: A segmentation result of the CT volume obtained by scanning blocks consisting of steel, aluminum and rubber. Left: cross-sections of the volume. Right: polygonized isosurface using our segmentation.

are the intersection points of the grid-edges and the isosurface, and the vertices are then connected with polygons in the grid-cells.

If two neighboring voxels denoted by v_j and v_k are associated to different material IDs ($s_j \neq s_k$), a mesh vertex is generated on the grid-edge whose end points are v_j and v_k . Let \mathbf{p}_j and \mathbf{p}_k be the coordinates of the voxels v_j and v_k , respectively. The coordinates of the mesh vertex \mathbf{p} are computed using the following linear interpolation.

$$\mathbf{p} = (w_k \mathbf{p}_j + w_j \mathbf{p}_k) / (w_j + w_k), \quad (3)$$

where the weights $w_j = g_j - \text{iso}(M_{s_j}, M_{s_k})$ and $w_k = \text{iso}(M_{s_j}, M_{s_k}) - g_k$.

If $w_j \cdot w_k < 0$, the isosurface does not intersect to the grid-edge, that means the material IDs contradict the isosurface as shown in the right image of Fig. 2. In this case, the mesh vertex is located at the middle point of \mathbf{p}_j and \mathbf{p}_k . As reported in the next section, this case happens only less than 10%.

4 RESULTS AND DISCUSSION

Results

Fig. 12, 13 and 14 show the results. The sizes of the input volumes are summarized in Table 1.

In Fig. 12, the input CT volume consists of 4 materials: steel, aluminum, rubber and air. The left four images show cross-sections of the volume. The right image shows a polygon mesh approximating the isosurface in the volume. The isovalues of the isosurface are adaptively selected in terms of the segmented volume.

Fig. 13 and Fig. 14 show the resulting isosurfaces representing engine parts. The input CT volumes consist of steel, aluminum and air.

Timing and quality

In Table 1, we demonstrate the speed of our segmentation algorithm. The timing is measured on a desktop

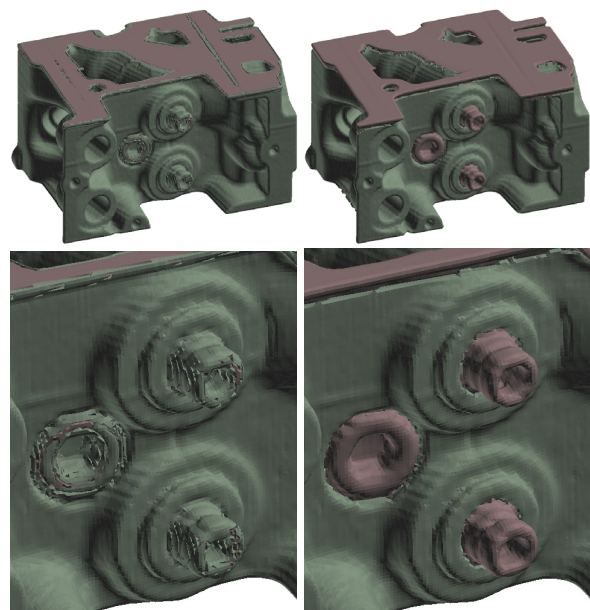


Figure 13: Isosurfaces of the CT scanned engine parts consisting of steel (pink) and aluminum (green). Left: the isosurfaces obtained with the globally constant isovalues. Right: the isosurface obtained with the adaptive isovalues.

computer equipped with Intel Core i7 4-cores of 2.93 GHz and 16 GB main memory. Since the algorithm can be parallelized in a straightforward way, the computational time proportionally improves with increasing the number of threads.

Table 2 shows the segmentation quality for isosurfacing. For each material ID, a polygon mesh is firstly generated by Marching Cubes [12] with the same grid-resolution as that of the input volume V . Then, the quality is computed as the rate of the mesh vertices which are successfully located on the isosurface ($w_j \cdot w_k \geq 0$ in equation (3)). In all results, more than 90% vertices are generated on the isosurfaces.

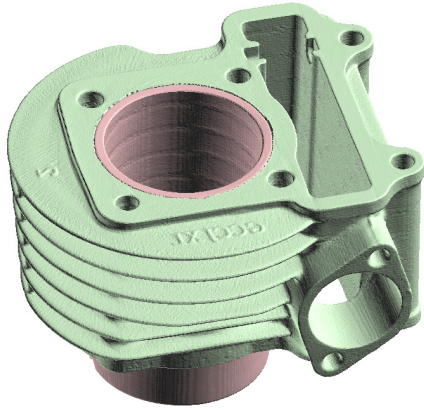
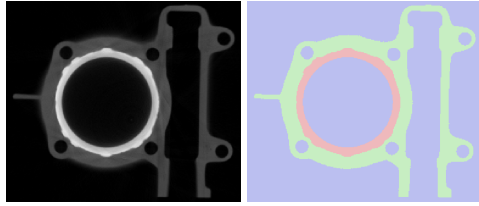


Figure 14: A result of the CT volume segmentation (the cylinder head of a motor-bike). Top: cross-sections of the input and segmented volume. Bottom: the isosurface obtained with adaptive isovalues.

Table 1: Computational time of volume segmentation.

Name	Size	#threads	Time (sec.)
Blocks (Fig. 12)	$200 \times 160 \times 50$	1	0.2082
		2	0.1264
		4	0.0766
Engine (Fig. 13)	$256 \times 256 \times 256$	1	1.7132
		2	0.9099
		4	0.5408
Cylinder (Fig. 14)	$512 \times 512 \times 178$	1	4.6464
		2	2.4587
		4	1.3674

Table 2: Isosurface quality.

Name	Material	#vertices	Quality (%)
Blocks (Fig. 12)	Air	67k	96.02
	Rubber	28k	91.20
	Aluminum	27k	95.36
	Steel	25k	99.07
Engine (Fig. 13)	Air	314k	98.98
	Aluminum	297k	97.32
	Steel	80k	94.42
Cylinder (Fig. 14)	Air	746k	99.84
	Aluminum	714k	99.81
	Steel	209k	99.92

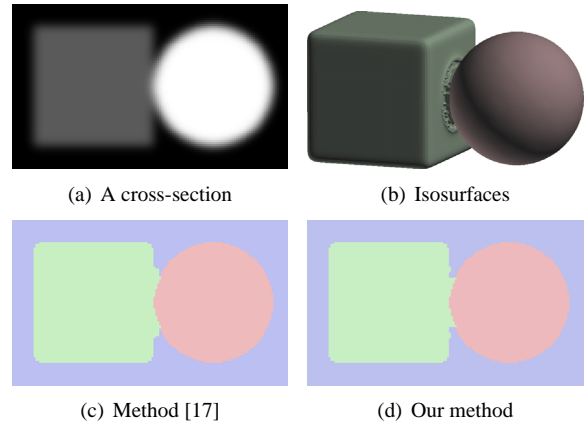


Figure 15: A comparison with the method [17] using an artificial volume.

Comparison

Fig. 15 shows a comparison with the graph-cut based method [17] using an artificial volume which simulates a scanning result of a sphere (high gray value) and a cube (middle gray value). We applied Gaussian convolution to produce blurring effects in CT scanning after rasterizing the cube and sphere. The variance of the Gaussian kernel is six voxels. The size of the volume is $288 \times 192 \times 192$.

To apply the method [17], we used the public available software *VolCut* [16]. Fig. 15(c) and (d) show cross-sections of the segmentation results. As summarized in Table 3, almost the same quality results for isosurfacing are obtained by our method and the method [17].

In order to evaluate the accuracy of the segmentation, we also counted the misclassified voxels by comparing the results and the rasterization of the cube and sphere. In the result of [17], about twice voxels which should be classified as the background are misclassified as the cube. The misclassification is mainly observed near the contact part of the cube and sphere where the boundary surface of the background has sharp features. This problem is caused by the smoothing effects of the graph-cut.

From the above comparison, we achieve almost the same level of the quality while the speed is much faster (more than ten times faster).

Parameter setting

In our segmentation method, the resulting segmentation depends mainly on the user-specified parameter ϵ . Fig. 16 shows a part of the cross-section of the volume in Fig. 14, and the effects of changing ϵ . Thin parts of materials in the CT image do not appear in the segmented image with too small ϵ (Fig. 16(b)), while unnecessary parts appear with too large ϵ (Fig. 16(d)). Since an appropriate ϵ value is not computed with the

Table 3: Quantitative comparison with the method [17] by using the volume shown in Fig. 15.

Method	Time (sec.) Single thread	Isosurface quality (%)			#Misclassified voxels		
		Sphere	Cube	Background	Sphere	Cube	Background
Method [17]	12.0	99.37	97.70	98.92	4,592	5,632	3,584
Our method	0.95	99.46	95.77	97.74	4,516	5,632	1,876

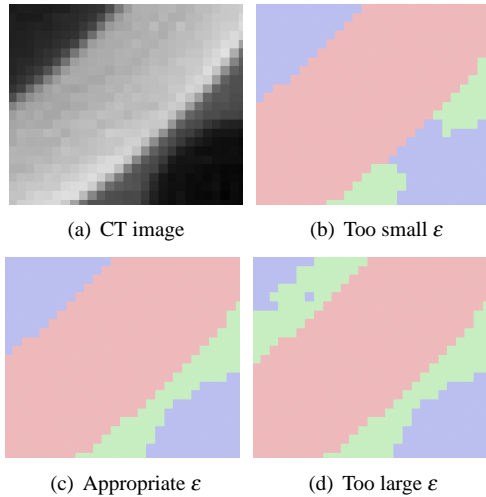


Figure 16: Effects of the gradient threshold ϵ .

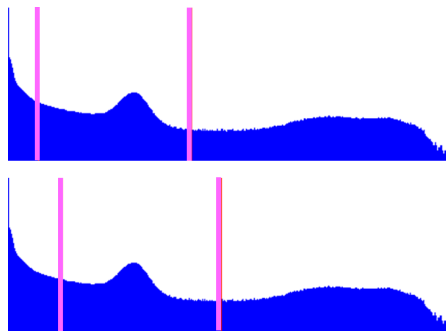


Figure 17: Two different selections of the initial $\{T_i\}$ both of which will converge to Fig. 5.

current method, a number of ϵ values must be examined to obtain a segmented image suitable for the user.

In the case representative CT values $\{m_i\}$ are unknown, the user must guess the initial thresholds $\{T_i\}$. Then, values $\{m_i\}$ are automatically computed through the iterative process described in Section 3.1. According to our experiments, a rough selection of $\{T_i\}$ is fine for computing the appropriate values for $\{m_i\}$. For example, Fig. 17 shows two different selections of $\{T_i\}$ on the histogram of gray values, and both of the selections give the same $\{m_i\}$ (shown in Fig. 5) within ten iterations.

Limitation and future work

Since our fast segmentation algorithm is specialized for CT volumes used for industrial applications, highly noisy volumes used for medial imaging might not be

segmented correctly. This problem is a drawback of the simplicity of the algorithm. We plan to make the algorithm more noise-robust in order to deal with various kinds of volumes while keeping its high-speed.

Another limitation of our algorithm is that it is difficult to correctly find thin-structures consisting of a few voxel thickness (tubes and sheets). The reason of this problem is that the CT value on such a thin-structure can not reach to the corresponding representative value because of blurring effects. According to our experiments, at least four or five voxel thickness is required to find appropriate seed voxels and doubtful voxels.

ACKNOWLEDGMENTS

Authors would like to thank for anonymous reviewers for valuable comments and suggestions. The CT volumes are courtesy of Toyota Motor Corporation (Fig. 2 and 12), Michael Bauer (Fig. 3), General Electric (Fig. 13), and VCAD System Research Program at RIKEN (Fig. 14). This work was partly supported by KAKENHI (22246018).

REFERENCES

- [1] R. Adams and L. Bischof. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647, 1994.
- [2] J.C. Anderson, C. Garth, M.A. Duchaineau, and K.I. Joy. Smooth, volume-accurate material interface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 16(5):802–814, 2010.
- [3] J. Bloomenthal. Polygonization of implicit surfaces. *Computer-Aided Geometric Design*, 5(4):341–349, 1982.
- [4] Y. Boykov and M.-P. Jolly. Minteractive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *International Conference on Computer Vision (ICCV)*, pages 105–112, 2001.
- [5] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.
- [6] A. Delong, A. Osokin, H.N. Isack, and Y. Boykov. Fast approximate energy minimization with label costs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2173–2180, 2010.

- [7] T. Fujimori and H. Suzuki. Surface extraction from multi-material ct data. In *Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05)*, pages 319–324, 2005.
- [8] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002)*, 21(3):339–346, 2002.
- [9] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987.
- [10] F. Labelle and J. R. Shewchuk. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, 26(3):#57, 2007.
- [11] S. P. Lloyd. Least square quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.
- [12] W. E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3D surface construction algorithm. *Computer Graphics (Proceedings of SIGGRAPH '87)*, 21(3):163–169, 1987.
- [13] S. Osher and R. Fedkiw. *The Level Set Method and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.
- [14] P. Powei Feng, T. u, and J. Warren. Piecewise tri-linear contouring for multi-material volumes. *Advances in Geometric Modeling and Processing (Lecture Notes in Computer Science)*, pages 43–56, 2010.
- [15] J. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.
- [16] H. M. Shammaa. Volcut. <http://sites.google.com/site/mhaithamshammaa/>.
- [17] H. M. Shammaa, Y. Ohtake, and H. Suzuki. Segmentation of multi-material ct data of mechanical parts for extracting boundary surfaces. *Computer-Aided Design*, 42(2):118–128, 2010.
- [18] H. M. Shammaa, H. Suzuki, and Y. Ohtake. Creeping contours: A multi-label image segmentation method for extracting boundary surfaces of parts in volumetric images. In *Asian Conference on Design & Digital Engineering*, pages 603–610, 2010.
- [19] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
- [20] I. Takahashi, S. Muraki, A. Doi, and A. Kaufman. Three-dimensional active net for volume extraction. In *Visual Data Exploration and Analysis (SPIE Proceedings of SPIE Volume: 3298)*, pages 184–193, 1998.
- [21] L. A. Vese and T. F. Chan. A multiphase level set framework for image segmentation using the mumford and shah model. *International Journal of Computer Vision*, 50:271–293, 2003.
- [22] Z. Wu and J. M. Sullivan, Jr. Multiple material marching cubes algorithm. *International Journal for Numerical Methods in Engineering*, 58(2):189–207, 2003.

APPENDIX

Algorithm 1 Adaptive-Thresholding

```

input:
n - the number of materials
f - n bit sequence
g - gray value of a voxel
output:
i - material ID

i ← 0
for k = 1 → n do
  if f(k) = 1 then
    if i = 0 then
      i ← k
    else
      iso ← (mi + mk)/2
      if g < iso then
        return i
      end if
    end if
  end if
  k ← k + 1
end for
return i

```

Photo Quality Assessment based on a Focusing Map to Consider Shallow Depth of Field

Dong-Sung Ryu, Sun-Young Park, Hwan-Gue Cho
Dept. of Computer Science and
Engineering, Pusan National
University, Geumjeong-gu BUSAN,
South KOREA, 609-735
dsryu99,parksy,hgcho@pusan.ac.kr

ABSTRACT

Proliferation and advances in digital cameras encourage people to take many photos. However, the number of photos that people can access is increasing exponentially. Good quality photo selection is becoming burdensome. In this paper, we propose a novel method to evaluate photo quality considering DoF (Depth of Field) based on a focusing map. The focusing map is a form of saliency map classified into four levels based on the spatial distribution of Canny edges. We implemented it in a CUDA environment to improve the speed of focusing map generation. In order to evaluate our method, we tested our feature on the four classified 206 photos; then, we compare our method to a photo set manually classified by a user. The proposed measure efficiently assesses the photos with DoF. Especially, the expert group who used DSLR camera agreed that our photo assessment measure is useful.

Keywords: digital photo, photo assessment, depth of field.

1 INTRODUCTION

Generally, high quality photos satisfy three principles: 1) a clear topic, 2) a focus of attention on the subject, and 3) the removal of objects that distract attention from the subject [4]. DSLR camera users control the camera parameters, for example, aperture and shutter speed, to take good quality photos. The assessment of these photos depends on how to arrange and present photo subject clearly. In the case of DoF photos, it is very easy to know which object is its subject due to their out-of-focussed background region. Therefore, it is important for DSLR camera users to evaluate DoF features in selecting good quality photos. In this paper, we define high quality photos as those that have shallow DoF features. Figure 1 shows the characteristics of good quality photo with DoF. The deep DoF photo is insufficient for a good quality photo, since its topic (the book) in Figure 1 (a) is not presented clearly (The region of the book is blurred).

Most studies related to good quality photo evaluation proposed combined measures of contrast, blur, and hue count to evaluate image degradation caused by noise, distortion, and compression artifacts. These studies are efficient in distinguishing defective (i.e. blurred) pho-



Figure 1: Examples of good quality photos considering difference in DoF. (a) A deep DoF photo. (b) A shallow DoF photo.

tos [2, 6]. In contrast to these works, we proposed a DoF measure based on the focusing map. The focusing map is a form of saliency map with four classification levels based on the spatial distribution of Canny edges. We calculated the total weight of all Canny edge pixels allocating four weight values corresponding to the focusing level.

2 RELATED WORK

Photo assessment studies measure color contrast and blur caused by camera shaking, overexposure and mis-configured camera settings. Recent studies deal with visual features in photography more importantly than ever before. Ke *et al.* designs high level image features to measure perceptual differences [2], considering the spatial distribution of edges, color distribution, hue count, blur and contrast. They combined these features using Bayes rules. Datta summarized 56 features to consider aesthetics in photography [1]. These classifiers are built using support vector machines and classification trees. That work focuses on the relationship

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

WSCG 2011 conference proceedings
WSCG'2011, January 31 – February 3, 2011
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

between emotions that pictures arouse in people, and their low-level content. Sun *et al.* proposed the RFA (Rate of Focused Attention) measurement based on a saliency map of computational visual attention model to consider human visual systems for photo assessment [5]. This model simulates the attention mechanism of a human visual system, most of which use a saliency map or a conspicuous map to describe how salient (interesting) a location in the visual field is. All these recent studies tried to find optimized visual features to mimic human perception in photo assessment.

Luo *et al.* designed a composition that depicts the organization of all graphical elements within a photo for professional photos [4]. They use a log-likelihood of derivatives with a blurring kernel of size $k \times k$ ($1 \leq k \leq 50$) to extract the subject region and to search the interest region of photos.

3 DEPTH OF FIELD PHOTO EVALUATION

Figure 2 depicts the framework of our method. It is important how a human detects a recognizable region in a photo. We first extract the Canny edge from the blurred photo to compute the focusing map to achieve this. The focusing map estimates the visual attention regions of a photo which consists of four regions. These visual regions are classified according to the count of Canny edge pixels in a designated mask. Then, we can calculate the score of the DoF photo comparing all extracted Canny edge pixels with the focusing map pixels.

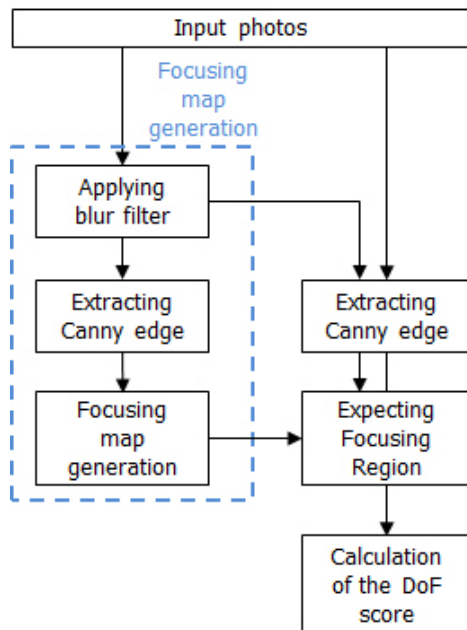


Figure 2: Framework of the proposed method.

3.1 Focusing Map Generation

Most studies use a saliency map to estimate the visual attention region, since it describes how salient a location in the image is. We used a similar concept of a saliency map [5]. However, we divide the photo regions into four classified pixel sets based on their intensity, R_k ($k \subseteq \{r, o, y, w\}$, each element depicting red, orange, yellow and white, respectively). These four levels are determined by the number of Canny edge pixels in a designated mask, m . Since Canny edge extraction considers the direction of pixels variation and double thresholds, human recognizable edges are extracted by using Canny edge detection algorithm. The description of the proposed focusing map generation is as follows. First, we apply the blur (Gaussian Filter) and Canny edge filter to the original images. This is to find the recognizable edge pixels. Then, we calculate the focusing value based on the count of Canny edge pixels in the designated mask (m) at each pixel. The mask size is $\sqrt{(w+h)/2}$, where w and h are the photo width and height, respectively. Third, in order to prevent the sequential Canny edge pixel counting from left top to right bottom image, all pixels in the focusing map are randomly shuffled. Then, they are sorted in the pixel value of the focusing map order. Finally, the sorted pixels are separated into four pixel groups (R_r , R_o , R_y , and R_w) whose sizes are 12.5%, 12.5%, 25%, and 50%, of the total number of pixels, respectively. These four pixel groups is determined by a naive experiment to be able to evaluate DoF photos in our 206 photo sets. This experiment is to maximize F-score of Section 4 Experiment. Note that each pixel on the image are divided into four pixel groups in the designated ratios. The role of these designated pixel subsets is to estimate which regions are focused in the entire image.

