18th International Conference in Central Europe
on
Computer Graphics, Visualization and Computer Vision

in co-operation with

EUROGRAPHICS

WSCG 2010

Full Papers Proceedings

Edited by

Vaclav Skala, University of West Bohemia, Czech Republic

18th International Conference in Central Europe
on
Computer Graphics, Visualization and Computer Vision

in co-operation with

EUROGRAPHICS

WSCG 2010

Full Papers Proceedings

Edited by

Vaclav Skala, University of West Bohemia, Czech Republic

Vaclav Skala – Union Agency

WSCG 2010 – Full Papers Proceedings

Editor: 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 978-80-86943-88-6*

WSCG 2010

Program Committee members

Adzhiev,V. (U.K.)
Balcisoy,S. (Turkey)
Benes,B. (USA)
Bengtsson,E. (Sweden)
Biri,V. (France)
Bittner,J. (Czech Republic)
Bouatouch,K. (France)
Brodlie,K. (U.K.)
Buehler,K. (Austria)
Csebfalvi,B. (Hungary)
Daniel,M. (France)
Davis,L. (USA)
de Geus,K. (Brazil)
Debelov,V. (Russia)
Ferguson,S. (U.K.)
Flaquer,J. (Spain)
Gavrilova,M. (Canada)
Gudukbay,U. (Turkey)
Gutierrez,D. (Spain)
Havran,V. (Czech Republic)
Chover,M. (Spain)
Jansen,F. (Netherlands)
Kruijff,E. (Austria)
Lee,B. (Korea)
Lee,T. (Taiwan)
Magnor,M. (Germany)
Mollá Vayá,R. (Spain)
Muller,H. (Germany)
Murtagh,F. (Ireland)
Pedrini,H. (Brazil)
Platis,N. (Greece)
Puppo,E. (Italy)
Purgathofer,W. (Austria)
Rojas-Sola,J. (Spain)
Rosenhahn,B. (Germany)
Rudomin,I. (Mexico)
Sakas,G. (Germany)
Sbert,M. (Spain)
Segura,R. (Spain)
Schumann,H. (Germany)
Sochor,J. (Czech Republic)
Stroud,I. (Switzerland)
Teschner,M. (Germany)
Theoharis,T. (Greece)
Tokuta,A. (USA)
Vergeest,J. (Netherlands)
Weiss,G. (Germany)
Zach,C. (Switzerland)
Zara,J. (Czech Republic)
Zemcik,P. (Czech Republic)
Zitova,B. (Czech Republic)

WSCG 2010

Board of Reviewers

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

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

Full papers

Modeling with Subdivision Surfaces

T. Ullrich, A. Schiefer, D. W. Fellner
Institut für ComputerGraphik und WissensVisualisierung, TU Graz, Austria
Fraunhofer Austria Research, Visual Computing Devision, Graz, Austria
Fraunhofer Institute for Computer Research and Technical University of Darmstadt, Germany

ABSTRACT

Subdivision surfaces are an established modeling tool in computer graphics and computer-aided design. While the theoretical foundations of subdivision surfaces are well studied, the correlation between a control mesh and its subdivided limit surface still has some open-ended questions: Which topology should a control mesh have? Where should control vertices be placed? A modeler – human or software – is confronted with these questions and has to answer them.

In this paper we analyze four characteristic situations. Each one consists of an analytical reference surface S and several variants of control meshes C_i. In order to concentrate on the topology of the control meshes, the geometrical positions of their control vertices have been determined and optimized automatically. As a result we identified the best topology of all C_i to represent the given surface S. Based on these results we derived heuristics to model with subdivision surfaces. These heuristics are beneficial for all modelers.

Keywords: Computer Graphics, Computer-Aided Design, Subdivision Surfaces, Modeling Techniques

1 INTRODUCTION

Subdivision surfaces have gained a lot of attention within the last decades. Especially artists and designers prefer subdivision surfaces to other free-form surface representations. The way how to model with subdivision surfaces is described in various tutorials and courses [4]. These tutorials pursue a plan which can be summarized by the sequence

idea → subdivision surface model → real 3d object,

whereas the last production step is omitted, if only the virtual object is of interest. Within the last few years subdivision surfaces are also used in reverse engineering at a progressive rate. Reverse engineering is the inverse situation of the sequence above. A real 3d object is the starting point and its corresponding subdivision surface model is the desired result. During the re-modeling phase the modeler – either human or algorithm – has to tackle two problems: geometry and topology. These are the two ingredients of a subdivision surface. While many articles are about geometric optimization, we concentrate on topological questions. Choosing the right topology to represent a given object by a subdivision surface reduces the number of needed control vertices significantly. The choice of a good topology is a distinction between an experienced, skilled modeler and a beginner. In this article we analyze some characteristic objects and test different, topological variants of control meshes. A geometrical optimization allows us to determine the best vertex positions automatically and to concentrate on topological aspects. Based on these optimizations we determine the appropriateness of each topological variant to represent a certain object. Consequently, we derive heuristics to model with subdivision surfaces.

WSCG'2010, February 1 – February 4, 2010
Plzen, Czech Republic.

2 RELATED WORK

Modeling with subdivision surfaces is the main topic of many tutorials and courses [16], [15]. A good overview on subdivision surfaces has been presented by WEIYIN MA [7] whereas details on the mathematical background can be found in the book *Subdivision Surfaces* by JÖRG PETERS and ULRICH REIF [11]. Furthermore a variety of articles is about topological robustness [1] or handling "special" situations such as filling n-sided regions [10].

In this article we concentrate on the more general situation from the reverse engineering point of view. From this perspective a subdivision surface reconstruction method has to tackle two problems, a topological problem (the layout of the control mesh) and a geometrical problem (the position of the control vertices in 3D). Unfortunately, in most articles about subdivision surface fitting [3], [2] the topological problem plays a minor role [8]. In this article we address this topological problem; i.e. we start with a topological setting; then we optimize the geometry of the control mesh by calculating the best vertex positions to approximate a given surface. The optimization results establish a relationship between topological layout and approximation quality, which allows to identify the best topology. In order to concentrate on the topological aspects the geometry is optimized automatically. The optimization uses a distance based error function [14], [12] which is minimized by a standard approach of numerical minimization [9], [5]. The minimization process is straight forward and operates on the vector of all vertex coordinates of all vertices to position. As the geometrical optimization is not within the scope of this article we

refer to RAINER STORN and KENNETH PRICE [13] for the algorithmic details of the minimization routine (see http://www.icsi.berkeley.edu/~storn/code.html for implementations).

3 MODELING

The case studies in the following section cover the four most important situations when modeling with Catmull-Clark subdivision surfaces:

Modeling Edges The first case examines smooth edges. While sharp edges can be modeled easily with special feature rules for subdivision surfaces, smooth edges offer at least two possibilities to be designed: with a row of control vertices along the edge or with two rows of control vertices alongside the edge.

Modeling Non-Quadrilateral Configurations Almost each type of subdivision surface has a topology for which it suits best. Loop subdivision prefers triangles, Catmull-Clark subdivision generates quads, etc. Unfortunately, not every object has a favorable geometric primitive to be modeled with. Several strategies for such a configuration are possible.

Modeling Curvature The third case inspects a surface with different curvatures: hyperbolic, parabolic and elliptic. It addresses the issue of quad orientation with respect to the surface's principle curvature directions.

Modeling Inflection Points The last case analyzes a surface which is defined by a cut curve. This is a standard situation in CAD modeling.

Each case examines different variants of control meshes that are adjusted by a number of parameters. These parameters are set by a numerical optimization routine based on *differential evolution* [13]. It minimizes the distance [14] between the nominal surface and the actual subdivision surface.

3.1 Smooth Edges

The first nominal surface is defined by the implicit equation

$$S_1 : \; x^6 + y^6 + z^6 = 1. \qquad (1)$$

The resulting surface looks like a cube with round edges and corners (see Figure 1). In order to analyze how to model beveled edges two variants are inspected.

Variant 1 The first variant of possible control meshes has a cube-like topological layout. It consists of 3×3 quads on each side. Due to symmetries the geometry is defined by only six parameters of three initial control points. Three parameters are used for the three coordinates (x_1, y_1, z_1) of the first control point. The fourth and the fifth parameter define the second control point.

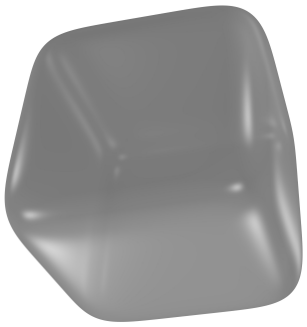


Figure 1: The implicit surface $x^6 + y^6 + z^6 = 1$ has a degree which cannot be reached by a cubic patch. Therefore, Catmull-Clark subdivision can only approximate it. The best way to approximate it is analyzed by testing different topologies.

In order to avoid self-intersections as well as optimizations with constraints, these parameters are defined as positive offsets to the first control point. Consequently, they have a fixed domain and the second control point is calculated via $(x_1 + x_2, y_2, x_1 + x_2)$. As the second control point defines an edge of the control mesh cube, and as the faces of the cube are connected at/over this edge, the x and z components have to be equal. Finally the last parameter x_3 again defines an offset from x_1. Due to symmetries this is the only value needed for the third control point: $(x_1 + x_3, x_1 + x_3, x_1 + x_3)$. As the third control point is at the corner of the control mesh cube, all coordinate components must be equal. Figure 2 shows the initial three control points that are generated by the six parameters.

The optimization routine minimizes the distance between the generated surface and the reference surface (Equation 1) based on uniformly-distributed samplings. The error function $f_{1,1}$ is a sum of point-to-subdivision surface distances. The optimum is

$$f_{1,1}(\; 0.33633, \; 0.33526, \; 0.99830, \; 0.66616, \; 0.32615, \; 0.66319 \;) = 7.45141.$$

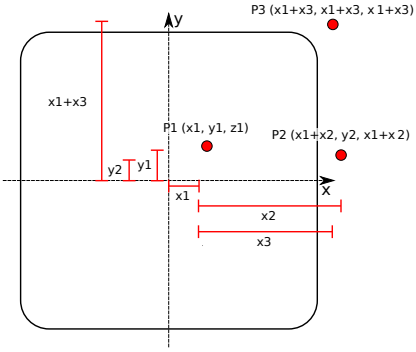


Figure 2: Due to symmetries six parameters are enough to define three different control points. The complete subdivision control mesh is composed of rotations and mirrorings of these points. Each of the six cube sides consists of 3×3 faces.

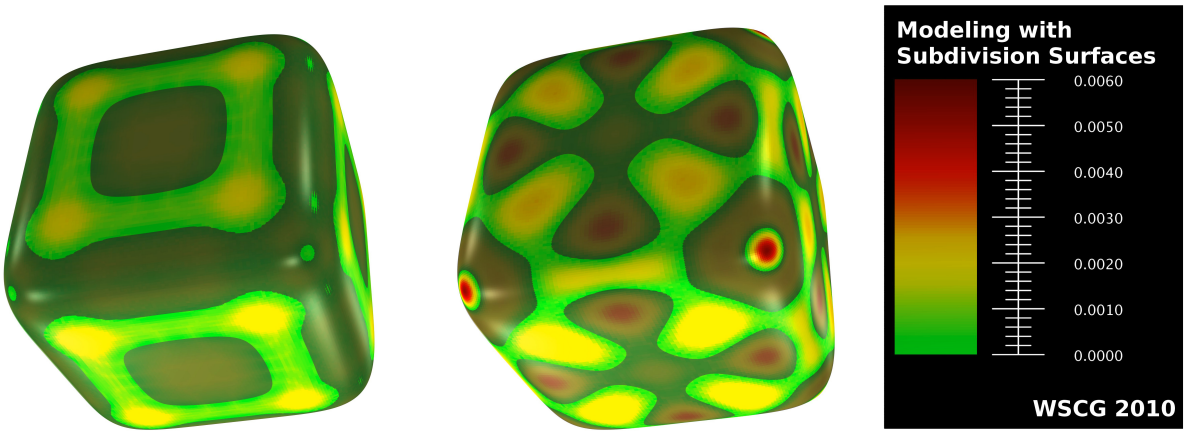


Figure 3: In comparison to each other the first variant performs better than the second one. Explicit modeling of beveled edges increases the risk of "over-modeling" – a high frequency fluctuation which is spread out from the beveled edge and which disturbs the low frequency parts of the surface.

The three control points, which generate the complete subdivision control mesh by rotations and mirrorings, have the coordinates

$$(0.33633, 0.33526, 0.99830),$$
$$(1.00249, 0.32615, 1.00249),$$
$$(0.99952, 0.99952, 0.99952).$$

Variant 2 Also the second variant is a cube-like control mesh derived from three initial control points but with five parameters. This time every side of the cube consists of four quads, the six sides of the cube are connected through beveled faces at the edges and with triangles at the corners. Figure 4 illustrates this control mesh.

The first parameter, z_1, is the z component of the first control point, which is located at $(0,0,z_1)$. Due to symmetries the first control point is centered on each side of the control mesh cube. The second and third parameters, x_2 and z_2, define the second control point at $(x_2,0,x_2+z_2)$. This time the sides of the cube are connected via beveled faces, therefore the offset z_2 is added to the z component of the second control point. Using the last two parameters, x_3 and z_3, the third control point is (x_3,x_3,x_3+z_3). The error of the best control mesh of this variant – according to the optimization routine – is 7.79202:

$$f_{1,2}(\ 0.99686,\ 0.83273,\ 0.14743,\ 0.74912,\ 0.33157) = 7.79202.$$

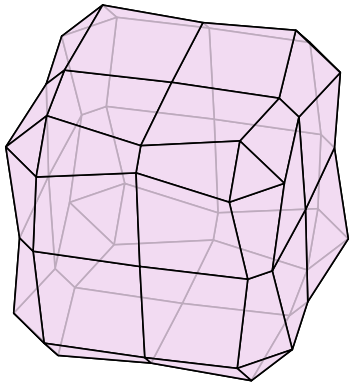


Figure 4: This variant uses a control mesh with explicitly beveled edges to approximate an implicit surface of degree 6.

Comparison Figure 3 illustrates both variants. Each variant is rendered with a color scheme indicating its distance to the nominal surface which is included in each rendering. It is visualized with a transparent, grayish style. The illustration shows that the first variant performs better than the second one. The second variant uses beveled edges to model the reference surface. This topology is not suitable to approximate the given surface. Even its geometrically optimized version shows "over-modeling" effects – a high frequency fluctuation which is spread out from the over-modeled parts. In this case the beveled edges disturb the low frequency parts of the surface. An in-depth analysis of the distances confirms the visualization. The average and the maximum of all distances between the nominal surface and points on the actual surface are:

	avg. distance	max. distance
Variant #1	0.00079	0.00204
Variant #2	0.00130	0.00519

3.2 Modeling Non-Quadrilateral Configurations

The second surface analysis investigates non-quadrilateral configurations. The reference surface is a so-called *monkey saddle*. It is a height field defined by

$$S_2(x,y) = \frac{x^3 - 3xy^2}{2}. \qquad (2)$$

This surface is axially symmetric at a degree of 120°; i.e. after a $\frac{2}{3}\pi$ rotation the surface is congruent to itself. The point at the origin is a parabolic, umbilic point. Within this analysis the parameters x and y may range from -1.0 to 1.0.

As the following variants have different numbers of control vertices, the area of every control mesh is chosen to be proportional to the number of its vertices. So control meshes with more vertices have to approximate a larger area of the reference surface.

Variant 1 The first variant has 19 vertices and seven parameters, which define their heights (z component). The symmetric configuration is illustrated in Figure 5 which shows the topology of the control mesh and the parameters (height) of each vertex. All vertices with $x = 0$ have a fixed height of 0.0. The best optimized

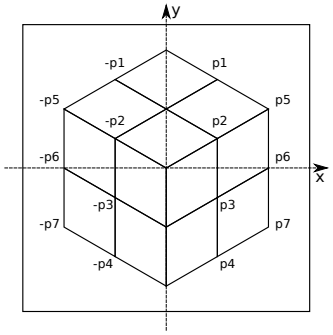


Figure 5: The topology of this control mesh is arranged in a 120° symmetric layout to adapt the subdivision surface to the reference.

version of this variant has an error of 0.44914. Its parameters are

$$f_{2,1}(\ \begin{matrix} -0.27195, & -0.00007, & 0.00001, \\ -0.27193, & -0.00007, & 0.27181, \\ -0.0001) & = 0.44914. & \end{matrix}$$

Variant 2 For the second variant a predefined mesh with 27 vertices is used as control mesh. The heights of its vertices are controlled by twelve parameters. Figure 6 shows the parameters and the topology which is (in the inner part) dual to the first variant. Again, all vertices with $x = 0$ have a fixed height of 0.0.

The smallest possible error of this topological configuration is

$$f_{2,2}(\ \begin{matrix} 0.00148, & -0.07632, & -0.25658 \\ -0.37829, & -0.07681, & 0.07507, \\ 0.00052, & -0.37787, & -0.25613, \\ 0.25710, & 0.37745, & -0,00072) \\ = 0.69362. & & \end{matrix}$$

Variant 3 The third variant for this surface does not adapt to the reference's axial symmetry of $\frac{2}{3}\pi$ in any way. It uses a regular quadratic grid centered on the reference surface. There are five vertices along each side of the grid, so 25 vertices in total. Fixing the vertices at $x = 0$ six parameters are enough to describe all vertices. Figure 7 illustrates this topology. The optimization returns the optimum of this topology with an error of $f_{2,3} = 0.80376$.

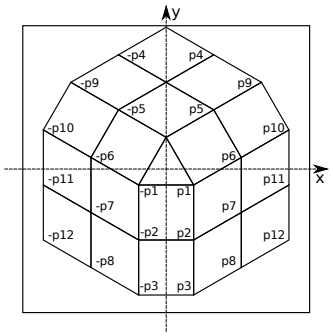


Figure 6: The topology of the second variant has a configuration which is dual to the first one (except for the border).

Comparison Taking the number of control vertices into account, each surface has been normalized; i.e. the area of every control mesh is proportional to the number of its vertices. Therefore, all subsequent error values are normalized and comparable to each other. In this situation the case study does not reveal a best topology but a worst one. The regular gird does not approximate the given surface well. The two variants whose topologies reflect the nominal surface's symmetry perform much better (see Figure 8):

	avg. distance	max. distance
Variant 1	0.00241	0.00624
Variant 2	0.00153	0.01037
Variant 3	0.00330	0.01555

3.3 Modeling Curvature

The third modeling study inspects a torus due to its characteristic curvatures: hyperbolic, parabolic, and elliptic [6]. The reference is described by the formula:

$$S_3(u,v) = \begin{pmatrix} (a + b\cos(2\pi v)) \cdot \cos(2\pi u) \\ (a + b\cos(2\pi v)) \cdot \sin(2\pi u) \\ b\sin(2\pi v) \end{pmatrix} \quad (3)$$

with major radius $a = 5.0$ and minor radius $b = 3.0$. The parameters u and v are within the range 0.0 to 1.0.

Variant 1 For the first variant the parameter domain is sampled at a fixed, regular grid to get the control point positions. Two parameters, that can be set by

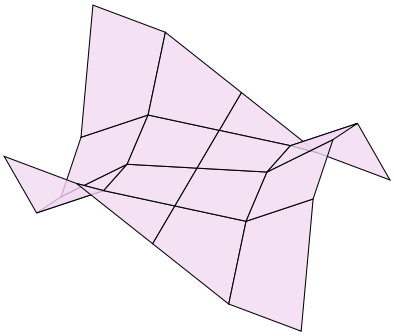


Figure 7: A rectangular topology leads to a very high error. Topologies that reflect the nominal surface's symmetry perform much better.

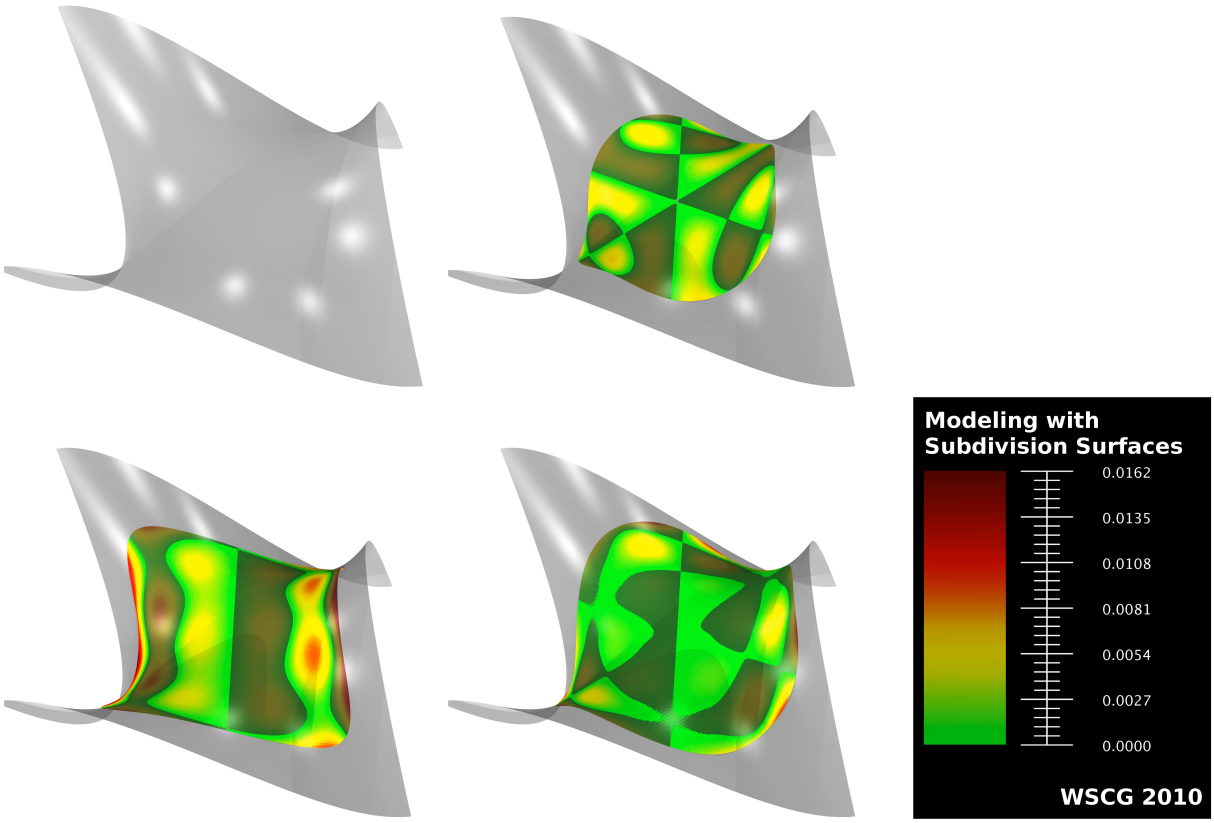


Figure 8: The nominal surface (upper left) is invariant to 120° rotations. The topology of variant #1 (upper right) and #2 (lower right) reflect this property whereas variant #3 (lower left) uses a simple rectangular grid. In this case topologies, which reflect the main symmetries of the reference surface, perform better.

the optimization, define the major and the minor radius of the control mesh torus. This is probably the most commonly used parameterization of a torus in modeling software.

The error of this variant after the optimization is 824.82012. The two parameters defining the major and minor radius have their optimum at 5.42375 respectively 3.38278. This variant is compared to slanted torus control mesh.

Variant 2 A slanted torus uses a slightly different parameterization than the first one. Again, the values u and v are a regular grid in the parameter domain, but this time an offset depending on u is added to v each time. For this variant, the offset is defined as $\frac{2}{3}u$. This introduces a slant in the control mesh.

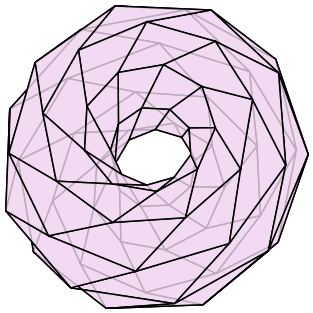


Figure 9: A slanted torus control mesh is a "non-standard" way to model a torus. This variant is compared to the most commonly used parameterization with a rectangular grid layout of the parameter domain.

The optimization routine calculates a minimum error of 1474.60738 for this variant. The parameters defining the resulting control mesh are $a = 5.62499$ and $b = 3.43444$. Figure 9 shows the slanted control mesh created with these parameters.

Variant 3 This variant of the torus control mesh is basically the same as variant 2, except that the offset defining the slant is defined as $\frac{4}{3}u$. Even the optimized geometry ($a = 6.24999, b = 3.56744$) leads to a large error $f_{3,3} = 4290.40374$.

Variant 4 As it is difficult to determine appropriate slant offsets, the fourth variant also optimizes this parameter; i.e. in addition to the two parameters defining the radii of the torus control mesh, it has a third parameter defining the slant offset. The optimization process returns

$$f_{3,4}(5.42399, 3.38277, 0.00000) = 826.65427.$$

Comparison The easiest way to model a torus seems to be the best way. All variations in topology increase the approximation error (see Figure 10). The in-depth analysis of distances confirms this heuristic:

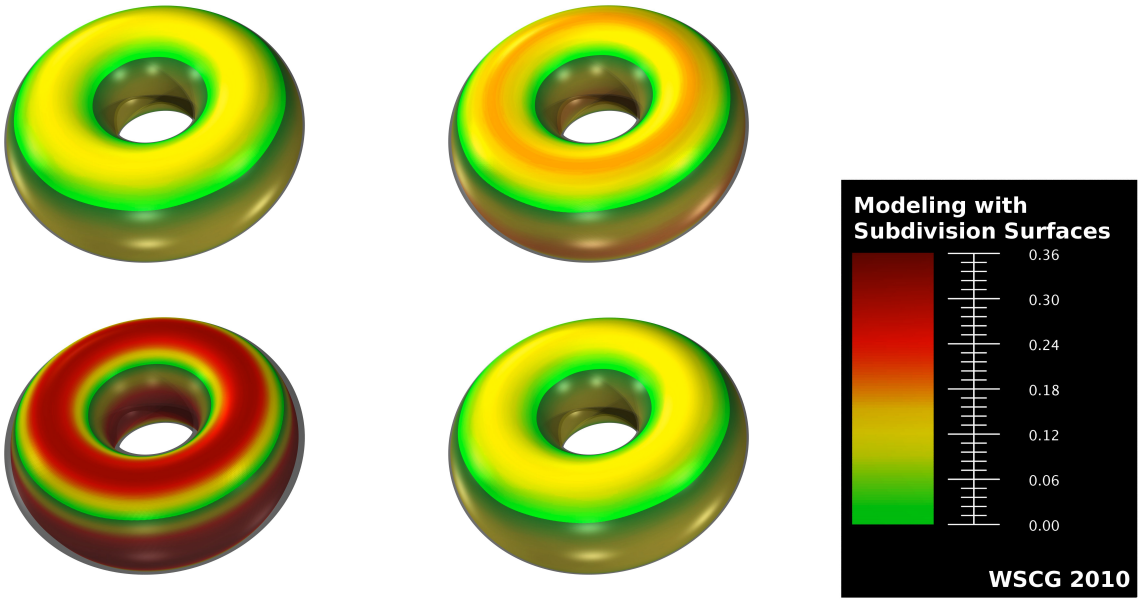


Figure 10: A torus can be modeled with a slant offset and without one (upper left). With increasing slant offset (upper right, lower left) the approximation error increases. If the optimization routine is allowed to set the slant offset by itself, it is set to zero (lower right).

	avg. distance	max. distance
Variant 1	0.07806	0.12925
Variant 2	0.11145	0.19450
Variant 3	0.19887	0.35382
Variant 4	0.07806	0.12904

3.4 Modeling Inflection Points

The last case analyzes a surface which is defined by a cut curve. Blended with a straight line the result has the formula:

$$S_4(x,y) = \frac{y}{10} \cdot \sin\left(\frac{2\pi}{3}\arctan x\right) \qquad (4)$$

The parameter x describes the cut curve in the range from -5.0 to 5.0 whereas the parameter y blends between the straight line and the curve from 0.0 to 10.0. The surface is plotted in Figure 11.

Variant 1 The control mesh of the first variant gets 20 parameters – 10 pairs of x position and height value z. The range of the x values (-5.0 to 5.0) has been partitioned into 10 equally-sized intervals. In each interval one and only one parameter pair is allowed. For each pair $p_i = (x_i, z_i)$ two control points are generated: one at $(x_i, 10, z_i)$ to approximate the cut curve and one at $(x_1, 0, 0)$ on the straight line. The error of the optimized variant is 6.73931:

$$\begin{aligned} f_{4,1}(\quad & -4.35546, \quad -0.30532, \\ & -3.05017, \quad -0.48656, \\ & -2.93309, \quad -0.50592, \\ & -1.58539, \quad -0.82026, \\ & -0.37453, \quad -1.29349, \\ & 0.38362, \quad 1.32794, \\ & 1.71154, \quad 0.78575, \\ & 2.30383, \quad 0.65748, \\ & 3.00657, \quad 0.48136, \\ & 4.02755, \quad 0.35095 \quad) = 6.73931. \end{aligned}$$

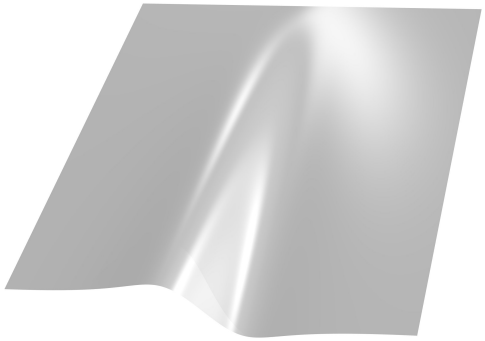


Figure 11: This surface is defined by a cut curve which is blended with a straight line. This is a standard situation in CAD modeling.

Variant 2 This variant uses a regular 10×3 grid covering the whole reference surface. All heights are specified as a parameter and the (x,y)-positions are fixed. Consequently, this variant operates on 30 parameters. The optimization calculates an error of 14.70749 for the best geometry. The best parameters for the control mesh are:

$$\begin{array}{lll} -0.00257, & -0.13120, & -0.27108, \\ 0.00131, & -0.17258, & -0.33910, \\ -0,00099, & -0.28033, & -0.55752, \\ 0,00048, & -0.36468, & -0.74336, \\ -0,00057, & -0.72072, & -1.40922, \\ -0,00022, & 0.72074, & 1.41115, \\ 0.00191, & 0.36363, & 0.74169, \\ -0.00244, & 0.28386, & 0.55652 \\ 0.00267, & 0.16798, & 0.34212, \\ -0.00105, & 0.13555, & 0.26656. \end{array}$$

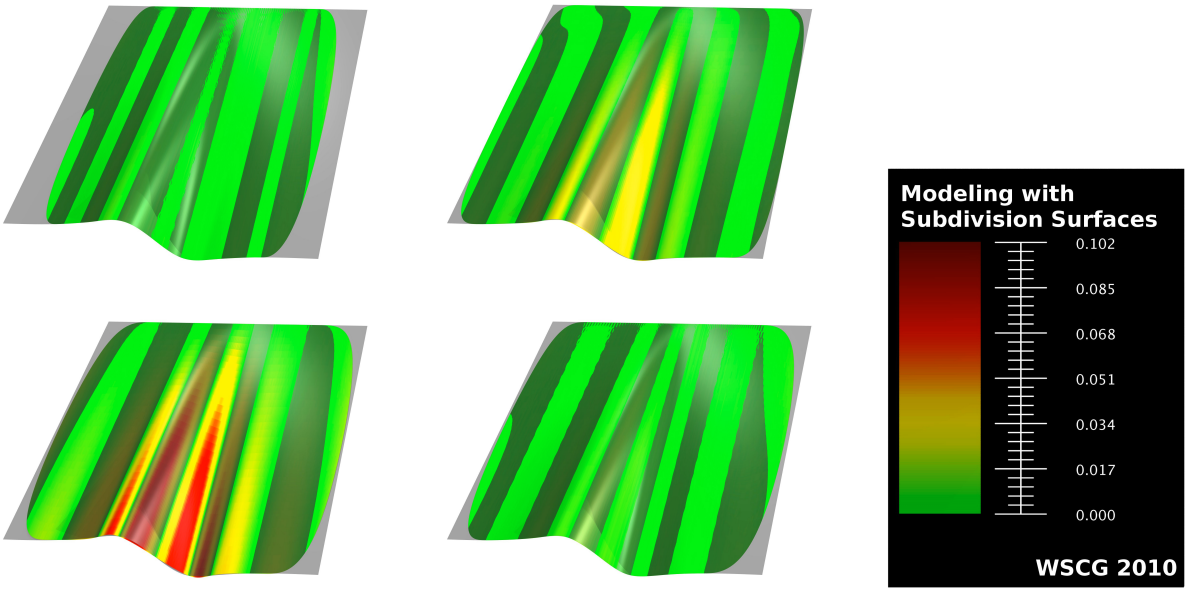


Figure 12: In this comparison approximations with fixed x positions are more erroneous than approximations with x positions as free parameters. The interval spacing (upper left) of the first variant and the prominent-values-of-the-cut-curve variant with additional offsets (lower right) are the best results, whereas the regular grid (upper right) and the fixed prominent-values-of-the-cut-curve (lower left) are the worst results.

Variant 3 The third variant has fixed x positions at prominent values of the cut curve: -5.00000 (end of range), -3.31764, -1.63528 (inflection point), -0.93159 (minimum), 0.00000 (inflection point, root), 0.93159 (maximum), 1.63528 (inflection point), 3.31764, 5.00000 (end of range). For each x position there is again one control point at $y = 0$ with height 0 and another one at $y = 10$ with the height value which has to be optimized. So this variant has nine height parameters, one for each x position. The result is

$$\begin{aligned} f_{4,3}(\quad & -0.22288, \quad -0.47240, \quad -0.67730, \\ & -1.37479, \quad -0.00353, \quad 1.37625, \\ & 0.67573, \quad 0.47305, \quad 0.22281) \\ & = 11.91346. \end{aligned}$$

Variant 4 The fourth variant is very similar to the third variant. It uses the same configuration but in contrast to fixed x positions, the optimization routine is allowed to modify these positions within an offset of $\pm\frac{1}{3}$. Therefore, this variant takes 18 parameters, nine for the height values at each x position and nine offsets for the initial x positions.

For this variant, the minimum error after the optimization is 6.17020. The parameters for this control mesh are

$$\begin{aligned} f_{4,4}(\quad & -0.25058, \quad -0.46790, \quad -0.74541, \\ & -1.18889, \quad -0.38835, \quad 1.29557, \\ & 0.76445, \quad 0.36971, \quad 0.26533, \\ \\ & 0.04882, \quad 0.33053, \quad -0.25836, \\ & 0.28352, \quad -0.17465, \quad -0.33332, \\ & -0.01570, \quad 0.32223, \quad -0.03218) \\ & = 6.17020. \end{aligned}$$

Comparison The in-depth distance analysis reveals the following values:

	avg. distance	max. distance
Variant 1	0.00264	0.01044
Variant 2	0.00718	0.04152
Variant 3	0.01826	0.09219
Variant 4	0.00323	0.01460

If the area of the approximated surface is taken into account, the fourth variant is even slightly better then the first one. The approximation with the highest error level is the variant which has fixed x positions at prominent values of the cut curve (see Figure 12 (lower left)). Consequently, the best result and the worst result can be created with the same topology and minor differences in geometry. The additional offsets which distinguish a good from a bad result are plotted in Figure 13. The figure shows some very important properties when modeling with subdivision surfaces.

- At the extreme values (minimum and maximum at ± 0.93159) the control points have been moved by the optimization routine towards the direction of higher absolute gradients.
- Having moved the control points, the control polygon has fewer intersections with the nominal curve than the fixed-x-positions version.
- All versions with a small error (also the first variant with interval spacing) intersect the reference curve very close to (at $x = -1.63528$ and at $x = 0.0$) or nearby (at $x = 1.63528$) inflection points.

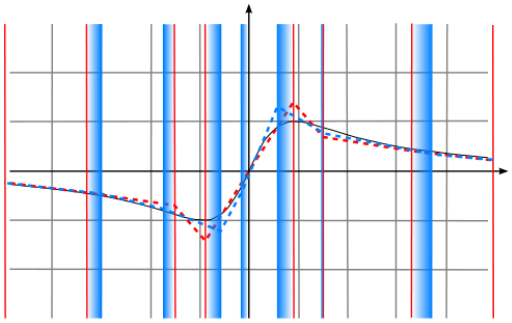


Figure 13: The cut curve which defines the fourth nominal surface is plotted in this diagram. Furthermore it shows the positions of the control vertices of the worst approximation (red) and the best approximation (blue). Their differences are visualized by gradients.

4 CONCLUSION

Based on the four case studies we arrive at several conclusions. The first case examines smooth edges and demonstrates the "over-modeling" effect. Modeling a local surface feature always takes the risk to disturb large parts of a model by spreading high-frequency fluctuations from the "over-modeled" parts. This effect can be avoided by a better topology, which does not model rounded edges explicitly, or by barrier lines – two or three consecutive lines of control vertices of low frequency, which suppress fluctuations due to the locally limited influence of a vertex to a subdivision surface.

While simple geometric properties – such as beveled edges – should not be considered in the topological layout of a control mesh, high-level geometric aspects play an important role. The second case study shows that global symmetries should be reflected in the control mesh. All control meshes, which did not reflect the global symmetry of the surface to approximate, caused higher errors than those with symmetrical layout. The third study approves this fact.

The last case analyzes a surface on the geometrical – not topological – level to explore the best positions for control vertices. Besides the conclusions already presented in the previous section the last study demonstrates the difficulties in predicting a good subdivision approximation without iterative optimization. All solutions with partly-fixed control vertices have a high error value. All heuristics derived from the last case reduce the error but do not reach the quality of the automatic optimization. As the optimization is time consuming (up to several hours for complex configurations) this is a problem for interactive modeling tools.

5 FUTURE WORK

Any surface can be approximated with a sufficient number of control points. Reducing this number keeps a subdivision surface model manageable. In the future we will concentrate on topological improvements and address the question whether it is possible to get accurate conclusions based on these case studies.

ACKNOWLEDGEMENTS

We gratefully acknowledge the generous support from the European Commission for the research project 3DCOFORM (3D COllection FORMation, 3D-coform.eu) under grant number FP7 ICT 231809.

REFERENCES

[1] Jianer Chen and Ergun Akleman. Topologically Robust Mesh Modeling: Concepts, Data Structures and Operations. *Technical Report*, 1:1–18, 2003.

[2] Kin-Shing Cheng, Wenping Wang, Hong Qin, Kwan-Yee K. Wong, Huaiping Yang, and Yang Liu. Design and Analysis of Optimization Methods for Subdivision Surface Fitting. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):878–890, 2007.

[3] Kin-Shing D. Cheng, Wenping Wang, Hong Qin, Kwan-Yee Kenneth Wong, Huaiping Yang, and Yang Liu. Fitting Subdivision Surfaces to Unorganized Point Data using SDM. *Proceedings of 12th Pacific Conference on Computer Graphics and Applications*, 1:16–24, 2004.

[4] Tony DeRose, Michael Kass, and Tien Truong. Subdivision Surfaces in Character Animation. *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 25:85 – 94, 1998.

[5] Nick Gould, Dominique Orban, and Philippe Toint. Numerical methods for large-scale nonlinear optimization. *Acta Numerica*, 14:299–361, 2005.

[6] Mark L. Irons. The Curvature and Geodesics of the Torus. *Technical Report, Raindrop Laboratories*, 20:1–19, 2005.

[7] Weiyin Ma. Subdivision surfaces for CAD - an overview. *Computer-Aided Design*, 37(7):693–709, 2005.

[8] Martin Marinov and Leif Kobbelt. Optimization methods for scattered data approximation with subdivision surfaces. *Graphical Models*, 67(5):452 – 473, 2005.

[9] J. C. Nash. *Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation*. Adam Hilger, second edition edition, 1990.

[10] Ahmad H. Nasri, Malcom Sabin, and Yasseen. Filling N-Sided Regions by Quad Meshes for Subdivision Surfaces. *Computer Graphics Forum*, 28:1644–1658, 2009.

[11] Jörg Peters and Ulrich Reif. *Subdivision Surfaces*. Springer, 2008.

[12] Jorg Peters and Xiaobin Wu. The distance of a subdivision surface to its control polyhedron. *Journal of Approximation Theory*, to appear:to appear, 2009.

[13] Rainer Storn and Kenneth Price. Differential Evolution: A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[14] Torsten Ullrich, Volker Settgast, Ulrich Krispel, Christoph Fünfzig, and Dieter W. Fellner. Distance Calculation between a Point and a Subdivision Surface. *Proceedings of 2007 Vision, Modeling and Visualization (VMV)*, 1:161–169, 2007.

[15] Denis Zorin, Peter Schröder, Tony DeRose, Leif Kobbelt, Adi Levin, and Wim Sweldens. Subdivision for Modeling and Animation. *SIGGRAPH 2000 Course Notes*, 1:1–116, 2000.

[16] Denis Zorin, Peter Schröder, and Wim Sweldens. Interpolating Subdivision for Meshes with Arbitrary Topology. *Proceedings of 1996 ACM Siggraph*, 23:189–192, 1996.

Selection of an Optimal Set of Discriminative and Robust Local Features with Application to Traffic Sign Recognition

Benjamin Höferlin
Universität Stuttgart, Germany
benjamin.hoeferlin@vis.uni-stuttgart.de

Gunther Heidemann
Universität Stuttgart, Germany
ais@vis.uni-stuttgart.de

ABSTRACT

Today, discriminative local features are widely used in different fields of computer vision. Due to their strengths, discriminative local features were recently applied to the problem of traffic sign recognition (*TSR*). First of all, we discuss how discriminative local features are applied to TSR and which problems arise in this specific domain. Since TSR has to cope with highly structured and symmetrical objects, which are often captured at low resolution, only a small number of features can be matched correctly. To alleviate these issues, we provide an approach for the selection of discriminative and robust features to increase the matching performance by speed, recall, and precision. Contrary to recent techniques that solely rely on density estimation in feature space to select highly discriminative features, we additionally address the question of features' retrievability and positional stability under scale changes as well as their reliability to viewpoint variations. Finally, we combine the proposed methods to obtain a small set of robust features that have excellent matching properties.

Keywords: Discriminative Local Features, Traffic Sign Recognition, SIFT.

1 INTRODUCTION

Over the past years, local features have become the preferred method in different fields of computer vision. Today they are applied in many tasks like panoramic imaging [BL07] or object recognition [Low04]. In this work we especially focus on scale-invariant and discriminative local features like the popular *SIFT* (scale-invariant feature transform) [Low04] or *SURF* (Speeded up robust features) [BTG06]. Such local features are commonly calculated in two stages. First, an interest point detector is used to find salient image regions at their characteristic scale. Then, a robust descriptor is extracted for each region.

Their ability to represent an image by the means of local patch descriptors, might be the reason for their success. Using local features, two images can be compared very fast, since only some salient regions are considered instead of the whole images. Besides this, local features like SIFT are more robust to various image transformations than global methods are. On the one hand, this is due to the extraction of image patches around salient interest points using a scale-invariant detector. On the other hand, the feature description is often covariant to a variety of image changes, too, including rotation or lighting changes. In contrast to global methods, local features can inherently cope with partial occlusions in the image. As result to their depicted strengths, discriminative local features have found their way into the field of traffic sign recognition (*TSR*), too. In literature, TSR is often split into the problems of traffic sign detection (TSD) and traffic sign classification (TSC). These applications are highly relevant to many recent advanced driver assist systems (ADAS), since the information annotated to the streets is primarily visually encoded in traffic signs. Due to the fact, that drivers tend to trust in the information provided by ADAS [BT05], TSR has to be reliable. This demand grows with the emergence of active ADAS-technologies like brake-by-wire. The demand for reliability leads to the application of local features for their illustrated vantages. But local feature approaches also involve weaknesses, like slow matching performance on large databases or mismatches due to features with poor saliency. In this work, we address these shortcomings and propose a method able to cope with them.

In detail, our contribution is the introduction of a novel method for the selection of a small but highly discriminative set of local features for TSR. We do not solely consider their discriminative power regarding a single image, but also their saliency regarding the traffic sign domain. In addition to their high saliency, the selected features have to be positionally stable and robust under large scale changes. Also the features' stability under viewpoint changes is evaluated in order to reduce the influence of features violating the local planarity assumption. Finally, a greedy algorithm is introduced to select a small and optimal set of local features according to their properties. Although, the feature set can be of arbitrary size, we go for a small amount to achieve fast feature matching.

The remainder of this paper is structured as follows: First, we briefly introduce how local features are applied to TSR and which specific problems arise within the traffic sign domain. In section 3, we then discuss research related to our work. As our main contribution, we propose in section 4 a new selection scheme that is able to gather a small and robust set of local features. Based on our approach the mentioned shortcomings of local features are alleviated, which we finally prove in section 5 by empirical results on two different datasets.

2 DISCRIMINATIVE LOCAL FEATURES FOR TRAFFIC SIGN RECOGNITION

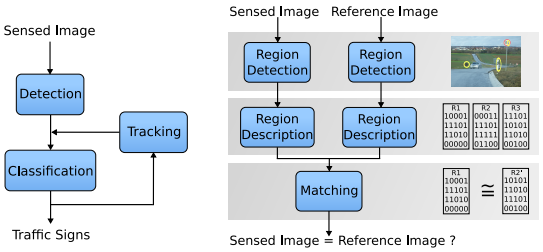
Traditionally, the task of TSR is divided into 3 stages: i) TSD, the detection of traffic sign candidates, ii) TSC, the classification of the candidate to the proper traffic sign class (i.e. the type of the road sign, e.g. a specific speed limit), and iii) the tracking of the candidate within the video sequence (cf. Fig. 1(a)).

Local features can be applied to all three stages of TSR. Since our considerations hold for every stage, we do not focus on a single one in this paper. The same applies to the choice of a specific local feature method. Although, the proposed selection approach is valid for several methods, for the evaluation of our approach we restrict ourselves to SIFT, since we identified SIFT to be most suitable for TSR. We make this decision based on the evaluation of feature descriptors provided by Mikolajczyk and Schmid [MS05]. In their evaluation, SIFT pointed out to be the strongest local feature in terms of recall and precision of all tested approaches. Additionally, we tested some recent local feature approaches for their suitability to TSR, among them SIFT [Low04], GLOH (gradient location and orientation histogram) [MS05] and some variants of SURF [BTG06]. The results are summarized in Table 1 and show the matching performance of these local feature approaches between sensed traffic sign images and traffic sign features stored in a database. This evaluation was done on a challenging testset (degraded traffic signs at low resolution) of 46 images with 99 traffic signs extracted from a 30 minute video sequence, captured while driving on a highway. Note that we do not provide a complete evaluation of local feature methods also considering more discriminative color versions of the mentioned techniques, since this is out of scope for this paper and does not affect our selection approach.

The common fashion, discriminative local features are used, is depicted in the 3 stages of Fig. 1(b). Basically the descriptors of both images, the sensed and the reference image, are calculated and compared to each other. Often the descriptors of the reference image are previously extracted and stored in a database. Note that we derive the terminology of sensed and reference image from the field of image registration. In the context of TSR, *sensed images* are the images of the traffic scene, which are commonly gathered by a front-view camera behind the car's windshield, while the term *reference image* refers to the image of which the descriptors are extracted from that define the traffic sign class.

	Recall	Precision
SIFT	33.68 %	78.05 %
GLOH	22.11 %	52.50 %
SURF	7.37 %	35.00 %
USURF128	25.26 %	72.73 %

Table 1: Performance of recent local feature approaches in the context of TSR.



(a) (b)

Figure 1: (a) Structure of TSR. (b) Stages of local feature matching.

As illustrated in Fig. 1(b), the matching process of two images can be divided into 3 parts. We now have a closer look into these parts and point out the specific issues arising with the application of discriminative local features in the domain of TSR.

First of all, an interest point detector is used to extract interest points that are likely retrievable. In the case of SIFT, this detector searches for extrema within the difference-of-gaussian pyramid, an approximation of the laplacian of the $2+1$D scale-space. In common, these regions are very salient and thus likely to be retrieved. An issue arises with the region's retrievability under different image scales. For example regions around coarse image structures can be retrieved from images with low resolution, while others cannot. This is due to the loss of detail caused by decreasing spatial resolution or increasing distance from the captured object (e.g. the traffic sign). Furthermore, salient regions may appear at certain scales of an image due to sampling artifacts. Feature descriptors calculated from these regions are often ambiguous and weakly discriminative and may cause false positive matches (*FP*) (cf. Fig. 5(e)). Hence, robustness to the image scale is important. Besides this, the regions extracted by an interest point detector have to be located at corners to be positionally stable, especially if the sensed image is captured from another perspective. In the traffic sign domain the positional stability is challenging, since traffic signs often include circular symbols, so that corners are

quite rare. This leads to unstable regions' locations and thus to feature descriptors that do not match with the corresponding features of the reference sign.

The second step in Fig. 1(b) depicts the descriptor calculation, which is performed after region detection. In the case of SIFT, the region patch is expressed by a 128 dimensional feature vector that is calculated from the region's edge orientation histogram. These features are often salient, that is rare in feature space, if they are calculated on a textured region. Unfortunately, in the domain of traffic sign recognition we have to deal with highly structured objects, which means the regions only include a few, very basic geometrical shapes. This leads to features with weak discriminative power. These features are unsuitable for TSR, since the following two problems may occur. If the sensed image does not contain the according object of the reference image, these features are likely to be confused with other features, resulting in false positive matches. The other problem is the inhibition of correct matches depending on the similarity measure used. Also challenging are the feature descriptions of regions that violate the planarity assumption, since they are viewpoint dependent. In the context of TSR, this issue is reduced to regions that overlap the traffic sign border, so that their features partially describe the background. This leads to features varying with the background captured around the traffic sign.

Finally, the feature descriptors of the reference image are compared with those extracted from the sensed image, as illustrated in the last stage of Fig. 1(b). This introduces a similarity measure to distinguish the features. Among the Mahalanobis distance, the Euclidean distance is very popular and was used for object recognition with SIFT by Lowe [Low04]. Lowe proposes a matching scheme with distance ratio $T_{ratio} = 0.8$ to the second nearest neighbor. So, matches have to satisfy

$$\frac{|\mathbf{v} - \mathbf{w_1}|}{|\mathbf{v} - \mathbf{w_2}|} < T_{ratio}$$

where $\mathbf{w_1}$ is the nearest neighbor and $\mathbf{w_2}$ the second nearest neighbor to feature vector $\mathbf{v}$. We compare every feature of the sensed image with those extracted from the reference sign (or stored in a database). Applying this order we receive a higher recall, than by matching vice versa. Additionally, this matching direction can more easily benefit from approaches for approximate nearest neighbor retrieval like *Best Bin First* [BL97].

A large number of FP may occur if this matching direction is used with a small number of features representing the reference image. This originates from the sparse feature space and can be alleviated by the introduction of an additional a distance threshold. We identified a distance threshold of $T_{dist} = 0.425$ to suit best the needs of TSR. This threshold is derived as 30 % of the maximal distance ($\sqrt{2}$) in the normalized feature space

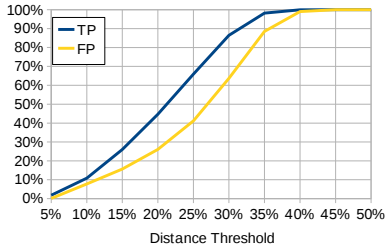


Figure 2: Matching performance in TP and FP (compared to TP and FP using only distance ratio matching) with additional distance threshold.

of $\mathbb{R}^d_+$ ($d > 1$) using the euclidean metric. Fig. 2 illustrates the dependency of true positive matches (*TP*) and FP according to the distance threshold T_{dist} obtained on a subset of the "Affine Covariant Regions Datasets" [MTS+05]. The problems arising in the matching stage are not restricted to the traffic sign domain only, but are rather common for most local feature applications. The main issue is the time spent for feature comparison, especially when the reference feature database is large. This is due to the search of the nearest neighbor for each feature vector of the sensed image. Hence, the number of features stored in the database for each traffic sign class is an important factor for the duration of the feature comparison. This number directly depends on the reference image's size and the complexity of the content as it is depicted in Fig. 3 for three different traffic signs. Thus, the issue is to select the right resolution of the reference image to receive a manageable number of features: not too many, due to speed reasons, but also not too few in order to safely recognize the traffic signs of the sensed image, even if the image is captured under severe conditions or is partially occluded.

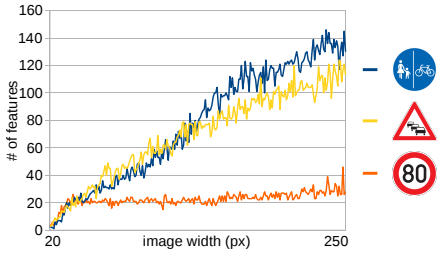


Figure 3: The number of features extracted from a reference image depends on the content as well as on its resolution.

3 RELATED WORK

Now that we have depicted the problems arising with the application of discriminative local features in the traffic sign domain, we give an overview of the research in this field. Recently, local features are utilized in the context of TSR in different ways. They are applied to the detection of traffic signs as well as to their classification.

In the field of TSD, weakly discriminative rectangle features in combination with a boosted classifier cascade are used to detect potential sign candidates [BV04, ERP07, BEV+09]. In the works of Bahlmann *et al.* [BZR+05] and Keller *et al.* [KSB+08] this technique is extended and the Haar-like features are calculated on 7 different chromatic bands to increase the discriminative power of these approaches. In contrast to the mentioned approaches, Höferlin and Zimmermann [HZ09] apply highly discriminative SIFT features to the problem of TSD. In their approach, a subsequent classification stage benefits from the estimation of the traffic sign class of the preceding SIFT detector. Contrary to this, Kus *et al.* [KGEU08] apply SIFT in combination with color features to the detection as well as to the classification of traffic signs.

Discriminative local features are adopted by Farag and Abdel-Hakim [FAH04] to classify traffic signs. They use SIFT features to classify the previously detected candidates. Ruta *et al.* [RLL07] introduce a distance transform based on color and use the resulting image representation for their local feature selection. Their approach for TSC was inspired by the trainable similarity measure used by Paclík *et al.* [PND06].

Local features are also applied to object tracking in video sequences. This topic is covered by the survey of Trucco and Plakas [TP06], to which we refer.

To reduce the amount of time spent in feature matching two methods are possible. First: the retrieval of the nearest neighbor can be accelerated. And second: the number of features can be reduced. For the first option a large number of methods already exist in literature. The problem is known as *Nearest Neighbor Search* and algorithms based on data structures like the kd-tree exist, which retrieves the nearest neighbor averagely in logarithmic time complexity for low dimensional spaces. But it is shown [IM98], that with the number of dimensions the nearest neighbor search approaches the expense of an exhaustive search. This issue is called the *Curse of Dimensionality* and can be alleviated by easing the restrictions and by avoiding the assurance of retrieving the exact nearest neighbor. The altered problem is called the *Approximate Nearest Neighbor Search*. There are also well-known methods to solve this problem like *Best Bin First* [BL97], which in most cases finds the nearest neighbor and otherwise retrieves another close candidate.