Let us explain an example of focusing map generation, as shown in Figure 3. We assumed the mask size is 3×3 . Figure 3 (a) shows the extracted Canny edge pixels. We execute the mask operation every photo pixel to calculate the number of Canny edge pixels in the 3×3 mask. Thus, we can obtain the focusing map, as shown in Figure 3 (b).

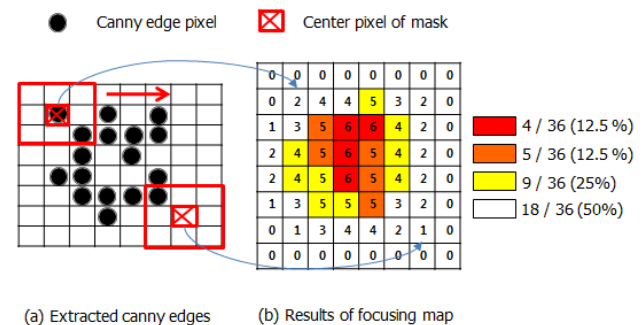


Figure 3: Example of the proposed focusing map.

The proposed focusing map is simple to implement. However, it has many iterative mask operations to scan all the pixels in a photo. Assume that we have a 1024×768 resolution photo, whose mask size is 100×100 . It will need 7,864,320,000 pixel traverse operations (a time-consuming task). We applied CUDA (Compute Unified Device Architecture) to implement this procedure to improve the speed of focusing map generation. This GPU implementation improves system performance more than approximately 15 ~ 20 times compared to CPU implementation. For example, the CPU implementation took about 33 s on average for the $1,024 \times 768$ resolution photos described in Table 1. With GPU implementation, it took approximately 1.9 sec on average.

3.2 Shallow and Deep Depth of Field Photos

Most professional photographers intentionally create a blurred area in the background region with a shallow DoF [3]. Therefore, a well-taken photo has a high possibility of having an intensive concentration of edges on an interesting object or region. We use the Canny edge detection algorithm to extract the human recognizable edges. The Canny edge considers the direction of pixel variation and double thresholds. We extract human recognizable edges from photos more readily than from other edge extraction methods. The main idea of our DoF photo assessment method is how to consider if the edges extracted from a photo image are in the in-focus or out-of-focus region. If edge pixels can be obtained from the in-focus region but no edge pixels can be obtained from the out-of-focus region, simultaneously, the photo is a shallow DoF photo. Figure 4 shows the pro-

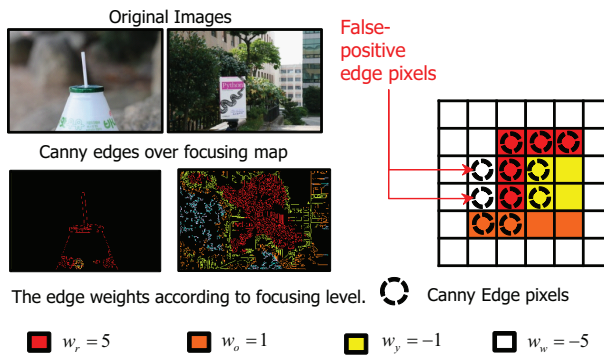


Figure 4: DoF evaluation example. We compute the number of the Canny edge pixels in the mask corresponding to the level of the focusing map. The weight values of the focusing map, w_r , w_o , w_y , and w_w are 5, 1, -1, -5, respectively. The negative weights on the yellow and white regions are false-positive pixels, since they represent the unfocused region.

cess of the DoF quality evaluation method to consider an edge distribution that depicts the level of focus. We compute the number of the Canny edge pixels in the

mask m corresponding to the level of the focusing map. The weight values for focusing map, w_r , w_o , w_y , and w_w are 5, 1, -1, -5, respectively. Our proposed focusing value $f(p)$ for each pixel p is defined by :

$$f(p) = \sum_{np \in m} \frac{\text{pixels}(\text{Canny}(np))}{|\text{dist}(p, np)|}, \quad (1)$$

where $\text{dist}(p_i, p_j)$ means the Manhattan distance between two pixels, p_i and p_j . $\text{Canny}()$ depicts the pixel sets extracted from Canny edge detection algorithm. Note that we assume the red (R_r) and orange region (R_o) is in-focus, whilst the other regions are out-of-focus. The other regions contain information that is less important than R_r and R_o , since there are less edge pixels. However, we also need the other regions (R_y , R_w) to determine the false-positive case, as shown in Figure 4. Finally, we calculate the total score of the edge pixels according to the level of focus by :

$$E_d(P) = \sum_{p \in P} (w_k(p) \cdot f(p)) / \text{pixels}(e_c(P)), \quad (2)$$

where e_c , $\text{pixels}(I)$ and $w_k(p)$ are the extracted edge pixels using Canny edge detection, the number of pixels stored in region I and the weights according to the focusing level, respectively. Figure 5 shows the result of our DoF assessment. You can see that the focused regions (R_r and R_o) are becoming clustered on the central area (near the rabbit objects) from left to right in the sequence .

4 EXPERIMENTS

We invited twelve digital camera users, divided into two groups based on their photo taking ability. The beginner group, $G1$, consists of four camera users who can take pictures using their compact digital camera controlling the embedded camera modes, for example, M (Manual), A (auto Aperture), S (auto Shutter speed) and P (all Programmed) modes. The expert group $G2$ of six users could control the detailed camera parameters, such as aperture, shutter speed, and ISO. They also have sufficient experience, shooting pictures using their DSLR cameras for more than two years.

We collected 206 photos from five categories, as described in Table 1. The general photo sets A and B consist of random shots of photos taken, while in motion (A), and still photos (B). Several experiment conditions were controlled for the experimental photo sets. We used DSLR cameras, controlling camera parameters, such as aperture and shutter speed, to shoot photo set C . The camera was shaken when we took photos for the final photo set, D .

The participants in our user study were asked to construct three photo sets according to the photo quality. The input photo sets used in this experiment were 30

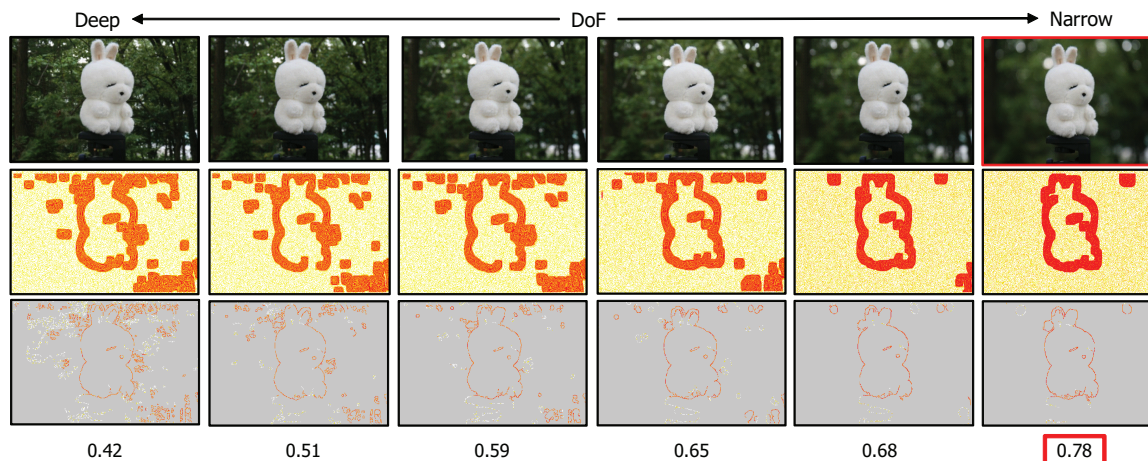


Figure 5: Evaluation results of contrast and DoF features. Experimental photos (C, D, and E in Table 1) for DoF. We took these photos controlling shutter speed and aperture parameters. The focus regions become clustered on the central object from left to right.

Purpose (# of photos)	Sets	Taken method	# of photos	User	eval. class	precision		recall		F-score
						avg.	std.	avg.	std.	
General photos (100)	A	traveling	69	G1	upper	0.75	0.12	0.88	0.14	0.81
	B	still shots	31		middle	0.77	0.14	0.82	0.07	0.79
Experimental photos (106)	C	DoF	60		lower	0.87	0.07	0.87	0.08	0.87
	D	shaking photos	46	G2	upper	0.86	0.08	0.85	0.09	0.85
			middle		0.85	0.10	0.89	0.07	0.87	
			lower		0.84	0.06	0.85	0.05	0.84	
				avg. score		0.82	0.11	0.87	0.10	0.84

Table 1: Input photo sets used in this experiment.

randomly selected from each of the shuffled input photo sets.

We calculate the average precision, and recall, as described in Table 2, to compare our classification result to those classified manually by the users. Precision and recall are two widely used metrics for evaluating the correctness of a pattern recognition algorithm. When using precision and recall, the set of possible labels for a given instance is divided into two subsets, one of which is considered “relevant” for the purposes of the metric. Recall is then computed as the fraction of correct instances among all instances that actually belong to the relevant subset, while precision is the fraction of correct instances among those that the algorithm believes to belong to the relevant subset. A measure that combines precision and recall is the harmonic mean of precision and recall. Since F-score is the measure recall and precision are evenly weighted, it means the reliability of their experiment. The proposed measure efficiently assesses DoF (its precision and recall exceed 0.82). Especially, the expert group agreed that our photo assessment measure is useful in evaluating the professional photos. However, since the beginner group, G1, prefers the high contrast photos, their experiment results (precision and recall) are somewhat poor (both values are less than 0.87).

Table 2: Photo quality assessment. The photo sets used are described in Table 1. We investigate precision and recall with their manual operation outcome to evaluate our metrics.

5 CONCLUSION

In this paper, we proposed a novel method to assess photo quality based on a focusing map. The focussing map is a form of a saliency map that is classified by Canny edge distribution. Its goal is to simulate the attention mechanism of a human visual system. This is simple to implement. It is implemented in CUDA to decrease the time required for image processing for a large resolution. We also conducted an experiment based on the precision, recall and F-score, with four photo sets (206 photos) to compare our performance with a user’s manual evaluation. The experiment shows the proposed measure perform well compared to manual user evaluation (The precision and recall exceed 0.8). Especially, the expert group agreed that our photo assessment metrics are useful to evaluate each photo.

In this paper, we considered only one criteria to assess the Depth of Field. However, it is insufficient to assess general photos, since there are many features to be considered (for example, blur, contrast and exposure). Measurements developed for other features would need a combination method for multiple features, for example, Ke’s naive Bayes classifier [2]. This method en-

ables various metrics that are dependent on each other. Therefore, we propose the development of further metrics for photo quality evaluation (for example, blur and contrast) and a combination method to integrate these metrics naturally. In this paper, we conducted on naive experiment to find the four classified levels of four focusing region, R_r, \dots, R_w . Although, we deduced it from a naive experimental result, it is also important to find the scientific reason of the four classified level ratio.

ACKNOWLEDGMENTS

This work was supported by the IT R&D program of MKE/MCST/KEIT (KI001820, Development of Computational Photography Technologies for Image and Video Contents).

REFERENCES

- [1] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. Studying aesthetics in photographic images using a computational approach. In *In Proc. of ECCV*, pages 7–13, 2006.
- [2] Yan Ke, Xiaoou Tang, and Feng Jing. The design of high-level features for photo quality assessment. In *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 419–426, 2006.
- [3] Renting Liu, Zhaorong Li, and Jiaya Jia. Image partial blur detection and classification. In *Proc. of IEEE Computer Vision and Pattern Recognition*, pages 1–8, 23-28 2008.
- [4] Yiwen Luo and Xiaoou Tang. Photo and video quality evaluation: Focusing on the subject. In *Proc. of the 10th European Conference on Computer Vision*, pages 386–399, Berlin, Heidelberg, 2008.
- [5] Xiaoshuai Sun, Hongxun Yao, Rongrong Ji, and Shaohui Liu. Photo assessment based on computational visual attention model. In *Proc. of the 17th ACM MM*, pages 541–544, 2009.
- [6] Niranjan Damera Venkata, Thomas D. Kite, Wilson S. Geisler, Brian L. Evans, and Alan C. Bovik. Image quality assessment based on a degradation model. *IEEE Transactions on Image Processing*, 9(4):636–650, 2000.

Image-based Animation

Biswarup Choudhury
Indian Institute of Technology
Bombay & ETRI, South Korea
biswarup@cse.iitb.ac.in

Ambareesha Raghothaman
Indian Institute of Technology
Bombay
ambareesha04@gmail.com

Sharat Chandran
Indian Institute of Technology
Bombay
sharat@cse.iitb.ac.in

ABSTRACT

Animation has evolved over the years – from the early days of 2D animation to the present technology of using GPU-based shader programs for providing complex, photorealistic lighting. One thing has, however, remained constant – a geometrical model has been considered essential in computer animation. In this paper, we propose an alternative.

An image-based framework is presented for creating arbitrary motions of an object using only captured images of the object; no geometry of the object or the environment is provided by the user. *Photorealism is an immediate side effect as a consequence.* Specifically we preprocess a set of images of a static object under a set of carefully chosen lighting configurations. Now given an arbitrary environment in the form of images again, and any arbitrary three-dimensional path that the object is desired to move, our algorithm creates a motion sequence of the object — realistically composed in the new environment.

Keywords: Animation, image-based rendering, visual hull, image-based relighting, virtual and augmented reality.

1 INTRODUCTION

Traditionally computer-generated animation requires knowledge of object geometry, reflectance functions, lighting configurations, and the desired motion sequence. All of these are used in conjunction with computationally intensive global illumination techniques to render each frame of the desired motion. In this paper, we show there is an alternative. We create the same animation with no knowledge of the geometry of the objects from the user, and no surface properties. Even the environment that the objects are expected to ‘live’ in is provided in the form of images.

1.1 Motivation

Behind the scenes of any computer animated movie, the “layout crew” choreographs the characters in the set and uses a virtual camera to create shots. This painstaking process of changing the lighting, viewpoint, and character motion is repeated several times to capture the emotion of each scene. To get an idea of how long it takes to do such rendering, we quote Pixar who use a Renderfarm where each frame takes about six hours to render. Pixar claims that some frames have taken as many as ninety hours [19]. These numbers may represent the time taken to render a full resolution frame. Nonetheless, the time taken for a lower resolution frame is high, and more significantly, the whole process is laborious.

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

WSCG 2011 conference proceedings,
WSCG’2011, January 31 – February 3, 2011
Plzen, Czech Republic.
Copyright UNION Agency – Science Press

- **Question:** Can we reduce the production time and resources substantially?

Answer: Yes, discounting the preprocessing time (which is highly dependent on the number and types of objects), our unoptimized implementation takes roughly a minute per frame.

- **Question:** If we had prior footage of an environment, can we realistically embed in it a shot, where a Ferrari is yanked, and hurled into the air ?

Answer: Creating such an animation would require the use of a Ferrari in a set of a studio (or its complicated geometric model), but is infeasible due to commercial production reasons. Our method achieves similar effects (please see the smaller Spaceship video in the supplementary material in which three Spaceships fly in the Uffizi Gallery [7]).

1.2 Contributions

Clearly it is desirable to be able to quickly create an animation of an object performing an “impossible” motion in director-mood dependent environments, edit (camera position, lighting, and object path), and again quickly re-render it. An image-based solution is a promising alternative, which would enable the director to create any desired animation using only a set of captured images of the object. This paper describes a method that combines existing techniques to achieve this framework. Specifically, the novelty in our work is summarized as:

1. **Motion:** Unlike prior image-based methods, our static object captured in the studio can move freely in space at the whims and fancy of the user and live in any virtual or camera captured real world. The framework allows any complex motion, and does not need any object geometry from the user.

2. **Lighting:** Since the object is rendered in a novel environment, we cannot simply copy the studio acquired images and stick it to a frame of the animation by applying matting techniques. Worse, since the ‘static’ object is moving in every frame of the animation, at every frame, the lighting directions (of the novel environment) with respect to the object changes.
3. **Visual Hull:** Any image-based framework inevitably has to balance storage costs with accuracy. We use a variation of the popular visual hull technique to create a dense but “lightweight” intermediate representation.

2 RELATED WORK

Considerable research has been done to create animation sequences from captured videos. These can be categorized into video-based rendering (VBR), and video-based animation (VBA) techniques. The aim of VBR techniques ([15, 24, 29, 27]) is to render novel views of dynamic scenes using multiple-view video capture. These techniques reconstruct (often across the time domain) the surface of the moving object, and then use the video and the reconstructed geometry to create novel view sequences. Some VBR techniques ([4, 20]) reanimate dynamic scenes using model-based reconstruction techniques, which fit a generic model to observations from multiple views, but suffer characteristic limited visual quality. On the other hand, VBA techniques ([21, 22, 2, 1]) provide a representation of dynamic scenes captured from videos allowing synthesis of novel image sequences. These novel sequences are synthesized by concatenating captured video segments based on a transition graph. These techniques use similarity metrics to identify frames in the video which are candidate transition points. Some techniques have been proposed which use the concepts of VBR and VBA techniques to render novel views of reanimated animation sequences. For example, techniques such as [9, 25] focus was on capturing the appearance of a person performing a specified cyclic action (walking or jogging), and then reproducing the character performing the same motion under variable viewpoint and illumination.

While the techniques mentioned above use videos as input, there are techniques which use images to create novel animation sequences. In [3], the authors create an illusion of realistic animation of an object from images captured at different instant of time, by inserting motion blur between the consecutive images. Most image-based techniques segment the source image into layers, and create animations by applying dynamic brush strokes [11] or displacement maps [11] to each layer. These layers are then recomposited to form the animated sequence. A recent work [28] uses a source image of a group of moving animals (or birds) to identify

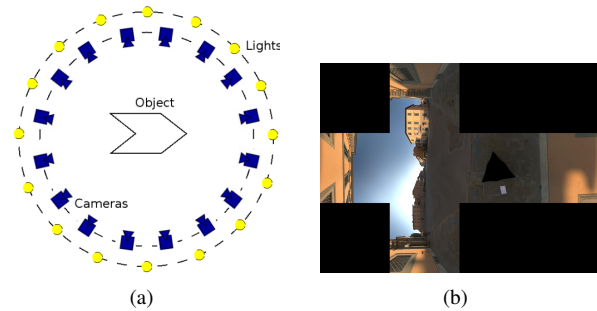


Figure 1: The first schematic is the cross-section view of our capture space composed of two concentric spheres, one with cameras (blue) and the other with point light sources (yellow). The novel environment (b) specified as a cube map, is used to relight each frame of the desired animation.

the *ordered cyclic sequence of poses* of any animal in the group while in motion. This sequence is then used to create a motion cycle, which can be used to create novel animation sequences of the animal(s). This technique assumes that all the “key-poses” of the animal in motion are present in the source image, or they need multiple images to identify all the key-poses of the animal.

In all the techniques mentioned above, reanimating the object is mostly limited, primarily to the repetitive cyclic motions (human walking/jogging, animals walking, birds flying, water flowing) as is present in the input videos/images. In contrast, we propose a framework to synthesize arbitrary, complex animation sequences of the object, without recording any motion of the object. We require only images of the static object.

Thus, image-based animation as we conceive with all its variety has not been sufficiently explored.

3 OVERVIEW

The general idea of image-based animation we propose in this paper can be implemented in various frameworks. Here we provide the following framework.

Inputs for Preprocessing: The primary input to our method are images of a single object captured from calibrated cameras which are distributed on a sphere (Figure 1(a)). For each acquisition camera, these images are captured under a set of lighting conditions sampled (Halton sampling as in [6]) on a concentric sphere.

The position of the (capture) cameras are recorded in a spherical coordinate system termed Acquisition Coordinate System (ACS), with the origin at the object centroid. The cameras may be assumed to be calibrated at the moment.

Inputs at Run-time: There are three inputs.

- The animator is presented with a GUI to specify in another coordinate system, the World Coordinate System (WCS), an arbitrary three-dimensional path along which he desires to animate the object. The

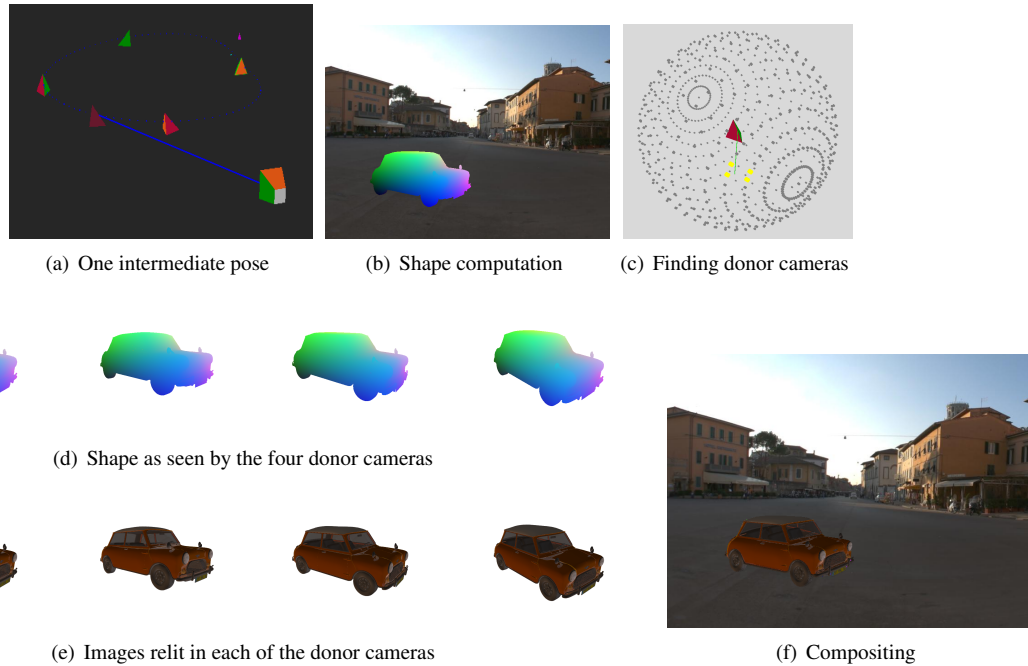


Figure 2: The method. In (a) we see a typical frame to be generated with the animation camera looking at a candidate position and pose. Using the specified view, we project a pre-computed visual hull to generate (b) the correct shape of the object of interest, in this case the MiniCooper car. The background is a novel environment given as input. In (c) donor cameras (shown in yellow) have been computed. In (d), the visual hull is conceptually projected on four donor cameras. The points P of the visual hull visible in (b) are also visible in the union (d) of these donor cameras, but not vice versa. In (e) we see how the MiniCooper would have appeared were pictures taken, not in the studio, but in the specified novel environment. Using some of these pixels, we see in (f) how the MiniCooper would look from the animation camera.

path is specified in the form of “key poses,” each of which is created using mouse-clicks in the viewports of the GUI. Each 3D point on the path is considered as a pose of the object, comprising of positional coordinates and orientations.

- The desired vantage point and orientation of observing the desired animation is also specified in the WCS. We term it the Animation Camera.
- The novel environment, specified as a cube map in which the object is bathed (Figure 1(b)).

Once the key poses have been specified by the animator, we generate the intermediate poses to create the final (virtually continuous) animation sequence path using standard B-spline based techniques [10]. Suitable editing capabilities are also provided to change the path once specified, the animation camera parameters, and the number of intermediate poses generated.

Consider, in the WCS, an arbitrary animation camera position X looking at the object centroid position Y along the specified path. We construct the reverse view vector from Y to X and map this view vector from the WCS to the ACS, resulting in a virtual viewpoint in the ACS. Using an efficient searching algorithm, the closest candidate viewpoints (capture donor cameras) in the

ACS are determined. The captured images from these donor cameras are used to synthesize the “view” as seen from the virtual viewpoint in the ACS (animation camera in WCS). This process is sketched in Figure 2.

Our method works because of the phenomenon of *relative motion*. An object moving in front of a stationary camera (in our case, the animation camera in the WCS) produces the same image as that of the camera moving in the “opposite direction” with respect to the stationary object (captured as input in the ACS, in the preprocessing phase.) One can therefore compute the correct camera parameters. Then *persistence of vision* realistically creates the illusion of motion.

4 THE ALGORITHM

The shape of the object as seen from the animation camera is computed using careful view interpolation techniques (Section 4.1). The color of the object in the novel environment is computed using the “basis” lighting conditions obtained as input (Section 4.2). This is repeated for all poses (Y) along the path of the object in the WCS, each generating a frame of the animation.

4.1 Shape

An immediate way of finding the view from the animation camera would be to interpolate among the im-

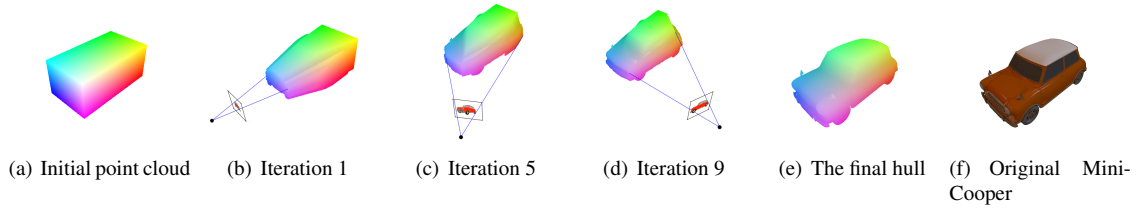


Figure 3: Visual Hull creation in progress: We start (Algorithm 1) with an initial point cloud. For each camera, using an image of the object silhouette, we determine the “relevant” points in the point cloud that needs to be stored. The remaining points in the point cloud are deleted. We keep iterating this procedure over all the cameras to produce our visual hull.

ages [5, 18], as viewed from the candidate closest view-points. However, a variety of rendering problems, including aliasing and ghosting artifacts manifest themselves. Thus the resulting image would most likely be unacceptable to the computer graphics community. To achieve higher visual fidelity, we could turn to techniques such as computing the point correspondences between the images of neighboring cameras [23, 13], or computing the optical flow between adjacent cameras [12]. Both of the above techniques are either highly time-consuming or are still error-prone. For highly complex models, visibility relationship from the novel viewpoint, and the ones taken earlier by the capture cameras should match.

Since we have the liberty to preprocess, we use the concept of image-based visual hulls [16, 17] as an intermediate step, for computing the view as seen from the animation camera. Note that in the input acquisition phase, and for each capture camera, we have the liberty of acquiring the image by placing a light source at the back of the object. This helps us to identify a proper silhouette of the object.

For each pose, the visual hull of the object is positioned and oriented in the WCS according to the specified pose parameters. The visual hull is then z-buffer projected on the animation camera (Figure 2(b)). At this point the shape of the object as seen from the animation camera is available. In the next section, we describe our novel visual hull computation algorithm, which computes a dense yet “lightweight” hull.

Visual Hull Revisited Various approaches [26, 14] to computing the visual hull of an object, given its silhouette data, have been proposed in the past. The idea behind all these algorithms is that if the camera parameters are known, the information from many 2D images can be used to obtain a good approximation of the shape of the object. This approximation is an upper bound estimate of the shape of the object. Since we get to arrange the acquisition setup, we use calibrated cameras; and for each camera, we capture an additional image of the object, in which the object can be well segmented from the background.

For computing the visual hull, we start with an initial cuboid point cloud (Figure 3(a)) of points. For each

Algorithm 1 CREATE-VISUAL-HULL (*CaptureCamList*)

```

1: PointCloud = Create a point cloud
2: for all camera in CaptureCamList do
3:   {SegmentedImage: Image with object segmented from background}
4:   for all P in PointCloud do
5:     (x,y) = Project P into camera
6:     if (x,y) = Background-pixel(SegmentedImage) then
7:       Delete P from PointCloud
8:     end if
9:   end for
10: end for
11: return SCOOP-OUT(PointCloud,CaptureCamList)

```

Algorithm 2 SCOOP-OUT(*PointCloud*, *CaptureCamList*)

```

1: Delete points from PointCloud that have neighbors
2: Mark remaining points in PointCloud as unseen
3: for all camera in CaptureCamList do
4:   capzbuf ← ZBUFFER(CaptureCam, VisualHull)
5:   for all P in capzbuf do
6:     Mark P as seen
7:   end for
8: end for
9: Delete all unseen points from PointCloud
10: return PointCloud

```

acquisition camera, we project each point in the point cloud into its image plane and check, using the well-segmented image, whether the corresponding pixel belongs to the object or the background. If the pixel belongs to the object, we keep it; otherwise we delete it from the point cloud. As the iteration proceeds (Figure 3) a conservative estimate of the shape of the object is obtained.

The novelty in our visual hull algorithm begins here. The hull, thus created, includes points that are in the silhouette but also points invisible to all capture cameras. We term such a hull as *not* lightweight. The number of points ‘inside’ a non-lightweight hull are orders of mag-

nitude larger than those strictly on the bounding surface (shell), As we shall see later, our algorithm requires frequent projection of points in the visual hull; therefore a lightweight hull is desirable. On the other hand, the hull should be dense to capture fine details and avoid holes.

Our solution is to scoop out the hull interior. The first step is to exploit the fact that the initial point cloud was populated as a matrix. We therefore delete a point with six neighbors on the hull. We then z-buffer project the remaining points in the hull to track ‘true positives,’ i.e., points that are visible from at least one camera. Points which are not visible can be safely deleted. (Bitmaps are used for efficient implementation.)

In summary, first we start with a sparse point cloud and use Algorithm 1 to create a hollow, but sparse, visual hull. This hull is lightweight, but may not be dense enough. Therefore, for each point in the sparse hull, we populate a cube of space around it with closely spaced points. This becomes our new point cloud, on which we iterate Algorithm 1. In Section 5, we provide qualitative and quantitative results of our visual hull algorithm.

4.2 Color

The color of the visible points of the visual hull as seen by the animation camera is obtained from the images of its k closest capture cameras (donor cameras) in the ACS, relit with the lighting configuration specified as a novel environment in the WCS. Note that as the object moves along the desired path, the relative orientation of the light sources (corresponding to the novel environment) with respect to the object (pose) constantly changes. Thus, to determine the correct color information of the object as it moves along its desired path, one must incorporate this change of light sources’ orientation for each pose of the object. For each pose of the object, we map the novel light sources’ directions (in the WCS) to the ACS, and then use this mapped set of light source directions (in the ACS) as the novel illumination configuration to relight each of the k donor cameras. The relit images thus computed are used for determining the color of the pixels in the synthesized view for the virtual viewpoint (animation camera). However, the following questions need to be answered.

- For a candidate pixel p as seen in the animation camera, which pixel q in the donor camera d is relevant? Should we consider more than one donor for the same candidate pixel p ?
- Overall, how many donor cameras are needed?

We answer these questions in an incremental, algorithmic fashion. At the end of shape computation we have the three dimensional coordinates of each visible point i in the visual hull. Starting with the closest donor camera, we determine the points in the visual hull visible to this donor camera. This procedure is incrementally, performed until all the visible points are tagged

with at least one donor camera. This is again done using a z-buffer (Figure 2(b) and 2(d)).

More specifically, consider donor camera d , and a 3D point P (corresponding to a candidate pixel p) that is visible in the animation camera. The image corresponding to the donor camera d contains pixels corresponding to a set of points Q_d of points in the visual hull that are visible from d . Three cases may arise in increasing order of complexity:

1. $P \in Q_d$ & $P \notin Q_i, \forall i \neq d$: We have a clear (pixel) match for P in d , whose color value (after relighting) is assigned to pixel p .
2. $P \in [Q_d, Q_{d'}, \dots]$, for some $d, d' \in [CaptureCamList]$: We perform a blend of the relit pixel values (as observed in all relevant donor cameras) to compute the color of pixel p .
3. $P \notin Q_d, \forall d \in [CaptureCamList]$: We assert a match for P in d , provided the depth value of a point in Q_d is close to the depth value of P .

Relighting At this stage we know which of the acquired images are relevant for the animation camera, i.e., the k donor cameras. We also know which pixels in these donor cameras map to the animation camera. Given our third input, the novel lighting environment specified as a cube map in the WCS, we obtain the lighting directions (in the WCS) and intensities of the 10 most significant light sources in the environment map (using HDRshop’s lightgen plugin [7].) Using these 10 light sources mapped into the ACS, we perform relighting on the images corresponding to the k donor cameras using an algorithm based on the method in [6]. Note that, for every frame of the animation, since the relative configuration of these 10 light sources changes constantly with respect to the pose of the object, these 10 light sources’ directions need to be mapped into the ACS for every frame (pose of the object).

5 IMPLEMENTATION AND RESULTS

We now describe some of the features of our implementation. All our experiments were performed on a Intel Centrino 1.73GHz Core Duo processor and 1GB RAM.

Preprocessing and Run-time: For proof of our concept, we simulated the capture setup by generating the input images using POV-Ray. In production, the system needs only photographs captured of the object on an acquisition setup as in [8] – which we do not have access to. Neither is the geometry of the object needed, and nor is there a need of a 3D renderer. Thus, our current experiments (using only the images of the objects as input) is indicative of the quality and resource (memory and computational) requirements of this animation framework for real world productions.

In our implementation, we use 762 cameras distributed on the bounding sphere. The number of basis



Figure 4: Visual hull creation results: Each set is composed of two images, the one on the left is the visual hull computed using our algorithm, while on the right is a snapshot image of the same object. Observe the similarity of features displayed by our computed hull.



Figure 5: The Axe, in a pose, rendered under four different illumination conditions. Note the distinct change of color on the blade of the Axe.

lighting conditions (sampled on a concentric sphere using Halton sampling [6]) are fixed at 100. As a preprocessing step, we also compute a dense, but memory efficient visual hull using multiple iterations of Algorithm 1. We observed from our experiments that in most cases, two iterations are enough to extract a good estimate. For a path specified by the animator, we generated approximately 50-100 intermediate poses (final output frames). The computation of the intermediate poses takes a few seconds.

Shape: Once the visual hull is computed, to compute an accurate shape description of the object, devoid of aliasing effects along its edges and boundaries, we create, in software, a high resolution z-buffer (upto 4 times the resolution of the frame buffer) for the animation camera, i.e., for every pixel in the output frame, there exists four samples in the z-buffer, each corresponding to a 3D point in the visual hull. Thus we get a supersampled image as an output, which is used to give antialiased results. Figure 4 shows the qualitative results of the visual hulls of various objects created using our visual hull algorithm. Notice the accuracy of the reconstructed geometries with respect to the the original object shape. Table 1 provides the quantitative results of our algorithm, while Table 2 depicts results of comparison with a conventional visual hull algorithm.

Color: The location of the capture cameras are stored in a lightweight k-d tree ($k=2$) to enable efficient nearest neighbor search for donor cameras, once a virtual viewpoint is determined (Section 4.2). For efficiently determining the color of each pixel in an output frame, we could compute and store on disk (in the Preprocessing stage) the depth map corresponding to all the capture cameras. This speeds up the match computation. However in our current implementation, we chose to do a real-time computation of the depth map, rather than an explicit store (with a consequence of an increase

in the final rendering time). For each pixel in the output frame, we chose to blend the color corresponding to the 4 samples in the z-buffer (of the animation camera) corresponding to it. This produces very realistic lighting and color blending effects. Figure 6 demonstrates screenshots of different animation sequences created with our algorithm on four objects, *Axe*, *Spaceship*, *MiniCooper*, and *Oak Leaf*. Please see supplementary material for generated animations and details.

Relighting: For our novel environments, we use the cube maps provided in [7], namely *The Uffizi Gallery*, *Florence*, *Glacier*, *Banff National Forest*, *Canada*, *Pisa courtyard*, *Italy*, and also generated a new environment, *woods*. As mentioned, using HDRShop lightgen plugin [7], we summarize the novel environment in terms of light sources. Figure 5 demonstrates the beautiful relighting effects faithfully reproduced on our *Axe* in the a fixed pose, but in different environments.

The entire run-time pipeline comprising of the continuous path (intermediate pose) generation, computing the view vector, shape determination, and color computation after relighting takes only around 1-2 minutes for a single output frame. Thus we are able to create the entire animation sequence within a few minutes.

6 FINAL REMARKS

In this paper, we have shown how to efficiently create realistic animations using only images as input, instead of traditional geometry-based input. We have employed four techniques to provide the alternative of animating an object on a specified arbitrary path in a novel environment. These techniques are, *B-Spline based interpolation* for the motion path specification, *classical coordinate transformations* for determining the correct pose of object, *visual hull* for view interpolation, and *lighting basis* for relighting.

Hull	Resolution	Initial Cloud	Before scoop-out	After scoop-out	Time(mins)
Axe	16	1.23×10^6	45463	18832	4
	32	9.6×10^6	344991	75321	10
	64	7.63×10^7	2794872	309013	31
Leaf	16	2.3×10^6	5045	5045	3
	32	1.8×10^7	39284	38037	4
	64	1.4×10^8	314928	222663	16
Mini	16	6×10^6	2143574	105098	37
	24	2×10^7	7220067	313382	120
Spaceship	16	7.4×10^6	901580	66329	40
	32	5.9×10^7	7213929	271692	120

Table 1: Quantitative results (size and preprocessing time) of our visual hull creation algorithm. The second column represents the sampling resolution of the initial point cloud, i.e., the number of points in a unit length; the third column depicts the total number of points in the initial cuboidal point cloud (Figure 3(a)); the fourth column indicates the number of points present (in the “solid” visual hull) in Algo.1 at step 10; the fifth column indicates the number of remaining points in the final visual hull after scooping the solid visual hull; the last column provides the total time taken in the studio to generate the visual hull, starting from the initial point cloud.

Hull	Resolution	Naive	Ours
Axe	64	1 hour	10 + 21 minutes
Mini	96	3 days	9 + 6 hours

Table 2: Preprocessing timing comparisons. **Naive** represents the time taken to create a visual hull beginning with a dense cuboidal point-cloud and using a classical algorithm. **Ours** represents by initially starting with a sparse point-cloud (Algorithm 1), and scooping out the hull interior. We report both times in the last column.

This animation framework could also be used as feedback for a final traditional CG animation. The director could quickly prepare a naive version of the final CG sequence by specifying a path, a novel lighting environment, a model; and later edit any of them to suit the mood of the story. Note that, not all three inputs need to be modified simultaneously at run time. For example, we might change the novel environment (see Figure 5), not the other two inputs. This would save man-hours spent by skilled personnel in the pre-production phase.

In future, research is required to incorporate non-rigid object motion into our animation framework. We also plan to incorporate special effects (for example, motion-blur, shadows) into the framework. Since IBR generates photorealistic outputs, the aesthetics of the models and the scene would be preserved.

REFERENCES

- [1] N. Ahmed, C. Theobalt, C. Ross, S. Thrun, and H.P. Seidel. Dense correspondence finding for parametrization-free animation reconstruction from video. In *CVPR '08*, pages 1–8, 2008.
- [2] Christoph Bregler, Michele Covell, and Malcolm Slaney. Video rewrite: driving visual speech with audio. In *SIGGRAPH '97*, pages 353–360, 1997.
- [3] Gabriel J. Brostow and Irfan Essa. Image-based motion blur for stop motion animation. In *SIGGRAPH '01*, pages 561–566. ACM, 2001.
- [4] Joel Carranza, Christian Theobalt, Marcus A. Magnor, and Hans-Peter Seidel. Free-viewpoint video of human actors. *ACM Transactions on Graphics*, 22(3):569–577, 2003.
- [5] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *SIGGRAPH '93*, pages 279–288, 1993.
- [6] Biswarup Choudhury and Sharat Chandran. Data-intensive image based relighting. In *GRAPHITE '07*, pages 155–162. ACM, 2007.
- [7] Paul Debevec. Light Probe Image Gallery. <http://www.debevec.org/Probes/>, 2008. Last visited: May, 2008.
- [8] Paul Debevec, Andreas Wenger, Chris Tchou, Andrew Gardner, Jamie Waese, and Tim Hawkins. A lighting reproduction approach to live-action compositing. *ACM Transaction on Graphics*, 21(3):547–556, 2002.
- [9] Per Einarsson, Charles F. Chabert, Andrew Jones, Wan C. Ma, Bruce Lamond, Tim Hawkins, Mark Bolas, Sebastian Sylwan, and Paul Debevec. Relighting human locomotion with flowed reflectance fields. In *EGSR '06*, pages 183–194, 2006.
- [10] Martin Lillholm Erik Dam, Martin Koch. Quaternions, interpolation and animation. www.itu.dk/people/erikdam/DOWNLOAD/98-5.pdf, 2008. Last visited: May 2008.
- [11] James Hays and Irfan Essa. Image and video based painterly animation. In *NPAR '04*, pages 113–120, 2004.
- [12] Price Bibliography Keith. Keith Price Bibliography: Surface Reconstruction from Optical Flow. <http://www.visionbib.com/bibliography/optic-f753.html#KK4747>, 2008. Last visited: August, 2008.
- [13] Price Bibliography Keith. Keith Price Bibliography: Virtual view generation, View synthesis, Image based rendering, IBR, Morphing. <http://www.visionbib.com/bibliography/describe487.html>, 2008. Last visited: August, 2008.
- [14] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994.
- [15] Marcus Magnor, Marc Pollefeys, German Cheung, Wojciech Matusik, and Christian Theobalt. Video-based rendering. In *SIGGRAPH '05 Courses*, page 1, 2005.
- [16] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. Image-based visual hulls. In *SIGGRAPH '00*, pages 369–374. ACM Press/Addison-Wesley Publishing Co., 2000.
- [17] Wojciech Matusik, Hanspeter Pfister, Addy Ngan, Paul Beardsley, Remo Ziegler, and Leonard McMillan. Image-based 3d photography using opacity hulls. *ACM Transactions on Graphics*, 21(3):427–437, 2002.