The second option is covered by Joly and Buisson, they extract Harris interest points from video data and consider only those features for matching, that are salient among all features in their database [JB05]. This way they speed-up the comparison process and obtain features of high quality. Their work inspired our method for the selection of discriminative features. They define the saliency of a feature according to Walker *et al.* [WCT98] as the likelihood of being misclassified with another feature. This definition leads towards density estimation in feature space to receive the probability density function of feature misclassification. In contrast to Joly and Buisson we also consider the robustness and the retrievability of features, that increases the matching performance, too.

Viewpoint invariance of local features is only little covered by research so far, e.g. [VS06].

4 SELECTION OF FEATURES

Based on the application of local features in TSR in section 2, we develop in this section a method for the selection of a set of robust and discriminative local features. We choose these features optimally with respect to the mentioned issues.

4.1 Retrievability and Localization Stability under Scale Changes

The retrievability as well as the positional stability of scale-adapted interest regions vary with the image scale and depend on the image structure that is covered by the region. Interest regions that can be retrieved under multiple scales are mostly located at coarse image structures (i.e. areas of low detail or low image frequency). The idea is to select those features which show the highest robustness and retrievability under scaling. We measure the retrievability of a feature by counting its occurrences at several scales. For every resolution we extract the features and match them to the features of the previous scales. Matching is done by searching for the nearest neighbor in a combined distance space of the feature vector and the keypoint's position, scale, and orientation. The distances are euclidean and both are separately thresholded to avoid false matches. If coinciding features are found, the arithmetical mean is calculated for their descriptors and positions, weighted by the number a feature was extracted in the previous scales (Fig. 4). The arithmetical mean centers the feature in image space and in feature space to reduce the influence of the interest point's localization instabilities and the variance of the region's descriptor.

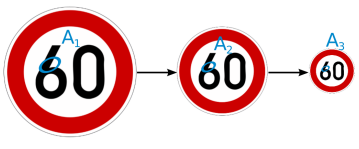


Figure 4: Selection of a region's descriptor that is robust to the change of image resolution.

Further, we measure the localization stability of a interest point under image scale changes. Brown *et al.* show, that common interest point detectors like SIFT lack of accuracy in repeatability of the interest point's

position, scale and orientation [BSW05]. They illustrate that such detectors extract the majority of their interest points with a positional variation between 0 and 3 pixels. In our tests, we experienced in average a jitter of about 0.4 % of the interest point's position, according to the image resolution. This corresponds to an expectation value of about 1 pixel based on the considered resolution range of 20 to 500 pixels in width. There is also a strong relation between the tapering of corners in the image and the accuracy of interest point localization. Hence, the average of the positional accuracy for traffic signs is about three times worse than for the natural images that we tested, although the maximal standard deviation of the interest point's position is quite similar among all tested examples (about 3 %). A similar effect is observed for the accuracy of scale and orientation. The regions' scales calculated by SIFT are disturbed in average by 0.2 % and we observed a maximal standard deviation of 0.5 % (0.05 %, 2.44 % for orientation).

Since the positional accuracy of interest points depends on the underlying image structure and varies between interest points, we also rate their quality in order to select a robust feature set. We derive the rating score from the standard deviation of the interest point's position, scale, and orientation, which is calculated on scaled instances of the reference image. Fig. 5(a) and (b) show the 5 best and the 5 poorest features according to their retrievability and positional stability under image's scale changes. As expected, features that are well retrievable among different image scales do more likely cover regions of coarse image structures. In contrast regions with poor retrievability are located at structures that show higher frequencies. Regions with high stability of position, scale, and orientation are small in size and generally extracted at tapering corners, while the regions of poor stability are located at less distinct corners (see Fig. 5(b)).

4.2 Viewpoint Variant Features

Recent discriminative local feature approaches like SIFT are invariant to rotation, scale and to some degree to affine deformations. In contrast to this, the most of these approaches are sensitive to the change of viewpoint under which the image is captured. This is especially a problem if the image patch of the detected interest point strongly violates the local planarity assumption, which is the case at ridges, corners, and occluding boundaries. Vedaldi and Soatto [VS06] try to cope with such regions by considering the scene geometry. But since the scene geometry is not always available or could be made available, we head for another method.

In our approach, we avoid additional knowledge of the scene geometry, even if it is simple like in the case of traffic signs. Therefore, we utilize a small set of training images that are captured from different viewpoints in order to measure the variation of the feature's descriptor. This method yields the additional advantage that the variation intensity is considered, too. Thus, violations to the local planarity assumption may be neglected, if they marginally affect the feature vector.

We determine the robustness to viewpoint changes of a feature by considering the spatial distribution of the distances between instances of features within the training set. For example, the subdivision of the feature patch into 16 subregions represents the spatial information of the SIFT descriptor. Thus, we measure the subregions' distance variation σ_{dist} for every instance of a particular descriptor within the training set:

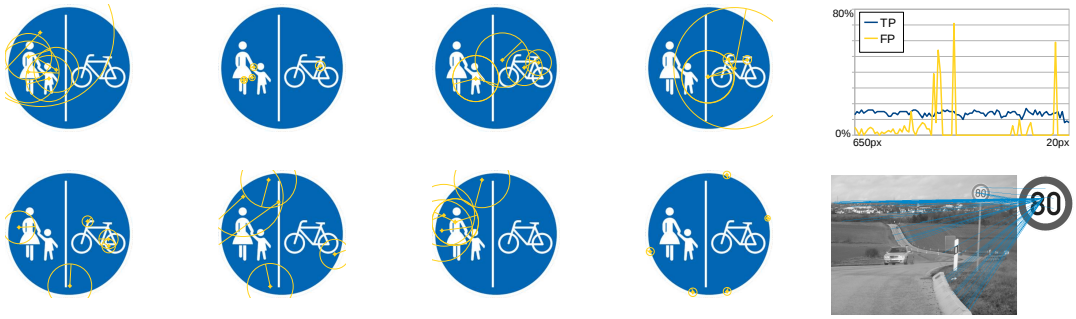
$$\sigma_{dist} = \sqrt{\sum_{k=1}^{r} (D_{k,k} - \hat{E})^2}$$

For the 128-dimensional SIFT descriptor, the distance of regions between two feature instances i and j is defined by $D_{1..r,1..r} = \sqrt[*]{(M_i - M_j)^T (M_i - M_j)}$, where M_i and M_j are their reshaped 8×16 feature matrices. The component-wise square root function is $\sqrt[*]{\cdot}$ and $\hat{E} = tr(D)/r$ represents the mean distance of the $r = 16$ subregions. The idea behind this concept is that patches with small distance variations are in common just jittered or misaligned, whereas large distance variations often indicate the inclusion of ridges, corners, and object boundaries. We use the maximum standard deviation $max(\sigma_{dist})$ of all instances of a single feature within the training set as measure for the region's stability to viewpoint changes. In the case of traffic signs, only occluding boundaries affect the region's stability with the change of viewpoint or background. Hence, stable region patches may only inclose the traffic sign area, while patches with poor stability to viewpoint changes overlap the sign boundaries (c.f. Fig. 5(c)).

4.3 Analyzing the Discriminative Power

Generally, it is desirable to select a set of highly discriminative features. This raises the probability for correct matches and reduces the likelihood of feature vectors to constrain each other's matching performance. During the evaluation of interest point detectors, we noticed the occurrence of features with marginal discriminative power at certain image sizes, originating from sampling artifacts. In Fig. 5(e) the matching performance of SIFT features at different scales is visualized. It points out that certain scales show sudden peaks of FP. These peaks originate from highly indiscriminative features as depicted in the bottom row of Fig. 5(e).

The discriminative power of a feature vector can be expressed by its saliency, which is equivalent to its rarity in feature space. Therefore, the probability density function of retrieving the wrong nearest neighbor is directly correlated with the density of the feature's neighborhood. This introduces kernel density estimation as



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

Figure 5: The 5 best rated features (circles, top row), the 5 poorest regions (circles, bottom row). Both depicted on an exemplary reference sign. (a) Retrievability under scale changes, (b) positional stability, (c) descriptor stability on viewpoint changes and (d) saliency of reference features among the features extracted from MIT-Highway set. Further explanation in the text. (e) Features with low discriminative power. Top: occurrence of these features at some scales. Bottom: interest point relation of these features.

appropriate measure for the discriminative power of a feature. We estimate the density function $\hat{f}$ at point v using a Gaussian kernel by

$$\hat{f}(v) = \frac{1}{|M|} \frac{1}{\sigma(2\pi)^{\frac{d}{2}}} \sum_{x \in \xi} e^{\frac{\|v-x\|^2}{2\sigma^2}} \qquad (1)$$

The set of features is represented by ξ and d refers to the dimensionality of the feature space which is 128 in the case of SIFT descriptors. We choose bandwith $\sigma = 0.3\sqrt{2}$ to be equal to the distance threshold T_{dist}. That corresponds to a window radius of 30 % of the maximum distance in normalized feature space. We populate the feature space ξ with features from typical traffic scene images, harvested from the MIT-Highway set. The bottom row of Fig. 5(d) shows an example of features with poor saliency. These are located along the traffic sign border and do not contain any discriminative image structures, while the top row regions include salient structures.

Besides feature's domain specific saliency, there are two other groups of features that are important in order to obtain a high matching performance. First, ambiguous features within the reference image should be avoided, since similar descriptors tend to inhibit each other, when nearest neighbor matching with distance ratio is used. Same holds for the second case: the saliency of features across the traffic sign classes. Obviously, a higher recall performance and also a better distinction between the sign classes can be achieved if features are selected that are discriminative across all classes. For both cases, we also apply the Parzen window approach according to 1, but either ξ is populated with the features of one traffic sign class or with the descriptors of all other traffic sign classes.

4.4 Choosing the Optimal Feature Set

Now that we have introduced a variety of criteria for robust and discriminative features, we select an optimal set of n features to represent the traffic sign class. We define the overall quality q of feature f by a weighted sum of its scores achieved for each criterion c_i:

$$q(f,S) = \sum_i w_i c_i(f,S)$$

The feature set can be adapted to a particular problem by adjusting the weights w_i of each criterion. Since the quality q of a feature depends on the other features selected for the representation set S, we have to solve a non-linear optimization problem. There are various approaches to approximate the solution of a non-linear optimization problem, e.g. evolutionary algorithms. We use a greedy algorithm to choose in every iteration step the feature with the highest fitness quality q, with respect to the features S selected in previous steps. For initialization, we start with an empty set S. This simple approach is very fast and yields good results, even if the detection of the global optimum is not ensured and thus the set may only be locally optimal.

5 EVALUATION

For the evaluation of the proposed method we first compare the feature sets obtained by several feature selection approaches for an exemplary traffic sign. Then, we compare the performance of our approach with that of the conventional representation of the traffic sign class as reference image of a certain size.

Fig. 6 shows the sets of 5 features selected by three different approaches: (a) the proposed approach, (b) selection by choosing a proper image resolution, and (c) selection of the most discriminative features, similar to Joly and Buisson [JB05]. Fig. 6(a) points out that the features retrieved by the proposed method are located on discriminative and coarse image structures, while the other both approaches either lack of the one or the other property. In Fig. 6(d) the matching performance of the 3 approaches is compared for the exemplary traffic sign class. It points out that especially

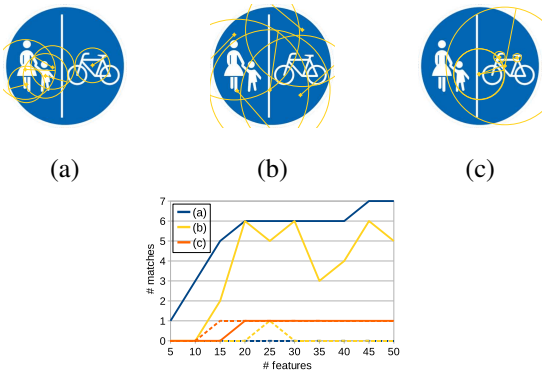


Figure 6: Exemplary feature set on a reference sign. Only 5 features shown for clarity. (a) Proposed method, (b) extracted for a certain reference image resolution, (c) selected at low density in feature space and (d) performance of these methods for increasing number of features. TP (solid line), FP (dashed line).

the features retrieved by method (c) have poor matching performance, since the most discriminative features of the reference sign often show low retrievability under scale changes.

We use two different testsets for further evaluation. The first contains 12 images (640x480 px) with 9 traffic sign classes, among them images captured under adverse conditions: blur, noise, perspective deformation and chromatic lighting. We call it the "German" testset, since it covers German traffic signs. The second set includes 30 images (360x270 px) of 3 classes. It is introduced in [GP03]. We refer to it as the "Dutch" set for the same reason. We compare the matching performance of the features selected with the proposed method (a set of training images per sign class) to those extracted from reference images of a certain size, as it is the conventional method to limit the number of features. That means, the selection parameter is the image resolution. For evaluation we restrict the number of selected features to 20. The results of 3 experiments for both testsets are presented in Fig. 7. The performance measure is provided by means of TP and FP.

The first experiment (Fig. 7(top row)) shows the matching performance of both methods with increasing feature count. The reference image's size was adapted in case of the conventional method in order to provide the proper number of features. Obviously, the matching performance benefits from the features selected by the proposed method. Especially, the low number of false positives for the "German" testset is remarkable, while the conventional method suffers from indiscriminative features at some resolutions. For the chart presented in the middle row of Fig. 7 we do not longer fix the number of features selected by the conventional method (for the proposed method we use a set of 20 features) but show the performance of the features

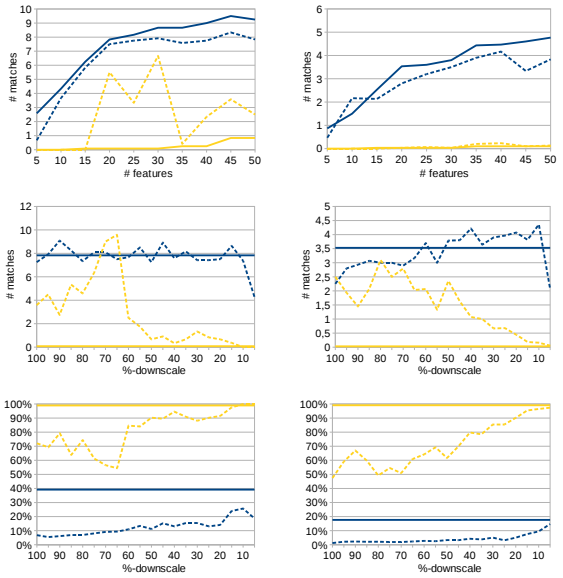


Figure 7: Average matching performance. Top and middle row: TP in blue/dark gray, FP in yellow/light gray. Bottom row: Recall in blue/dark gray, precision in yellow/light gray. Proposed method (solid line), conventional method (dashed line). (a) "German" testset, (b) "Dutch" testset.

extracted at different scales. We see that it is difficult to choose a certain reference image size with high TP and low FP. This becomes even more obvious if we involve the number of features and thus express the performance by means of recall and precision (c.f. Fig. 7(bottom row)). Our method also reduces the time spent for matching, since the desired number of features can be defined by the user (cp. Fig. 3). Hence, feature dependencies to the reference image's resolution are eliminated and the matching duration becomes appreciable.

6 CONCLUSION

We presented a method to select a small set of discriminative local features (e.g. SIFT features) with excellent matching properties. Applying our approach we were able to increase the matching performance by speed, recall and precision. Hence, issues inherent to local feature matching were alleviated. Further work has to consider other selection approaches to retrieve the globally optimal set of features. In addition, new criteria like the spatial distance of the interest points or the coverage of the reference object by the set of features can be introduced, to increase robustness against occlusion, too.

REFERENCES

[BEV+09] X. Baró, S. Escalera, J. Vitrià, O. Pujol, and P. Radeva. Traffic Sign Recognition Using Evolutionary Adaboost Detection

and Forest-ECOC Classification. *IEEE Transactions on Intelligent Transportation Systems*, 10(1):113–126, 2009.

[BL97] JS Beis and DG Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings to IEEE CVPR'97.*, pages 1000–1006, 1997.

[BL07] M. Brown and D.G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, 2007.

[BSW05] M. Brown, R. Szeliski, and S. Winder. Multi-image matching using multi-scale oriented patches. *IEEE CVPR 2005.*, 1:510–517 vol. 1, 2005.

[BT05] VTT Building and Transport. Analysis of context of use and definition of critical scenarios, 2005. EU project HUMANIST. Reference: AVTT-030305-T1-DA(1).

[BTG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. *SURF: Speeded Up Robust Features*, volume 3951 of *Lecture Notes in Computer Science*, chapter Computer Vision - ECCV 2006, pages 404–417. Springer Berlin / Heidelberg, 2006.

[BV04] X. Baró and J. Vitrià. Fast traffic sign detection on greyscale images. *Recent Advances in Artificial Intelligence Research and Development*, pages 69–76, 2004.

[BZR+05] C. Bahlmann, Y. Zhu, Visvanathan Ramesh, M. Pellkofer, and T. Koehler. A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. *IEEE Intelligent Vehicles Symposium.*, 2005.

[ERP07] Sergio Escalera, Petia Radeva, and Oriol Pujol. Traffic sign classification using error correcting techniques. *VISAPP 2007*, pages 281–289, 2007.

[FAH04] Aly A. Farag and Alaa E. Abdel-Hakim. Detection, categorization and recognition of road signs for autonomous navigation. *Proceedings of Acivs 2004*, pages 125–130, 2004.

[GP03] C. Grigorescu and N. Petkov. Distance sets for shape filters and shape recognition. *IEEE Transactions on Image Processing*, 12(10):1274–1286, 2003.

[HZ09] B. Höferlin and K. Zimmermann. Towards reliable traffic sign recognition. In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 324–329, June 2009.

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *18th ACM Symposium on Theory of Computing*, pages 604–613. ACM New York, NY, USA, 1998.

[JB05] Alexis Joly and Oliver Buisson. Discriminant local features selection using efficient density estimation in a large database. In *Proceedings to 7th ACM SIGMM MIR*, pages 201–208, 2005.

[KGEU08] M.C. Kus, M. Gokmen, and S. Etaner-Uyar. Traffic sign recognition using Scale Invariant Feature Transform and color classification. In *Computer and Information Sciences, 2008. ISCIS'08. 23rd International Symposium on*, pages 1–6, 2008.

[KSB+08] Christoph Gustav Keller, Christoph Sprunk, Claus Bahlmann, Jan Giebel, and Gregory Baratoff. Real-time recognition of u.s. speed signs. In *IEEE IV'08*, 2008.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[MS05] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10), 2005.

[MTS+05] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *Int. J. Comput. Vision*, 65(1-2):43–72, 2005.

[PND06] P. Paclík, J. Novovicova, and RPW Duin. Building road-sign classifiers using a trainable similarity measure. *IEEE Transactions on Intelligent Transportation Systems*, 7(3):309–321, 2006.

[RLL07] A. Ruta, Y. Li, and X. Liu. Traffic sign recognition using discriminative local features. *Lecture Notes in Computer Science*, 4723:355, 2007.

[TP06] E. Trucco and K. Plakas. Video tracking: a concise survey. *IEEE Journal of Oceanic Engineering*, 31(2):520–529, 2006.

[VS06] A. Vedaldi and S. Soatto. Viewpoint induced deformation statistics and the design of viewpoint invariant features: Singularities and occlusions. *Lecture Notes In Computer Science*, 3952:360, 2006.

[WCT98] KN Walker, TF Cootes, and CJ Taylor. Locating salient object features. In *BMVC'98*, pages 557–566, 1998.

A Simple and Effective CAPTCHA by Exploiting the Orientation of Sub-images Cropped from Whole-size Photos

Woo-Keun Chung, Seung-Hyun Ji, Jong-Woo Kim and Hwan-Gue Cho
Graphics Application Laboratory,
Dept. of Computer Science and Engineering
Pusan National University, Republic of Korea
{wkchung, shji, jwkim73, hgcho}@pusan.ac.kr

ABSTRACT

Automated detection of image orientation has previously been studied as an important problem in intelligent image processing and computer vision. For this problem, numerous methods and tools have been developed by adopting approaches such as objects segmentation, color feature analysis and machine learning e.g., Support Vector Machines(SVMS). But conversely, the difficulty of image orientation can be used to examine the robustness of a CAPTCHA(Completely Automated Public Turing test to Tell Computers and Human Apart). The automated image orientation problem previously only had been studied and solved using typical photos which almost include important semantic cues such as people, bright sky, dark ground and vertical edges. In this paper we propose a simple prototype CAPTCHA, which exploits the hardness of orienting sub-images cropped from a whole digital photo. Our CAPTCHA takes 8 sub-images from base-photos and rotates them randomly. Then we present them to the user, who is required to find the correct orientations of the 8 sub-images. The true orientation is easily obtained since most current high-end digital cameras have an automatic mechanism to store its orientation in EXIF. Thus we can simply and easily obtain the image orientation without applying complicated computation. For our experiment, we have collected about 1850 base photos that provide more than 100,000 different sub-images. Experiment showed that the accuracy of our CAPTCHA with humans is about 95%. We think this sub-image orientation is hard to solve by an automated procedure since all previous machine learning procedures have only considered whole photos with enough semantic cues, rather than partial image segments. Another advantage of our system is that user interaction is simpler(there are four choices) and more intuitive than a common text-based system or the previous image orientation method with arbitrary rotation. Experiment showed that common users performed at most two rotations for each sub-image. The total time to complete orienting the 8 sub-image orientation was less than 15 seconds which is significantly shorter than that of previous image-based CAPTCHAs.

Keywords: CAPTCHA, Sub-image, Image Orientation, Image Classification, Machine Learning.

1 ISSUES IN CAPTCHA

1.1 Motivation

Recently, there have been numerous automated software bots and automated scripts that exploit public web services. The problems caused by automated SPAM generators are becoming serious for public web bulletins. So the user is commonly required to solve a Turing test problem, namely a "Completely Automated Public Turing test to tell Computers and Humans Apart(CAPTCHA), before they are allowed" to use web services. Also HIP(Human Interaction Proof) terms are widely accepted in this subject. Therefore the main goal of a HIP or CAPTCHA is to discourage script attacks by raising the computation and development cost of breaking a HIP or CAPTCHA to an unprofitable level[10]. One criteria of a CAPTCHA is that each puzzle should be easy for most people to solve, but difficult for automated bots to solve. We can easily construct a CAPTCHA that meets the criteria, by customizing each CAPTCHA problem manually, such as "Find a very funny picture among these photos". But there is another criteria for a CAPTCHA: each problem should be efficiently generated and evaluated by an automated procedure[5]. This is a contradictory issue in every CAPTCHA system. This paper addresses one simple procedure to make an automated image-based CAPTCHA from a small set of user photos which can be efficiently collected and refined. Also our CAPTCHA system is hard to break using the previous tools depending on training-based machine learning tools.

A brief introduction to our system is provided by the following figure. The puzzle is to guess the correct orientation of a cropped sub-image from a whole-size photo. We define a whole-size photo as an image file totally recorded in a single shot by a modern digital camera. In general we assume that every whole-size photo includes a few meaningful objects we can recognize easily and immediately.

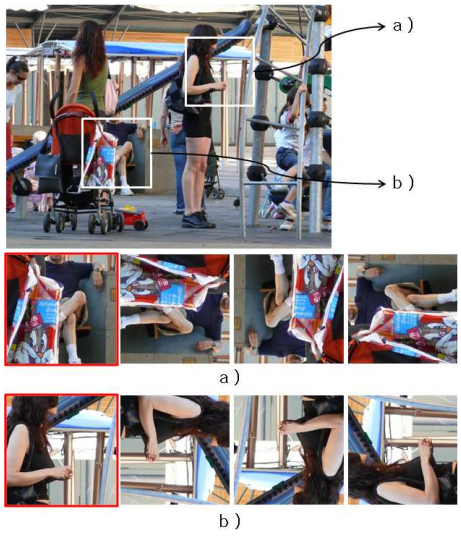


Figure 1: For a common given photo, we randomly crop sub-images. Next we show then to a human solver, who is required to find the correct orientation when the whole image is not shown.

The top image of Figure 1 shows a whole-size photo taken in a public children's park. We crop two sub-images (a) and (b) randomly, then we present them to the user, who is required to find the correct orientation. Each user can freely rotate the image 0, 90, 180 and 270 degrees in order to find the right position. Figure-1(a) and (b) shows four possible orientations for two square sub-images. As shown, a human observer can easily solve this problem (the leftmost one is correct) at a single glance. But previous automated tools for image orientation cannot deal with these partial images efficiently, since they have only considered the structures of whole photos, which include abundant semantically meaningful whole objects, rather than partial scenes. So the random chance to solve the two sub-images in Figure1 is 1/16.

1.2 Recent Advances on CAPTCHA

We simply survey the most recent work on CAPTCHAs for a comparison in terms of user-friendliness and system performance. Traditional CAPTCHAs asked users to identify a series of letters which were geometrically transformed to defeat a character recognition system[3, 10]. But recently, many smart character recognition softwares have succeeded in deciphering plain text-based CAPTCHAs. One way of breaking a noised text image is projection-based segmentation to clear artificial line noise, which have been demonstrated to break MSN and YAHOO[6]. And a careful shape analysis and machine learning approach also works well in deciphering text-based CAPTCHA [10, 12].

Since spammers are very eager to utilize academic algorithms to break CAPTCHAs, the defeating tools are easily exploited and distributed by spammers. Thus in order to defend against deciphering tools, text-based CAPTCHAs have increased the amount of noise and introduced different types of noise such as global warp into text images. But unfortunately this kind of hard-noised text puzzle makes it harder for humans as well as computers. Consequently a highly noised and wrapped text-based CAPTCHA leads to lower success rates(less than 60% accuracy) and frustrates common users[3, 5]. The drawbacks and pitfalls are well explained with typical examples in [2]. And the usability issues in character-based CAHPTCHAs were deeply discussed in one notable work[9].

In order to overcome the weak points of text-based CAPTCHAs, new Image-Based CAPTCHAs(IBCs) have been introduced recently. One great advantage of an image-based puzzle is that it is natural language-independent. There are many variants of IBCs[4]. These include selecting an appropriate label for an image and selecting an abnormal image for a set of subject images. One notable work has revealed the performance and disadvantages of image understanding CAPTCHAs [4]. Another interesting example of IBCw is identifying cats in 12 photos of both cats and dogs[7]. It was reported that humans solve this task 99.6% of the time in less than 30 seconds. Others showed that calculating a simple math problem can be used as a reasonable IBC[2].

Unfortunately most previous image-based CAPTCHAs have a basic problem in automatic generation, since they all require a priori knowledge such as the object name in the image or the true orientation of an image taken. For example if we ask for a good name(label) for a given image ,then we must specify the correct name(label) in advance or a priori, which is a burdensome work for a human generator. Also the name of the image may be dependent on the human generators. And if the size of the image database is not large or contains highly restricted objects, eg., cats and dogs only, then a well-known powerful machine learning model such as Support Vector Machines can break it after a moderate training and adaptation procedure[13], which insists that they can distinguish cats and dogs with more about 82.5% accuracy. Rather than a static image, Srikanth proposed a real-time image recognition CAPTCHA[16].

2 DETECTING IMAGE ORIENTATION

2.1 Image Orientation Problem

The image understanding problem has long been studied as a fundamental issue in computer vision. One typical application of image understanding is how to detect the human faces. Numerous computation models and tools and testing data on face detection have been well introduced in [21]. Or there is a specialized

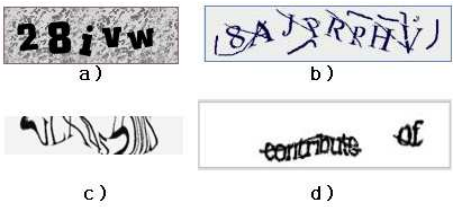


Figure 2: Typical types of text-based CAPTCHAs (a)text on noisy back ground. (b) noisy line segment added. (c) global warp applied. (d) warped after noise curve inserting.

cognition method for only one class of car objects[14]. As a subproblem of image understanding, automatic image orientation detection has been studied recently for content-based image organization and automated retrieval. But this type of image orientation detection is believed to be a very difficult problem, since state-of-the-art computer vision techniques still can not infer high-level abstraction of photo images[17]. For instance, it is very hard to recognize a dog partially hidden by a tree.