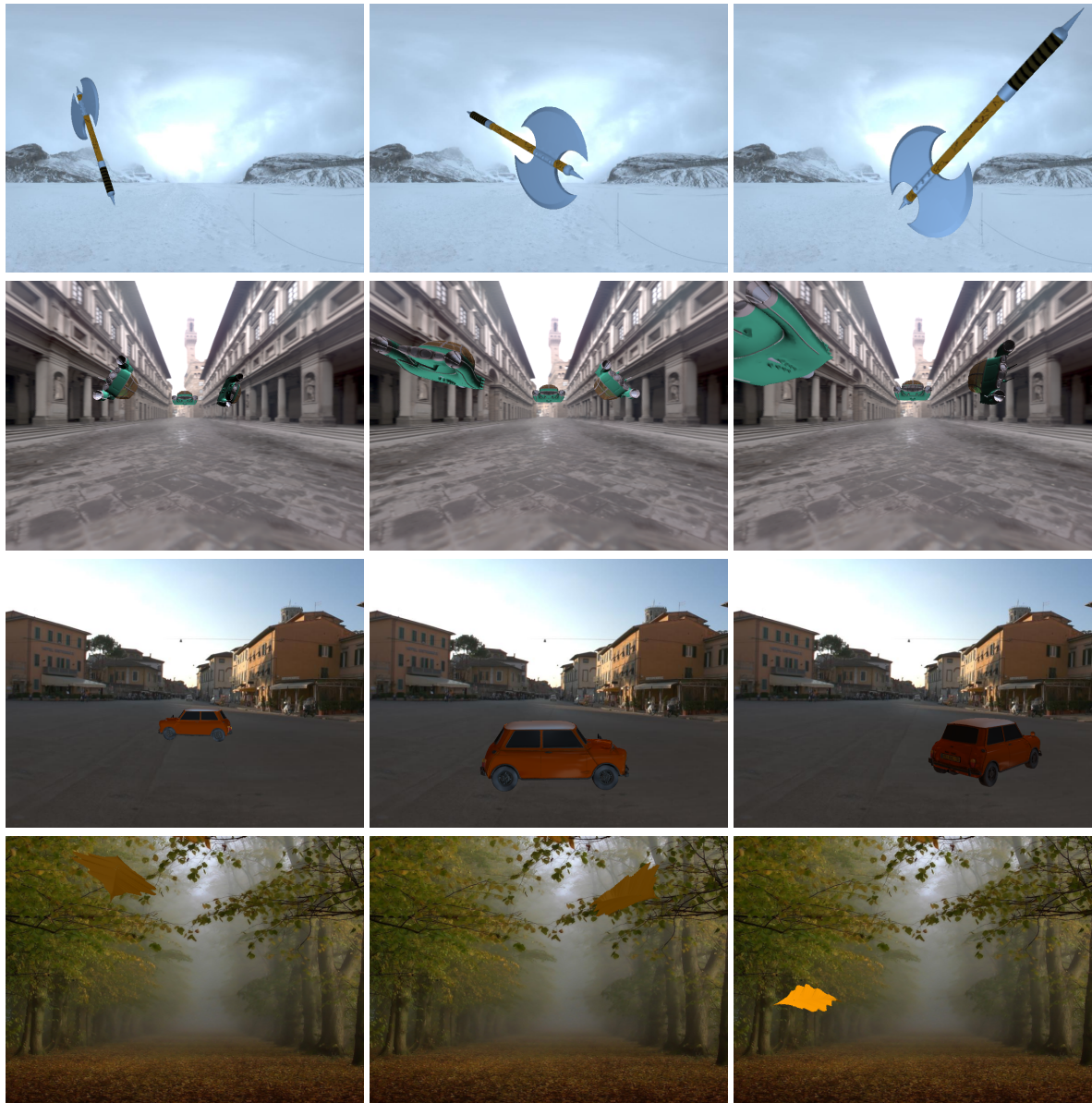


Figure 6: Images of Axe (row 1), Spaceship (row 2), MiniCooper (row 3) and Oak Leaf (row 4) rendered in different poses, i.e., while in motion, each under novel lighting conditions: Glacier, Uffizi, Pisa and Woods respectively.

- [18] Leonard McMillan and Gary Bishop. Plenoptic modeling: an image-based rendering system. In *SIGGRAPH '95*, pages 39–46, 1995.
- [19] Pixar. The Pixar Process. <http://www.pixar.com/howwedoit/index.html>, 2008. Last visited: May 2008.
- [20] R. Plankers and P. Fua. Articulated soft objects for video-based body modeling. *ICCV '01*, 1:394–401 vol.1, 2001.
- [21] Arno Schödl and Irfan A. Essa. Controlled animation of video sprites. In *SCA '02*, pages 121–127, 2002.
- [22] Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa. Video textures. In *SIGGRAPH '00*, pages 489–498, 2000.
- [23] Steven M. Seitz and Charles R. Dyer. View morphing. In *SIGGRAPH '96*, pages 21–30. ACM, 1996.
- [24] J. Starck and A. Hilton. Virtual view synthesis of people from multiple view video sequences. *Graphical Models*, 67(6):600–620, 2005.
- [25] J. Starck, G. Miller, and A. Hilton. Video-based character animation. In *SCA '05*, pages 49–58, 2005.
- [26] Marco Tarini, Marco Callieri, Claudio Montani, Claudio Rocchini, Karin Olsson, and Therese Persson. "marching intersections: An efficient approach to shape-from-silhouette". In *Proceedings of the Conference on Vision, Modeling, and Visualization (VMV 2002)*, pages 255–262, 2002.
- [27] Sundar Vedula, Simon Baker, and Takeo Kanade. Image-based spatio-temporal modeling and view interpolation of dynamic events. *ACM Transaction on Graphics*, 24(2):240–261, 2005.
- [28] Xuemiao Xu, Liang Wan, Xiaopei Liu, Tien-Tsin Wong, Lian-sheng Wang, and Chi-Sing Leung. Animating animal motion from still. *ACM Transactions on Graphics*, 27(5):1–8, 2008.
- [29] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM Transaction on Graphics*, 23(3):600–608, 2004.

Performance Considerations When Using a Dedicated Ray Traversal Engine

Tomáš Davidovič
Saarland University and DFKI
Campus E1 1
DE, Saarbruecken
davidovic@cs.uni-saarland.de

Lukáš Maršálek
Saarland University and IVCI
Campus E1 1
DE, Saarbruecken
marsalek@cs.uni-saarland.de

Philipp Slusallek
Saarland University and DFKI
Campus E1 1
DE, Saarbruecken
slusallek@cs.uni-saarland.de

ABSTRACT

In the recent years we have witnessed massive boost to hardware graphics accelerators (graphics cards), not only in the raw performance, but also in their programmability, introducing the concept of GPGPU. However, despite this, the current architectures still favor feed-forward algorithms over recursive ones. While shading is, in this sense, a feed-forward algorithm, ray tracing, and specifically ray traversal, is a recursive rather than feed-forward algorithm. Adding a dedicated hardware ray traversal engine should therefore prove to be an interesting option. Also, with dedicated hardware, we can perform many optimizations on arithmetic units due to their fixed interaction. This can reduce the area well below a simple sum of areas of the individual units. In this paper we offer for consideration analysis of memory requirements for combination of a dedicated hardware ray traversal and intersection engine with highly parallel general purpose processor used for shading. We show results and requirements of such a combination on scenes of moderate complexity, with regard to speed, bandwidth and latency.

Keywords: ray tracing, hardware, GPU, bandwidth

1 INTRODUCTION

CPU ray tracing has received tremendous speed ups in the past decade, with works on acceleration structures, traversal algorithms, and packet (or frustum) traversal [7, 17, 16, 12, 6]. While these advancements brought ray tracing into the real-time area [9], the speed is still generally not competitive to rasterization.

One of the chief problems causing this gap is the dedicated rasterization hardware, which is something ray tracing lacks. The modern graphics processing units (GPU) consist from a relatively small number of fixed function raster engines (four in NVIDIA's Fermi architecture) and a large number of general purpose processing elements. Typically the general purpose part of GPU prepares triangles (vertex and geometry shaders) for raster engines, which dice triangles into a stream of fragments that are fed back into the general purpose part that performs fragment shading on them.

There are many papers [10, 6, 2] on GPU ray tracing, as well as some deliverable products [8]. All such approaches use solely the general purpose part, and most of them focus on overcoming limitations of their contemporary architectures. In [15] Seiler et al. present ray tracing on Intel's Larrabee, also utilizing only general purpose elements of the chip.

The opposite approach, a stand-alone ray tracing hardware, has also been explored [14, 18, 19, 13, 11]. While such solutions have potential to deliver the best performance, the development in rasterization hardware shows necessity of general purpose shading capabilities.

Surprisingly little research has been done on the middle ground, replacing the GPU raster engines with another kind of fragment generator. The only significant work on this has been done by Caustic Graphics [3], who proposed a separate ray tracing accelerator board, that can utilize present GPU for shading.

A recent paper by Aila and Kerras [1] focuses on bandwidth issues of complex scenes. The paper assumes a separate ray tracing engine, but presents results for a whole NVIDIA Fermi GPU used as such unit. However, it leaves open possibility of different implementations, including dedicated hardware.

In this paper we present a solution, which proposes to place a small dedicated hardware ray traversal engine (RTE) either directly on the GPU die, or as a co-processor on the graphics card.

2 RAY TRAVERSAL ENGINE

Our ray traversal engine (RTE) implementation is based on the well-documented FPGA implementation of the DRPU by Woop [18]. We will first give a brief overview of the basic blocks used, then discuss frequency, area scaling, and the connected design decisions.

2.1 Design blocks

The RTE (Figure 1) is a heavily pipelined unit with fine grain multithreading. It uses B-KD trees [20] for its

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.

acceleration structure. Our B-KD tree implementation offers a two-level hierarchy, where each leaf of the top tree can contain either a triangle or a transformation matrix and pointer to a subtree.

The basic computation unit is a thread, which processes four rays at once. Each pipeline stage can accommodate a different thread, and there is no overhead in switching threads. The whole RTE can process up to 64 threads (256 rays) at once. While there is no strict requirement for coherence of rays within the same thread, it can bring performance boost. Also, coherence between all processed rays improves cache hit rates.

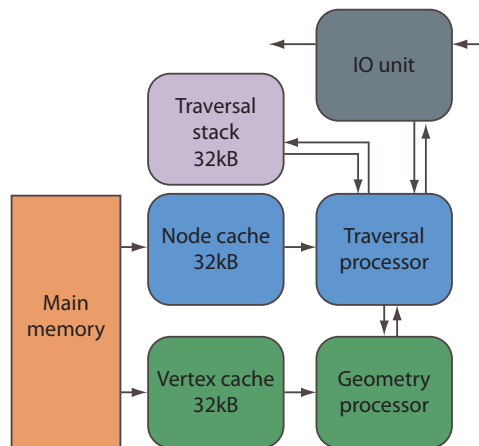


Figure 1: **RTE block diagram.** The RTE core consists of traversal and geometry units. Each unit is connected to main memory via 32kB cache. Traversal unit has an additional 32kB memory for traversal stack and connection to the IO interface to return results and receive new queries.

The IO unit strongly depends on the particular implementation details of integrating RTE with the rest of the system. We will therefore not elaborate on its implementation, but we can imagine it as a kind of memory controller that handles ray and result queues in the main memory, akin to what is assumed in [1]. The basic functionality is to fetch rays for idle threads and store results from finished ones.

The actual RTE core consists of two units: traversal and geometry. Each is connected to the main memory via a 4-way associative 32kB read-only cache. A single larger cache has been considered, but there would be either high contention for its single port, or it would have to be two port cache, resulting in higher complexity. Also, as each unit requires different data, there is no redundancy in fetches from main memory that would lower the efficiency.

Another two parts, not shown in the Figure 1 are per-thread stack memory and ray state buffer. The per-thread stack is accessed with up to 1 read and 1 write each cycle, where the read occurs when a thread is submitted to traversal unit, and the write occurs when the

result of traversal stage is that both children should be intersected. Each item in stack consists of a node address (4B) and near and far values for each ray (32B), for the total of 36B per item. For convenience we consider maximum scene depth of 32. Making the internal stacks shorter and spilling overflow into main memory, as described in [1], is also an option.

Ray state buffer is per-ray storage with current closest hit distance, ID of the closest triangle and barycentric coordinates.

For each thread, the traversal unit fetches the required node. If the node is a leaf, it is sent to the geometry unit. For inner nodes it intersects the child nodes and, based on the result, it either pops a new node from stack, traverses the single hit child, or traverses the closer child and pushes the farther child onto the stack. If the stack is empty it informs the IO unit that the ray has finished.

The geometry unit computes both the intersection with triangles and the transformation of rays for traversing the bottom tree when a two-level hierarchy is used. This is possible because both operations require very similar functional units with a minimal reconfiguration. When this unit receives a leaf node, it first fetches three vertices (for intersection) or matrix rows (for transformation). The thread waits until all three memory fetches are finished before further execution. Each ray of the thread occupies 2 consecutive pipeline stages, therefore it takes 8 cycles to process the whole thread.

The unit is, for some of the measurements, augmented with a 1MB 4-way associative L2 read-only cache. This cache has latency of 100 cycles, and assumes 512bit wide access to memory (fetches 64B at once). When L2 cache is not present, the L1 caches are connected directly to the memory and assume 128bit wide access. The main memory is assumed to have latency 600 cycles.

2.2 RTE synthesis

We first synthesized all the major arithmetic sections of both units as well as the cache logic using proprietary IBM 90nm process [4]. All synthesized parts were able to run at frequencies over 2GHz. This was done without any special fusion of dependent arithmetic units. Also, considering the already quite high latency of both traversal (14 cycles) and geometry (36 cycles) units, more stages could be added to further stabilize the frequency, without a significant impact on the overall performance. While both optimizations can lead to higher frequency than reported, we opted against their implementation as the preliminary results are more than satisfactory.

We have not explicitly synthesized all the required memories, instead we borrowed statistics from similar memories from Cell/B.E.TM processor [5]. This processor has 32kB L1 cache, running at 3.2GHz, as well as 1-read 1-write port local store memory running at the

same frequency. As these two types of memory fulfill all requirements we have on our memory blocks, we conclude that memory frequency will not be an issue.

The area of RTE is not of the prime concern for our results, so we present only rough upper bound on RTE area. We used a range of area estimation techniques to confirm that RTE comfortably fits into the area of less than 15mm². Further reduction of size is possible by aforementioned fusion of arithmetic units.

3 SIMULATOR ARCHITECTURE

We have performed two distinct simulations. The first simulation was on the actual low-level VHDL code, that is used as base for our synthesis results. While this confirms that the design is correct and the synthesized frequencies valid, the simulation itself is very slow.

To solve this we also designed a cycle-accurate SystemC model of RTE. SystemC is a C++ library that allows using many high-level language constructs not available in VHDL, while at the same time allowing to simulate exactly the same timings, to the cycle levels. We integrated this RTE model as a part of our ray tracing framework, essentially replacing its standard traversal and intersection routine.

Normally our framework uses the following workflow. First, a primary ray is generated from camera. It is then intersected with the scene and the closest geometry is found. If there is any hit, integrator is invoked that, based on its type, queries material for its BRDF, scene for the lights, performs shading and possibly shoots other rays. Once the primary ray is finished and the final color is computed, another primary ray is generated, until the whole image has been rendered.

While this sequential process works very well for CPU rendering with low amount of parallelism (4-16 threads at once), it is ill-suited for massively parallel hardware acceleration. We have therefore modified the algorithm as follows. First, all primary rays are generated and put into input queue of the RTE. The RTE emulation engine is then started, and after each cycle its output queue is checked. If there is a ray in the output queue, it is passed to the integrator that processes the ray. This can, and generally does, generate other rays, that are in turn fed back into the input queue. Where there are no more rays in the input queue and no active rays in the RTE, the frame has finished. Should generating all primary rays occupy too much memory at once, it is also possible to split the image into several blocks or batches, based on the memory configuration.

3.1 Shader models

Unfortunately, this system does not lend itself easily to the standard recursive shaders, nor to any kind of shader where the integrator needs result of a traced ray. There are two possible solutions to the problem. First,

adopted by [3], is to employ tail recursion. Here the integrator generates all required rays in a fire-and-forget manner, attaching to them all the necessary information. Shadow rays would, for example, work in such a way, that all lights are counted as contributing. Shadow rays are then generated with a flag marking them as shadow, maximum length and weight for the particular light. When they do not hit anything, no integrator code is invoked and the light contribution stays added. If they hit, the integrator simply accumulates the negative weight to the pixel, effectively subtracting the light contribution. In this way, the original integrator does not have to wait for shadow ray results, nor do we need special *no hit* integrator to add light contribution when shadow ray does not hit anything.

This approach has a drawback of spawning large and poorly controllable number of rays. The other approach is to use continuations. Here the integrator is effectively split at each call for *traceRay* and its state is stored. Once the ray tracing has been finished, the state is fetched from a global storage and the integrator continues. This can be easily implemented by a finite state machine, but requires stack in the case ray bifurcation is allowed. One of the advantages is that we do not need to sum over all rays contributing to a single pixel, as all the computations concerning one pixel are very self-contained. Disadvantages include larger per-ray storage and higher sensitivity to latency.




3.2 Acceleration structure partitioning

To increase cache coherence, Aila and Kerras [1] introduce concept of partitioning the acceleration structure into multiple treelets (subtrees), each of which fits into the cache. As we consider this approach very relevant for our results, we adapted it for B-KD-trees and included it in our measurements.

The basic principle is that the whole structure is split into many small subtrees, called treelets. Each of the treelets has its own ray queue and when a ray crosses a treelet boundary, its traversal is stopped and it is put into corresponding treelet's input queue. While the original paper introduced several methods for scheduling treelets to ray traversal engines, we use only the simplest one called *lazy scheduler*. It simply takes the treelet with largest queue and processes it until the queue is empty. Then it switches to another treelet, with the currently largest queue. The more complex methods are meant to balance situation where there are multiple ray traversal engines, which is something we do not currently consider for our scenario.

4 RESULTS

We tested on three scenes of moderate complexity, the **Conference** (283k triangles), **Fairy forest** (174k triangles), and **Kitchen** (253k triangles). We test both

	Conference	Fairy forest	Kitchen
			
Triangles:	282759	174117	253433
Res:	512 × 512	512 × 512	600 × 450

Continuations shaders						
L2	off	on	off	on	off	on
L1 hit rate [%]	68/53	69/55	55/44	56/45	83/77	89/85
L1 bandwidth [GB s ⁻¹]	6.7/4.7	15.7/11.0	5.0/3.4	11.7/7.9	11.2/7.6	16.7/11.3
L2 hit rate [%]	-	95/84	-	88/73	-	90/80
L2 bandwidth [GB s ⁻¹]	-	4.8/4.9	-	5.2/4.3	-	1.8/1.7
Ray bandwidth [GB s ⁻¹]	0.14	0.14	0.14	0.14	0.14	0.14
Mem bandwidth [GB s ⁻¹]	4.4	4.2	4.3	7.5	3.7	2.1
Latency [cycles]	6.0 M	2.7 M	12.0 M	5.4 M	3.3 M	2.3 M
Throughput [MRays/s]	47.8	112.4	20.8	48.1	66.6	100
Tail recursive shaders						
L2	off	on	off	on	off	on
L1 hit rate [%]	76/64	78/67	63/53	64/55	87/82	85/80
L1 bandwidth [GB s ⁻¹]	8.3/5.8	16.3/11.2	6.0/4.1	12.5/8.4	12.6/8.5	16.8/11.4
L2 hit rate [%]	-	93/79	-	86/71	-	93/86
L2 bandwidth [GB s ⁻¹]	-	3.6/3.7	-	4.5/3.8	-	2.6/2.3
Ray bandwidth [GB s ⁻¹]	0.14	0.14	0.14	0.14	0.14	0.14
Mem bandwidth [GB s ⁻¹]	4.4	4.1	4.1	6.2	3.3	2.1
Latency [cycles]	4.9 M	2.6 M	10.0 M	5.0 M	3.0 M	2.3 M
Throughput [MRays/s]	61.1	117.4	25.0	52.3	75.4	99.2

Table 1: **Results without treelets.** We measure all three scenes with both tail recursive and continuation shaders and both with and without 1MB L2 cache. The reported results are L1 cache hit rate, required L1 bandwidth in GB s⁻¹, the same for L2 cache (if applicable), ray traffic bandwidth in GB s⁻¹, total required memory bandwidth, both from cache and rays, latency in cycles (Latency) and throughput in million rays per second. The L1 and L2 cache results are given in format vertices/nodes.

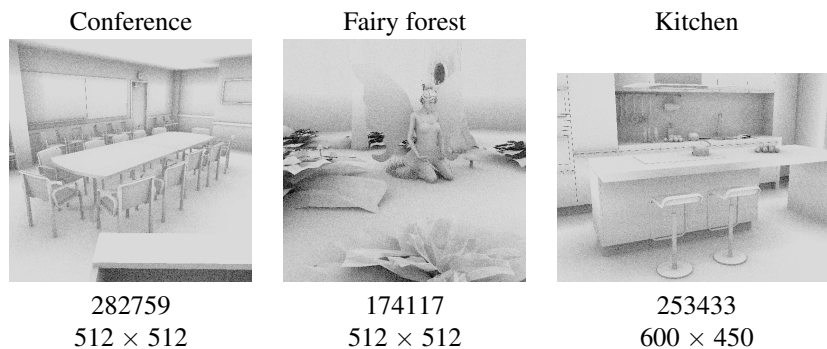
shader approaches, using 16 rays per pixel ambient occlusion. We focus on the cache hit rates, throughput (rays per second), ray latency (important for continuation shaders), and the bandwidth requirements of our unit, measured both with and without an L2 cache.