Generally speaking, a few well-established machine learning approaches such as MDL, LVQ, PCA, LDA(Linear Discriminant Analysis) and HDR(Hierarchical Discriminating Regression) are mainly applied for image orientation problem with a reliable set of training images. Others deeply studied the upper limit of orientation in terms of human psychological aspects [8]. Others have revealed the characteristics of image statistics in determining of image orientation[15], which was an essential factor leading to SVM-based automated image orienting system. Support Vector Machines are the most common learning tool for image orientation [11, 15, 19, 20].

It is worth to noting the psychophysical features in human image orientation perception[8]. For a randomly collected set of 1650 images five observers were asked to provide the correct orientation of the given image at various resolutions. The experimental results show that for typical images the orientation accuracy was close to 98% when all available semantic cues in high resolution photos were used. For coarser-resolution images, the accuracy was around 84%. Which means that upper limit of human orientation accuracy is about 84-95% depending on the image quality. Interestingly sky, ground and standing people are the most useful and reliable cues to find the correct orientation among other semantic cues including buildings, ceilings, color, grass, roads, texture and trees. This implies if the semantic cues are not observed or partially available in the image, then image orientation is a difficult problem.

The hardness of image orientation by computers can be a good basis for the usefulness of IBCs. Recently, detecting image orientation was used by a CAPTCHA system[5]. The IBC asks for an upright position of a circular form of image which was maximally inscribed in the image. This involved collecting a set of image s from a large image repository. Some images which have with an ambiguous upright orientation such as balls, guitars and general textures were excluded.

2.2 Machine Learning Approach for Image Orientation

Several machine learning approaches have long been applied to obtain the image orientation. The preprocessing step in this work involves transforming each photo into a vector with more than 1000 features[5]. For example, for an input feature vector, [5] used 1,965 means and 1,965 variances from 91 disjoint sub-regions. Next, the experimenters carefully collected a set of training photos in order to make a power classifier over unknown data. After applying numerous training steps, we finally obtained learned machine(program). In this step, it is very important to provide typical photos representing orientations that are very familiar to humans. Currently, Support Vector Machines are widely applied, and other methods combined SVMs with boosting models in order to obtain more correct and reliable results[18].

We describe the latent limitations in the machine learning approach for image orientation. First of all, input training images are commonly wide rectangular images with a height:width=ratio of 4:3. The preprocessing step for machine learning is where, photo images are usually divided. So we assume that we should divide a wide rectangle image into n by m grid-base sub-images. The top rows of an upper square sub-image are likely to be similar in general photos as was illustrated in the examples in Ref.[5, 17]. Thus if the training set includes this kind of plain images, it is easy to classify simply by observing the number of similar adjacent sub-grid images. Therefore, though the machine learning approach uses more than 2000 features of photosincluding RGB, YIQ, color intensity, vertical and horizontal edges, etc, we believe that the most crucial features of image vectors are smaller(30-50?) than we expected.

2.3 Problem of Sub-image Orientation

According to [8], the most important semantic cue(spatial region) is sky in the upper part of an image or grass or dark brown ground images in the lower part of a whole photo, as shown in Figure 1. So if the training images are not taken from a whole photo such as a partial region, then the most all kinds of machine learning approach fail, since a partial image hardly provides any kinds of semantic cues or environmental

information helpful to obtaining the correct orientation. So if we restrict all the testing photos to exactly square photos, then the current machine learning approach does not show the good performance that was obtained in the previous work. We show another example of this issue in Figure 3. We know it is very easy to obtain the orientation of whole photos(top), but for their 4 sub-images(shown below)it is hard to obtain the correct orientation, since the sub-images do not include any whole faces. We have tested these four sub-images by applying well-known face detection tools and systems. Most face detection systems do not recognize the partial faces shown in Figure 3 (a)-(d).

Common users never take photos such as those shown in (a)-(d), and they were not used in any training set for machine learning tools where the sub-images do not provide any meaningful information to an automated machine. However, interestingly a human can recognize the right orientation of the 4 sub-images immediately, since a human has a very broad and abstract knowledge of partial face images. We have simply tested these 4 sub-images with 15 undergraduate students, who completely identified the orientation.

But most face recognition tools and systems did not recognize these four face sub-images, since the computing principles used in these face recognition trained machines were only developed using whole, even more, straight-view human faces. In fact most of the sub-images do not have the other semantic cue that the upper one is significantly brighter than the lower one in most all photos, which is a significant human perception cue to decide the orientation[18].

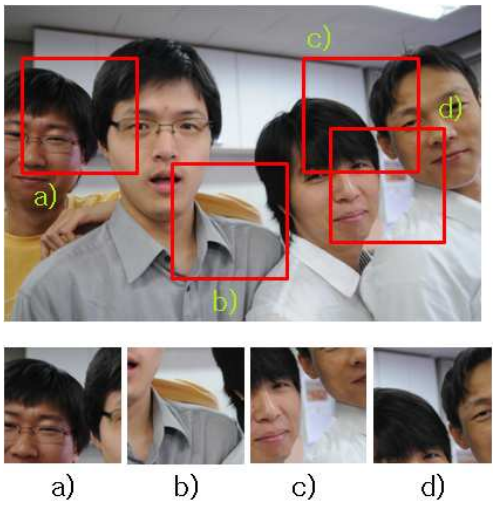


Figure 3: 4 different sub-images cropped from a photo. Though they do not have any single complete face, humans can easily obtain the correct orientation.

We provide another example to show that orienting a sub-image without whole images is hard for a computer, but easy for a human. See the other cropped sub-image which was taken from a travel photo in Figure 4. A Computer can easily orient this photo, since it has very typical semantic cues (one face and the a brighter upper region and darker lower region). But if only a sub-image of a partial face is used, then orientation is a very hard task because it does not contain enough semantic cues. But humans can do it without any hesitation.

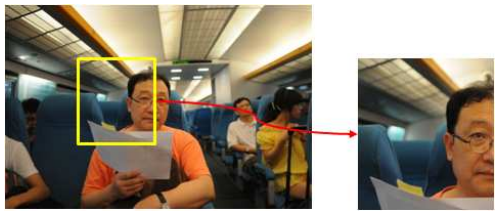


Figure 4: Computers can easily orient the left-hand whole photo but they cannot easily solve the right-hand partial subimages. Humans can do it immediately.

2.4 Image Orientation in EXIF of High-End Digital Camera

We believe the image orientation problem is easier due to the recent development of high-end digital cameras. Most of the high-end DSLR cameras from Canon, Nikon, Panasonic, Sony and SamSung have been equipped with internal automated orientation correction hardware. This information is stored in the EXIF field of each digital camera. EXIF denotes the Exchangeable file Image Format, which includes information about camera settings such as the timestamp, focal length, ISO, flash status etc. In the future, we will not have to apply a complicated machine learning system to obtain the orientation automatically, unless the photographer intentionally disturbs the image orientation. So in this paper, we assume that the true orientation can be easily obtained by retrieving the EXIF information of each digital photo. Thus in our CAPTCHA experiment, we only considered photos with orientation information recorded automatically by the mechanical sensor installed in digital cameras.

3 STRUCTURE OF OUR SYSTEM

3.1 Overview of Our System

In this section we explain the overview of our CAPTCHA system. Our system consists of four parts: the two photo databases DB1 and DB2, the random selector for image and cropping, the filtering module and the user interface. The n-plate is a problem including k sub-images cropped from our data base. If a human solver can provide at least k correct answers

from n sub-images, then we say that a human can solve a k-plate CAPTCHA problem.

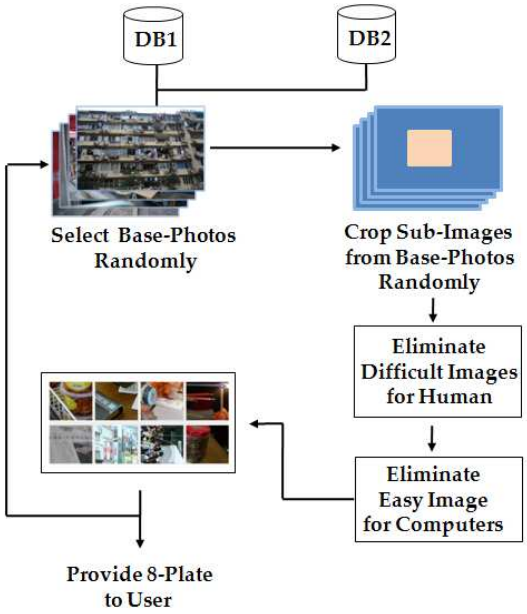


Figure 5: Overview of Our CAPTCHA and Data flow

DB1 was established by selecting photos which were previously taken during travel. And we intentionally took about 650 photos for use in our CHAPTCHAs, which were stored in DB2. We did not include any regions of sky, ground, paths and human faces. Also, we tried to avoid objects having too many vertical or horizontal edges such as straight buildings. If the inclusion of straight edges was inevitable, then we slightly rotated the viewing angle so as to avoid detection by edge segmentation tools automatically. But the rotation was limited to +10 to - 10 degrees to avoid frustrating a human solver's perceptions. One simple way to make DB2 photos is taking a scene with 5 to 10 small familiar stationeries, which are placed randomly on a desk within a distance of 1.0-2.0m from the camera lens. So after gathering around 600 photos we manually refined them to obtain the final photos. In the following, the base photos are the set of all photos stored in DB1 and DB2.

3.2 Eliminating Bad Sub-images

Though we carefully selected the base photos, bad sub-images can be included in the final 8-plate problem, since a smaller region of a sub-image may contain a scene that is very hard for human solvers. When we crop subimages from the base photos, we automatically exclude those with one of following two conditions, since they do not provide any partial semantic cues to human.

- A photo with a large color segmentation region, which is called a monotone region in the following paper. Thus a monotone sub-image implies that it has no semantic objects. A plain texture without any object may belong to this category. Since a monotone region does not provide any orientation cues, it cannot be solved either by a computer or a human. Then the remaining problem is how to decide if a photo has a monotone region. We quantized the whole image into regions of 25 different natural colors by the color segmentation algorithm of [1]. They cleverly proposed and defined 25 representative colors including Black, Sea green, Light green, Aqua. Rise Yellow, Pink, Orange and White, for general photos favoring human perceptual sensing. So we can decompose a whole photo into at most 25 different colored regions. If the maximal area of one(from 25)-colored color segment is larger than 20% of the whole photo's area, then we discard the photo, since it is a hard problem for humans. The elimination criteria can be improved by experiment, but in this paper we do not address this problem further and we adopt a very simple rule. For instance, all photos including a larger sky region or ground(grass) region or water and wide walls are discarded(See 6).

(a) (b)

Figure 6: Two whole-size photos including larger monotone segments. If a red square is cropped, then it is not possible for a human to obtain the orientation due to the absence of meaningful objects(even partial ones)

- A photo including numerous human faces at a recognizable size and resolution. The most crucial semantic cue in a general photo is a human face, since the human eyes and month completely determine the image orientation in most cases(except when people are lying on the ground). Numerous face detection systems are available and can be obtained easily, such as OpenCV tools. We have utilized the tools released from *Applied Device* available on http://www.applieddevice.com/facedb/fs.php. If the size of the human face segments detected by the above tool is smaller than the area of the sub-image cropped by our system, then we disregard the photo, since the cropped sub-image completely contains the face image segment. Figure shows two photos belonging to this category. We exploited available codes to eliminate images with faces.

Figure 7: Examples of sub-images eliminated by our system. The left-hand two images were eliminated by the monotone region and the right-hand two were eliminated due to the faces it contained.

4 EXPERIMENT

4.1 Human Accuracy Analysis

In order to evaluate our system, we performed several user studies. We collected 350 base-photos which were taken in during overseas travel in Shanghai and Europe. Also we intentionally took more than 350 photos for use with this CAPTCHA. After refinement we finally initialize DB1 and DB2 with 1650 photos. Since we took the sub-images from the base photos, we can took more than 10,000 sub-images which were mutually different more than 70% of the time in terms of the spatial region.

We have evaluated our system with 10 human testers. We gave each tester, we gave 50 "8-plates", which consist of 8 different sub-images in on screen. We regard the answer as correct if a human finds more than 7 correct orientations from the given 8 images. The probability to break this 8-plate problem by random chance is given $\binom{8}{8} \cdot (1/4)^8 \cdot (3/4)^0 + \binom{8}{7} \cdot (1/4)^7 * (3/4)^1 = 25/(2^{16}) \approx 0.000381$

In order to find the optimal image size, we varied the crop size by 0.1*height of the base photos. In the following Table, *SIZE* denotes the size of the sub-images cropped from the whole-size photos. $SIZE = 01$ means the width and height of cropped sub-image is $0.3 * height$ of the whole-size photos. So if the width:height ratio of a photo is 3:2, then the area of a $SIZE = 0.3$ sub-image is $1/15 = 0.06$. So it is 6% of the area of a whole-size image, which is quite small. So if we can exclusively extract sub-images from a photo with $SIZE = 0.3$, we obtain get 15 totally different(disjoint) sub-images. If we allow some overlaps in the cropped sub-images, then we can get more than 30 sub-images, which is one of the advantages of our system.

We us show the main experiment plate of our system in Figure 8 which shows six different sub-image $SIZE = 0.1, 0.15, 0.2, 0.3, 0.35, 0.4$. The overall accuracy rate was more than 90% $SIZE > 0.3$. Since it is generally believed that a CAPTCHA with higher than 90% human accuracy is acceptable, we insist that $SIZE = 0.35$ provides quite good performance with an accuracy 97%, which so satisfactory in practice.

Detail statistics for the 10 testers are provided in Table 1. We did not apply the Partial Credit Algorithm

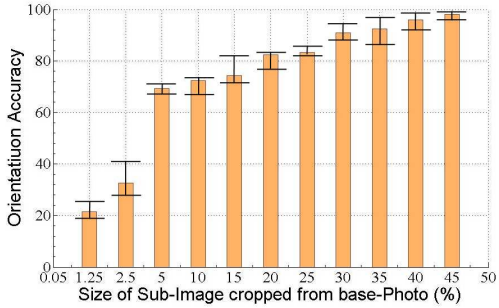


Figure 8: Experiment Graph

SIZE	0.1	0.15	0.20	0.25	0.30	0.35	0.40
tester 1	77.4	80.2	85.7	84.9	94.9	97.6	98.5
tester 2	73.4	76.4	80.7	82.1	88.8	98.6	98.1
tester 3	78.4	79.1	80.1	87.2	88.1	97.7	100.0
tester 4	75.1	74.5	83.9	86.3	91.5	98.4	98.2
tester 5	72.6	81.2	80.1	83.2	91.1	96.8	98.1
tester 6	75.9	78.2	79.4	86.9	93.5	97.5	100.0
tester 7	74.9	75.1	82.4	85.7	88.3	98.7	98.5
tester 8	75.5	74.3	81.2	81.2	94.8	96.4	100.0
tester 9	73.9	80.2	78.5	81.2	88.1	96.4	98.1
tester 10	74.6	77.3	79.2	87.1	93.1	97.2	98.4
Average	75.1	77.6	81.1	84.4	91.2	97.4	98.7

Table 1: 10 testers have tried to solve around 30-50 $8-plate$ problems at their work place for sub-images SIZE.

Figure 9: Example an 8-plate with SIZE=0.10

introduced in [7], since it may weaken security as was pointed out in [7].

We explain the resolution of the base-photos as follows. The base-photo resolutions are various, ranging from a low resolution 1000*1000 photo to a high 4000*4000 one which was taken from a high-end DSLR. If the size of a sub-image is greater than 0.30, then the orientation accuracy does not depend on the image resolution. The accuracy is likely dependent on the relative size, rather than the absolute size(=the number of pixels) of the cropped sum-images. Figures 9 and Figure 10 show examples of 8-plate snapshots where SIZE=0.1 and 0.30. Figure-11 is the corresponding solution of the problem Figure=10.

Figure 10: Example an 8-plate with SIZE=0.10

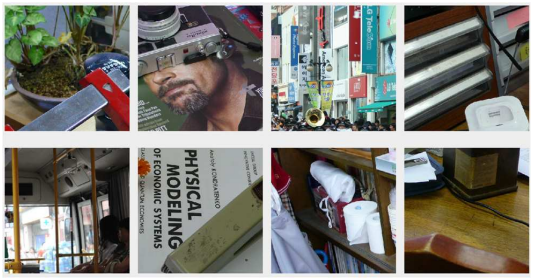


Figure 11: Corresponding correct orientations of 8-plate subimages in Figure 10

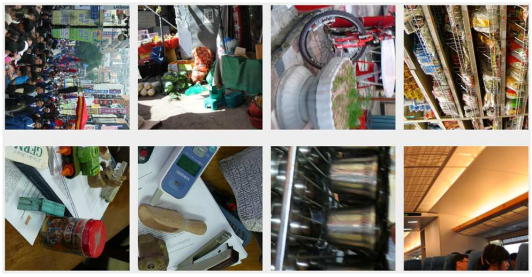


Figure 12: A plate of eight sub-images cropped with SIZE=0.40. From top-left, the photos show a crowded street during rush hour(more than 20 tiny faces), flee market, bicycle leaning against a street wall, shelf in a small market, five objects on a desk, objects(phone and stapler) on a desk, cups arranged on a restaurant shelf and a meeting room with a bright ceiling.

We show an example an 8-plate with SIZE=0.4 in the following Figure 12.

Intuitively we observe that an 8-plate of SIZE=0.4 is easier to solve by visual inspection than one with SIZE=0.30, since the larger sub-image may contain more semantic cues to help obtain the correct orientation.

4.2 Security and User Interaction

We discuss the usability of our system. There is no solid security analysis model for CAPTCHA, since it depends on very diverse human solvers.

The generally accepted figure in the previous work is that the system should admit automated bots with less than 1/10,000 probability within 30 seconds and only a small amount of human effort it is needed. Let the $(n,k)-plate$ problem denote that we regard the answer as right if a responder identified more than k sub-images from a total of n sub-images in the n-plate. The probability of random chance for the $(8,8)-plate$ is $1/4^8 = 1/65,536$, which is quite acceptable in practice. The probability of a random attack for the $(8,7)-plate$ is $(24+1)/4^8 \approx 3/10,000$. In practice we hope to use the $(9,8)-plate$, for which the random chance to break it is $(1+27)/4^9 \approx 1/10,000$.

Image SIZE	Accuracy %	Clicks	Time(Sec.)
0.10	75.1	0.81	6.2
0.15	77.6	1.12	10.4
0.20	81.1	1.15	14.3
0.25	84.4	1.46	16.4
0.30	91.2	1.41	13.7
0.35	97.4	1.14	9.2
0.40	98.7	1.11	6.5

Table 2: Average success rate and average user clicks for each sub-image and average time for completing an 8-plate according to SIZE=0.1 - 0.5

We consider the user-friendliness of our system. We have checked the number of total clicks and elapsed time to clear an $(8,7)-plate$ in practice in Table 2. The average time for an 8-plate is around 13 seconds, which is faster than the previous text-based CAPTCHAs and the other IBCs, because the user only interacts with one mouse click, which rotates the image 90 degrees clockwise. It is interesting that solving the plate of with the small SIZE=0.1 and big SIZE=0.40 requires shorter time since a human does not need to rotate the sub-image due to the absence of any semantic cue objects or even partial images. Contrary to the small sub-image, about 8/4 sub-images in an 8-plate can be in a correct orientation if it is bigger(SIZE > 0.35). Thus it takes a shorter time to complete a plate.

Experiment showed that our 4-way discrete orientation CAPTCHA is quite efficient for a user compared with sliding rule-based image rotation[5]. Less then 2 clicks are needed to obtain the correct orientation, which implies that the interactions of our system are very human-friendly. So our system is very compatible to with mobile devices, e.g., touch-based cellular phones, where the user is allowed only a simple touch panel.

5 CONCLUSION

This paper introduced another kind of CAPTCHA by exploiting the orientation of cropped sub-images from whole-size base photos. We summarize the notable contributions of our idea.

- One fundamental disadvantage of the previous work on image-based CAPTCHAs is that they assume *a*

priori knowledge such as image labels and semantic meanings. But in our model, the correct orientation is obtained entirely from the EXIF information of digital photos without any human intervention.

- It is hard for an automated procedure to obtain the correct orientation of a sub-image of a whole-size digital photo, because it has only partial semantic cues, which can be very difficult to learn in the machine learning model. But experiment showed that only 10% of a whole-size photo region is good enough for a human to easily and successfully provide the correct orientation with around 98% accuracy.

- The probability of random chance to break the $(9,8) - plate$ is about $1/10000$, which is low enough to be used in practice.

- Unlike the previous image-based CAPTCHA systems, we can generate more than $20 * N$ sub-images easily from N base-photos. Thus 1,000 base-photos provide more than 20,000 sub-images. Also, it is quite computationally expensive for an attacker to reconstruct a whole base-photo by gathering the sub-images used.

- The user interaction of our system is quite easy and simple, since the user chooses one from four possible orientations, taking one from four possible rotations, which can be done within 10 seconds for an 8-plate problem. So our model is quite effective, especially in a mobile environment where no keywords and control icon boxes are given.

We need to evaluate our system and rigorously compare it with the previous machine learning-based breaking tools using $k-$ plates. For practical applications, we are preparing more than 5,000 base-photos by filtering out bad images for humans and good images for computers. A web-based testing platform will be announced in the near future.

6 ACKNOWLEDGEMENT

This work was supported by the IT R&D program of MKE/MCST/IITA (2008-F-031-01, Development of Computational Photography Technologies for Image and Video Contents).

REFERENCES

[1] Prasad B. G., Biswas K. K., and Gupta S. K. Region-based image retrieval using integrated color, shape, and location index.

[2] H. Castro C. Javier and A. Ribagorda. Pitfalls in captcha design and implementation: The math captcha, a case study. pages 1–17, July 2009.

[3] Kumar Chellapilla, Kevin Larson, Patrice Simard, and Mary Czerwinski. Designing human friendly human interaction proofs (hips). In *Proc. of the SIGCHI conference on Human factors in computing systems*, pages 711–720, 2005.

[4] Monica Chew and J.D. Tygar. Image recognition captchas. In *Information Security Conference*, volume 3225, pages 268–279, 2004.

[5] Rich Gossweiler, Maryam Kamvar, and Shumeet Baluja. What's up captcha?: a captcha based on image orientation. In *Proc. of the 18th international conference on World wide web*, pages 841–850, 2009.

[6] Bell G. Huang S.Y., Lee Y.K. and Ou Z.H. A projection-based segmentation algorithm for breaking msn and yahoo captchas. In *Proc. of the International Conference of Signal and Image Engineering*, 2008.

[7] J. Howell J. Elson, JR. Douceur and J. Saul. Asirra: a captcha that exploits interest-aligned manual image categorization. In *Proc. of the 14th ACM conference on Computer and communications security*, pages 366–374, 2007.

[8] A. Singhal M. Boutell J. Luo, D. Crandall and R. Gray. Psychophysical study of image orientation perception. 16(5):429–457, 2003.

[9] Yan Jeff and El Ahmad Ahmad Salah. Usability of captchas or usability issues in captcha design. In *Proc. of the 4th Symposium on Usable Privacy and Security*, pages 44–52, 2008.

[10] P Simard K Chellapilla. Using machine learning to break visual human interaction proofs (hips). In *Advances in Neural Information Processing Systems 17*, 2004.

[11] Hong-Jiang Zhang Lei Zhang, Mingjing Li. Boosting image orientation detection with indoor vs. outdoor classification. In *Proc. of the Sixth IEEE Workshop on Applications of Computer Vision*, pages 95–99, 2002.

[12] G. Mori and J. Malik. Recognizing objects in adversarial clutter: breaking a visual captcha. In *Proc. of Computer Vision and Pattern Recognition.*, June 2003.

[13] Golle Philippe. Machine learning attacks against the asirra captcha. In *Proc. of the 15th ACM conference on Computer and communications security*, pages 535–542, 2008.

[14] B. Leung S. M. Bileschi and R. M. Rifkin. Towards component-based car detection. In *In ECCV Workshop on Statistical Learning and ComputerVision*, 2004.

[15] Lyu Siwei. Automatic image orientation determination with natural image statistics. In *Proc. of the 13th annual ACM international conference on Multimedia*, pages 491–494, 2005.

[16] V. Srikanth, C. Vishwanathan, Udit Asati, and N. Ch. Sriman Narayana Iyengar. Think-an image based captcha mechanism (testifying human based on intelligence and knowledge). In *Proc. of the International Conference on Advances in Computing, Communication and Control*, pages 421–424, 2009.

[17] A. Vailaya, H. Zhang, Changjiang Yang, Feng-I Liu, and A.K. Jain. Automatic image orientation detection. *IEEE Trans. on Image Processing*, 11(7):746–755, July 2002.

[18] Lei Wang, Xu Liu, Lirong Xia, Guangyou Xu, and Alfred Bruckstein. Image orientation detection with integrated human perception cues (or which way is up). In *Proc. of ICIP*, volume 2, pages 539–42, Sept. 2003.

[19] Yongmei Wang and Hongjiang Zhang. Content-based image orientation detection with support vector machines. In *IEEE Workshop on Content-Based Access of Image and Video Libraries*, 2001.

[20] Yongmei Michelle Wang and Hongjiang Zhang. Detecting image orientation based on low-level visual content. *Computer Vision and Image Understanding*, 93(3):328 – 346, 2004.

[21] Ming-Hsuan Yang, David J. Kriegman, and Narendra Ahuja. Detecting faces in images: A survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(1):34–58, 2002.

Feature preserving mesh smoothing algorithm based on local normal covariance

Miroslav Svub, Premysl Krsek, Michal Spanel,
Vit Stancl, Radek Barton, Jiri Vadura
{svub,krsek,spanel,stancl,barton,ivadura}@fit.vutbr.cz
PGMed@FIT
Department of Computer Graphics and Multimedia
Faculty of Information Technology
Brno University of Technology, Brno, Czech Republic

ABSTRACT

Our goal is to develop a smoothing algorithm, which would be feature preserving and simple to use without the need of extensive parameter tuning. Our method does the smoothing of vertices based on local neighbourhood character, which is modeled by a covariance matrix of neighbourhood triangle normals. The eigenvalues and eigenvectors of the covariance matrix are used for local weighting of the displacement vector of laplacian operator. This way the method is locally auto-tuned.

Keywords: smoothing, polygonal mesh, covariance, edge preserving, noise removal

1 INTRODUCTION

One of the key phases of modelling a 3D polygonal mesh is smoothing. It can substantially reduce the amount of artifacts and noise within the mesh. A rather important features of a smoothing algorithm is how it treats different mesh features. A *feature preserving algorithm* should leave **corners** and **sharp edges** untouched while smoothing and flattening the other areas of the mesh.

Different algorithms have approached this with varying success and there is no general algorithm, which works reliably in all cases. Moreover, the best algorithms often require tuning of several parameters. Our approach attempts to address these problems and offer a simple to use, yet relatively powerful smoothing method.

1.1 Related work

One of the basic approaches to smoothing is *laplacian operator* [7] which works by averaging position of the vertex with it's neighbourhood, defining a vector by the previous and average position and then moving the vertex in this direction by a fraction of the vector length. The algorithm is very simple and is easily tunable by only one parameter. The main disadvantages are volume shrinking and reducing sharp edges and corners. Therefore, an *improved laplacian smoothing* algorithm presented in [1], which improves the results of laplacian operator by pushing the smoothed vertex a bit back, thus reducing the volume shrinking effect. The performance of this algorithm can be adjusted by two parameters. Another improvement was presented in [3] and [4] which operates in alternating inward and outward diffusion of vertices in order to maintain the shape of the mesh. Again, the algorithm is controlled by two parameters. The *bilateral mesh denoising* approach from [5] and similar method from [6] has been quite successful. It is essentially a bilateral filter applied on a mesh topology and works by filtering vertex positions in directions of their normals. Adjustable filter parameters can affect the output.