The reason why we provide not only cache hit rates, but also their bandwidth is that one should not automatically assume that higher hit rates are more desirable. Should we devise the perfect ray tracing algorithm, that only ever needs each node and triangle exactly once, our caches would have hit rate 0%, but the cache bandwidth would be significantly lower. Moreover, looking at the bandwidth is still not sufficient, because the most obvious way to lower bandwidth while keeping the same hit rates is to reduce the overall throughput of the engine, which is not at all desirable. We therefore have to look at the ray throughput, cache hit rates and bandwidth together and draw conclusions only from the combination of the three, as we will see below.

At the beginning in Section 4.1 we provide results for RTE without the use of treelets, follow up with Section 4.2 commenting results using treelets and close with Section 4.3 on using BVH instead of B-KD-tree.

4.1 Standard implementation

Table 1, providing results for the standard RTE implementation, offers several insights. First and most important is, that the L2 cache helps in all the scenes, giving us speed up factor 1.3-2.3×. The reason for this is somewhat obvious, the L2 cache provides large portion of the scene with lower latency (100 cycles) than the main memory (600 cycles). The L2 cache also exhibits relatively high hit rate, above 85 % for nodes and over 70 % for vertices. It is important to note, that while the L2 hit rate is higher for continuations shaders, this is actually caused by the lower L1 cache hit rate, which causes more requests to L2 cache, increasing the bandwidth as noted at the beginning.



Continuations shaders						
L2	off	on	off	on	off	on
L1 hit rate [%]	83/70	86/73	79/65	81/66	89/82	91/85
L1 bandwidth [GB s ⁻¹]	9.3/6.5	14.0/9.8	8.0/5.4	12.1/8.1	11.7/7.9	15.3/10.3
L2 hit rate [%]	-	79/62	-	72/57	-	76/66
L2 bandwidth [GB s ⁻¹]	-	2.0/2.7	-	2.3/2.8	-	1.4/1.6
Ray bandwidth [GB s ⁻¹]	1.3	1.3	2.2	2.2	1.4	1.4
Mem bandwidth [GB s ⁻¹]	5.5	7.1	5.8	9.5	4.2	4.8
Latency [cycles]	2.8 M	2.2 M	5.8 M	4.6 M	3.6 M	3.1 M
Queue switches [-]	104718	103067	196828	199421	130285	129507
Avg. queue size [-]	87	88	86	85	79	79
Throughput [MRays/s]	67.2	101.2	33.0	49.3	69.1	90.4
Tail recursive shaders						
L2	off	on	off	on	off	on
L1 hit rate [%]	96.2/92.3	97/93	95/89	96/89	97/95	98/95
L1 bandwidth [GB s ⁻¹]	14.0/9.8	15.4/10.7	13.9/9.3	15.5/10.3	15.0/10.2	16.1/10.9
L2 hit rate [%]	-	73/61	-	70/64	-	70/62
L2 bandwidth [GB s ⁻¹]	-	0.4/0.7	-	0.6/1.1	-	0.4/0.5
Ray bandwidth [GB s ⁻¹]	1.2	1.2	2.0	2.0	1.3	1.3
Mem bandwidth [GB s ⁻¹]	2.5	2.8	3.7	4.4	2.3	2.5
Latency [cycles]	15.8 M	15.4 M	30.0 M	28.7 M	24.3 M	23.7 M
Queue switches [-]	13220	13176	19118	19252	15092	15255
Avg. queue size [-]	659	661	829	823	653	647
Throughput [MRays/s]	103.6	112.8	57.8	64.4	90.2	96.7

Table 2: **Results with treelets.** We again measure all three scenes with both tail recursive and continuation shaders and both with and without 1MB L2 cache. The reported results are L1 cache hit rate, required L1 bandwidth in GB s⁻¹, the same for L2 cache (if applicable), ray traffic bandwidth in GB s⁻¹, total required memory bandwidth, both from cache and rays, latency in cycles (Latency) and throughput in million rays per second. Two treelet specific statistics are the number of queue switches and the average size of queue that has been scheduled for processing.

The ray traffic bandwidth, created by reading and writing rays from input and into output queues, is in all the measurements an order of magnitude lower than the total bandwidth to the main memory and thus relatively insignificant. We can also see that with the total memory traffic between 2 and 6 GB s⁻¹, we are well beneath the peak performance of current GPU memory systems.

Another very important thing is the ray latency (noted in the table simply as *Latency*). This represents the average number of cycles between receiving a ray into the input queue and writing the result into the output queue. The latency goes from 2 million to 12 million cycles (1-6ms at 2GHz) for both tail recursive and continuation shaders. This effectively prohibits any kind of active

or passive waiting on the shading side. By active waiting we mean an actual spin loop that checks ray status. By passive waiting we mean not scheduling the thread, akin to when threads are waiting for global memory access on NVIDIA GPUs. We would therefore keep a work queue of rays to be processed and whenever tracing a ray is required, we would store the whole shader state, submit the ray query and fetch a different ray from the work queue. Considering that with ray bifurcation the shader state actually contains a ray stack, the whole process becomes significantly more involved than the tail recursive shaders.

Also, looking at the ray throughput, we can see that the tail recursive shaders lead to almost universally bet-

ter results than continuation shaders, and never actually perform significantly worse. The improvement always corresponds to increase in L1 cache hit rate and bandwidth, suggesting that the tail recursion gives us noticeably more coherent rays.

So far we have concluded that the overall best choice would be using tail recursion with L2 cache, giving us 50-100 million rays per second. However, assuming that the L2 cache is occupied roughly equally by both nodes and vertices, it can hold up to 25 % of the whole scene, for each of our scenes. The cache therefore effectively lowers demand on ray coherence, but considering larger scenes, this effect would become less pronounced. We would therefore prefer to increase the coherence itself rather than mitigate the impact of incoherence.

4.2 Treelet implementation

Towards this goal we implemented the treelet approach introduced by Aila and Kerras [1], as described in Section 3.2. The results are summed up in Table 2.

We present the same statistics as in Table 1, but on top of that provide two statistics that are specific to the treelet mechanism. The first is the number of input queue switches. It represents the how many times was the RTE switched from working on one treelet to another. Obviously, the lower the number the better, as each switch effectively means cache invalidation (whole different treelet is loaded). Corresponding to that is the average queue size, measured when the queue's treelet was switched to active. It represents the number of rays that are processed between the cache invalidations. Here, the larger the number the better.

Looking at the L1 hit rate, bandwidth and the overall performance, we can conclude that the treelets do provide overall improvement over the standard implementation. Because the rays are moved to and from the RTE on each treelet boundary crossing we can see an order of magnitude increase in ray traffic bandwidth. This is offset by the fact that due to the increased coherence between rays, the total bandwidth to the memory (including the ray traffic bandwidth) is actually lower than in the implementation without treelets.

We can see drop of about 10 % in the L2 hit rate, combined with a significant drop in the L2 bandwidth. This is also manifested by much closer ray throughput between the versions with and without L2 cache.

The tail recursive shaders clearly and consistently provide better results than continuation shaders, mainly due to the fact that they have much more rays in flight that can be sorted into treelet queues. As result, there are about 10× less queue switches with queues being on average 10× larger than when using continuations.

The only drawback is significant increase in ray latency (to 15-30 million cycles, i.e. 7.5-15ms), which, however, is not so important when tail recursion is used.

4.3 Using BVH

While using B-KD-tree treelets improved the situation significantly for both **Conference** and **Kitchen** scenes, the **Fairy forest** scene showed unsatisfactory results. The treelets did indeed balance the performance between versions with and without L2 cache, but the absolute performance was still only slightly above half of what we could achieve in the other two scenes.

We suspected that the B-KD-tree might be poor fit for the scene and modified our RTE to handle BVH instead. We modified only the simulation engine itself, without any considerations for the changes in area or frequency.

We implemented two different approaches to the BVH. The first approach we call *Node BVH*, where each node contains its own bounding box and only indices to the children. The traversal then checks whether ray hits a node, and if so always proceeds to both children, determining the first one based on node split plane and ray direction. The second approach we call *Child BVH*, where each node contains bounding boxes of both its children and we only descend to the child the ray intersects.

While there is no principal difference between the approaches, two things have to be considered. First, the *Child BVH* needs to perform 12 ray-plane intersections, while the *Node BVH* needs to perform only 6. This essentially means a factor of 2× in terms of area requirements for ray traversal unit. The other thing to consider is treelet implementation. Should we choose to use *Node BVH*, ray can descend to a child that resides in another treelet only to discover it does not intersect the child, thus generating two unnecessary treelet transitions.

	L2	off	on
B-KD-tree [MRays/s]		57.7	64.4
Node BVH [MRays/s]		50.0	54.9
Child BVH [MRays/s]		65.0	73.5

Table 3: **Acceleration structures.** We show performance in million rays per second on the Fairy forest scene, using tail recursive shaders, treelets and three different acceleration structures.

The Table 3 shows that the *Node BVH* drawback of unnecessary treelet transitions outweighs any performance gain by using BVH acceleration structure. The *Child BVH* approach offers approximately 12 % speed up, both with and without L2 cache, but further tests showed that a similar speed up is achieved in the other two scenes as well.

Given the fact that going from very light weight B-KD-tree nodes to BVH nodes required for *Child BVH* approach would introduce significant changes in the whole design, we did not pursue this any further.

We conclude that the lower performance in **Fairy forest** is not due to the acceleration structure itself, but rather due to the complex traversal paths of rays we

generated. This is also supported by the highest number of queue switches as well as the longest average ray latency among the three scenes.

5 CONCLUSION AND FUTURE WORK

We introduced a hardware implementation of a ray traversal engine (RTE), that could act as a fragment generation unit in GPU, in lieu of current rasterization engines. The RTE has been confirmed to run at frequencies above 2GHz and can fit into area less than 15mm², and achieves performance of over 100 million rays per second while keeping bandwidth to the main memory below 5GBs⁻¹.

The unit was tested in two variants, with and without 1MB L2 cache. While the L2 cache is beneficial for the overall performance by mitigating impact of ray incoherence, we implemented a treelet approach to actually reduce the incoherence.

We tested not only on our basic acceleration structure, B-KD-tree, but also on BVH with two versions of traversal, showing that the B-KD-tree offers only slightly lower performance than the significantly more involved of the traversals and is actually superior to the less involved one.

Two competing shader styles are compared, tail recursive shaders and continuations shaders. The first is found to be almost universally better, as even with 16 rays per pixel the memory consumption and bandwidth are very reasonable, while at the same time it provides a large pool of rays that nicely complements the treelet approach.

In the future, analysis of combination and cooperation of multiple such units would prove to be very useful. While testing on significantly more complex scenes proved to be challenge for our simulator implementation, the preliminary results show that feeding the treelet queues with only one unit is not optimal.

In conclusion, we propose that using tail recursive shading does match the feed forward scheme used in the current GPUs rather well, and that hardware ray traversal engine using tail recursive shaders with treelets would be an interesting, useful, and not very demanding addition to the current GPUs.

APPENDIX A

The ray life cycle when tail recursion with treelets is used is the most complicated scheme we are using. We will therefore describe it in more detail. Please refer to Figure 2. The numbers mark data paths used in each step and correspond to the step numbers below:

1. Shader stores ray into Root Queue, a queue associated with root treelet.
2. The input output (IO) unit picks the largest queue for processing.

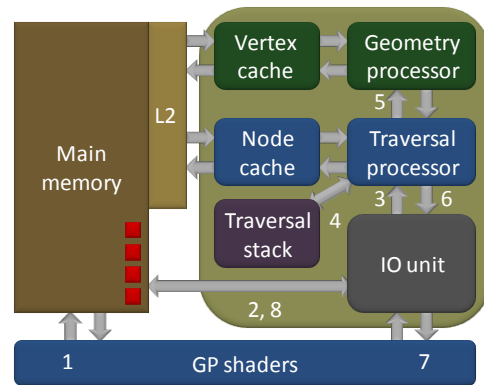


Figure 2: **RTE block diagram.** We use this diagram taken from the presentation to explain life cycle of ray in the tail recursion with treelets configuration.

3. Rays from this queue are sent to traversal processor to traverse ray in the treelet. This is not a block transfer, they are sent only when unit has free processing capacity.
4. During the traversal we use per-ray stack.
5. When ray encounters a leaf, ray triangle test is performed in Geometry processor.
6. When ray encounters a treelet boundary, ray goes back to IO unit.
7. If ray is completely finished, it is sent to shaders and initiates new shader code.
8. Otherwise it is stored in queue corresponding to the treelet it wants to traverse next.
9. When all rays from the processed queue are finished, algorithm goes to step 2. When all queues are empty, algorithm terminates.

REFERENCES

- [1] Timo Aila and Tero Karras. Architecture considerations for tracing incoherent rays. In *Proc. High-Performance Graphics 2010*, 2010.
- [2] Timo Aila and Samuli Laine. Understanding the efficiency of ray traversal on gpus. In *Proc. High-Performance Graphics 2009*, pages 145–149, 2009.
- [3] Caustic Graphics, Inc. CausticRT platform. <http://www.caustic.com/>, 2009.
- [4] Tomas Davidovic, Lukas Marsalek, Nicolas Maeding, Markus Kaltenbach, Peter-Hans Roth, and Philipp Slusallek. Ray Tracing Element for cell/b.e. In *Poster, High Performance Graphics (HPG) 2009, New Orleans*, August 2009.

- [5] Flachs et al. A Streaming Processing Unit for a CELL Processor. In *IEEE International Solid-State Circuits Conference*, pages 134–135, 2005.
- [6] Johannes Günther, Stefan Popov, Hans-Peter Seidel, and Philipp Slusallek. Realtime ray tracing on GPU with BVH-based packet traversal. In *Proceedings of the IEEE/Eurographics Symposium on Interactive Ray Tracing 2007*, pages 113–118, September 2007.
- [7] Vlastimil Havran. *Heuristic Ray Shooting Algorithms*. Ph.d. thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, November 2000.
- [8] Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. Optix: A general purpose ray tracing engine. *ACM Transactions on Graphics*, August 2010.
- [9] Daniel Pohl. Quake wars gets ray traced. *Visual Andrenaline*, 0, 2009.
- [10] Stefan Popov, Johannes Günther, Hans-Peter Seidel, and Philipp Slusallek. Stackless kd-tree traversal for high performance gpu ray tracing. *Computer Graphics Forum*, 26(3), September 2007. (Proceedings of Eurographics), to appear.
- [11] Karthik Ramani, Christiaan P. Gribble, and Al Davis. Streamray: a stream filtering architecture for coherent ray tracing. In *ASPLOS '09: Proceeding of the 14th international conference on Architectural support for programming languages and operating systems*, pages 325–336, New York, NY, USA, 2009. ACM.
- [12] Alexander Reshetov, Alexei Soupikov, and Jim Hurley. Multi-level ray tracing algorithm. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 1176–1185, New York, NY, USA, 2005. ACM.
- [13] Jörg Schmittler. *SaarCOR - A Hardware-Architecture for Realtime Ray Tracing*. PhD thesis, Saarland University, 2006.
- [14] Jörg Schmittler, Ingo Wald, and Philipp Slusallek. SaarCOR – A Hardware Architecture for Ray Tracing. In *Proceedings of the ACM SIGGRAPH/Eurographics Conference on Graphics Hardware*, pages 27–36, 2002.
- [15] Larry Seiler, Doug Carmean, Eric Sprangle, Tom Forsyth, Michael Abrash, Pradeep Dubey, Stephen Junkins, Adam Lake, Jeremy Sugerman, Robert Cavin, Roger Espasa, Ed Grochowski, Toni Juan, and Pat Hanrahan. Larrabee: a many-core x86 architecture for visual computing. In *ACM SIGGRAPH 2008 papers, SIGGRAPH '08*, pages 18:1–18:15, New York, NY, USA, 2008. ACM.
- [16] Ingo Wald. *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Computer Graphics Group, Saarland University, 2004.
- [17] Ingo Wald, Carsten Benthin, Markus Wagner, and Philipp Slusallek. Interactive rendering with coherent ray tracing. In Alan Chalmers and Theresa-Marie Rhyne, editors, *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2001)*, volume 20, pages 153–164. Blackwell Publishers, Oxford, 2001. available at <http://graphics.cs.uni-sb.de/wald/Publications>.
- [18] Sven Woop. *DRPU: A Programmable Hardware Architecture for Real-time Ray Tracing of Coherent Dynamic Scenes*. PhD thesis, Saarland University, 2006.
- [19] Sven Woop, Erik Brunvand, and Philipp Slusallek. Estimating Performance of a Ray-Tracing ASIC Design. In *Proceedings of IEEE Symposium on Interactive Ray Tracing 2006*, pages 7–14, September 2006.
- [20] Sven Woop, Gerd Marmitt, and Philipp Slusallek. B-KD Trees for Hardware Accelerated Ray Tracing of Dynamic Scenes. In *Proceedings of Graphics Hardware*, pages 67–77, 2006.

Virtual LEGO Modelling on Multi-Touch Surfaces

Daniel Mendes
DEI/IST/Technical University of
Lisbon
Av Rovisco Pais, 1
1049-001 Lisboa
Portugal
danielmendes@ist.utl.pt

Alfredo Ferreira
INESC-ID/IST/Technical
University of Lisbon
Rua Alves Redol, 9
1000-029 Lisboa
Portugal
alfredo.ferreira@inesc-id.pt

ABSTRACT

Construction of LEGO models is a popular hobby, not only among children and young teenagers, but also for adults of all ages. Following the technological evolution and the integration of computers into the everyday life, several applications for virtual LEGO modelling have been created. However, these applications generally have interfaces based on windows, icons, menus and pointing devices, the so-called WIMP interfaces, thus being unnatural and hard-to-use for many users. Taking advantage of new trends in of interaction paradigms we developed an innovative solution for virtual LEGO modelling using a horizontal multi-touch surface. To achieve better results, we selected the most common virtual LEGO applications and performed a comparative study, identifying advantages and disadvantages of each one. In this paper we briefly present that study and describe the application developed upon it.

Keywords: 3D Modelling, 3D Object Manipulation, Multi-touch Interaction, LEGO Models

1 INTRODUCTION

With the popularization of multi-touch technology and the decreasing cost associated with its construction, multi-touch interaction is now common among general public. Indeed, from mobile devices to large multi-touch surfaces, a wide range of devices are available. These devices introduced new interaction paradigms, different than those provided by traditional interfaces based on windows, icons, menus and pointing devices, the WIMP interfaces [16]. While some of these paradigms already reached a maturity level, others are still in its infancy. For instance, the manipulation of two-dimensional objects in multi-touch surfaces has been clearly defined and a *de facto* standard is recognized for rotation, translation and scale [6]. On the other hand, several approaches to interaction with three-dimensional objects and 3D scene navigation have been proposed, but none were recognized as a definitive solution.

Due to this obstacle, the number of available multi-touch applications using 3D environments for the common user is limited. During the development a virtual LEGO modelling tool for multi-touch surfaces, we propose an application for manipulating 3D objects that we

believe to be easy to understand and natural to use for a generic audience.

Throughout childhood, many people played with building blocks that fit together to create what the imagination dictated. LEGO is the most famous manufacturer of such toys, globally known for its building blocks and beyond. In reality, building LEGO models is an activity shared by people of all ages. As a complement to traditional plastic blocks, there are now several applications that allows the construction of virtual models.

Although it was shown that building with virtual blocks using a mouse is considerably slower than building LEGO in the real world [1], there are also studies that show that the multi-touch interfaces based on gestures are more efficient than traditional ones. In [8] it was compared the performances of direct-touch, bi-manual and multi-finger for a task of multi target selection. It was concluded that "direct touch with a single finger provides a large performance benefit over over using a mouse and that bi-manual interaction provides a smaller additional benefit." In [2] was tested the selection, translation, rotation and scale of objects in a 3D scene. Results show that, using a multi touch interface, users can perform faster than using WIMP interfaces. We believe that using multi-touch surfaces for virtual LEGO modelling will provide an improved interaction experience for users through more natural and easy to use interfaces.

In the following sections we present a selection of existing virtual LEGO modelling tools, together with a comparative study that identifies advantages and disadvantages of each application. Then, we introduce the current status of multi-touch interfaces regarding ma-

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

nipulation of 3D objects. Next, we describe in detail our approach, with a special focus on LEGO blocks representation in memory and their manipulation in a multi-touch tabletop. Finally, we present some conclusions drawn from the work developed so far and identify the path for future work.

2 VIRTUAL LEGO MODELLING APPLICATIONS

Currently, several applications allow the creation of virtual LEGO models, each presenting peculiarities in relation to others: some want to give the feeling of building a LEGO model as it would be in reality, while others follow a more technical approach, oriented to three-dimensional modelling, some use open-source library parts, while others are proprietary and use their own parts system.

LEGO Digital Designer (LDD)¹, is a proprietary application of the LEGO Company. The LEGO modelling is done in a three-dimensional environment and has an efficient system of connecting parts, preventing their overlap.

This application displays all the pieces available using a list with their previews. It has always a visible grid, simulating a base plate, as can be seen in Figure 1 (a). The movement of parts is done exclusively in the grid plane, to which they adapt. Their rotation is done only along two axes, using intervals of 90 degrees. The camera movement is done by orbiting around a point and never tilting.

Mike's Lego CAD (MLCad)² is a CAD system applied to building LEGO models. It uses four viewports with different views of the model: three orthogonal (one top, one front and one side) and a perspective, as seen in Figure 1 (b). However, it only allows changes to the model made in the orthogonal views, and the perspective view is for viewing only. It uses the LDraw³ open-source parts library.