1.2 Structure of the paper

In second chapter, we would like to show a matrix feature of a vertex and suggest how it can be used to improve the results of *laplacian operator*. This approach will be further described in chapter 3. In the fourth chapter, we will look at experimental results of our method on geometrical primitives and non-primitive models. Evaluation metrics will be introduced and results will be summarized. The paper will be concluded in the fifth chapter and some directions of future work will be pointed out.

2 VERTEX PROPERTIES FROM NORMAL COVARIANCE MATRIX

In this section we will describe the background for our method. The main idea lies in altering the

laplacian operator effect for each vertex type **corner**, **edge**, **flat areas**. First, let us define some basic notations. Let v denote the vertex that is being smoothed and n it's normal. Consequently let t_i and n_i denote i-th face(triangle) and normal of v's neighbourhood. Let N be a matrix with n_i forming it's columns.

Let us take a look at matrix C :

$$C = NN^T$$

We can see that this is a zero-mean covariance matrix (covariance matrix for average normal zero $(N-0)(N-0)^T$) of normals in the neighbourhood of v. This matrix has been used in [2] for construction of *quadric error metric* and was shown to have a couple of its interesting properties. We can use C as a robust local vertex feature, which reflects the character of the vertex neighbourhood. The zero-mean covariance matrix C can be interpreted as a *quadratic form* defined for v and forms a quadric centered at v. Since quadric matrices are always symmetric, we can perform it's spectral decomposition into A and X, where the columns of A are the eigenvectors of C and λ_i are the corresponding eigenvalues :

$$C = AXA^{-1}, where X = diag(\lambda_1, \lambda_2, \lambda_3)$$

Now let's take a look at the *eigenvalues* of C. We can distinguish three main cases :

- if $\lambda_1 \cong \lambda_2 \cong \lambda_3$, then there is a large variation of normals in every direction around the vertex and it can be assumed to be a **corner**.
- if $\lambda_1 \cong \lambda_2 < \lambda_3$, then there is a direction of maximum covariance of neighbourhood normals and the direction corresponds to an **edge**.
- if $\lambda_1 < \lambda_2 \cong \lambda_3$, then there are two directions of maximum covariance and the vertex lies on a **flat area**.

Let e_1, e_2, e_3 denote the *eigenvectors* of C. In the second case, the e_3 is aligned with the direction of the edge. In the third case, e_3 will be perpendicular to the ideal plane of the flat area. Let us define vertex v' which is the average of vertices from a neighbourhood of size M.

$$v' = \frac{1}{M}(v_0 + v_1 + \ldots + v_M)$$

Then $w = v - v'$ would be the vector along which the vertex would move using the *laplacian operator*.

The *laplacian operator* based smoothing method computes the new position $\hat{v}$ of v as $\hat{v} = v + a\,w$. In our method, we project w into the vector space with base vectors being the columns of A, then weight the projected vector by inverse eigenvalues in direction of each base vector and project it back to obtain the final displacement position :

$$\hat{v} = v + a AYA^{-1}w, Y = diag(\lambda_1^{-1}, \lambda_2^{-1}, \lambda_3^{-1})$$

3 USING VERTEX NORMAL COVARIANCE FOR SMOOTHING

Consider a vertex and it's zero mean local covariance matrix C. The eigenvectors of C form an orthogonal basis with the center at the current vertex. We have designed following procedure for smoothing of a current vertex v (see figure 1 graphical interpretation of the scheme in 2D) :

1. compute position difference vector w after applying the laplacian operator
2. compute eigenvalues and eigenvectors of zero mean covariance matrix for v's neighbourhood. We have been using the neighbourhood of 3 surrounding triangle layers, because if less layers are used, the covariance statistic is less robust. For example on a very noisy but otherwise flat surface, using only one surrounding layer would result in wrong C, which would resemble that of a corner vertex. However using the larger the neighbourhood, the more C approaches the correct form (that of a flat area vertex).
3. express w within the basis formed by eigenvectors of C to obtain w' expressed in new basis coordinates
4. weight the coordinates of w' by inverse eigenvalues of C to obtain w'_1
5. multiply w'_1 by the smoothing factor a, which regulates the amount of smoothing
6. express w'_1 back within the canonical basis of three dimensional space to get the final displacement vector for the current vertex

These steps are to be performed on the whole mesh. Our algorithm is also iterative, so multiple runs are often required to achieve desired result.

3.1 Improving the eigenvalue weighting by heuristic

When we look at our weighting formula for a vertex, we can see, that we are using the inverse eigenvalues of C as the weights for *laplacian operator*. This works in theory, however on real models, we have to adjust the weighting so that extreme values

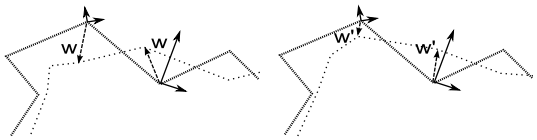


Figure 1: left : w denotes the vector to the new position of vertex, right : w' denotes the same vector after having its coordinates weightened by eigenvalues of C in the local coordinates of C's eigenvectors

and therefore extreme deformations are not permitted. Let f_w denote a function mapping a three dimensional vector onto another. Let l denote the vector of eigenvalues. Then, the smoothing formula will take the following form :

$$\hat{v} = v + aAYA^{-1}w, Y = diag(f_w(l))$$

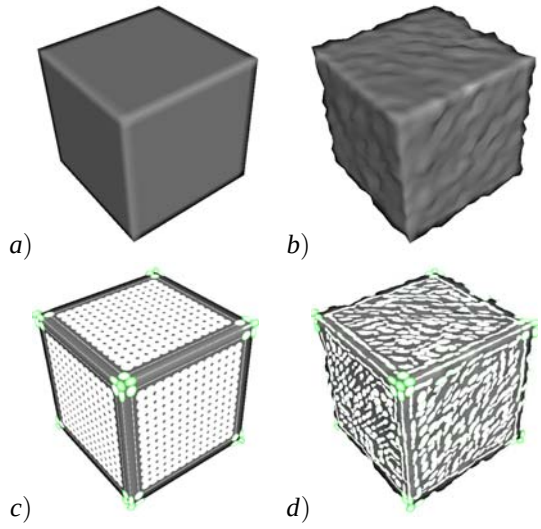


Figure 2: visualizing vertex weights a) original mesh b) noisy mesh c) covariances as degenerated elipsoids d) visualization on noisy mesh

The weighting function we have been using is a heuristic. The function is $f_w(l_1, l_2, l_3) = normalize(l)^{-\frac{2}{3}}$. The normalization of eigenvalues will ensure, the weights sum to one and the vertex displacement will not be greater than using the standard *laplacian operator*. The $-\frac{2}{3}$ power reduces the large differences between weights for a vertex.

3.2 Vizualizing vertex weights

The algorithm therefore treats directions of eigenvector differently. The smaller the weight corresponding to a direction, the stiffer the movement of this vertex in that particular direction will be. Figure 1 shall give us better idea behind the algorithm. Suggested method of visualizing C can be found in [2]. C is the main part of *quadric* - an ellipsoid centered at the current vertex and with it's principal axes aligned with the eigenvectors of C and its principal radii proportional to the inverse eigenvalues. We can use the same principle but the ellipsoid will be slightly deformed by our weighting function f_w.

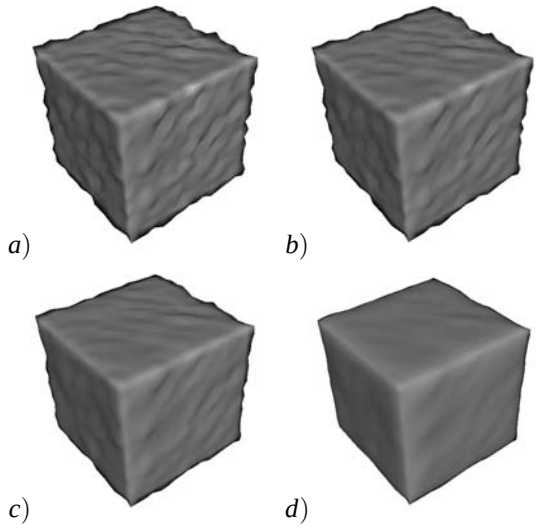


Figure 3: iterations of our algorithm on cube model a) original mesh b) iteration 1 c) iteration 3 d) iteration 7

Figure 2 shows the visualization. At the corner of the cube (green), the ellipsoid almost reduces to a sphere. This means, that the weights are close to being equal in all directions and the movement of this point should be restricted the most. At an edge vertex, the ellipsoid is reduced to elongated line, which is aligned along the direction in which the vertex can move without damaging the edge. At a flat area vertex, the ellipsoid reduces to a disc. The left set of images show a cube that is no longer regular, but we can see, that the ellipsoid characteristics still hold. Figure 3 shows the result of smoothing the noisy cube using our algorithm.

4 EXPERIMENTAL RESULTS

We have tested our method and compared its results with the algorithms presented in section 1 ([4], [5], [1]). We had a collection of 3D meshes, which include geometrical primitives (cube, sphere, cylinder, torus, etc.) to test the feature preservation properties and a collection of real models (Stanford bunny, Stanford dragon, skull model, etc). The rough data for the smoothing algorithm were obtained by adding a *gaussian noise* to the mesh. For the s^2 of the noise, we have chosen values between 0.05 to 0.1 times the mesh size. The noise was added in normal direction (see fig. 3). Each model was well tessellated.

The metric chosen for algorithm performance evaluation was histogram based. Since the original, noisy and smoothed models are topologically equivalent we have chosen following metrics :

- Histogram of *differences between normal angles of corresponding faces*. The method, which outperforms the others should have minimal amount of corresponding normal angle differences (H_N).
- Histogram of *distances between corresponding vertices*. This metric reflects the ability of the method to preserve shape of the model (H_V).

Since we are using *gaussian noise* to alter the mesh, it is safe to assume, that the angle differences and distance differences should have approximately a *gaussian distribution* with mean and deviation. After the smoothing has been performed on an altered mesh, we are going to observe that the new histogram will show, that the mean shifted towards zero and the deviation shifted towards zero as well. This should be easily observed on the histograms. We will also include a table of the mean and deviance approximations for each algorithm at the end of this section. For the methods that require parameter tuning, we have adjusted the parameters to perform as good as possible for every particular model.

4.1 Cube model analysis

The figure 4 shows the results obtained on the cube model. The $H_N(our)$ shows that our algorithm tends to minimize the larger differences better than *hc* while the majority of differences are zero (contrary to *taubin*). This shows, that our method was able outperform the others as far as edge preservation is concerned.

The figure 5 shows H_V histograms of the cube model. According to this metric, our method is outperformed by *hc* and *tabuin*. If we compare $H_V(our)$ with $H_V(bmds)$, we see that better shape preservation was achieved with our method, than with *bmds*. Figure 10 shows visual comparison of the results on the cube model.

4.2 Real model analysis

The cube model discussed above was an example from the set of ideal model with clearly distinguishable features. We have also tested our method on models that are not geometric primitives. Following histograms were obtained from the model of *the Stanford bunny*. Figure 6 shows that the shape similarity of $H_N(original)$ and $H_N(our)$ is good (see fig. 8 for visual comparison).

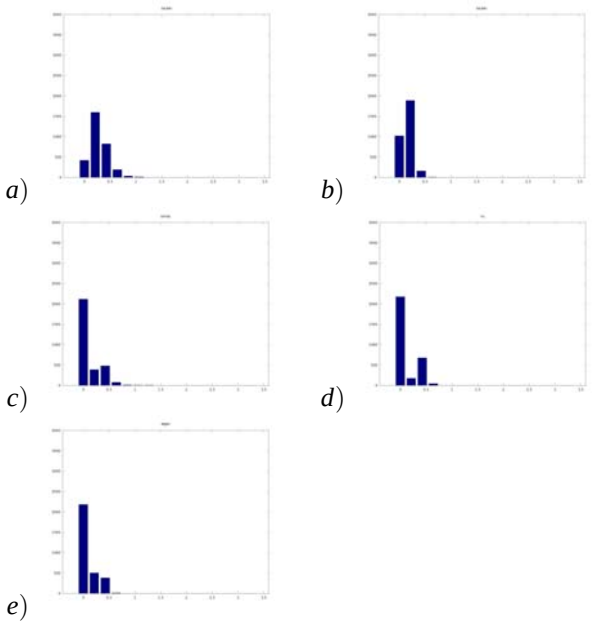


Figure 4: H_N Histogram of corresponding normal angle differences. The sixth bar corresponds to a difference of around $\frac{pi}{3}$ rad. a) $H_N(noisy)$ b) $H_N(taubin)$ c) $H_N(bmds)$ d) $H_N(hc)$ e) $H_N(our)$

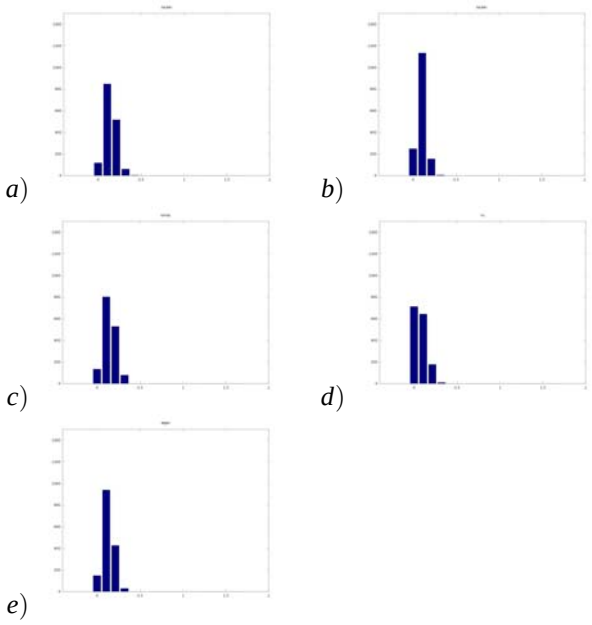


Figure 5: H_V Histogram of distances between corresponding vertices. The histogram is scaled for the maximum allowed displacement on the right (concidering the noise parameters). a) $H_V(noise)$ b) $H_V(taubin)$ c) $H_V(bmds)$ d) $H_V(hc)$ f) $H_V(our)$

The comparison of H_V histograms (figure 7) shows that the best shape preservation was achieved by *hc* algorithm. *our* method has shown

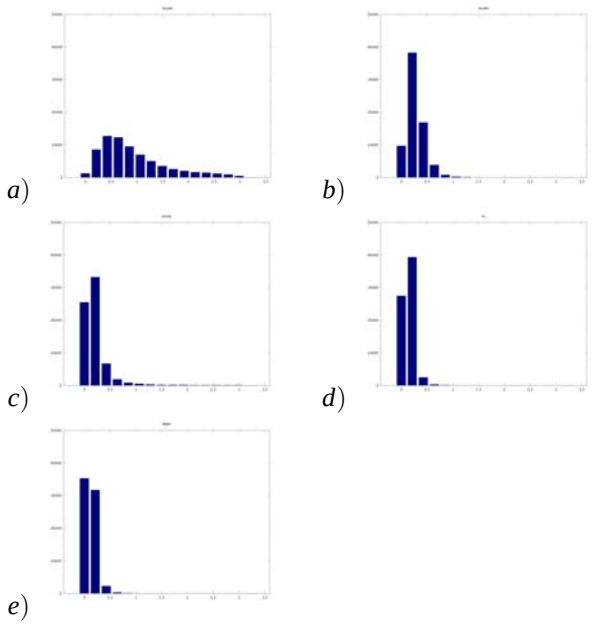


Figure 6: H_N Histogram of corresponding normal angle differences. The sixth bar corresponds to a difference of around $\frac{pi}{3}$. a) $H_N(noisy)$ b) $H_N(taubin)$ c) $H_N(bmds)$ d) $H_N(hc)$ e) $H_N(our)$

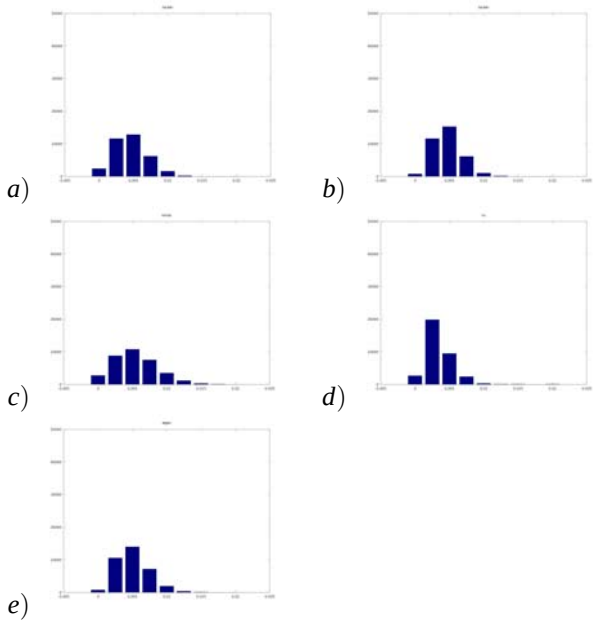


Figure 7: H_V Histogram of distances between corresponding vertices. The histogram is scaled for the maximum allowed displacement on the right (concidering the noise parameters). a) $H_V(noisy)$ b) $H_V(taubin)$ c) $H_V(bmds)$ e) $H_V(hc)$ e) $H_V(our)$

comparable results with *taubin* method and clearly outperformed *bmds*.

4.3 Summary

Tables 1 and 2 sumarize the results on several models we have used. In order to quantify the results, we have reduced the histogram metric to computing mean and standard deviation approximation (m,s^2) of the data. The key idea which was outlined at the begining of this section is, that if the original model and the smoothed one are similar, then the angles between corresponding face normals should be small and the distance between corresponding vertices should be small as well.

	taubin	bmds	hc	our
cube (m)	0.156	0.13	0.148	0.117
cube (s^2)	0.09	0.181	0.174	0.125
cylinder (m)	0.106	0.076	0.084	0.084
cylinder (s^2)	0.103	0.166	0.157	0.155
bunny (m)	0.270	0.216	0.143	0.128
bunny (s^2)	0.171	0.228	0.094	0.103
dragon (m)	0.214	0.277	0.17	0.182
dragon (s^2)	0.252	0.335	0.231	0.274
bull (m)	0.121	0.145	0.143	0.11
bull (s^2)	0.114	0.165	0.083	0.108
skull (m)	0.38	0.53	0.25	0.267
skull (s^2)	0.401	0.62	0.256	0.302

Table 1: Angles of corresponding face normals(rad)

For the models to be similar as far as feature preservation is concerned, we need the mean angle difference to be small. The edge preserving ability manifests itself through the standard deviation of angles (this can be observed on ideal meshes). For the models to be similar in shape, we expect the corresponding vertices distance to be small. Therefore the smaller average distance, the better the models correspond to each other shape-wise.

	taubin	bmds	hc	our
cube (m)	0.099	0.146	0.076	0.129
cube (s^2)	0.045	0.07	0.056	0.06
cylinder (m)	0.067	0.073	0.075	0.071
cylinder (s^2)	0.035	0.038	0.046	0.038
bunny (m)	0.046	0.053	0.034	0.05
bunny (s^2)	0.021	0.032	0.022	0.023
dragon (m)	0.031	0.044	0.033	0.031
dragon (s^2)	0.015	0.023	0.018	0.015
bull (m)l	0.572	0.41	0.356	0.389
bull (s^2)	0.255	0.2	0.19	0.192
skull (m)	0.31	0.455	0.562	0.713
skull (s^2)	0.747	0.277	0.346	0.446

Table 2: Distance between corresponding vertices

On the ideal models (cube, cylinder), the best feature preserving algorithms are *bmds* with *our* method being only slightly less effective. On real models, the best shape preserving methods are *hc* and *our* method. With the overall shape preservation, our algorithm outperforms *bmds* and is comparable with *taubin* and *hc*.

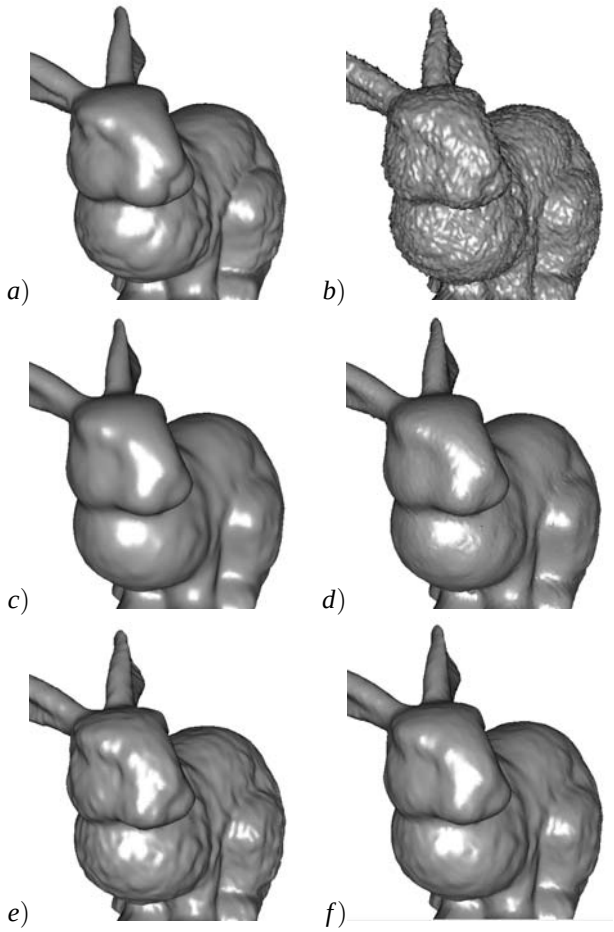


Figure 8: the stanford bunny a) *original* b) *noisy* c) *taubin* d) *bmds* e) *hc* f) *our*

5 CONCLUSION

We have been trying to develop a smoothing method, that would aim to perform well on different types of meshes and would not require extensive amount of tuning to do so. Our method is not the best performing in all cases however it's performance is rather stable and comparable to the best methods for particular case. Another advantage of our method is it's robustness, since we are using broader vertex statistics (zero mean covariance).

5.1 Future work

We would like to focus on developing a robust smoothing method based on local normal covariance. Our approach currently extends the standard *laplacian operator* by locally modifying it's results to reflect local mesh characteristics. In the future, we would like to explore following possibilities for improving the method performance :

- Explicitly clustering the vertex characteristics (eigenvalues) into categories edge, corner, flat and designing and applying custom smoothing function based on vertex type.
- Finding connectivity between vertices of the same type and using the average computed from vertices of the same type.
- Improving our zero-mean covariance metric. The matrix is constructed from a fixed size neighbourhood of the vertex. We would like to develop an adaptive approach, that selects the neighbourhood shape, so that the metric itself isn't damaged.

Another important task for the future will be evaluating the time complexity of the algorithm. Since the method uses lot of computations, this should become an important criterion. The optimization of the algorithm to be less time consuming will also be one of our proximate tasks. We would also like to explore other applications of the vertex normal covariance feature, for example using it for other mesh related tasks (matching, registration, etc.)

REFERENCES

[1] Vollmer J. , Mencel R., Muller H.: Improved laplacian smoothing of noisy surface meshes, Research report No. 711 /1999, June 1999

[2] Garland M., Heckbert, P.: Surface simplification using quadric error metrics. In: Proceedings, Siggraph 97, USA, 1997, s. 209-216

[3] Taubin G.: A signal processing approach to fair surface design. In: Proceedings of SIGGRAPH 1995

[4] Taubin G.: Geometric signal processing on polygonal meshes: Eurographics 2000 State of The Art Report(STAR), September 2000.

[5] Fleishman, S., Drori I., Cohen-Or, D. : Bilateral mesh denoising. In: Proceedings of SIGGRAPH 2003

[6] Kai-Wah, L., Wen-Ping, W. : Feature-Preserving Mesh Denoising via Bilateral Normal Filtering, In: Proceedings of 9th International Conference on Computer Aided Design and Computer Graphics 2005

[7] Field, D.A. : Laplacian Smoothing and Delaunay Triangulations, Communications in Applied Numerical Methods, Wiley, Vol 4, pp.709-712, 1988

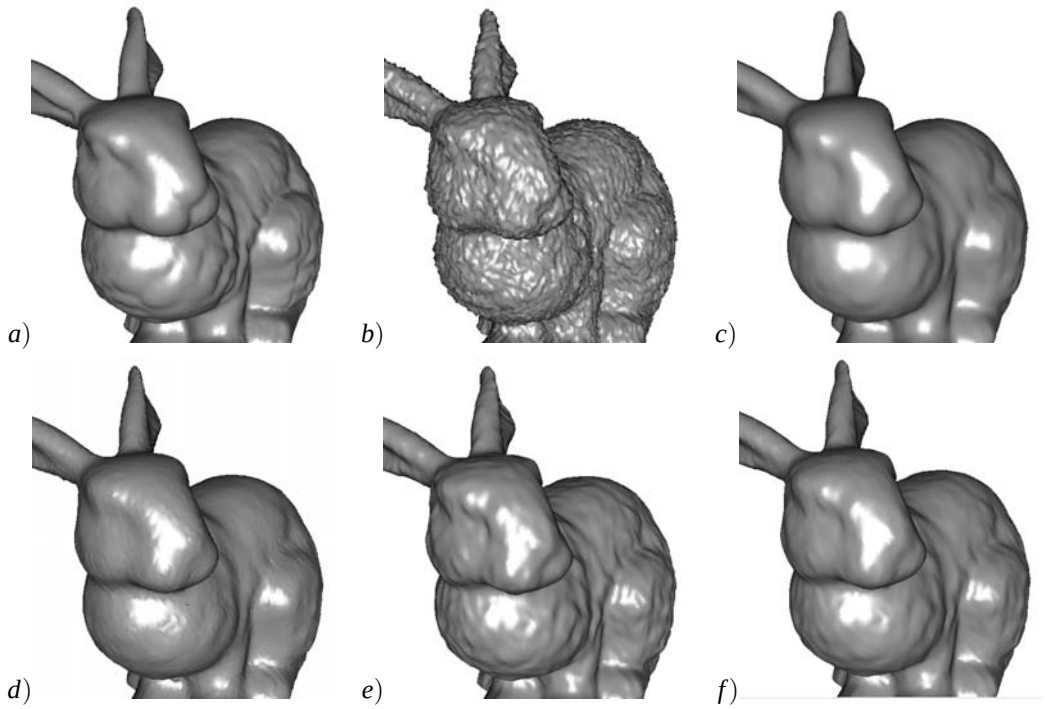


Figure 9: the stanford bunny a) *original* b) *noisy* c) *taubin* d) *bmds* e) *hc* f) *our*

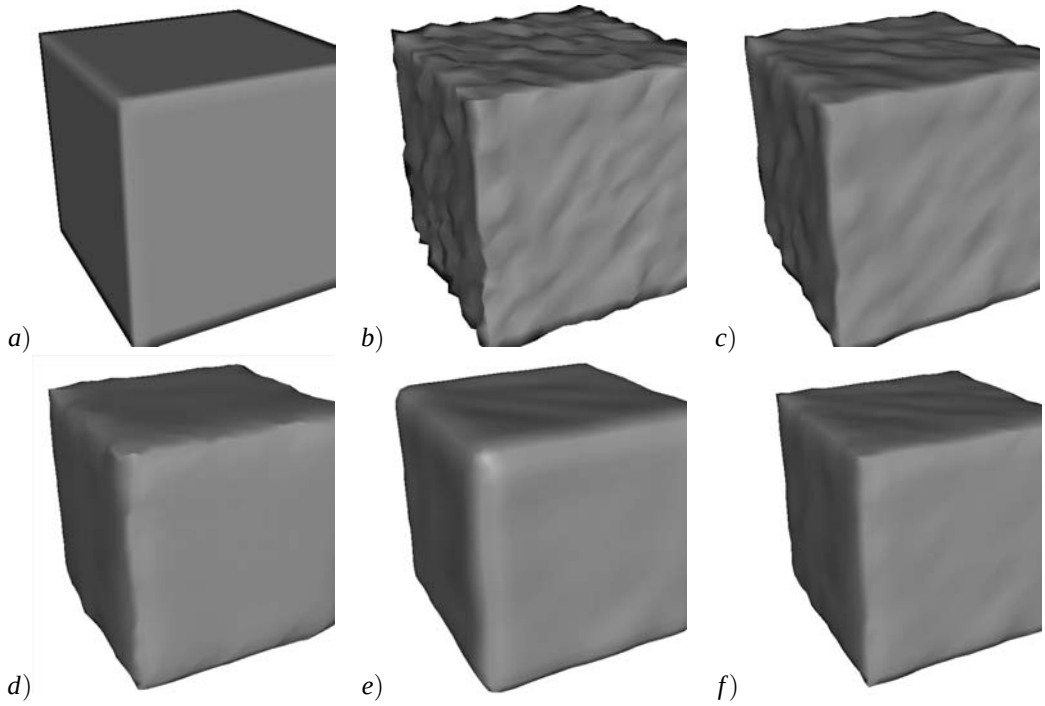


Figure 10: cube a) *original* b) *noisy* c) *taubin* d) *bmds* e) *hc* f) *our*

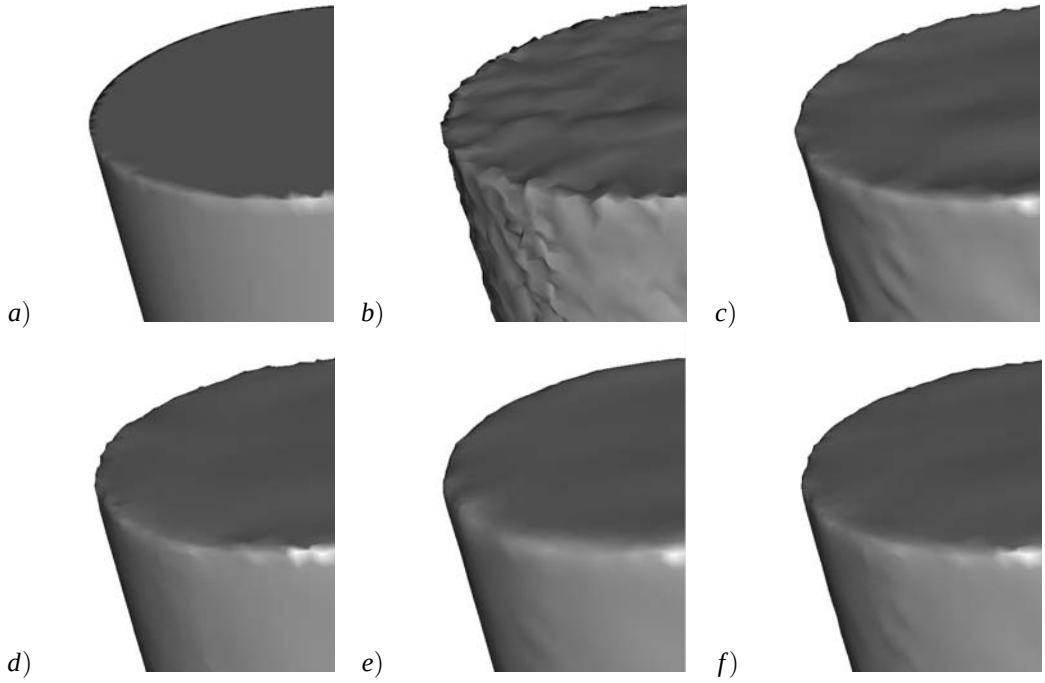


Figure 11: cylinder a) *original* b) *noisy* c) *taubin* d) *bmds* e) *hc* f) *our*

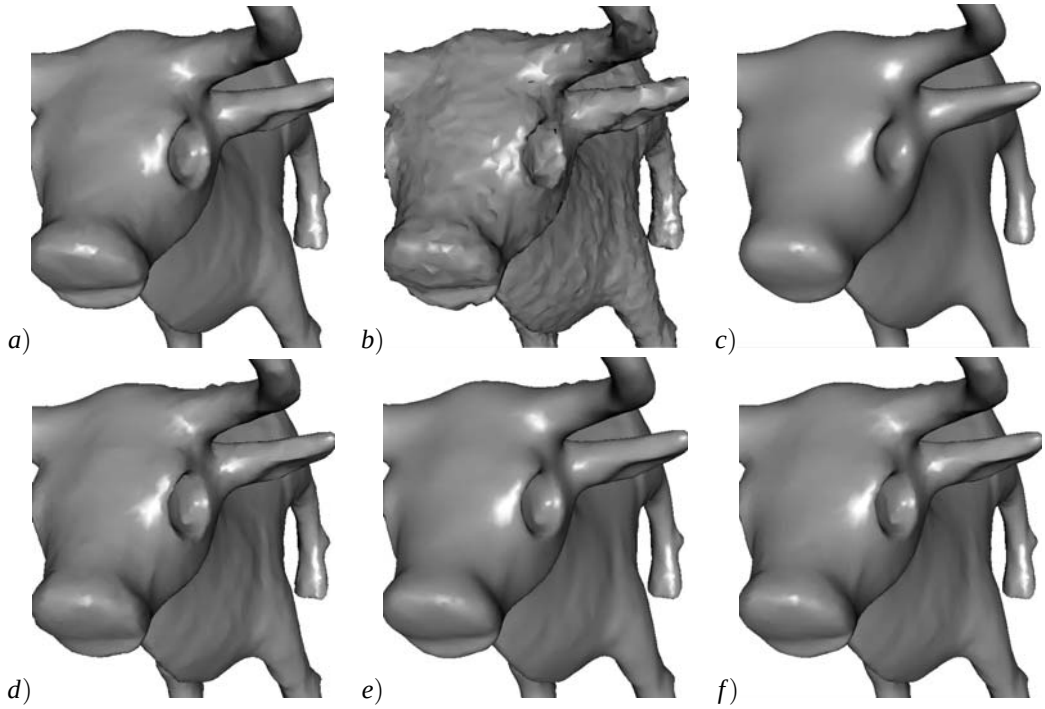


Figure 12: bull a) *original* b) *noisy* c) *taubin* d) *bmds* e) *hc* f) *our*

A Framework for Motion-Based Mesh Sequence Segmentation

Romain Arcila
romain.arcila@inrialpes.fr

S.Kartik Buddha
BUDD0001@ntu.edu.sg

Franck Hétroy
franck.hetroy@grenoble-inp.fr

Florence Denis
fdenis@liris.cnrs.fr

Florent Dupont
florent.dupont@liris.cnrs.fr

ABSTRACT

In this paper, we present a complete framework to produce motion-based geometric segmentation of mesh sequences: each segment corresponds to a mesh region that tends either to move rigidly, or to be stretched in a uniform way. We take a sequence of meshes with varying number of vertices as input and match them by pairs. This allows us to find the approximated displacement vector of each vertex of the first frame all along the sequence. Using these displacement vectors, an initial segmentation is performed on the first pair of frames, clustering vertices into static, stretched and rigidly moving regions. This segmentation is then refined after each matching. Our segmentation is computed on the fly, and lets us find the exact frame when each transformation (rotation, translation, stretch) appears. We have validated our method both on dynamic meshes and unconstrained mesh sequences.

Keywords: Mesh Sequence; Mesh Animation; Segmentation; Matching.

1 INTRODUCTION AND RELATED WORK

1.1 Introduction

In computer graphics, segmentation of surface meshes is an important and challenging problem which involves partitioning a mesh into smaller segments of homogeneous characteristics. By doing so, a simplified representation of the mesh is obtained. This representation is more meaningful and easier to analyse. Segmentation plays an important role in diverse applications like texture mapping [LPRM02], compression [KG00], mesh simplification [GWH01] and 3D shape retrieval [Zuc02] among many others. The result of the segmentation required in each case may vary depending on the type of application.

While segmentation of static meshes is a well-known topic (see [Sha08] for a survey), segmentation of mesh sequences is a more recent research field. Here "segmentation" can have two different meanings: *motion-based geometry segmentation* aims at clustering vertices or faces based on their motions while *temporal segmentation* clusters meshes of the sequence (also called *frames*) into meaningful sub-sequences. In this paper, we focus on motion-based geometry segmentation.

1.2 Related Work

Mesh sequences can be split into two categories. A *dynamic mesh* is a sequence with fixed number of vertices and fixed neighbourhood relationships between vertices. In other words, only the 3D coordinates of the vertices change through time. An *unconstrained mesh sequence* (or *time-varying mesh* or *3D video*) is represented by a sequence of meshes where the number of vertices, as well as the number of faces and connectivity, may vary along the sequence. Typically, dynamic meshes are created from modelers and animation software, while unconstrained mesh sequence are captured from video or computed from (e.g. fluid) simulation.

A few methods for motion-based geometry segmentation of dynamic meshes have been proposed [Len99, AS07]. Some of them ([dATTS08, LWC06]) segment a dynamic mesh into rigid components.

Segmentation of unconstrained sequence meshes is a difficult task since there is no one-to-one mapping between vertices of two successive frames. This mapping has to be explicitly computed for each pair of frames. Cuzzolin et al. [CMK+08] and Yamasaki et al. [LTA08] have performed such segmentations for mesh sequences, however the former computes only protrusions (extremities) segmentation while the latter uses an additional skeleton. [SC06] has been designed to solve this problem using voxel clustering, but it suffers from inherent voxelisation problems (low resolution leads to poor results while hight resolution leads to high computation time). [KSMH09, JZvK07, MHK+08] compute the mapping between any two meshes, but their methods are not designed for sequences so they do not take the temporal coherency into account. Starck et al. [SH07] computes

the matching for distant frames in a sequence, but uses additional video information. Varanasi et al. [VZBH08] tracks the displacement of vertices of the first mesh all along the sequence, but does not provide a matching for all frames.

In this article, we propose a framework (Section 2) to perform a stretched and rigid motion-based geometry segmentation of a mesh sequence, i.e. with varying connectivity and variable number of vertices. A sequence of meshes, without global topological change (the genus of all meshes of the sequence are supposed to be identical), is taken as input. A matching is performed between each pair of meshes (Section 3), which allows to extract the approximated displacement vectors and compute segmentation on the fly (Section 4). A cluster in the resulting segmentation represents a connected region (*patch*) of the 3D object which either moves almost rigidly, or is stretched in a uniform way. Results are discussed in Section 5.

2 PIPELINE OVERVIEW

In this section, we describe the complete pipeline of the segmentation process (see Figure 1). The process can be roughly split, at each time step, into 2 parts:

- matching (Sect.3).
- creation or refinement of the segmentation (Sect.4).

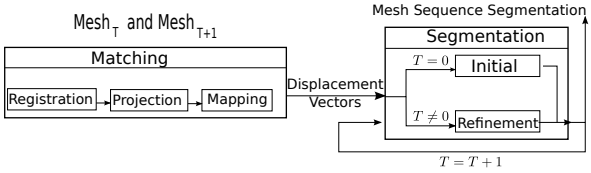


Figure 1: Overall Pipeline of the segmentation.

The algorithm takes as input a sequence of meshes with constant global topology. It registers the first mesh with the second. Displacement vectors between the vertices of the two frames are then computed using the results of the registration.

Based on the displacement vectors, an initial segmentation of the first mesh into (i) stretched (and translated), (ii) static and (iii) rigid regions is computed.

For each pair of successive frames in the sequence, we then apply the same matching process and compute the displacement vectors. The initial segmentation is then refined by analyzing the patches motion.

The segmentation of the sequence corresponds to the segmentation on the last frame, remapped on all frames of the sequence.

3 MATCHING PROCESS

The matching process consists of three successive stages: registration, projection and mapping, which are described in the following subsections. At the end of the matching process, we have the displacement vectors of the vertices of the first frame all along the sequence and the matching of vertices in frame i with vertices in frame $i+1$ for $1 \leq i < N$.

3.1 Registration

It takes as input an unconstrained mesh sequence and registers it by pair of frames with the Coherent Point Drift algorithm (CPD) [MSCP07]. CPD is a probabilistic method for non-rigid registration of point sets. This method is robust to noise and outliers. Registration is performed between the vertices of all adjacent frames - vertices in frame F_i are registered with those in frame F_{i+1} for i ranging from 1 to N-1 (see Fig 2 for the notation) to obtain registered point sets R_i which are the best alignment of the vertices in F_i with those in F_{i+1}. The number of points in the resulting registered point set R_i is equal to the number of vertices in F_i.

3.2 Projection

Each registered point set R_i obtained from the previous step is projected onto the surface of the subsequent frame F_{i+1} so as to attain a better correspondence between the adjacent frames F_i and F_{i+1}. This projection is performed in the following way:

- For each vertex on the surface of F_{i+1}, its normal vector to the surface of the mesh is estimated.
- For each point α in R_i, the vertex β which is closest to it in F_{i+1} is first searched. For each projected point, the L2 norm with all the nodes on the surface is calculated and the node with the minimum norm is taken as the closest point of that projected point on the surface.
- Finally, every point α in R_i is projected onto the surface of F_{i+1} along the normal vector of the corresponding closest point β on the surface of F_{i+1}. The projected point is denoted by P_i.

Using the above method, every point in F_i is projected onto the surface of F_{i+1} to obtain P_i for i ranging from 1 to $N-1$. The next step is to find the mapping between the vertices of the two adjacent frames.

3.3 Mapping

For sequences with constant number of vertices throughout the sequence, the correspondence between vertices of adjacent frames is easy to obtain. Every vertex in a frame has a corresponding vertex in the other frames in this case. However, such correspondences are not valid anymore when the number of vertices varies at every frame. In such cases, to establish correspondences between two adjacent frames F_i and F_{i+1}, each point in R_i is considered separately. The vertex β

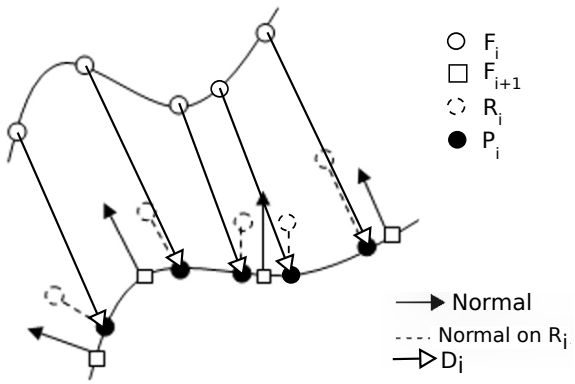


Figure 2: Matching Scheme.

in F_{i+1} which is closest to a point $R_i(\alpha)$ corresponds to the vertex $F_i(\alpha)$. In this way, a unidirectional mapping of all the vertices of F_i with the vertices of F_{i+1} can be computed to obtain F_i^{i+1} , for $i \in [1, N-1]$. Such a mapping is essential if the trajectory of a vertex in a frame has to be followed along the successive frames of the sequence.

Two possible scenarios arise when registering adjacent frames with unequal number of vertices. The first case is when the number of vertices in F_i is greater than the number of vertices in F_{i+1}. In this case, more than one point in F_i may be assigned to a vertex in F_{i+1}. On the other hand, in the case when F_i has fewer vertices than F_{i+1}, some vertices in F_{i+1} may not have any corresponding vertices in F_i. The unidirectional mapping obtained between all adjacent frames can be extended to find the mapping of the vertices in a frame in all the subsequent frames using a recursive procedure. For each vertex α in frame F_i, the index of the corresponding vertex in frame F_j is given by:

$$F_i^j(\alpha) = F_{j-1}^j((F_i^{j-1}(\alpha))) \text{ where } j \geq i \qquad (1)$$

The displacement vectors D_i for each vertex in F_i are calculated as follows: $D_i = P_i - F_i$, $i = 1 : N-1$, where the points of P_i are the projected points of F_i on the surface of F_{i+1}. Once these displacement vectors are obtained, the displacement map of each vertex in the first frame can be constructed using the correspondences from the F_i^j arrays. The displacement map is the mapping of vertices in the succeeding frames for each vertex in the first frame, i.e. the displacement map gives the corresponding vertex in any succeeding frame for each vertex in the first frame.

4 SEGMENTATION

4.1 Matching Process and Segmentation

In this section, we present our segmentation method. Our segmentation method relies on correct matching information (i.e. displacement vectors and matching points). As long as this information is provided, any matching method can be used (not only the one we described in section 3). For instance, we also test our segmentation method using [KSMH09], where the matching is computed on the embedding of the meshes using Laplacian embedding (which "unfold" mesh). Segmentation results using this matching methods are discussed in Sect.5.2.

4.2 Outline

Our segmentation is a motion-based segmentation. We cluster vertices in rigid components, but contrary to previous segmentation methods, we are also able to detect stretch motion. We are using a refinement method:

1. using the first two meshes and the associated displacement vectors, we compute an initial segmentation (section 4.4).

2. for all other frames of the sequence, we refine the previous segmentation by analyzing the current clusters and splitting them if necessary using displacement vectors and vertex matching (section 4.5).

This refinement approach allows us to:

- process long sequences with lots of vertices. We require only two meshes at a time in memory.

- track transformations. We know at which frame a cluster is created and its type of transformation.

The segmentation of the sequence is the segmentation computed on the last frame. This segmentation is then remapped on the whole sequence using the matching information (section 4.6).

4.3 Errors

In the segmentation process, we use a metric denoted as E_T to sort clusters by errors. We use the method proposed by Horn [Hor87] to compute the best possible transformation between two set of points P_i and P_j, where $card(P_i) = card(P_j)$ and points from P_i have direct correspondent in P_j. Horn's method returns a translation C_T and a rotation C_R such as $P_j = C_R * P_i + C_T$. For (P_i, P_j), E_T can be defined as either:

- Average error: $AE(P_i,P_j) = \frac{1}{card(P_i)} \sum_{p_i \in P_i} \|C_R p_i + C_T - p_j\|$.

- Max error: $ME(P_i,P_j) = \max_{p_i \in P_i} \|C_R p_i + C_T - p_j\|$.

- Difference error: $DE(P_i,P_j) = AE(P_i,P_j) - ME(P_i,P_j)$.

4.4 Initial segmentation

The initial segmentation is applied on the first mesh F of the sequence. Using the displacement vectors obtained from the matching between the first two frames, we cluster vertices based on three categories, and clustering is applied successively:

1. *static vertices*: these vertices have no motion (clusters of static vertices are denoted as $\mathscr{C}_{static}$).
2. *stretched vertices*: these vertices are in a stretched region (clusters of uniformly stretched vertices are denoted as $\mathscr{C}_{stretched}$).
3. *rigid vertices*: motion of these vertices can be described by a rigid transformation (clusters of rigid vertices are denoted as $\mathscr{C}_{rigid}$).

Clusters are computed as follows. Static vertices are defined by displacement vectors whose norms are under a threshold T_{static}, vertices with a norm greater than this threshold and with collinear (and in the same direction) displacement vectors correspond to stretched vertices. They are gathered into connected regions. Rigid vertices are harder to compute. We use a modified version of the method described in [HAWG08]. To cluster them, we define the *extended cluster* $\tilde{C}_i$ of C_i as C_i plus its n closest points (in our experiments, $n = 3$ is sufficient).
The algorithm is:

1. Initialization: each vertex is a cluster.
2. For each cluster, compute the optimal transformation [Hor87] which maps $\tilde{C}$ of first frame to second frame. Points of second frame are reconstructed from points of $\tilde{C}$ and their associated displacement vectors. Take the cluster C_i with the smallest transformation error E_t (defined in section 4.3).
3. For all clusters C_j which intersect $\tilde{C}_i$, apply the transformation found for C_i to each point of C_j. If the error is under a threshold T_{transf}, merge C_j to C_i. If no merging happens, take the cluster with the smallest error apart from C_i, and do the same. Here the error is defined as the maximum distance between the transformed point of C_j and the displaced point of C_j.
4. Repeat steps 2 and 3 while merging is possible.

As previously outlined, the order in which the clustering is performed is important. The algorithm first computes $\mathscr{C}_{static}$ and $\mathscr{C}_{stretched}$. It then builds a submesh $F_s = F \setminus (\mathscr{C}_{static} \cup \mathscr{C}_{stretched})$. Rigid clusters are computed at the end on F_s, in order to avoid overlapping of rigid regions over static and stretched ones. To avoid small clusters, i.e whose size is inferior to 3% of the mesh, due to noise or small areas around articulations, we merge them to the neighbouring cluster which minimizes the error as in step 3 of the rigid clustering process.

At the end of the initial segmentation process, the mesh of the first frame is decomposed in clusters of different types based on the displacement vectors between the two first frames. The segmentation is then refined using following frames, after each matching.

4.5 Refinement

The refinement aims at finding new regions which may appear on a new frame. Each cluster is decomposed (if necessary) into sub-clusters. The process starts from the clusters found at the previous step, i.e. the initial segmentation for frame 2, or the previous refinement in the other cases. It also uses the displacement vectors computed during the matching process. The refinement is done in two steps:

- check previous clusters.
- find new clusters.

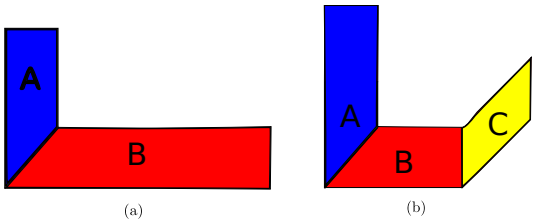
A quick checkout step is used to avoid to re-create the same cluster. We check that:

- for static clusters, all displacement vectors norms in a cluster are below the threshold T_{static};
- for stretched clusters, displacement vectors are still collinear and with the same direction;
- for rigid clusters, the best transformation between the points of cluster C_i at frame F and frame $F+1$ is valid. This transformation is computed using Horn's method [Hor87] (applied to points reconstructed from the previous frame and from the displacement vectors) If the associated error is above T_{transf}, the transformation is considered as not valid.

All clusters which are still valid are kept unchanged. All other clusters correspond to regions where new transformations are starting. These clusters are decomposed into sub-clusters:

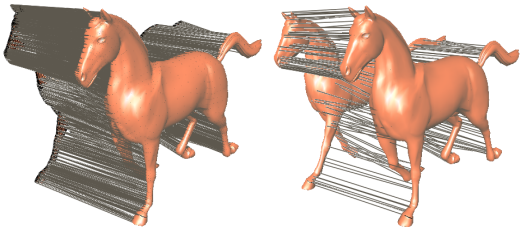
- If the new transformation applies to a whole cluster, this cluster is kept and not decomposed.
- Else, this cluster is decomposed.

This decomposition is similar to the one applied in the initial segmentation: for each cluster C_c of the list, the algorithm computes static vertices, stretched vertices and rigid vertices. As before, static and stretched subclusters are computed first and added to the sets of static $\mathscr{C}_{static}$ and stretch $\mathscr{C}_{stretch}$ clusters. $F_s = F \setminus (\mathscr{C}_{static} \cup \mathscr{C}_{stretch})$ is built. Contrary to initial segmentation, the rigid clustering is not applied on F_s, but on $F_s \cap C_c$, to make sure that the transformation applies on C_c. These clusters are added to the set of rigid clusters (See Figure 3).



(a) (b)

Figure 3: Refinement example: (a) Segmentation at time F. (b) Segmentation at time $F+1$. Cluster B is refined into sub-cluster B and C because of the rotation.



(a) (b)

Figure 4: Matching result: (a) Full matching, (b) Partial matching.

4.6 Remapping

Once all frames have been segmented, we report the segmentation of the last frame on the whole sequence using matching information. As some points can be unmatched, when we report the segmentation from one frame to the other, we apply a region growing method to avoid holes in the segmentation. For each unmatched point p_i, we search for the cluster C which contains the maximun number of points in p_i neighbourhood and set p_i in C.

5 RESULTS AND DISCUSSION

5.1 Matching

Matching results are shown in Figure 4 where the correspondences of vertices between different frames of the horse gallop sequence are displayed. The displacement vectors of these corresponding vertices in the succeeding frames are then used in the segmentation process. On a standard personal computer (core 2 duo, 2.33 GHz), matching of one mesh (8199 vertices) with another vertices (7254) takes 13 minutes. The implementation is performed with MATLAB using a fast implementation of the CPD algorithm based on a fast Gaussian transform. In the matching process, the registration of adjacent frames is the major bottleneck as compared to the projection and mapping parts.

5.2 Segmentation

Setting of Thresholds and Errors

To handle noise present on unconstrained mesh sequence, we use the threshold T_{static}. For the decimated horse sequence (see Fig.8), we use a threshold of 1×10^{-4}, the bounding box of the mesh is 12.3177 . The threshold T_{transf} corresponds to the motion amplitude tolerated for a cluster on dynamic mesh between two frames. For unconstrained mesh sequence, this threshold should incorporate the slight error of the projected points which contain a small error contrary to models created in a modeler. Segmentation can be refined using different values for T_{transf}: small values produce more clusters as shown in Fig.6. This threshold does not depend on vertex density but depends on the amplitude movement between 2 frames.

In our experiments, the Difference Error should be preferred when dealing with dynamic meshes. Indeed, Difference Error avoids to merge clusters with large errors. Clusters with large errors in dynamic meshes are due to change in transformation. On presence of noise and outliers, the Difference Error is too sensible and Average Error becomes the best error type: large error can result from outliers/noise in clusters. Such cluster should not be discarded if it has few outliers. Fig.5 shows the different errors applied on a dynamic mesh and on a dynamic mesh perturbed with some noise (noise is applied independently on each frame).

Results

Fig.6 shows the result of our algorithm applies to a dynamic mesh compared to [LWC06] and [dATTS08]. As can be seen, our method yields similar results. The main difference with [LWC06] deals with cluster's boundaries: they are not smooth. We do not apply a post-processing stage as boundaries are not clearly defined on an unconstrained mesh sequence (see next section for a discussion on boundaries).

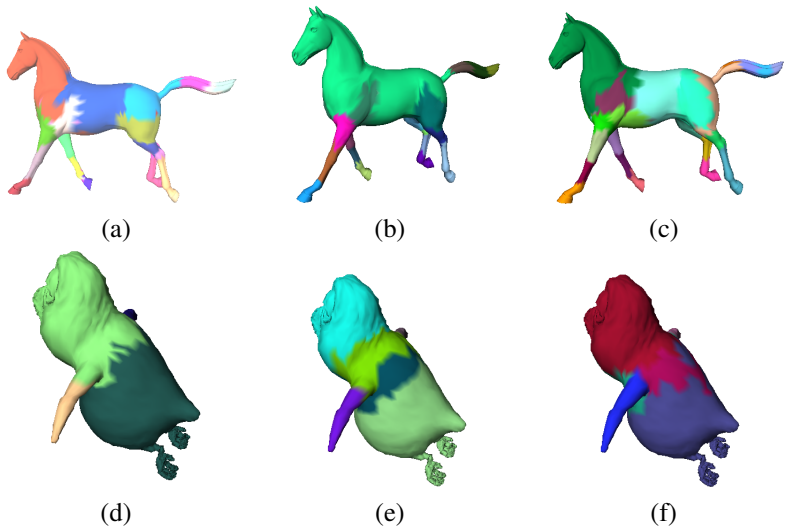
Contrary to previous methods, we handle stretch deformations. We show some results of such clustering on Fig.7. Some parts of the object follow a rigid transformation while others are stretched.