The search for the parts is carried out using a textual list of names and another for their previews. The movement of the parts is carried out only in a plane parallel to the orthogonal views, which do not have any kind of auxiliary grid and there is no restriction on the parts position. Parts rotations are made at intervals of 90 degrees accordingly to three axes.

LeoCAD⁴ uses the LDraw parts library, as the previous application. Its options provide a grid to which the pieces fit into each half unit, and various sets of viewports. Both the grid and the viewports are not active by

default but can be activated if necessary. To switch between different functions, such as moving parts or rotating the camera, it is always necessary to use the buttons on the interface.

The pieces are presented through a list of groups whose identification, both as parts and their groups, is done just by text, although, after selecting a part in the list, there is a window with a its preview. The pieces translation is done in a horizontal plane, and to move in a vertical plane is needed a special action. For the rotation of the parts it followed the same approach as proposed in [14] to specify the orientation using a traditional mouse, visible in Figure 1 (c). Performing the rotation by clicking on the circumference causes the rotation to be done exclusively on an axis, at intervals of 30 degrees. This application offers many ways to rotate the camera: rotate around her own axis, orbiting around a point and tilt.

There are more applications to create virtual LEGO models, but these three are the most popular between those that have a WIMP interface. LSketchIt [12], built upon LeoCAD, has the particularity to enable the construction of LEGO models using sketches. The search and selection of parts is made by drawing sketches of the pieces to be obtained in the place that the user wants to put it. Given this outline, the program presents a list of suggestions and allows the user to make modifications to the piece, giving new suggestions according to that modification.

Currently there is no application for the creation of virtual LEGO models developed for a multi-touch based interaction.

3 COMPARATIVE STUDY

With the aim of trying to find out the best approach in interacting with objects in multi-touch surfaces, including virtual LEGO building blocks, we started by analysing existing solutions, which use the usual WIMP interfaces, described in the previous section.

3.1 Users Testing

We developed a set of tests, which had the participation of twenty one users, to evaluate the three most common virtual LEGO modelling applications: LEGO Digital Designer, MLCad and LeoCAD. In order to determine the main positive and negative aspects of the various applications we conducted a series of tests with users, that followed the methodology suggested by Preece et al. in [11].

At the beginning of the experiment, the main features of each application were presented to the user, and it were followed by a training phase, consisting of a period of three minutes to interact freely with each application.

After the training phase, each user was asked to complete a task on each application. The task was to con-

¹ LEGO Digital Designer: <http://ldd.lego.com> last visited on October 20, 2010.

² MLCad: <http://www.lm-software.com/mlcad>, last visited on October 20, 2010.

³ LDraw: <http://www.ldraw.org>, last visited on October 20, 2010.

⁴ LeoCAD: <http://www.leocad.org>, last visited on October 20, 2010.

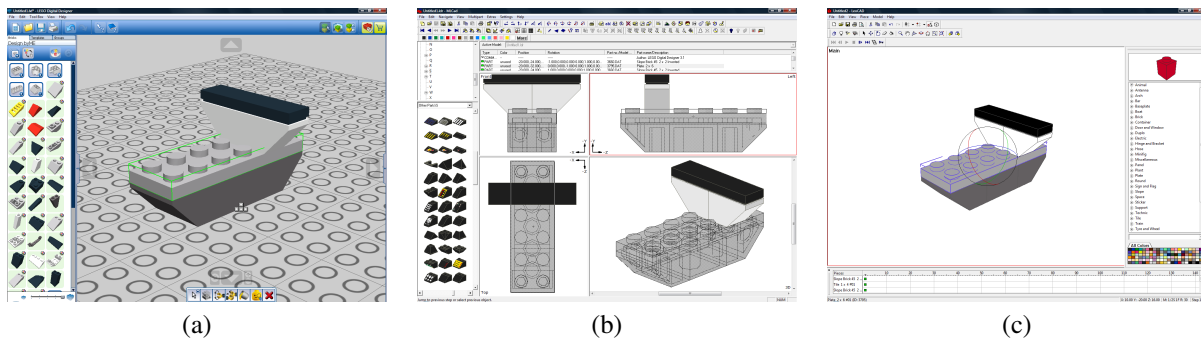


Figure 1: Most common virtual LEGO applications: (a) LEGO Digital Designer; (b) MLCad; (c) LeoCAD.

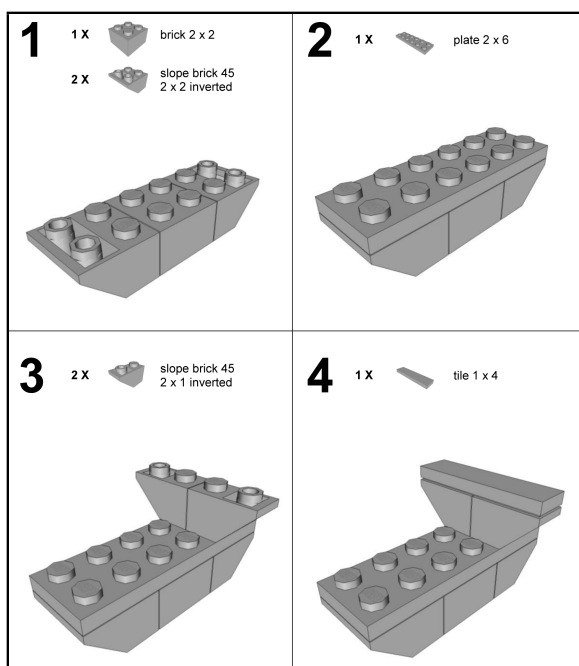


Figure 2: Requested task for users to complete.

struct a simple model, illustrated in Figure 2, which involved searching, selecting, manipulating, rotating and placing various parts. The order in which the users used the applications to carry out the task was randomly selected for each one. In this step, users were asked to think aloud as they performed the task, thus we can better understand what were the main difficulties that users encountered during the task. Since our main goal was to discover what most appeals to users, we did not use any performance measure because it is incompatible with the think aloud technique.

Having accomplished the task in the three applications, each user completed a short questionnaire that focused on the experience with each one. It was hoped thus to identify the strengths and weaknesses in each one of them, with particular emphasis on manipulation of the parts and the camera. The questionnaire consisted on a first set of multiple choice questions to identify the user profile, like age, qualifications and com-

puter experience with image editing, 3D modelling and virtual LEGO tools. It was followed by a second set with five questions to classify the various aspects of each application using a Likert scale with four values and an open-response question so that users could leave comments for each application. These five questions were related to translation and rotation of parts, camera manipulation, searching and selecting parts and overall satisfaction.

3.2 Results

Through the analysis of the qualitative evaluation performed by the users, extracted from their feedback, both written in the questionnaire and said during the task execution, we identified important factors to consider in developing an application for visualization and manipulation of LEGO building blocks. Several of these factors may also be valid in another type of three-dimensional modelling applications.

The results obtained followed a normal distribution, demonstrated by application of the Shapiro-Wilk test, and the average classification was different for each application, proved with an One-Way ANOVA. Results showed that LEGO Digital Designer was the most appealing to users, in all aspects, followed by MLCad and, finally, LeoCAD, with negative ratings for everything except for searching parts. In [10] there are detailed results and analysis thereof. The main advantages and drawbacks we found in each application are summarized in Table 1.

The presented study revealed that users expect a system that tries to simulate the behaviour of the pieces in the real world, contributing to a more familiar interaction with the application. For example, it is desirable to prevent two parts from overlapping each other and pin their positions to a grid. On the other hand, we realize that too much freedom can lead to user frustration, as the case of an application that provided rotation of the pieces according to the three axes simultaneously.

From the conclusions of this study we were able to develop our solution taking advantage of strengths found in the existing applications, as well as sugges-

LDD	<ul style="list-style-type: none"> + Search parts by image + Effective part positioning system + 90 degrees interval for parts rotation + Simple orbiting camera - No part size information - Only two axes for the rotation of parts - No information about which rotation axis is being used
MLCad	<ul style="list-style-type: none"> + Viewports good for experienced users + Bricks rotations clearly identified by axis and direction + Allows the rotation of parts around three axes + 90 degrees interval for parts rotation - Groups of parts identified by name - Does not prevent parts overlapping - Viewports bad for novice users - Need to use interface buttons to rotate parts
LeoCAD	<ul style="list-style-type: none"> + Allows rotation of parts around three axes - Search of parts only by name - Does not prevent parts overlapping - Allows rotation of parts around more than one axis simultaneously - 30 degrees interval for parts rotation - Camera controls, other than orbiting, are frustrating - Constant need to use interface buttons

Table 1: Main advantages(+) and drawbacks(-) of tested applications.

tions made by the users. Our solution combines this strengths and suggestions with a multi-touch interface.

4 MULTI-TOUCH INTERACTION

Following the evolution of computers, interaction methods also change. According to van Dam [16], unlike the predictable evolution of computer components described by Moore's law, interfaces suffer from long periods of stability followed by an abrupt transition. Currently the most popular interfaces are based on windows, icons, menus and pointing devices, such as the mouse, the so-called WIMP interfaces. Since these interfaces are the most popular for over twenty-five years and the technologies capable of providing an interaction more natural and powerful, closer to human interaction, being more common, there is now a need to develop the next generation of interfaces, which the author calls the Post-WIMP.

In [13] it is said that multi-touch based interfaces offer more exciting and efficient ways to interact with information. These interfaces have demonstrated natural metaphors, allowing a coordinated manipulation to replace what would have been multiple actions controlling a cursor.

With the recent work of Jeff Han [3], the interest in touch sensitive surfaces has increased and has accelerated the development of more accessible technologies to create such surfaces. Today there are surfaces that

can detect multiple points of contact simultaneously, using different technologies.

4.1 Multi-Touch Surfaces

With the emergence of multi-touch surfaces, the first step was, naturally, to create new forms of interaction to existing applications. In [7] and [15] an interaction with a multi-touch surface to control Google Earth and Warcraft 3 was implemented. In that context, a set of gestures was developed, both with one and two hands, allowing a more natural way to control the navigation within the applications.

In [18] the authors explored both visualization and interaction techniques to support shared environments and privacy. They created a prototype for organizing a furniture plant, RoomPlanner, which is based on a diverse set of gestures using one finger, two fingers, one hand and two hands.

These are just some examples showing the capabilities of multi-touch surfaces and interaction possibilities. A plethora of applications for this type of surfaces exists, though none was made for virtual LEGO modelling.

4.2 3D Objects Manipulation

Although there are a *de facto* standard to manipulate two-dimensional objects using multi-touch surfaces, the manipulation of three-dimensional objects does not have a definitive solution yet. Regarding this challenge, there are already several approaches.

Hancock et al. [4, 5] presented a set of techniques for manipulating 3D objects. These techniques consist in maintaining the finger always in touch with the initial point of contact in the object. Using one, two and three touches, the user can simultaneously control up to six degrees of freedom. Test results show that an interaction with more contact points tends to be faster, and with just a touch users end up spending more time to carry out cognitive processing.

Two different approaches for moving objects in a three dimensional space are suggested by Martinet et al. in [9]. The first uses multiple viewports, each one responsible for moving objects in its corresponding viewing plane. The second, called Z-technique, consists of a single perspective view in which, with one touch, the user can move the object in a plane parallel to the view plane and, with a second touch, can adjust the depth of the object.

Andrew Wilson [17] follows a different approach, in which there is no need to program a specific interaction through gesture recognition. His solution is to create virtual proxies in the scene with the shape of the user's contact zone with the surface and then use a physical simulation to control these proxies and, consequently, the manipulation of other objects in the scene. However, given the constraints inherent to LEGO building

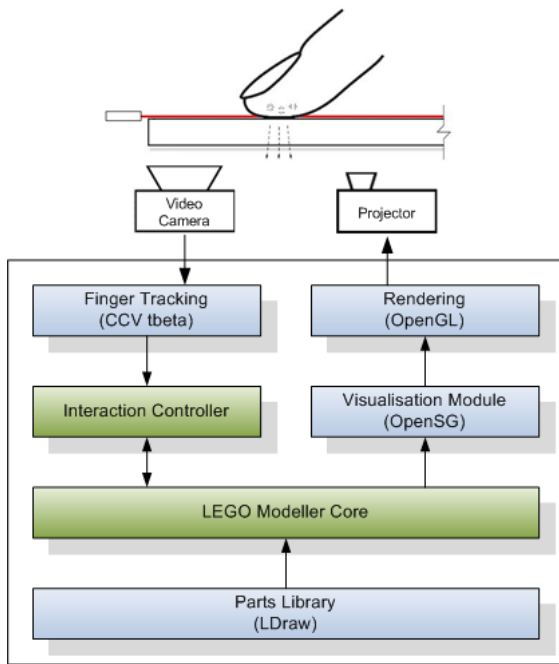


Figure 3: LTouchIt architecture.

blocks, such as the model visualization with free perspectives and the need to move objects in depth, this latter approach does not seem to be most appropriate.

5 LTOUCHIT

Based on the study existing applications, described previously, and on the state of the art regarding the handling of multi-touch surfaces, the development of our solution has been initiated. Next we describe the architecture used, the representation of LEGO pieces and how the interaction can be done.

5.1 Architecture Overview

Our solution is to develop the modules represented in green in the architecture illustrated in Figure 3. The first and most relevant to the job is aimed at the interaction. This is responsible for analysing the movements of the user's fingers from the CCV, in order to determine the gestures made and what action he wants to do. These actions are then passed to the second component, the LEGO modeller. This is the core of the application and it's where all of its logic is programmed. It also contains information about LEGO pieces, the model that is being built and the camera.

All the data related to the scene, including the representation of the used pieces, their position, orientation and details of the visualization, is stored using OpenSG, which undertakes the conversion to OpenGL primitives and the rendering process. The information regarding the pieces to be used in the modeller is gathered from the LDraw parts library. LDraw is an open standard for applications designed to create virtual LEGO models offering a vast library of parts.

Using LEGO modelling as the context, the purpose of this application is the study and development of natural interaction techniques for manipulating three dimensional objects in virtual environments, using multi-touch surfaces.

5.2 LEGO blocks in OpenSG

For the manipulation and visualization of the different LEGO pieces, these have to be converted from its original LDraw format to one that offers features such as transformations and rendering. For this purpose, we have chosen to work with OpenSG⁵, an open source scene graph system to create real-time graphics programs. In this conversion we chose to follow an approach as close as possible to the specification of LDraw. Therefore, we shall briefly explain its structure.

The files of this format are formed by six types of data: triangles, quadrilaterals, lines, optional lines, inclusion of sub-files and META commands. The latter are intended to specify the particularities of the current file or the command that will follow. Due to the large amount of these commands, they will not be explained here, except those referring to back-face culling. The inclusion of sub-files is intended to include parts or sub-parts defined in another file within the geometry of the current part. Each included sub-part is associated with a transformation matrix relative to the including part.

The remaining types consist of the representation of the part itself. The optional lines are lines that should only be drawn as some points are visible or not and serve, for example, to draw the contours of cylinders. This type of line, given the complexity of its implementation and its limited relevance, was disposed of in this implementation for now.

Thus, in our application, each piece will have a structure like the one visible in Figure 4. Basically each part is a scene graph, being the part itself the primary node which contains the transformations, as translations and rotations, to be applied to the part. Thus, by applying a transformation to this node, this transformation is automatically applied to all of its children. This node is also the parent node of the geometry nodes of the part and all of its sub-parts. There are two OpenSG nodes for information on the geometry of the piece. The first one has the part geometry itself - triangles and quadrilaterals, and the second has the geometry of the lines. To disable the drawing of lines we simply have to disable this second node.

In each piece there is also a reference to the structure corresponding to each of its sub-parts, in order to be able to change the color of a piece. To do so, we have to change the color of the geometry of the part and the geometry of each of its sub-parts recursively.

⁵ OpenSG: <http://www.opensg.org>, last visited on October 20, 2010.

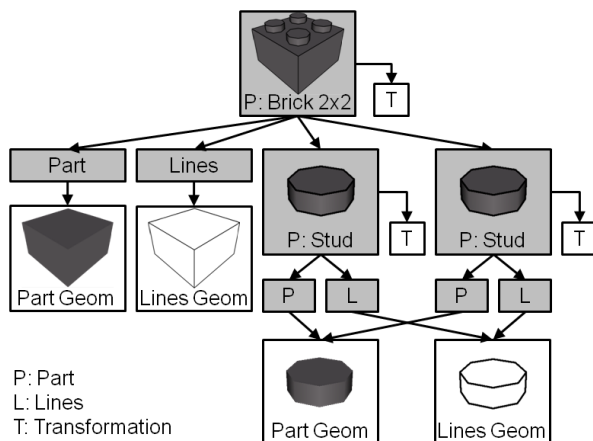


Figure 4: Our LEGO part structure for a simplified 2x2 brick. The grey filled boxes represent OpenSG nodes, while the white ones are the nodes' core.

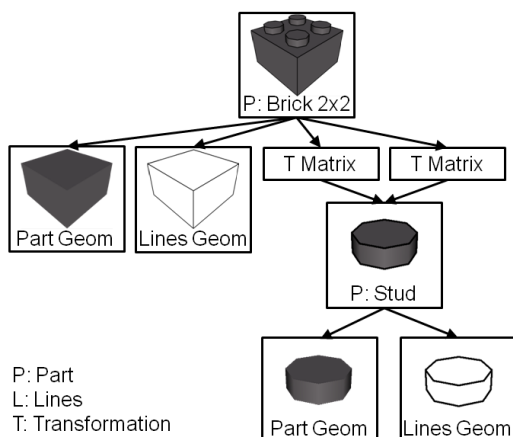


Figure 5: Our LEGO part cache structure for the same 2x2 brick.

To achieve more efficiency in terms of memory and since many parts have sub-parts in common, we built a cache of parts, illustrated in Figure 5, which is fully loaded at the beginning of the application execution. Contrarily to the part structure described above, this cache is not a scene graph. It has, for each part, the geometries required for each color and the transformation matrices to be applied to each sub-parts, also in cache. With this approach, it is possible that only one geometry is created for each part, being shared by all parts that include it, requiring only the application of the transformation matrix for each case.

As mentioned, the only LDraw META commands that will be explained here will be the ones regarding back-face culling. Other ones have been disregarded. The specification of the LDraw file lets one decide using META commands which culling should be made, according to the vertices of the polygon are specified clockwise or counter-clockwise. In addition, there is a META command that allows to reverse the test for the clipping of polygons in the middle of the file itself. Thus it was necessary that the insertion order of

the polygon vertices in the OpenSG geometry was different depending on the specification of each particular situation, in order to be able to support this diversity.

Regarding the conversion of parts of the LDraw format for OpenSG remains only the fact of both having different coordinate systems. While the former uses a right-handed coordinate system, the latter uses a left-handed one. To circumvent this problem it is necessary for the main node of each piece, and not in any of its sub-parts, to be applied a scale transformation with the values of 1, -1, -1 for x, y and z respectively, i.e. an inversion of the part relatively to the y and z axes.

5.3 Current set of LEGO blocks

Although the LDraw parts library contains more than three and a half thousand pieces, not all of them were considered in our application. This measure was undertaken with the aim of reducing the complexity of the implemented solution, since the primary focus of this work is the manipulation of three dimensional objects in multi-touch surfaces, and not to achieve the total diversity of the supported parts.

We discarded all the pieces that have geometries that contain incorrectly specified polygons. These are polygons that contain points that are not co-planar, non convex angles and/or vertices that are not defined in the correct order. We also accept only pieces that support some sort of back-face culling and that use only the default LDraw color, so that the piece is all coloured alike. Whenever any piece includes some sub-part that does not meet the requirements cited above, it is also discarded.

With these restrictions, we currently support more than eight hundred pieces and more than eighty colours, including colours with alpha. It is expected, however, that the number of used parts will decrease, given the complexity of some of them regarding the bricks adaptation and collision detection, something that will be further developed.

5.4 LEGO Modelling on a Tabletop

A first approach to interact with LEGO bricks, described next, is already implemented. Our solution is being developed in a multi-touch table that uses the technology Laser Light Plane (LLP). This table has dimensions of 180x120 cm and uses a projector with 720p resolution.

LLP technology allows the detection of multiple points of interaction using the dispersion caused by the user fingers on the laser beams placed over the surface. This technology has a lower construction cost and is more accurate than other optical technologies like FTIR, DI and DSI.

Our application, illustrated in Figure 6, is designed for Microsoft Windows environment, using C++ programming language and OpenSG.



Figure 6: User interacting with LTouchIt.

Inside the table, the camera receives the rays reflected by the fingers of the user and the obtained image is sent to the CCV tbeta. This program processes this image, recognizes the touches and sends their positions for the Interaction Controller, using the TUIO protocol. Whenever the controller receives a message from CCV tbeta, it originates an event. This event is then processed and the position of the fingers over time is analysed to determine the gestures of the user, transmitting to the corresponding actions to the Core, such as moving a piece or rotate the camera.

For the generation of the image to be projected on the table, OpenGL goes through the scene graph of the model being worked on the LEGO Core, originating all the OpenGL commands needed to render the scene, such as the geometries of all the objects in the scene and the position of the camera and the light sources. In order to have the geometry information of all the LEGO bricks, these are all loaded and stored in the Core, according to a procedure described in the previous sections.

It is already implemented the internal representation of LEGO pieces and the interaction techniques next described.

5.5 Manipulation of virtual LEGO blocks

To interact with three dimensional objects on multi-touch surfaces, which in our case are LEGO bricks, we followed some of the techniques already presented.

Concerning the translation of the pieces, we turned to two approaches. The first is used to move them in an horizontal plane. This approach consists of, after selecting a piece, dragging it by moving a finger in contact with it, making the piece follow the touch, while remaining in this plane. This is the technique that can be found in several applications currently available for creating LEGO models. We chose this approach instead of moving the object in a plane parallel to the camera

plane [9][5], since we believe that this movement can become confusing when the camera plane is not perpendicular to one of the scene axis, due to the translation being made in the three axis simultaneously.

For the movement of the pieces according to the third dimension is used a technique similar to the Z-technique [9]. Keeping a first finger in contact with the piece, it is used the vertical motion of a second finger to regulate its depth in relation to the camera, as can be seen in Figure 7.

The application also has a grid that underlies the construction in which the parts fit after every movement.

Regarding the control of the camera while viewing the model, we followed an approach used in some existing applications of virtual LEGO. While keeping a point of the model centred, the user can orbit around that point by dragging a finger, whose touch was started out of a selected piece. The movement is made like dragging a spherical surface, moving sideways, up and down, with the particularity that it never tilts. To zoom in and out the scene, we again use the vertical motion of a second touch, as shown in Figure 8. This gesture is identical to the Z-technique, thus diminishing the vocabulary needed to interact with the application.

6 CONCLUSIONS AND FUTURE WORK

In this paper we presented an innovative application for for constructing virtual LEGO models. This application takes advantage of multi-touch interfaces on a distinct approach to LEGO modelling. However, the current prototype is not a final version. Indeed it stands more as a proof-of-concept prototype with which we seek to find the most natural way to interact with three dimensional objects - the LEGO blocks - in multi-touch surfaces.

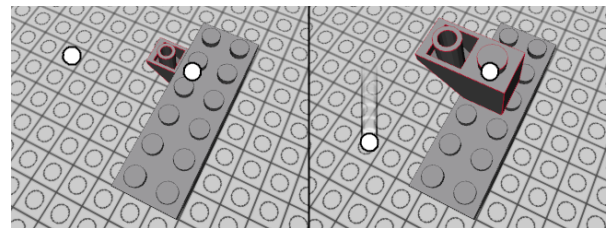


Figure 7: Depth manipulation using two touches.

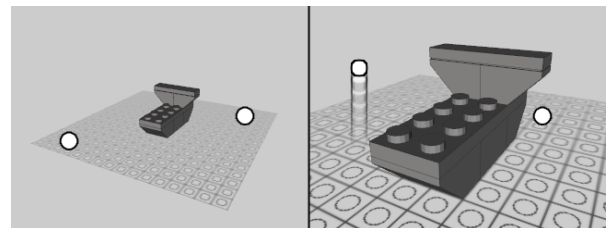


Figure 8: Camera zoom using two touches.

The development of this prototype was based on the results of a comparative study of existing virtual LEGO modelling applications. This procedure guided us into developing a solution easy-to-use for all users. However, as stated before, we are aware that such application still in its infancy and a considerable effort must be dedicated to augmented it with additional features.

Indeed, the next step of our work will focus on extensive user evaluation of our LTouchIt prototype. With this evaluation we intend not only to validate the proposed approach for virtual LEGO modelling, but also to identify further enhancements in the prototype. We believe that we are on the correct path for introducing a simple and easy-to-use 3D manipulation on multi-touch surfaces that will allow users to build LEGO models in a virtual environment as fast as they do in real world.

7 ACKNOWLEDGEMENTS

The work described in this paper was partially supported by the Portuguese Foundation for Science and Technology (FCT) through the project 3DORuS, reference PTDC/EIA-EIA/102930/2008.

REFERENCES

- [1] H. Baradaran and W. Stuerzlinger. A Comparison of Real and Virtual 3D Construction Tools with Novice Users. *CGVR'06*, 2006.
- [2] D. Fiorella, A. Sanna, and F. Lamberti. Multi-touch user interface evaluation for 3d object manipulation on mobile devices. *Journal on Multimodal User Interfaces*, 2009.
- [3] J. Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. *Proceedings of the 18th annual ACM symposium on User interface software and technology*, 2005.
- [4] M. Hancock, S. Carpendale, and A. Cockburn. Shallow-depth 3d interaction: Design and evaluation of one-, two- and three-touch techniques. *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2007.
- [5] M. Hancock, T. ten Cate, and S. Carpendale. Sticky tools: Full 6dof force-based interaction for multi-touch tables. *ITS '09: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, 2009.
- [6] M. S. Hancock, S. Carpendale, F. D. Vernier, D. Wigdor, and C. Shen. Rotation and translation mechanisms for tabletop interaction. In *TABLETOP '06: Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, 2006.
- [7] S.-G. Kim, J.-W. Kim, K.-T. Bae, and C.-W. Lee. Multi-touch interaction for table-top display. *Advances in Artificial Reality and Tele-Existence*, 2006.
- [8] K. Kin, M. Agrawala, and T. DeRose. Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. In *GI '09: Proceedings of Graphics Interface 2009*. Canadian Information Processing Society, 2009.
- [9] A. Martinet, G. Casiez, and L. Grisoni. The design and evaluation of 3d positioning techniques for multi-touch displays. *IEEE Symposium on 3D User Interfaces 2010*, 2010.
- [10] D. Mendes and A. Ferreira. Estudo comparativo de aplicações para a construção de modelos lego. In *Actas da 4ª Conferência Nacional em Interacção Humano-Computador*, 2010.
- [11] J. Preece, Y. Rogers, and H. Sharp. *Interaction Design: Beyond Human-Computer Interaction*, chapter 14. John Wiley and Sons Ltd, 2002.
- [12] T. Santos, A. Ferreira, F. Dias, and M. J. Fonseca. Using sketches and retrieval to create lego models. *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*, 2008.
- [13] T. Selker. Touching the future. *Communications of the ACM*, 2008.
- [14] K. Shoemake. Arcball: a user interface for specifying three-dimensional orientation using a mouse. In *Proceedings of the conference on Graphics interface '92*, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.
- [15] E. Tse, C. Shen, S. Greenberg, and C. Forlines. Enabling interaction with single user applications through speech and gestures on a multi-user tabletop. *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, 2006.
- [16] A. van Dam. Post-wimp user interfaces. *Communications of the ACM*, 1997.
- [17] A. Wilson. Simulating grasping behavior on an imaging interactive surface. *ITS '09: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, 2009.
- [18] M. Wu and R. Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. *Proceedings of the 16th annual ACM symposium on User interface software and technology*, 2003.

New path planning method for computation of constrained dynamic channels in proteins

Petr Beneš
Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic
xbenes2@fi.muni.cz

Ondřej Strnad
Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic
xstrnad2@fi.muni.cz

Jiří Sochor
Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic
sochor@fi.muni.cz

ABSTRACT

Collision-free paths in the geometric model of a protein molecule reveal various dependencies between the structure of the molecule and its function. The paths which connect a biochemically important part of the protein molecule with the surface of the molecule can serve as egression or access paths for small molecules which react in the active site. The geometric method introduced in this paper is designed to compute such paths in the dynamic models of protein molecules. The paths have to satisfy additional constraints such as valid flow of time which allows us to distinguish between access and egression paths, minimum width and others. Possibly, the method may be used not only for protein molecules but also for similar environments with high density of spherical obstacles. The method was tested on real protein data and the results indicate that if there is a path present, it is detected by our method.

Keywords: protein, path planning, collision-free path, constrained dynamic channel

1 INTRODUCTION

Investigating the behaviour of protein molecules is one of the main tasks that help biochemists fully understand how the real micro-world works. One of the real applications is the development of new and the improvement of existing drugs. These days, we can utilize the computational power and perform simulations instead of wasting time in the laboratory.

With the increase in computational power the molecular dynamics simulation which simulates the behaviour of the protein molecule is capable of capturing longer time intervals. The resulting vast amount of data has to be processed and for chemists, it is desired to reveal the dependency between structure and function of a protein molecule. One of the crucial analyses is to find the collision-free paths connecting the active site of the protein molecule with its surface. These paths are called channels and may be used by a small molecule as access paths to or egression paths from the active site. Chemists need to focus on these paths to assess whether the active site is well or poorly accessible.

In the protein molecule, each atom is represented geometrically as a sphere on given three-dimensional coordinates with appropriate Van der Waals radius. To

reflect the real situation, we have to consider the movement of atoms in the molecule. This movement certainly influences channels and their properties. The movement is typically obtained by a molecular dynamics simulation and is stored as a set of snapshots – states of the molecule over time. The simulation may generate thousands of snapshots. The set of snapshots is denoted by chemists as a trajectory.

Previous approaches to channel computation were able to compute channels in a single snapshot only. Even if the approach was applied to each snapshot, the solution did not exactly reflect the real situation in which the small molecule (substrate or product) passes into or leaves the active site. The ongoing research demanded that collision-free paths which satisfy certain constraints and connect the active site with the surface of the molecule are determined. Although other possibilities exist, in this paper, the surface of the molecule is understood as the convex hull boundary.

In this paper, we introduce new method, which joins the information about an empty space in different snapshots into one large graph and finds collision-free paths connecting the active site with the surface of the molecule. These paths are called dynamic channels. Applying additional constraints on these paths allows us for instance to consider the flow of time and distinguish between access and egression paths. These paths are referred to as constrained dynamic channels.

The method we propose is based on computational geometry principles and could be used to find collision-free paths for a spherical robot in an environment with high density of moving obstacles. We expect these ob-

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.

stacles to be spherical too. Our specific implementation is used for solving the issue of finding dynamic channels in a protein molecule.

2 BASIC DEFINITIONS

A channel in a protein molecule is defined by the centerline and the volume [MBS07]. The centerline is a three-dimensional continuous curve and the volume is formed by the union of spheres with centers on this centerline and with appropriate radii so that none of the spheres intersects any atom in the molecule. The bottleneck radius of the channel (i.e. the minimum sphere inserted) limits the size of a molecule which is able to pass through a channel. In addition to previous definition, a dynamic channel contains for each point on the centerline the reference to a particular snapshot in the trajectory.

Definition 1: Let $M = \{m_1, \dots, m_n\}$ be a time-ordered set of snapshots. Dynamic channel T is defined as $T = \bigcup_{x \in a_T} s(x, r, i)$ where a_T is the centerline of T (a three-dimensional curve) and $s(x, r, i)$ is a sphere with center x and radius r which does not intersect nor contains inside any atom in the reference snapshot m_i .

The definition of a dynamic channel is general. For the purposes of this paper there is a need to apply various constraints. The constraints may be geometrical (such as minimal width) or time (such as the order of snapshots). For each dynamic channel we are able to determine if it satisfies the given constraints. For example, if a dynamic channel with the centerline a_T , connecting an active site with the surface is to be an egression path, the basic constraint which should be satisfied is that the reference snapshot numbers should be increasing from the active site. More formally, for each $a, b \in a_T$ such that a is closer in a_T to the active site than b , if the reference snapshot number for a is smaller than reference snapshot number for b , then the path satisfies the egression path constraint. Otherwise, the constraint is not satisfied.

3 RELATED WORK

3.1 Motion planning methods

In the environments with static obstacles, there are various approaches that can be used to find a collision-free path from a starting position to a destination position. Possible methods are cell decomposition [Lin04], visibility graphs [JLK95], Minkowski sum [Gho90] and potential fields [HA92]. The issue of path planning gets more complicated when considering environments with moving obstacles.

In the dynamic environments, there are approaches which are designed for the two dimensional case. As an example, spatial indexing [FS88], velocity obstacles method [FS98] and hierarchical strategies [FS89] can be mentioned.

The methods designed for three-dimensional environments with moving obstacles are much rarer. As an example the following methods introduced in [YJSHJ06], [BKZ⁺07] and [HKcLR00] can be mentioned.

Our solution of path planning is done in an environment with different settings. Firstly, obstacles in our environment are always spherical and all of them are moving. Secondly, all obstacles are of similar size and their movement is known a priori. In this environment, our method tries to find paths connecting the starting position with a destination position which satisfy user-defined constraints. For each path the maximum size of the spherical robot which is able to pass through can be determined.

The proposed solution utilizes computational geometry principles Voronoi diagram and Delaunay triangulation [SU00] and is based on the breadth first search algorithm (BFS) [CLRS01].

3.2 Channels

The issue of computation of channels in one snapshot of the molecule was described in [POB⁺06] and the improved approach based on computational geometry was proposed in [MBS07]. Other approaches which appeared later [PKKO07, YFW⁺08] are based on similar concepts.

The extension to dynamics where channels are computed separately in each snapshot was proposed in [BMS09]. It should be stressed that the cluster analysis which was utilized in this issue is not directed to find a dynamic channel as defined above since there is no explicit notion of continuous movement through the set of channels clustered together.

4 PROPOSED METHOD

The input data for the proposed method include the geometric model of the molecule which is composed of a set of spheres in three dimensions. The molecule contains thousands of atoms; each atom corresponds to a sphere in the model. Additionally, the movement of atoms is sampled with a fixed frequency, producing a set of snapshots (a trajectory). Each snapshot represents the positions of atoms in a certain instance of time and in fact is a static model. The capture of snapshots is dense enough so that no important movements of atoms are missed. This implies that there is a certain limit for the translation of each atom between consecutive snapshots.

In addition, it is necessary to specify the starting point and the set of destination points. For protein analysis, starting point should be located in the active site cavity and destination points may be automatically determined as points lying on the surface of the molecule.

It should be noted that the trajectory typically describes local movement of atoms only, while the global

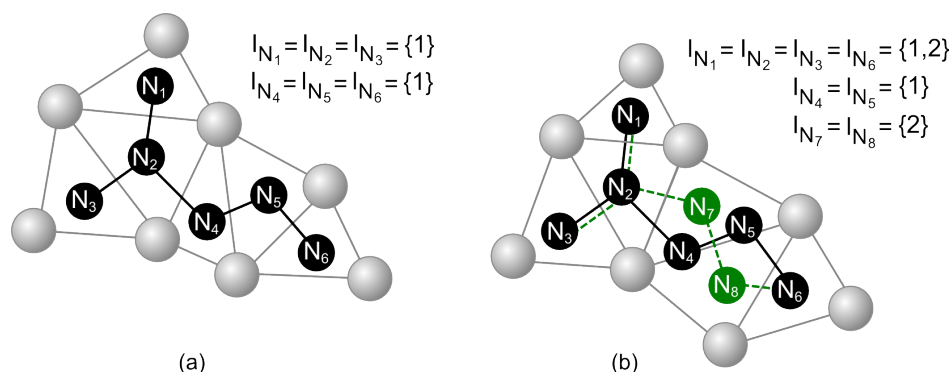


Figure 1: Construction of a graph from the Delaunay triangulations. (a) Graph after processing of the first snapshot. (b) In the next snapshot, the movement of atoms caused changes in the Delaunay triangulation. New edges (dashed green) and nodes (green) were added into a graph after processing this snapshot.

movement of the molecule is not present. Various alignment tools can be used in case the data do not satisfy this condition.

The proposed method is purely geometrical. It processes the geometric model of the molecule which consists of a set of spheres in three dimensions; each atom corresponds to one sphere. No biochemical properties are considered. This makes the method general and applicable in similar situations and similar dense environments in which feasible paths are to be computed. For protein analysis, the method can be adjusted so that biochemical properties are taken into account to find the most prospective and biochemically relevant paths.

The method is composed of two main phases – the graph creation phase and the graph processing phase. In the first phase, the Voronoi diagrams (or dual Delaunay triangulations) are merged into a large graph. This graph captures the behaviour of the geometric model over time. In the second phase the graph is processed searching for feasible paths which satisfy defined constraints.

4.1 Phase 1: create graph

Recall that $M = \{m_1, \dots, m_n\}$ be a time-ordered set of snapshots. For each snapshot m_i , we compute the Delaunay triangulation DT_i that maintains the space partitioning of the molecule. The Delaunay triangulation is computed using the QuickHull algorithm [BDH96]. It should be stressed that each tetrahedron in the triangulation contains the indices of four atoms in the molecule. In general, all triangulations DT_i for M are merged into one multi-edge graph G .

Due to the fact that there is a fixed number of different atom radii and that the radii do not vary greatly, the Delaunay triangulation for atom centers can be used. For general situations in which the size of spheres (obstacles) would differ more, the additively weighted Delaunay triangulation would have to be computed.

The algorithm starts with an empty graph G . For each snapshot m_i and for each tetrahedron $t \in DT_i$, a

new node N containing a reference to t , is created, $Ref(N) = t$. Two nodes are considered equal if the atom indices of their reference tetrahedra are equal. If G does not contain an equal node, then N is inserted into G . This ensures that the associated tetrahedra for nodes are unique.

Moreover, for each node N a set I_N of indices of snapshots is maintained. If $i \in I_N$, then $Ref(N)$ was a tetrahedron in the Delaunay triangulation in the snapshot m_i .

Let N_1 and N_2 be the two nodes in G . For each snapshot m_i in which tetrahedra $Ref(N_1)$ and $Ref(N_2)$ share a face, a new edge e connecting N_1 and N_2 is inserted into G , then $e = (N_1, N_2, i, v)$ where i is the snapshot number and v is the representation of width. For short, let $Snap(e)$ denote the snapshot number of e , i.e. $Snap(e) = i$. The process of computing the width representation follows.

For each edge $e = (N_1, N_2, i, v)$ in G , a Voronoi edge exists which is dual to the shared face between two tetrahedra $Ref(N_1)$ and $Ref(N_2)$ in DT_i . The value of v can be computed as the distance between the corresponding Voronoi edge associated with e and the surface of the nearest atom in i -th snapshot. The issue of computation of the values on edges for a Delaunay triangulation from only one snapshot was addressed in [MBS07]. The referred value v represents the maximum possible radius of a spherical probe which is able to pass along the corresponding Voronoi edge without colliding with any of the atoms. As a result, a multi-edge graph G is created. Note that there may be more than one edge between two nodes in G if associated tetrahedra for these nodes were present in the Delaunay triangulation in more than one snapshot or even no edge if both associated tetrahedra did not share a face in any of the triangulations DT_i . The example of such a graph is shown in Fig. 1.

To keep the number of edges in G reasonable, it is convenient to filter the edges in G by their value. If the computed value v for an edge is less or equal to

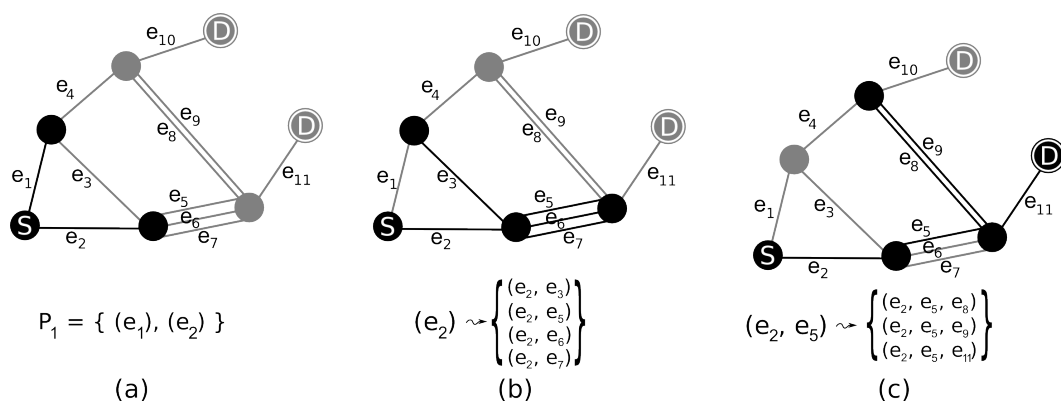


Figure 2: Modified BFS. Starting node (S), destination nodes (D). (a) First step of the algorithm in which paths of length one are created. (b) Extension of a path $p = (e_2)$ with edges emanating from the last node of p . (c) Extension of path $p = (e_2, e_5)$ during which a path between starting and destination node $p_{new} = (e_2, e_5, e_{11})$ is found.

user-defined limit, the edge is not added to G . Without filtering, all edges are used regardless of their value.

Once the multi-edge graph is computed, it can be stored for further processing. This graph represents the discrete evolution of the proximity information about the geometric scene over time.

The starting node in G from which the search for paths begins is selected by its associated reference tetrahedron. For the purposes of protein analysis, a node whose tetrahedron encloses three-dimensional coordinates of the active site is selected. In other applications, an arbitrary node may be selected.

For the purposes of protein analysis, the paths which end on the convex boundary of the molecule are important. Each node in G whose associated tetrahedron appeared on such boundary of the molecule in at least one snapshot is marked as a destination node. In other applications, a set of destination nodes may be selected according different criteria.

4.2 Phase 2: process the graph and find constrained paths

Given multi-edge graph G and starting node and a set of all destination nodes we search for paths satisfying a set of constraints. The straightforward solution in which no path is excluded is to generate all possible paths which emit from a starting node. To find such paths, G is processed using the modified breadth-first search (BFS) algorithm.

A path p consists of edges from G , $p = (e_1, \dots, e_m)$ where $e_i \in G$. The length of a path is denoted as the number of edges in p . The modified BFS generates paths in steps; in each step, paths of length increased by one are created.

The paths which consist of consecutive edges in G and connect the starting node with one of the destination nodes can be mapped to a centerline of a dynamic channel. The mapping is described later.