Our algorithm is able to segment unconstrained mesh sequences as shown in Fig.8. Each frame of the horse sequence is randomly decimated, and we segment this sequence. The number of vertices varies from 2624 to 4696. Segmentation is similar to the dynamic one.

Discussion

Our whole process deals with two frames at a time, meaning:

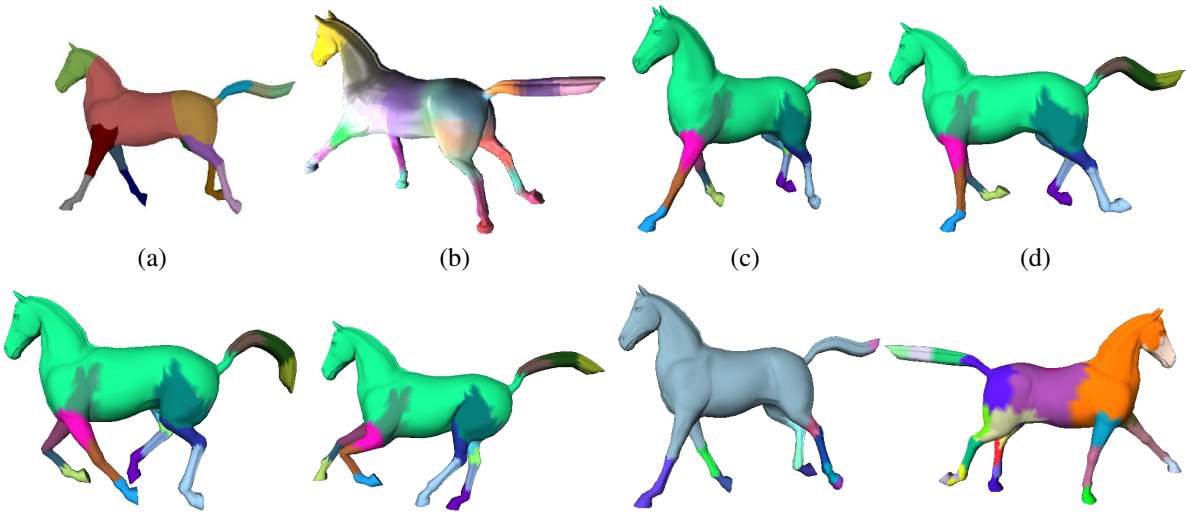
- mesh sequences with a high number of frames and/or vertices can be processed.
- the apparition and the decomposition of each cluster is controlled.



(a) (b) (c)

(d) (e) (f)

Figure 5: Comparison of error type on segmentation result. (a) Average Error, (b) Difference Error and (c) Max Error on dynamic mesh. (d) Average Error, (e) Difference Error and (f) Max Error on noisy mesh.



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

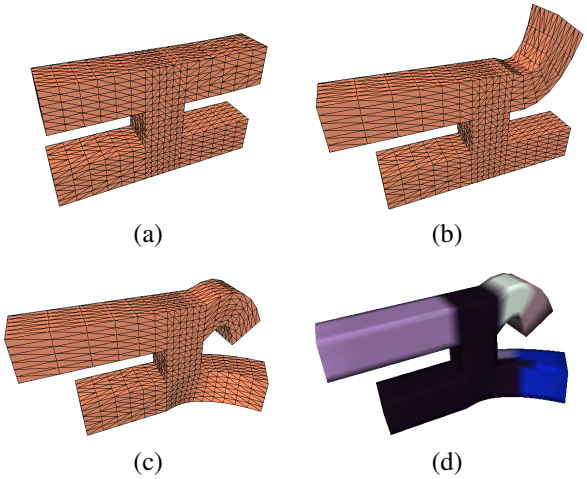
(e) (f) (g) (h)

Figure 6: Horse Segmentation: (a) [LWC06], (b) [dATTS08], (c) with our method. (d,e,f) Frames of the sequence. (g) Coarse, $T_{transf} = 0.0015$ and (h) refined, $T_{transf} = 0.0007$ result, to compare with (c), $T_{transf} = 0.0009$.

Boundaries There are two closely related issues on cluster boundaries: they are not smooth and boundaries of clusters on articulation are not necessarily on the middle of the articulation as with other methods. They can be around, a little bit after or before (see Fig.9 for illustration). Clusters are merged while the transformation (see section 4.4) found can be applied, this transformation becomes invalid after/before the boundaries. As we deal with meshes with varying connectivity, boundaries move from one frame to another. As we want to work on these original sequences (i.e with no modification), this is acceptable. Otherwise, a post-processing step can correct this problem by inserting points on boundaries in all frames.

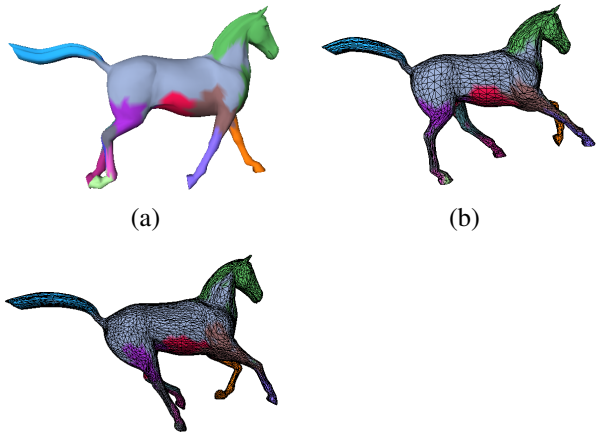
Noise and outliers The presence of some outliers or badly matched points is similar to noise. Usage of average error merge them with other vertices. The remaining outliers are on their own cluster, and these clusters are smalls. This means that they are handled by the forced merge of small clusters. However, in presence of many grouped outliers, our algorithm produces incorrect results, as they are clustered together (but the segmentation is coherent with the matching information). It also produces incorrect results when many badly matched points are present: it forces user to select large threshold value. This means that it can produce only a coarse segmentation. In this case the segmentation is correct on the first frame, but it is incorrectly remapped on following frames.

Errors coming from matching and outliers which cause a cluster subdivision at F_i are passed on all



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

Figure 7: Stretch: (a), (b), (c) frames of the sequence. (d) segmentation.



(a) (b) (c)

Figure 8: Segmentation on a decimated horse sequence. (a) Result. (b,c) Minimum and maximum number of vertices.

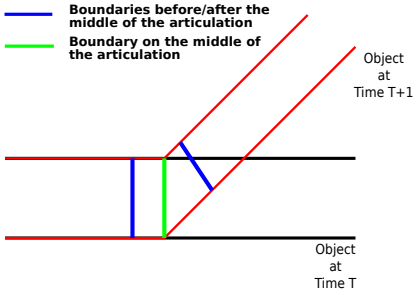


Figure 9: Boundaries on articulation.

following frames. This is due to the nature of our algorithm.

Matching Methods We have chosen to use projected points to obtained the displacement vectors. The consequence is that we have "real" displacement vectors, but with an expensive computation cost. Using an algorithm as in [MHK+08], which does not provide projected points, we have to use the matched point to compute the displacement vectors. This implies an error which is more important than with our method. On a dense mesh, this is not be important, as matched point are close to the projected one. However, on sparse meshes, the error can be big enough to either:

- give incorrect result.
- force user to use large T_{transf} (and obtained a coarse segmentation).

Implementation The prototype for the segmentation part is written in C++ using CGAL, while the matching part is written in Matlab.

Future Work The proposed method can be improved in several ways:

- we did not test our method with objects facing global topological changes. We do not see any trouble in handling them: for each pair of successive frames, detect topological changes between them and split clusters concerned by the topological change of the object;
- The algorithm currently works using two successive frames, which means that we can miss slow motions: if the deformation between two frames is too small, the checkout on the refinement process will always be true and the algorithm will not refine the cluster. The threshold T_{transf} limits this problem but working with a small time window (as it can easily fit in memory) can improve this. We have some preliminary results, but the main difficulty consists in automatically detecting the size of this time window. Time window may help to correct invalid cluster subdivision by inspecting following frames.
- Detect new motion type such as twist.

The main objective of our next work is to define a validation framework, using user validation and defining error criteria to validate mesh sequence segmentation [BVLD09, CGF09].

6 CONCLUSION

We have presented a complete framework to segment mesh sequences with varying number of vertices in rigid and stretch components. Our segmentation method works on the fly and allows us to track deformation all along the sequence. Extension to handle objects facing global topological changes, time window to segment slow motion and temporal matching to take advantage of temporal coherency are planned for the near future. This method can be integrated in an automatic production chain, however the threshold needs to be determined manually before. There is no limit to the number of vertices or frames that can be treated by our method, but since errors are passed from frame to frame, long sequence should be splitted into shorter ones.

7 AFFILIATIONS

Romain Arcila:[1],[2],[3],[4]
S. Kartik Buddha,Florence Denis, Florent Dupont:[1],[2]
Franck Hétroy:[3],[4]
[1] Université de Lyon, CNRS
[2] Université Lyon 1, LIRIS, UMR5205, F-69622, France
[3] Université de Grenoble & CNRS, Laboratoire Jean Kuntzmann, Grenoble, France
[4] INRIA Grenoble - Rhône-Alpes, Grenoble, France

8 ACKNOWLEDGMENT

This work is partially supported by the ANR (Agence Nationale de la Recherche) through MADRAS project (ANR-07-MDCO-015).
A static mesh and mesh sequence repository can be found at http://www-rech.telecom-lille1.eu/madras/

REFERENCES

[AS07] R. Amjoun and W. Straßer. Efficient compression of 3d dynamic mesh sequences. In *Journal of the WSCG*, 2007.

[BVLD09] Halim Benhabiles, Jean-Philippe Vandeborre, Guillaume Lavoué, and Mohamed Daoudi. A framework for the objective evaluation of segmentation algorithms using a ground-truth of human segmented 3D-models. In *IEEE International Conference on Shape Modeling and Applications (Shape Modeling International 2009)*, Beijing, China, June 26-28 2009. short paper.

[CGF09] Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), August 2009.

[CMK+08] F. Cuzzolin, D. Mateus, D. Knossow, E. Boyer, and R. Horaud. Coherent laplacian 3-d protrusion segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[dATTS08] E. de Aguiar, C. Theobalt, S. Thrun, and H.P. Seidel. Automatic Conversion of Mesh Animations into Skeleton-based Animations. volume 27, pages 389–397, 2008.

[GWH01] M. Garland, A. Willmott, and P.S. Heckbert. Hierarchical face clustering on polygonal surfaces. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 49–58, 2001.

[HAWG08] Q. Huang, B. Adams, M. Wicke, and L.J. Guibas. Non-rigid registration under isometric deformations. *Computer Graphics Forum*, 27(5):1449–1457, 2008.

[Hor87] B.K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4(4):629–642, 1987.

[JZvK07] V. Jain, H. Zhang, and O. van Kaick. Non-rigid spectral correspondence of triangle meshes. *International Journal on Shape Modeling*, 13(1):101–124, 2007.

[KG00] Z. Karni and C. Gotsman. Spectral compression of mesh geometry. In *Siggraph 2000, Computer Graphics Proceedings*, pages 279–286, 2000.

[KSMH09] David Knossow, Avinash Sharma, Diana Mateus, and Radu Horaud. Inexact matching of large and sparse graphs using laplacian eigenvectors. In *GbRPR '09: Proceedings of the 7th IAPR-TC-15 International Workshop on Graph-Based Representations in Pattern Recognition*, pages 144–153. Springer-Verlag, 2009.

[Len99] J.E. Lengyel. Compression of time-dependent geometry. In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 89–95, 1999.

[LPRM02] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. In *ACM SIGGRAPH conference proceedings*, pages 362–371, 2002.

[LTA08] N.S. Lee, T.Yamasaki, and K. Aizawa. Hierarchical mesh decomposition and motion tracking for time-varying-meshes. In *ICME*, pages 1309–1312, 2008.

[LWC06] T.-Y. Lee, Y.-S. Wang, and T.-G. Chen. Segmenting a deforming mesh into near-rigid components. *Vis. Comput.*, 22(9):729–739, 2006.

[MHK+08] Diana Mateus, Radu P. Horaud, David Knossow, Fabio Cuzzolin, and Edmond Boyer. Articulated shape matching using laplacian eigenfunctions and unsupervised point registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[MSCP07] A. Myronenko, X. Song, and M. Carreira-Perpinan. Non-rigid point set registration: Coherent point drift. In *Advances in Neural Information Processing Systems (NIPS) 19*, pages 1009–1016. 2007.

[SC06] A. Sundaresan and R. Chellappa. Acquisition of articulated human body models using multiple cameras. *Lecture Notes in Computer Science, Proceedings of AMDO*, 4069:78–89, 2006.

[SH07] J. Starck and A. Hilton. Correspondence labelling for wide-timeframe free-form surface matching. In *ICCV*, pages 1–8, 2007.

[Sha08] A. Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008.

[VZBH08] K. Varanasi, A. Zaharescu, E. Boyer, and R.P. Horaud. Temporal surface tracking using mesh evolution. In *Proceedings of the Tenth European Conference on Computer Vision*, volume Part II, pages 30–43, 2008.

[Zuc02] E. Zuckerberger. Polyhedral surface decomposition with applications. *Computers and Graphics*, 26(5):733–743, 2002.

Histograms of Oriented Gradients for 3D Object Retrieval

Maximilian Scherer
Interactive Graphics Systems Group
TU Darmstadt, Germany
maximilian.scherer@gris.tu-darmstadt.de

Michael Walter
Interactive Graphics Systems Group
TU Darmstadt, Germany
michael.walter@gris.tu-darmstadt.de

Tobias Schreck
Interactive Graphics Systems Group
TU Darmstadt, Germany
tobias.schreck@gris.tu-darmstadt.de

ABSTRACT

3D object retrieval has received much research attention during the last years. To automatically determine the similarity between 3D objects, the global descriptor approach is very popular, and many competing methods for extracting global descriptors have been proposed to date. However, no single descriptor has yet shown to outperform all other descriptors on all retrieval benchmarks or benchmark classes. Instead, combinations of different descriptors usually yield improved performance over any single method. Therefore, enhancing the set of candidate descriptors is an important prerequisite for implementing effective 3D object retrieval systems.
Inspired by promising recent results from image processing, in this paper we adapt the Histogram of Oriented Gradients (HOG) 2D image descriptor to the 3D domain. We introduce a concept for transferring the HOG descriptor extraction algorithm from 2D to 3D. We provide an implementation framework for extracting 3D HOG features from 3D mesh models, and present a systematic experimental evaluation of the retrieval effectiveness of this novel 3D descriptor. The results show that our 3D HOG implementation provides competitive retrieval performance, and is able to boost the performance of one of the best existing 3D object descriptors when used in a combined descriptor.

Keywords: 3d model, retrieval, hog, gradient, histogram

1. INTRODUCTION

The increasing number of available 3D models is accompanied by the need for ever more sophisticated methods to retrieve those models. There are several ways to tackle this task[16]. One prominent approach, which we also adhere to, is the computation of high-dimensional *feature vectors* to describe the *global shape* of 3D models. The (dis-)similarity between two models can then be expressed by the distance between two feature vectors. So under these conditions, the remaining task at hand is the extraction of feature vectors, that are able to capture and effectively discriminate the shape of 3D models. Note that for the remainder of this paper, we will use the terms feature vector and descriptor interchangeably, as feature vectors are the only descriptors we consider.

In the past, several methods that have proven to be effective in image analysis were successfully adapted to the task of 3D object retrieval and classification[5, 18]. Intuitively, this can be explained by the fact that the human perceptual system judges *similarity* of 3-dimensional shapes to a large extent based on 2-dimensional projections. This strongly motivates our approach of adapting the successful "**H**istograms of **O**riented **G**radients"[6] feature computation from 2D images to 3D models.

The contribution of this paper is twofold. First, we extend the HOG descriptor to 3D mesh models, evaluate its performance, and make the implementation publicly available. Second, we provide a framework for the extraction of different descriptor instances based on the gradient computation method employed. By choosing a method to compute gradients from meshes, our implementation may be easily adapted to compute corresponding feature vectors (see Section 3 for details).

The remainder of this paper is structured as follows. After discussing related work in Section 2, we discuss the HOG descriptor in Section 3. Specifically, the definition of HOG in the 2D image domain is reviewed, as well as our approach to apply it in the 3D domain is discussed. The main challenge of defining suitable gradients on 3D meshes is considered in detail. Section 4 demonstrates the effectiveness of our descriptor by evaluating it on several well-known 3D retrieval benchmarks. A detailed discussion of the obtained results is provided in Section 5. Section 6 concludes and outlines options for future work in this area.

2. RELATED WORK

There is a wealth of available approaches for 3D model retrieval [16], which can be classified into several different categories. In terms of the addressed retrieval problem, methods can be distinguished by supporting global or partial model similarity. Global methods determine the overall similarity of the shape of whole models, while partial methods analyze for local similarities. Note that in both cases, as similarity is a rather fuzzy and domain-dependent concept, there are no absolute solutions to neither the global nor the local 3D similarity problem. Consequently, all methods proposed so far in 3D object retrieval are of heuristic nature, and their usefulness for solving a specific retrieval task needs to be evaluated experimentally.

In our work, we consider the global 3D similarity problem. Our approach compares the global shape of 3D models by relying on histograms of 3D *gradients*. Gradient is not used in a strict mathematical sense throughout this paper, but rather to describe a notion of *intensity* at a given 3D coordinate in a certain direction (see Section 3). Therefore, our approach relates to histogram-based and gradient-based shape description methods.

A histogram-based approach is proposed by Zaharia et al in [21], called *Shape Spectrum Descriptor*. There, the histogram contains values based on the curvature of several points on the model surface. We will briefly come back to such an approach when describing future work (Section 6). In [1], the authors decompose the object space into cells and compute a global *3D Shape Histogram*, where the frequency of 3D points that fall into a given cell form one bin in the shape histogram. A recent approach that relies on surface normals called *Multishell Extended Gaussian Images* was proposed by Wang et al in [4]. In contrast to our approach, these normals are not encoded in histograms. Rather, they are mapped onto the unit sphere and this spherical function is then expanded into spherical harmonics. The resulting coefficients form the basis for this descriptor.

In [8], Glomb extends the Harris-Operator[9] to 3D meshes. He faces similar challenges as we do, since corner detection using the Harris-Operator relies on the aggregated value of gradients in a specific region. Accordingly, he proposes several methods to define gradients on meshes and also evaluates them. We will come back to his results later (see Section 6).

Our approach utilizes Euclidean distance fields for gradient-computation, which is a well-studied field with many differing applications. For further details please refer to [7, 11].

Due to the heuristic nature of the definition of 3D shape descriptors, experimental evaluation of their retrieval performance is important. Several 3D retrieval benchmarks have been established to date. The benchmarks each consist of a set of 3D mesh models along with a classification of models into similarity classes (ground truth). Generic benchmarks such as the Princeton Shape Benchmark [14] or the Konstanz 3D Shape Benchmark [3] contain generic models such as humans, cars, animals, furniture etc. More specialized benchmarks containing e.g., mechanical engineering models [10] or furniture models [20] exist. These benchmarks can be used to compute metrics such as Precision and Recall to compare the retrieval performance of individual 3D descriptors.

3. 3DHOG DESCRIPTOR

In this section, we first recall the 2D HOG algorithm, and then introduce the proposed concept to adapt it to the 3D domain. Alternatives for calculation of 3D gradients from a distance field representation of 3D mesh models are discussed. An implementation of the 3DHOG descriptor is described and made available.

3.1 The HOG Descriptor for Images

In the 2D image domain, successful methods like Harris-Corner detection [9] and the well-known Scale-Invariant-Feature-Transform (SIFT) algorithm [12] rely on aggregated gradients. The HOG descriptor organizes gradients into histograms. As the first step, the gradient image is computed by convolving the input image with an appropriate filter mask. Then, a grid of histograms is constructed, where each histogram organizes the respective gradients into bins according to their orientation. Each gradient votes into the corresponding bin using its length. To preserve locality, such a histogram is computed for each cell in an evenly spaced grid. Accordingly, each cell contains the same number of gradients (depending on the cell size), and gets assigned exactly one histogram. The cells themselves are then organized in rectangular blocks, which might even overlap. The histogram values of all cells within one block are concatenated to form a vector. The vector of each block is then normalized and subsequently, the concatenation of all those block-vectors yields the final feature vector. Note that the order of concatenation is arbitrary but fixed, hence an entry of the resulting feature vector always contains information about the same image region (the *block* it originates from). Further details and an evaluation of the 2D HOG descriptor can be found in [6].

3.2 Extending HOG to 3D Meshes

Extending the HOG approach to 3D mesh models mainly consists of two steps. First, we need to extract *gradients* from our mesh. Second, we need to organize these 3-dimensional gradients into bins using appropriate histograms computed over uniformly

spaced grid-cells. The second step is straightforward, as we simply extend the grid and histogram dimension each by one. Then, we can convert each gradient into spherical coordinates (eq. 1) and bin it according to its orientation (*zenith* $\theta \in [0,\pi)$ and *azimuth* $\phi \in [0,2\pi)$).

$$\begin{pmatrix} \theta \\ \phi \\ r \end{pmatrix} = \begin{pmatrix} \arccos \frac{z}{\sqrt{x^2+y^2+z^2}} \\ \text{atan2}(x,y) \\ \sqrt{x^2+y^2+z^2} \end{pmatrix} \quad (1)$$

The first step, however, does not generalize as easily.

To compute the image-gradient, several approaches are considered in [6]. According to their evaluation, the convolution of the image with a 1D $[-1,0,+1]$ filter mask is most suitable. This approximates the partial first-order derivative according to:

$$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)^T \quad (2)$$

$$\approx \begin{pmatrix} f(x+1,y) - f(x-1,y) \\ f(x,y+1) - f(x,y-1) \end{pmatrix} \quad (3)$$

To extend this to 3D, we need a pre-processing step to compute a notion of *neighborhood* and *intensity* from a polygonal mesh. We do this by computing a 3-dimensional *Euclidean distance field* as a real valued function defined on a discrete, regular 3D grid spanning the entire volume the mesh occupies. We will denote grid-cells as *voxels*. Each voxel contains the distance from its center to the surface of the mesh. To formalize this statement, let us define a distance function $f : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \mapsto \mathbb{R}$, which maps each voxel, defined by its indices $x,y,z \in \mathbb{N}$, to its corresponding value:

$$f(x,y,z) = \min_{\mathbf{x} \in \Sigma} \|\mathbf{x} - center(x,y,z)\|_2, \quad (4)$$

where Σ is the set of all points on the surface of the mesh and $center : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \mapsto \mathbf{p}$ is a function that returns the center coordinate of a given voxel.

Once the distance field is computed, the gradient computation can be carried out analogously to the gradient computation on images by convolving the distance field with the $[-1,0,+1]$ filter mask in three dimensions.

Note that the distance field strongly depends on position and size of the object. This demands normalization of the mesh prior to distance field computation. We rely on established methods for mesh normalization. Specifically, we provide translation invariance by moving the center of mass of the mesh to the origin; we provide scale invariance by scaling the mesh into the unit cube. Finally, we normalize for rotation using a weighted PCA analysis. For details on these normalization steps, see [18].

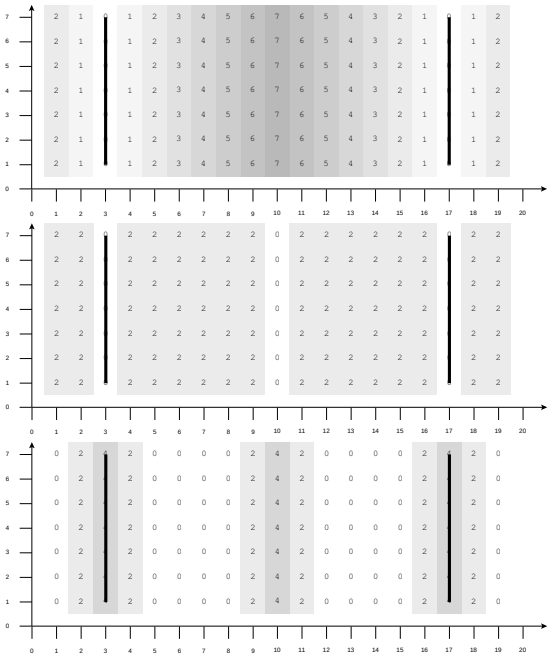


Figure 1: 2D illustration of a distance field and derived gradient fields for a simple structure consisting of vertical bars. The top image shows the distance field. The gradient field using the $[-1,0,+1]$ mask is shown in the middle image, while the bottom image shows the gradient field using the $[1,0,-2,0,1]$ mask.

3.3 Alternative Gradient Definition

While distance fields are very convenient for adopting HOG to the 3D domain, we found the first-order gradients approximated by the simple edge detector described above not to be the best option for our purposes. To demonstrate this, we visualize a simple, 2-dimensional distance field (see figure 1, top). The mesh in this case only consists of two walls. The respective gradient field computed using the $[-1,0,+1]$ convolution mask is shown in the middle. The numbers and grey-scales represent the magnitude of the gradients. Note that the gradients are mainly distributed among the empty space and disappear at the local extrema of the distance field, in particular at the local minima, which represent the walls in the distance field. Intuitively, this does not provide a proper analogy to gradients in images. A pixel contains information about the reflected light at a certain point in the world. Accordingly, the gradient reveals abrupt changes in intensity and an image of walls for example, would certainly exhibit strong gradients near the edges of the walls. To elaborate this aspect, we consider a different definition of *gradient*. Usually, the first order derivatives in each dimension (see equation 2) constitute the gradient. To compute gradients from the 3-dimensional distance field, we propose to use the second order derivative in each dimension. This modified gradient can be approximated by applying the simple edge detector twice, which results in the same gradients, which a convolution with a $[1,0,-2,0,1]$

mask would have yielded (see equation 5). At the bottom, figure 1 shows the resulting gradient field when computed with the $[1,0,-2,0,1]$ convolution mask. Such a gradient field provides a proper analogy to image-gradients. We further elaborate on this in Section 5.

$$\nabla f(x,y,z) = \left(\frac{\partial f}{\partial x^2}, \frac{\partial f}{\partial y^2}, \frac{\partial f}{\partial z^2}\right)^T \quad (5)$$

$$\approx \begin{pmatrix} f(x-2,y,z)+f(x+2,y,z)-2\cdot f(x,y,z) \\ f(x,y-2,z)+f(x,y+2,z)-2\cdot f(x,y,z) \\ f(x,y,z-2)+f(x,y,z+2)-2\cdot f(x,y,z) \end{pmatrix} \quad (6)$$

3.4 3DHOG Extraction Algorithm

Summing up the previous subsections, we present the algorithm we implemented to extract the 3DHOG feature vector from triangular meshes. The actual Java source code of our implementation is also freely available for download[1]. For brevity's sake the details of interpolation are omitted here, for further information please refer to the actual source code. Please also refer to figure 2, which summarizes the main processing steps of our 3D HOG extraction pipeline.

1. choose Size of Voxel, Cell, Block
2. FOREACH Model m DO
3. normalize Trans., Scale, Rot. of m
4. FOREACH Voxel $v_{i,j,k}$ IN DistanceField $\mathbf{D}$ DO
5. computeShortestDistance($v_{i,j,k}, m$)
6. GradientField $\mathbf{G}$ = conv3($\mathbf{D}, [1,0,-2,0,1]$)
7. FOREACH Gradient $\vec{g}$ IN $\mathbf{G}$ DO
8. transformToSphericalCoordinates($\vec{g}$)
9. interpolateNeighborhoodOf($\vec{g}$)
10. insert $\vec{g}$ into its CellHistogram $h_c(\theta,\phi)$
11. FOREACH CellHistogram h_c DO
12. with h_c as $\vec{c}$
13. append $\vec{c}$ to its BlockVector $\vec{b}$
14. FOREACH BlockVector $\vec{b}$ DO
15. append normalize($\vec{b}$) to $\vec{f}_{3dhog}$
16. save $\vec{f}_{3dhog}$ for model m

4. EVALUATION

We conducted experiments on several established benchmarks to evaluate the retrieval performance of the 3DHOG descriptor. We are interested in assessing the relative performance of the two alternative gradient definitions. We also compare the performance of 3DHOG with established existing 3D descriptors. Finally, we assess the retrieval performance achievable when combining 3DHOG with complementary descriptors.