The modified BFS search algorithm works as follows. In the first step, paths of length one (containing a single edge which originates from the starting node) are created. In the next step, all paths from previous step are extended with edges emanating from the last node in the path being extended. To prevent cycles, a path is extended with edges whose end node is not already present in the path. With this approach the set of all feasible paths of variable length from the starting node to all other nodes are computed.

The generation of new paths may terminate once a path connecting the starting node with any of the destination nodes is found. Alternatively, the generation of paths may continue to find more than one path. Also, if multiple destination nodes were selected, it may be crucial to find all suitable paths ending in some of these nodes. Otherwise, the computation terminates after paths to all reachable nodes are checked.

The described generation of feasible paths produces a vast number of paths. If there was an unlimited computational power and storage, it would be possible to search the whole graph for all paths connecting starting and destination nodes. Currently, this is not possible and various restrictions have to be applied to keep the number of paths reasonable. The description of various techniques which help us to cope with the number of paths follows, including the concept of constraints.

As mentioned above, paths are generated in steps. In the first step, the set of paths P_1 of length one is created; each of these paths has exactly one edge which emanates from the starting node. Then, in $(i + 1)$ -th step, each path $p = (e_1, \dots, e_i)$ from P_i is extended with every edge e_{new} which emanates from the last node in the path p (except for cases when p already contains the end node of e_{new} to prevent cycles) resulting in a new path $p_{new} = (e_1, \dots, e_i, e_{new})$ which is inserted into P_{i+1} . Finally, the set P_i is discarded and no longer maintained.

The example progress of the algorithm is shown in Fig. 2.

To keep the number of paths reasonable, it is convenient to filter non-prospective paths during the extension process. More precisely, in the i -th step, each path of length i can be extended by many edges to form paths of length $i + 1$. The concept of constraints is utilized to decide whether an extension is appropriate instead of blindly creating extended paths and filtering them afterwards.

Let us now explain various constraints and their application in the process of extending a path $p = (e_1, \dots, e_m)$ with an edge e_{new} , resulting in a new path $p_{new} = (e_1, \dots, e_m, e_{new})$. The following list describes important constraints which take into account the minimum clearance of the path, proper flow of time, maximum speed of a robot, waiting of a robot and a limited curvature of the path. The list, however, is not exhaustive and one can design other constraints that can be applied. The selection of constraints below is motivated by the computation of the widest straight-leading paths which are safe to pass along over time.

- **Minimum radius of a path.** If the width of e_{new} is lower than the minimum value specified, the extension does not happen and p_{new} is not created. See Fig. 3 (b).
- **Increasing snapshot number.** For paths, which should serve as egression paths from the active site, the snapshot numbers for edges in the path have to be increasing. If $Snap(e_{new}) < Snap(e_m)$, the extension does not happen since the extended path would not satisfy the constraint. Additionally, for the access paths, the decreasing order of snapshot numbers has to be used. This constraint is the most important for the validity of the path over time. See Fig. 3 (c).
- **Speed of the robot.** If there is more than a certain number of edges having the equal snapshot number in the path, it would be impossible for a robot to pass through in such a short time. Let $time$ be the time between the each two consecutive snapshots in the trajectory. If there is a certain number of edges having the equal snapshot number in the path, the sum of lengths of these edges divided by the speed of the robot cannot exceed $time$. In case all edges in the multi-edge graph are of approximately the same length, the constraint can be formulated in the following simpler form. If the extension of a path causes that a last certain number (experimentally derived) of edges have equal snapshot number, the extension does not happen. See Fig. 3 (d).
- **Waiting of a robot in a graph node.** For this constraint, the additional information stored for each

node in I_N is utilized. Note that I_N contains snapshot numbers in which it was safe for the robot to wait in node N (Fig. 1). Let N be the node shared by e_m and e_{new} . If I_N contains all integer numbers from interval $\langle Snap(e_m), Snap(e_{new}) \rangle$ and the radius of the inscribed sphere to a tetrahedron $Ref(N)$ is larger than the size of the robot, the transition between e_m and e_{new} is safe and the robot is able to wait in N . See Fig. 3 (e).

- **Curvature of the path.** For every new edge e_{new} to be added we are able to compute an angle α between e_{new} and e_m . If α is above a user specified value the extension does not happen. If desired, this allows us to exclude paths exceeding a user-defined curvature threshold. See Fig. 3 (f). Note that this definition limits the angle between each pair of consequent edges in the path only. For our purposes this definition is sufficient. However, other definitions which consider the whole path can be used.

For the practical case, the number of paths may be extremely large for huge multi-edge graphs even if previously mentioned constraints are applied. One of the other possibilities to reduce the number of paths is adding new constraints. There are also other techniques which do not belong to the category of constraints but still help us to reduce the number of paths so that the computation is feasible when only a limited amount of memory is available. The techniques typically result in the loss of paths.

In i -th step, when paths are created, we can stop the extension and proceed to the next step if the number of generated paths of length i is larger than a user defined value. It is obvious, that this approach prefers such paths which are to be found earlier in the process of generating paths.

The alternative approach we propose is to leave only one path for each node in G . This way, a number of paths is limited by the number of nodes in G . It is crucial that the best candidate path is kept for each node to reduce the number of lost paths. Let $P(N)$ be a set of paths whose last node is N . The selected candidate path $p \in P(N)$ for a certain node N is such a path $p = (e_1, \dots, e_m)$ for which $Snap(e_m)$ is minimal. This is the safest way which ensures that if there exists a path satisfying the most important time flow constraint (increasing / decreasing order of snapshots) it will be generated.

Our algorithm is general enough that the output of paths can also be done during the extension process. If the ending node of e_{new} is marked as destination, it is clear that a path which connects the starting node with the destination node exists. The path is stored and is no longer extended. However, the process of generating paths may continue to find other different paths connecting the starting node with a destination node.

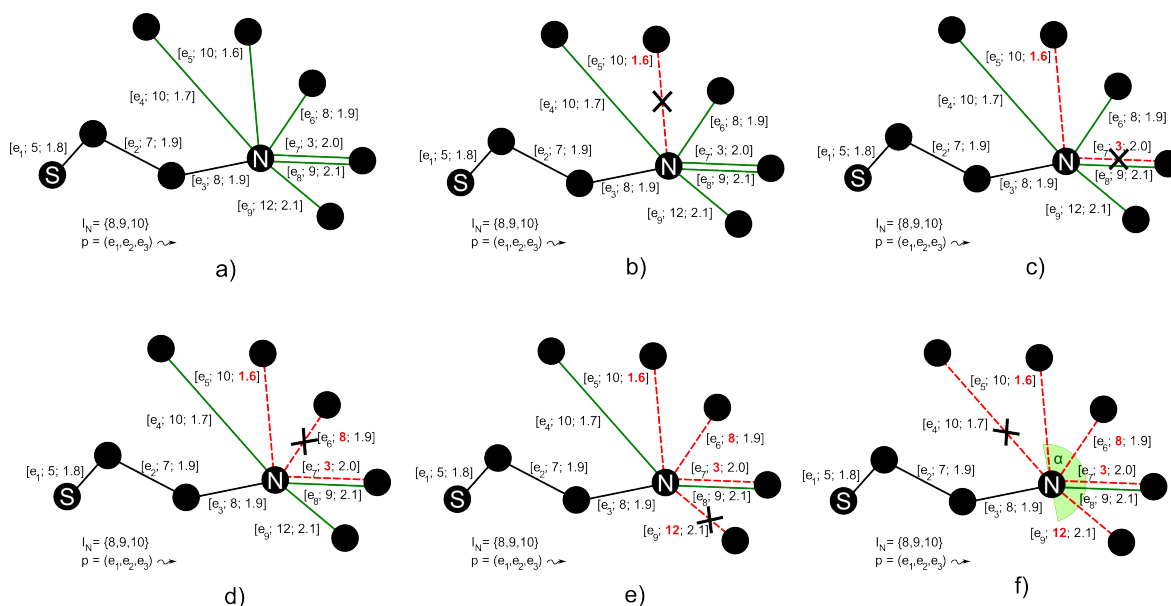


Figure 3: Extension of a path $p = (e_1, e_2, e_3)$ with $e_4, e_5, e_6, e_7, e_8, e_9$ and applying constraints. Crossed dashed lines denote the edges which do not satisfy given constraints. (a) Extension without constraints. (b) Minimum-width constraint with value of 1.7. Edge e_5 does not meet the minimum width requirement. (c) Time-flow constraint. The path is treated as egression and the numbering of snapshots should be increasing. Newly added edge e_7 does not satisfy the constraint. (d). Speed of the robot. In this case, we do not allow two consecutive edges to have the same snapshot number. Edge e_6 does not satisfy the constraint. (e) Waiting of the robot. Edge e_9 does not satisfy the constraint as some of the snapshot numbers between 8 and 12 is missing in I_N . (f) Angle constraint. Edge e_4 does not satisfy the constraint of maximum angle of 90 degrees shared by e_3 and e_4 . The angle α which satisfies the constraint is emphasized.

5 BIOCHEMICAL APPLICATION: MAPPING A PATH TO A DYNAMIC CHANNEL

A path connecting the starting node with a destination node is mapped to a dynamic channel in the following way. The centerline of a dynamic channel consists of the Voronoi edges dual to the faces shared by tetrahedra associated with the nodes in G and new edges are added to ensure that the resulting centerline is continuous to meet the definition of a dynamic channel.

More precisely, each edge $e_1 = (N_1, N_2, s_1, w_1) \in G$ after which follows $e_2 = (N_2, N_3, s_2, w_2) \in G$ is mapped to a centerline as a Voronoi edge ve_1 dual to the face shared by $Ref(N_1)$ and $Ref(N_2)$ in snapshot s_1 . If both e_1 and e_2 originate from different snapshots, additional edges have to be mapped to the centerline. For each pair of successive i_1, i_2 in the ordered set of integer numbers from $I_{N_2} = \{s_1, \dots, s_2\}$, a new edge ge_i is added to the centerline which connects Voronoi vertex dual to $Ref(N_2)$ in snapshot i_1 and the Voronoi vertex dual to $Ref(N_2)$ in snapshot i_2 . The whole process is demonstrated in Fig. 4

6 RESULTS

The method was tested on trajectories which were computed using a computer simulation of a protein

molecule. In the visualization of these data, it can be clearly visible that a small molecule (a ligand) leaves the active site defining an egression path. In addition, the molecule does not remain open during the egression which means that recent approaches which compute channels separately in each snapshot fail to detect such a path.

For the analysed trajectories, the ligand was naturally excluded from the computation. The above mentioned constraints were used searching for egression paths; their settings were identical for all analysed trajectories. After constrained dynamic channels were found, these channels were compared against the egression path of a ligand. Table 1 shows the results of the comparison. We have inspected the bottleneck radius (in Angstrom, \AA , $1 \text{\AA} = 10^{-10} \text{ m}$) of the dynamic channel and the distance between dynamic channel and the ligand position in the appropriate snapshot. Average value of this distance is shown in the table. In the trajectories 1-3, the egression path was computed by our method which was almost identical with the egression path of the ligand. For trajectory 4, our method has detected wider possible egression path which exists, whereas the ligand in the simulation chose different path. In this case, the size of the ligand (Table 1) is smaller and more possible paths exist. In all cases, the bottleneck radius of computed dynamic channel was the same or slightly smaller than

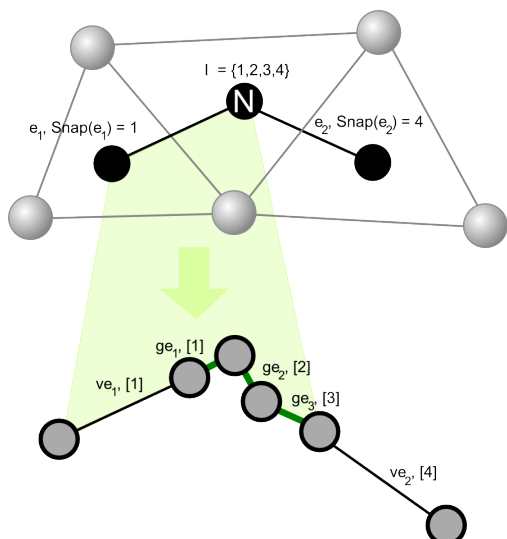


Figure 4: The example mapping of an edge e_1 in a path (upper part) to the part of a dynamic channel centerline (lower part). For the edge e_1 , the Voronoi edge ve_1 is added to the centerline together with edges ge_1 , ge_2 and ge_3 . The ge edges show the waiting of the robot and ensure that the resulting centerline is continuous.

the radius of a ligand bounding sphere. This was caused by the non-spherical shape of the ligand which can pass through a channel only when properly oriented. It can be noticed that for trajectory 4, the number of computed paths was significantly larger than for other trajectories. Since all these paths satisfied the used constraints, additional filtering is convenient to select the most important of them.

The example visualization of a dynamic channel in selected snapshots using pyMol visualization software [Del02] is shown in Fig. 5. In each snapshot, only a part of the dynamic channel which is valid in this particular snapshot is visualized. Standard visualization method of insertion of empty spheres on the centerline is used. Notice that the path is blocked by atoms in the second and fourth selected snapshot – this is exactly the case in which methods that compute channels in each snapshot independently fail to detect the channel.

It should be stressed that in the testing data the behaviour of the molecule over time is sampled densely enough so that the translation of each atom in the two consecutive snapshots is kept within a reasonable limit and no crucial movement is omitted. This ensures that it would be feasible for a small molecule to pass through a computed path without any collision.

The time required to find paths satisfying the constraints depends on the number of edges in G and on the constraints applied. However, in the worst case the modified BFS algorithm may process all paths which may require vast amount of time and memory. More

precisely, let c be the number of nodes in G and k be the number of snapshots. Then, the maximum number of edges between a pair of nodes is k . In the worst case, the number of all paths of length d is $\prod_{i=1}^d k(c-i)$. With the constraints and the cutting of paths applied after each step, the number of paths generated is significantly smaller. For protein trajectories, the time required to find paths with all mentioned constraints was hours on a common desktop computer (single core 2GHz, 2GB of memory). With the reduction of paths in each step to the number of nodes in G , the time even decreased rapidly to approx. 20 minutes still generating reasonable paths. It should be noted that the processing time is not crucial since the molecular dynamics simulation takes days to compute.

7 CONCLUSION

We have presented a new geometric method, which is capable of finding constrained paths in a three dimensional space with a huge number of moving spherical obstacles. These obstacles may not only move arbitrarily, but also may overlap. Various constraints were designed which can be applied to filter non-prospective paths. We have presented the modified breadth-first search algorithm which generates feasible paths with the constraints applied during the generation process.

We have also demonstrated the application of the method to an interesting biochemical problem of finding a dynamic channel in a trajectory. Due to limited computing resources currently available, trajectories do not cover large period of protein life and that egression may not happen in the short interval of simulation. Despite this, we expect the method to have a great potential once the data is available. We have also shown that for the available data with egression paths confirmed by other means, these collision-free paths were found by our method.

8 FUTURE WORK

In the future, we would like to focus on the validation of biochemical relevance of computed paths on large protein trajectories. Moreover, in cooperation with biochemists it would be necessary to include various additional biochemical constraints to improve the relevance of the results if still a large number of paths exists.

So far, the method assumes that the radii of obstacles do not vary greatly. The modification of the method for environments with variable-size spherical obstacles could be accomplished using the additively weighted Voronoi diagram and the corresponding triangulation.

9 ACKNOWLEDGMENTS

This work was supported by The Ministry of Education of The Czech Republic, Contract No. LC06008 and by The Grant Agency of The Czech Republic, Contract

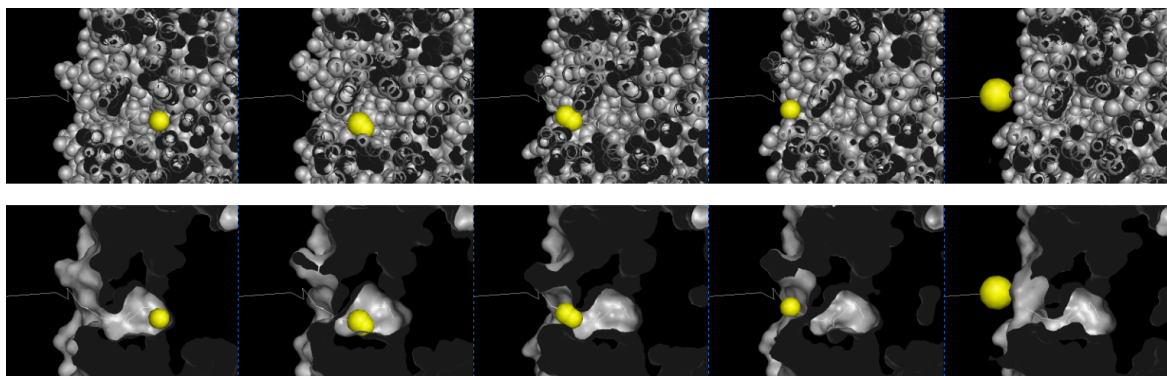


Figure 5: Example visualization. Side view of a protein molecule, cut with a cutting plane perpendicular to the view. The resulting egression path is visualized using yellow spheres defining the safe collision-free positions of a spherical robot inside the molecule over time (example for six selected snapshots). The molecule is visualized using two basic visualization methods (upper panel – sphere model, lower panel – surface model).

id	molecule (codename)	ligand egression	dynamic channel bottleneck radius	number of paths found	average distance between ligand egression and one of the paths computed
1	LinB WT	cyclohexanol (OCX)	2.7 Å	59	0.8 Å
2	LinB L177W	cyclohexanol (OCX)	1.4 Å	43	3.8 Å
3	LinB L177W	2-bromoethanol (BEO)	1.3 Å	1	2.01 Å
4	LinB L177W	bromide ion (Br ⁻)	1.4 Å	1588	7.7 Å

Table 1: The properties of computed constrained dynamic channels for selected trajectories.

No. P202/10/1435. We would also like to acknowledge Lada Biedermannova from Heidelberger Institut for Theoretical Studies for the supplied trajectories.

REFERENCES

- [BDH96] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, 1996.
- [BKZ⁺07] P. Broz, I. Kolingerova, P. Zitka, R. Apu, and M. Gavrilova. Path planning in dynamic environment using an adaptive mesh. *Spring Conference on Computer Graphics, ACM proceedings*, pages 172–178, 2007.
- [BMS09] P. Beneš, P. Medek, and J. Sochor. Computation of channels in protein dynamics. In *Proceedings of the IADIS International Conference Applied Computing 2009*, 2:251–258, 2009.
- [CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.
- [Del02] W. L. Delano. The pymol molecular graphics system, 2002.
- [FS88] K. Fujimura and H. Samet. Path planning among moving obstacles using spatial indexing. *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 1662–1667 vol.3, apr. 1988.
- [FS89] K. Fujimura and H. Samet. A hierarchical strategy for path planning among moving obstacles [mobile robot]. *Robotics and Automation, IEEE Transactions on*, 5(1):61–69, feb. 1989.
- [FS98] Paolo Fiorini and Zvi Shillert. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17:760–772, 1998.
- [Gho90] Pijush K. Ghosh. A solution of polygon containment, spatial planning, and other related problems using minkowski operations. *Comput. Vision Graph. Image Process.*, 49(1):1–35, 1990.
- [HA92] Y.K. Hwang and N. Ahuja. A potential field approach to path planning. *Robotics and Automation, IEEE Transactions on*, 8(1):23–32, feb. 1992.
- [HKcLR00] David Hsu, Robert Kindel, Jean claude Latombe, and Stephen Rock. Randomized kinodynamic motion planning with moving obstacles, 2000.
- [JLK95] J.A. Janet, R.C. Luo, and M.G. Kay. The essential visibility graph: an approach to global motion planning for autonomous mobile robots. volume 2, pages 1958–1963 vol.2, may. 1995.
- [Lin04] F. Lingelbach. Path planning using probabilistic cell decomposition. volume 1, pages 467–472 Vol.1, apr. 2004.
- [MBS07] P. Medek, P. Beneš, and J. Sochor. Computation of tunnels in protein molecules using delaunay triangulation. *Journal of WSCG*, 15(1-3):107–114, 2007.
- [PKK007] Martin Petřek, Pavlína Košinová, Jaroslav Koča, and Michal Otyepka. Mole: A voronoi diagram-based explorer of molecular channels, pores, and tunnels. *Structure*, 15(11):1357–1363, November 2007.
- [POB⁺06] Martin Petřek, Michal Otyepka, Pavel Banáš, Pavlína Košinová, Jaroslav Koča, and Jiří Damborský. Caver: a new tool to explore routes from protein clefts, pockets and cavities. *BMC Bioinformatics*, 7(1):316+, June 2006.
- [SU00] J.-R. Sack and J. Urrutia. *Handbook of computational geometry*. North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 2000.
- [YFW⁺08] Eitan Yaffe, Dan Fishelovitch, Haim J. Wolfson, Dan Halperin, and Ruth Nussinov. Molaxis: Efficient and accurate identification of channels in macromolecules. *Proteins: Structure, Function, and Bioinformatics*, 73(1):72–86, 2008.
- [YJSHJ06] Kwon Kyoung Youb, Cho Jeongmok, Kwon Sung-Ha, and Joh Joongseon. Collision avoidance of moving obstacles for underwater robots. *Journal of Systemics, Cybernetics and Informatics*, 4(5):86–91, 2006.