[1] http://www.gris.informatik.tu-darmstadt.de/projects/vsa/3dhog/3dhog.zip

4.1 Experimental setup

To evaluate the retrieval precision of 3D descriptors, several established benchmark data sets exist. For our experiments, we used the Princeton Shape Benchmark Test Partition (PSB) [14], the 2009 SHREC Generic Shape Retrieval Contest dataset (SHREC) [13], and the Konstanz 3D shape database (KN-DB) [3]. Each of these datasets includes generic 3D mesh models along with classification information grouping the models into similar classes. 3DHOG descriptors were calculated for all models, and retrieval experiments were performed by producing nearest neighbor rankings for all query objects in each benchmark data set, using the L1 norm on the respective feature vectors. *Precision versus Recall* diagrams were obtained by averaging over all queries in a given data set. In addition, several well-known single value retrieval precision measures (*nearest neighbor*, *first tier*, *second tier*, *e-measure*, *discounted cumulative gain*) [14] were calculated from the rankings. We use these measures to compare the retrieval precision of 3DHOG.

We extracted 3DHOG descriptors for all benchmark data sets using our implementation described in Section 3.4. The 3DHOG parameterization we used and a discussion thereof is given in Section 5.

We use the following alternative 3D descriptors to evaluate 3DHOG against: A 438-dimensional Depth-Buffer descriptor (*DBD438*), a 300-dimensional Silhouette-based descriptor (*SIL300*), and a 136-dimensional descriptor based on Radial Extent function (*RSH136*). Descriptor details can be found in [18]. We also consider a 472-dimensional descriptor which statically combines the aforementioned three descriptors (*DSR472*) [19]. Note that descriptors based on Depth Buffers to date are among the most effective 3D descriptors [5, 3], and that the hybrid DSR descriptor has been shown to substantially improve over the Depth Buffer descriptor [19].

4.2 Experimental Results

To confirm our intuition about the alternative 3DHOG gradient definitions from Section 3, we first extracted Precision-Recall-Curves for both variants of our descriptor. Comparison of the results obtained by first order gradients and second order gradients indicated significantly better results using second order gradients. Subsequently in all following experiments, second order gradients were used. Figure 5 shows Precision-Recall-Curves of the 3DHOG descriptor compared to the DBD, SIL, RSH, and DSR descriptors. As expected, the combined descriptor DSR consistently outperforms the other descriptors on all benchmarks. Our 3DHOG descriptor performs quite well compared to the single descriptors. While it yields similar precision results on the KN-DB benchmark as the RSH descriptor does, on the PSB benchmark the RSH de-

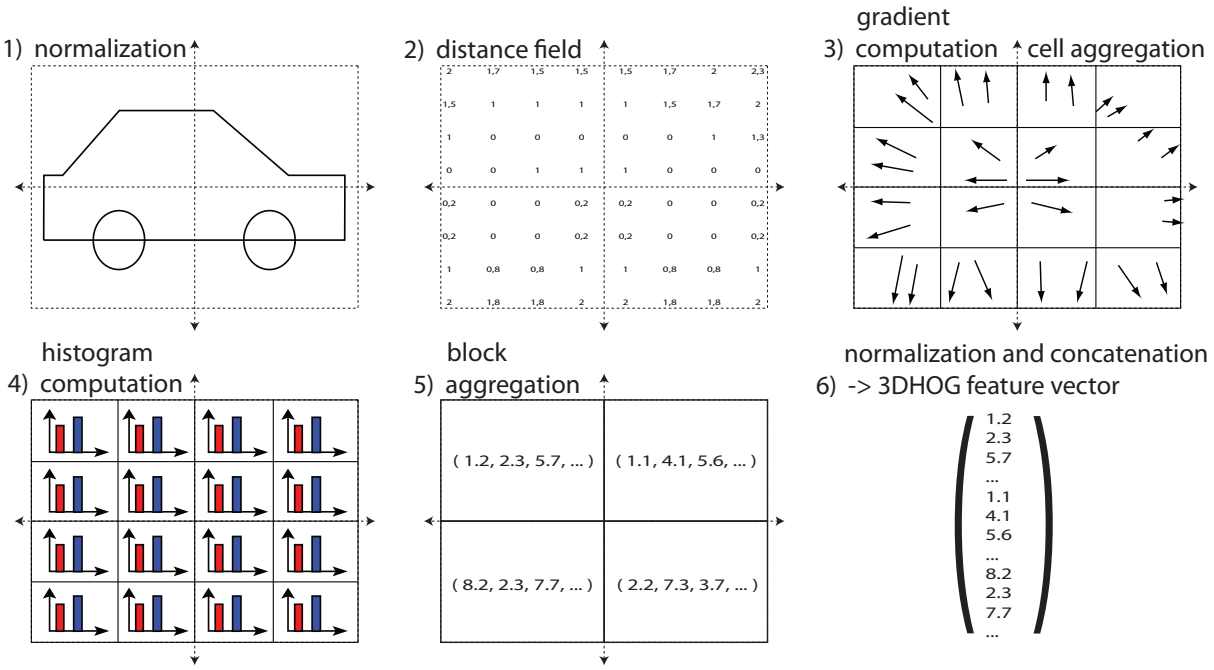


Figure 2: The major processing steps of the 3DHOG feature vector extraction pipeline

Benchmark	Descriptor	NN	FT	ST	E	DCG
KN-DB	DSR472	0.70	0.42	0.54	0.29	0.69
	DBD438	0.61	0.34	0.43	0.23	0.60
	SIL300	0.58	0.31	0.41	0.22	0.59
	RSH136	0.56	0.28	0.37	0.20	0.55
	3DHOG	0.59	0.29	0.36	0.20	0.55
PSB-Test	DSR472	0.65	0.40	0.51	0.29	0.66
	DBD438	0.60	0.33	0.42	0.24	0.59
	SIL300	0.54	0.30	0.40	0.23	0.58
	RSH136	0.51	0.25	0.34	0.20	0.53
	3DHOG	0.58	0.27	0.35	0.21	0.55
SHREC	DSR472	0.82	0.50	0.64	0.43	0.78
	DBD438	0.72	0.39	0.50	0.34	0.70
	SIL300	0.71	0.39	0.53	0.36	0.70
	RSH136	0.68	0.35	0.47	0.32	0.67
	3DHOG	0.75	0.41	0.52	0.35	0.71

Table 1: Scalar retrieval precision measures for the descriptors on the benchmarks.

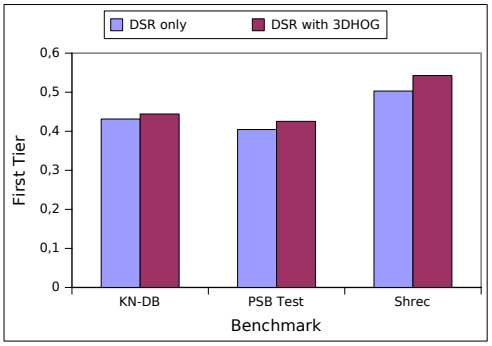


Figure 3: Precision on first tier of DSR (standalone) and in combination with 3DHOG. Better retrieval performance indicates that 3DHOG captures complementary, descriptive 3D features.

scriptor is outperformed by our descriptor. On the SHREC benchmark, 3DHOG outperforms all the individual descriptors (RSH, SIL, DBD). Table 1 gives the scalar precision metrics for the descriptors on the benchmarks, supporting this observation.

We also compared the retrieval precision of 3DHOG to its strongest competitor descriptor in our setting (DSR472), considering not per-benchmark average but per-class average metrics. Figure 4 shows the results for the SHREC benchmark. The results on the other benchmarks are similar. The chart shows that there exist classes of objects in the benchmark which the 3DHOG descriptor manages to retrieve more effectively than the DSR. This indicates that for certain query classes, 3DHOG is among the best 3D descriptors in our setting.

Finally, experiments were carried out combining the DSR and the 3DHOG descriptor in a hybrid descriptor. For combination, a static distance-based aggregation was used after distance normalization based on the variance as described in [17]. Figure 3 compares the precision on the first tier metric of the DSR472 descriptor alone, and in combination with the 3DHOG descriptor on all three benchmarks. Note that the retrieval precision is consistently improved on all three benchmarks by including 3DHOG. Precision-Recall-Curves were extracted as well and support this point (Figure 6). Since the DSR descriptor is already a combination of three single descriptors, these results suggest that the 3DHOG descriptor is able to capture meaningful, complementary 3D features, not captured by the other three descriptors. Clearly, 3DHOG is a

valuable contribution to the set of candidate 3D descriptors, as by combination of feature vectors, we are able to improve the precision of 3D retrieval systems. Note that improvements in retrieval precision usually cannot be obtained only by upscaling system hardware, but foremost require methodological improvements. This work contributes toward this end.

5. DISCUSSION

The retrieval results presented in the previous section demonstrate the applicability of the 3DHOG. The main point for discussion, as well as future work in this context, is the definition and computation of *gradients*.

We experimentally found that using second order derivatives improves over the use of first order derivatives. We attribute this to the fact that for effective 3D description, not only information about local extrema near the mesh surface, but also information about extrema inside the mesh are useful. The second order derivate reveals precisely such extrema by high values and sets monotonical regions to zero. This could also explain why this computation of gradients yielded superior results to using, for example, surface normals as gradients, which was also evaluated but not presented. Those only assign high values near the mesh surface, but always zero in and around the mesh (as they are not defined there).

Apart from the gradient definition, the choice of parameters has a significant impact on the performance of the 3DHOG method. The first parameter to set is the number of voxels in the distance field by choosing the edge-length $r_{x,y,z}$ of each voxel. During this discretization process we loose less information if we choose a small edge-length – naturally this induces a higher computational time. Our implementation uses a straight-forward algorithm to compute the distance field. Optimization is necessary to compute high-resolution distance fields efficiently.

The next parameter is the cell-size $c_{x,y,z}$, which determines how many gradients – computed from the distance field – are aggregated in one histogram. Thereby we can choose the *degree of locality* the descriptor exhibits.

The histograms themselves are defined by the number of bins they provide for the gradient's zenithal and azimuthal orientation $(\theta, \phi)^T$ (see eq. 1). This introduces the usual trade-off between accuracy and stability.

The last major parameters we have to choose are the size $b_{x,y,z}$ of the blocks as well as their overlap $o_{x,y,z}$. These determine how many neighboring cells are normalized with each other.

For our evaluation good first parameter-space guesses based on the results by Dalal and Triggs [6] were used and some further exploration was carried out. The parameters yielding the best descriptor we found and its dimensionality can be found in table 2. Surprisingly, the block overlap did not have the positive influence it had in the experiments of Dalal and Triggs [6], so it was set to 0.

Parameter	Value
$r_{x,y,z}$	$\frac{2}{52}$
$c_{x,y,z}$	12 vxl
bins(θ)	9
bins(ϕ)	9
$b_{x,y,z}$	2 cells
$o_{x,y,z}$	0 cells
dimensionality	5184

Table 2: Parameter settings used to extract 3DHOG descriptors

We are aware of two efficiency implications of 3DHOG: High dimensionality of the feature vector, and the expensive computation of the distance field in our simple implementation. Dimensionality could be tackled by subsequently applying a dimensionality reduction technique. E.g., in [18] PCA was indicated to improve retrieval efficiency without significantly reducing retrieval effectiveness. Accordingly, appropriate application of PCA can even improve effectiveness due to noise filtering effects. Efficient computation of distance fields is already well-studied and subject to current research [11, 15]. consequently, accelerating 3DHOG feature vector extraction is possible by using more advanced algorithms.

6. CONLUSION & FUTURE WORK

In this paper we derived and implemented a new 3D descriptor (3DHOG) based on the HOG feature vector, which is well-known in 2D image processing. The main challenge of defining meaningful gradients was looked at in detail. One concept to implement 3DHOG was presented in general – an actual open-source implementation is made publicly available. The performance of the descriptor was evaluated against established retrieval benchmarks and results indicated competitive retrieval performance. The descriptor captures 3D object information complementary to those of existing edescriptors, and in combination with them manages to boost retrieval performance.

The concept of using HOG for 3D model retrieval seems promising, and we like to further explore the parameter space and alternative gradient definitions. Preliminary experiments by defining gradients as the local curvature of the mesh's surface were conducted. First results of this approach could not match the original implementation. This can be explained by the fact, that curvature is only defined on the mesh surface. Accordingly combination with second-order derivate distance

fields could be the key to further enhancement: Using curvature to describe the mesh near the surface, and using the distance derivatives to describe the mesh's interior.

As mentioned in related work (Section 2), particular similarities of the challenges faced can be drawn with the work by Glomb[8], who extends the Harris-operator to 3D meshes. The methods proposed by Glomb include *gaussian functions*, *quadric surfaces*, *Haussdorff distance* and *fitted surface distances*. All of these methods differ drastically from our approach, as they require the definition of a set of neighbors for each vertex. His findings suggest that fitting quadric surfaces is more suitable than the other extensions. Accordingly, we plan to adapt this approach for use with 3DHOG. Recalling the algorithm for feature computation as described in Section 3.4, we only have to substitute gradient computation on the distance field by the actual derivative of the fitted quadric at the specified position. The retrieval performance of the 3DHOG using gradients defined this way will also be evaluated.

Additional drawbacks of *exact* Eucleadian distance transforms occur, because the distance function is not differentiable at points that are equidistant from the mesh surface [2]. This may lead to undesirable outliers when convolving the distance field to approximate second order derivatives. Accordingly, we will examine the use of *smooth distance transforms* [2], which sacrifice a certain amount of accuracy for differentiability at each point.

ACKNOWLEDGEMENTS

We thank Bernt Schiele and the Multimodal Interactive Systems group for inspiring discussions on the HOG method and options for porting it to 3D. Furthermore, we thank colleagues in the Interactive-Graphics Systems Group and in Fraunhofer IGD A2. We thank Christian Wojek for providing a HOG implementation for images; Thomas Kalbe for supplying a render-engine for distance fields; and Matthias Bein for providing source code for curvature computation.

This work was partially supported by the German Research Foundation DFG within the LIS Leistungszentrum PROBADO under grant INST 9055/1-1.

REFERENCES

[1] Mihael Ankerst, Gabi Kastenmüller, Hans-Peter Kriegel, and Thomas Seidl. 3d shape histograms for similarity search and classification. In *Spatial Database*, page 207–226. Springer, 1999.

[2] Arpan Biswas and Vadim Shapiro. Approximate distance fields with non-vanishing gradients. *Graph. Models*, 66(3):133–159, 2004.

[3] B. Bustos, D. Keim, D. Saupe, T. Schreck, and D. Vranic. An experimental effectiveness comparison of methods for 3D similarity search. *International Journal on Digital Libraries, Special Issue on Multimedia Contents and Management*, 6(1):39–54, 2006.

[4] H.-S. Wong D. Wang, J. Zhang and Y. Li. 3d model retrieval based on multishell extended gaussian image. In *VISUAL*, page 426–437, 2007.

[5] Y. T. Shen D. Y. Chen, X. P. Tian and M. Ouhyoung. On visual similarity based 3d model retrieval. In *Proc. of Eurographics Workshop*, volume 22, pages 223–232, 2003.

[6] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR (1)*, pages 886–893, 2005.

[7] R. Fabbri, L. Da F. Costa, J. C. Torelli, and O. M. Bruno. 2d euclidean distance transform algorithms: A comparative survey. *ACM Comput. Surv.*, 40(1):1–44, 2008.

[8] Przemysław Głomb. Detection of interest points on 3d data: Extending the harris operator. In *Computer Recognition Systems 3*, volume 57 of *Advances in Soft Computing*, pages 103–111. Springer Berlin / Heidelberg, 2009.

[9] C. Harris and M. Stephens. A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

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

[11] Mark W. Jones, J. Andreas B?rentzen, and Milos Sramek. 3d distance fields: A survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):581–599, 2006.

[12] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[13] NIST. Shape Retrieval Contest (SHREC), New Generic Shape Benchmark Dataset. http://www.itl.nist.gov/iad/vug/sharp/benchmark/shrecGeneric/, 2009.

[14] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The princeton shape benchmark. *Shape Modeling and Applications, International Conference on*, 0:167–178, 2004.

[15] Avneesh Sud, Miguel A. Otaduy, and Dinesh Manocha. Difi: Fast 3d distance field computation using graphics hardware. *Comput. Graph. Forum*, 23(3):557–566, 2004.

[16] Johan W. H. Tangelder and Remco C. Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia Tools Appl.*, 39(3):441–471, 2008.

[17] Tobias Schreck. *Effective Retrieval and Visual Analysis in Multimedia Databases*. PhD thesis, Universität Konstanz, Universitätsstr. 10, 78457 Konstanz, 2007.

[18] D.V. Vranić. *3D Model Retrieval*. PhD thesis, University of Leipzig, German, 2004.

[19] Dejan V. Vranic. Desire: a composite 3d-shape descriptor. In *ICME*, pages 962–965, 2005.

[20] R. Wessel, I. Blümel, and R. Klein. A 3d shape benchmark for retrieval and automatic classification of architectural data. In *Eurographics 2009 Workshop on 3D Object Retrieval*, pages 53–56, 2009.

[21] T. Zaharia and F. J. Preteux. 3d-shape-based retrieval within the mpeg-7 framework. In *Society of Photo-Optical Instrumentation Engineers Conference Series*, page 133–145, 2001.

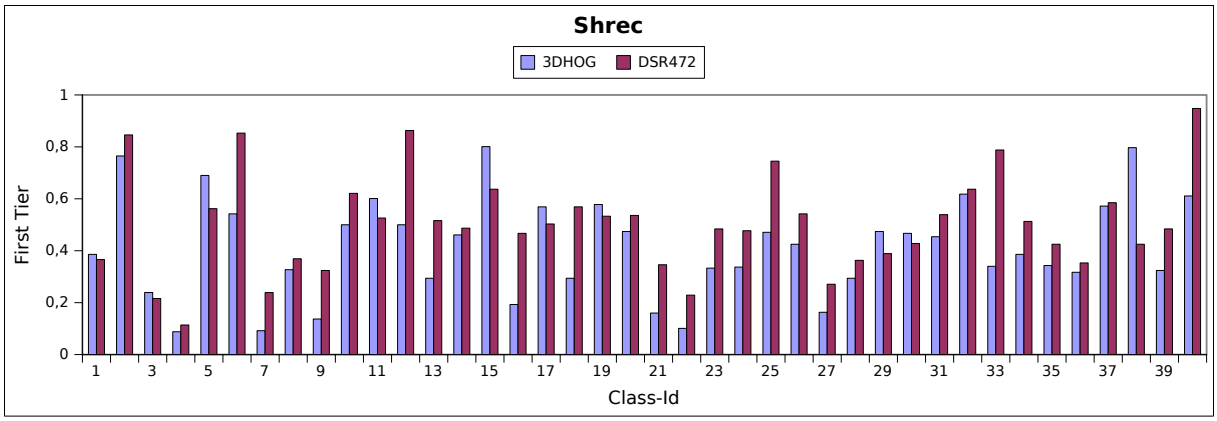


Figure 4: Precision on first tier of individual classes of the SHREC benchmark. In several query classes (e.g., 11, 15, 38), 3DHOG outperforms one of the best hybrid 3D descriptors (DSR).

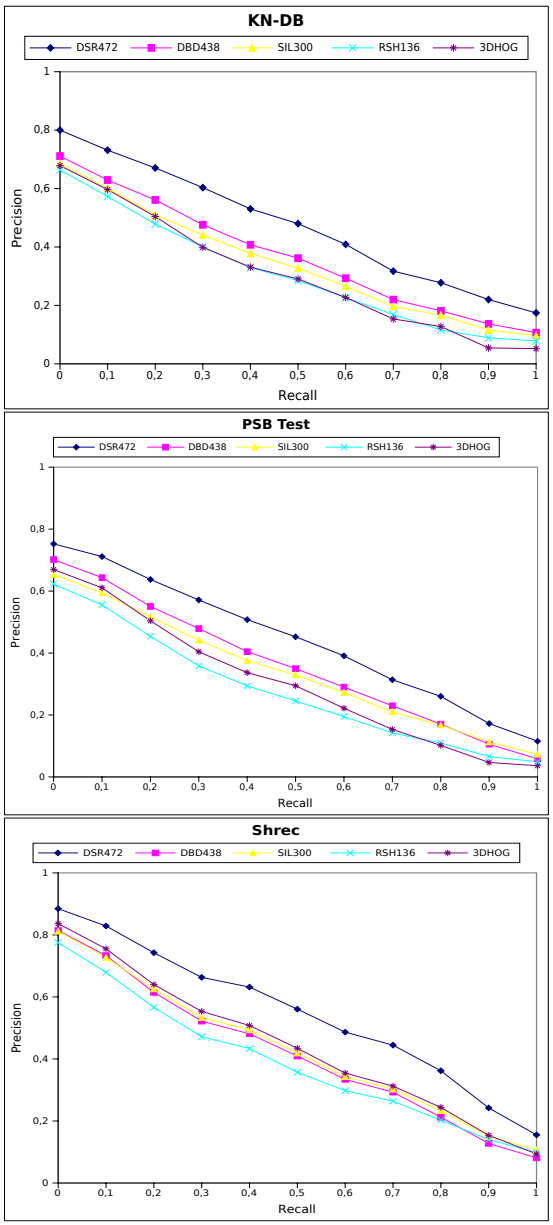


Figure 5: Precision-Recall-Curves on the benchmarks.

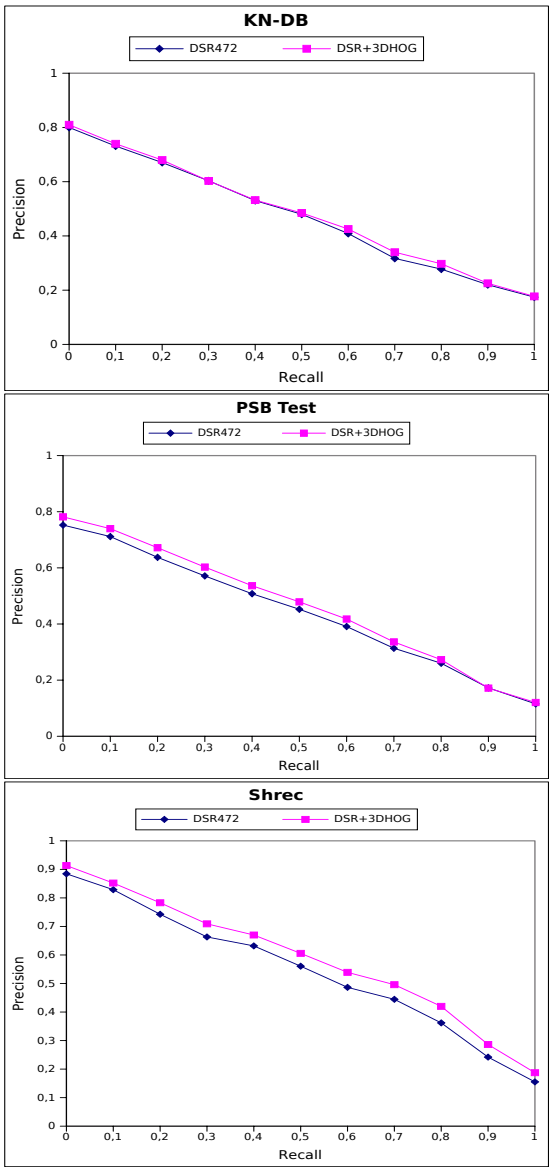


Figure 6: Precision-Recall-Curves of DSR (standalone) and in combination with 3DHOG on the benchmarks.

Dynamic Visual Effects for Virtual Environments

Matthias Haringer
University of Hamburg, Germany
haringer@informatik.uni-hamburg.de

Steffi Beckhaus
University of Hamburg, Germany
steffi.beckhaus@uni-hamburg.de

ABSTRACT

We introduce a high level interface capable of instantly adding and manipulating a multitude of visual effects for any object or area in VE scenes. This enables a controlling person or an automated system to react to the in scene situation and the user's actions by dynamically changing the scene's appearance. Such a high-level scene and effect access is also a powerful tool for story telling engines and scene authors to support the intended plot or impact. To achieve this smoothly and effectively, our interface allows fading of effects to generate soft effect transitions and grouped effects for controlling complex effect combinations in a simple way. We also describe our fully functional implementation and the changes necessary to realize our concept in a scene graph based VR-system. We further introduce a simple script interface to add new effects to the available effect pool. We present, but are not limited to, three visual effect types: shader effects, post-processing effects, and OpenGl based effects. Our implementation supports multi-pipe displays, multi-pass rendering, and an arbitrarily deep per-object post-processing effect graph.

Keywords

Effect control, dynamic effects, post-processing, shaders, virtual environments

1 INTRODUCTION

Visual effects are nowadays commonplace in VR applications and games. Up to now their usage is mostly predefined by scene authors, but in principle many of those effects are accessible and modifiable at run-time. The effects we look at are not only high level effects like depth of field, heat haze, high dynamic range rendering, or lens flare, but also simple changes of brightness, contrast, color balance, and basically everything to change the appearance of the scene. The potential of controlling such effects dynamically in an application is rarely used, because there is no universal and real-time interactive access to most of them. Authoring tools available for games and some VR-systems allow scene creators to place and test effects of a scene, but do not provide for a general and intuitively controllable effect integration. They help to create scenes and introduce some dynamics via scripts, but they do not provide run-time access and effect control for the scene.

An analogy to our effect control concept from theater and stage performances is the light designer and the light board operator. The operator is able to control lighting and special effects with sliders and buttons. He can use one slider to control multiple effects, preprogram spot locations, and fade between effects. Usually a fixed schedule is prepared by the light designer. But at live performances the operator has to instantly react to the stage and audience and activates appropriate effects. As we look on interactive VE's, this live performance case is what we aim at.

Our vision is to create an interface capable of influencing the scene's appearance via many implemented effects. Operators of this interface only have to know what the effects can do, how they can be influenced, which user reactions can be expected, and which mood they might create - like in the light board case. Such an interface is not only useful for human control; it can be as well used for automated story engines or can be connected to user assessment systems to automatically react to the users state. Our system is not limited to visual effects, but as the visual effects were the first to be fully implemented, this paper concentrates on the visual part of the work.

The reduction of the brightness of the sky, for example, is achieved in our system by selecting the sky object, adding a brightness effect from the list of available effects, and reducing the brightness value via a slider. Other examples, which can be applied in a similar way are: making waves and clouds move faster, fading to night lighting conditions (by moving one slider which controls many effects), introducing fog, and correlating the inverse distance to a house to a weather change or a local glow effect on that house.

This paper presents our concept and implementation of a system for such a dynamic effect based scene ma-

nipulation. We introduce related work in Section 2 and give an overview over the whole system in Section 3. Section 4 introduces different types of effects and in Section 5 we present the implementation of the system. Section 6 lists some performance measurements for different effect usages, and in Section 7 a summary and outlook is given.

2 RELATED WORK

Various fields and methods informed the concept and the resulting system presented in this paper. The most important of them are discussed in the following. A major feature of our approach is the manipulation of different kinds of effects using a general interface. Conrad et al. introduced a tuner application for virtual environments to be able to manipulate several aspects of virtual environments via a GUI [2]. Modifiable parameters were object transformations, material properties, light source properties, and selecting the navigation metaphor (fly, drive, and walk). Conrad et al. use a fixed set of modifiable parameters. In our approach we want a more general and extendable solution, where new effects can be added easily.

To add effects at run-time, we have to be able to access the scene's objects. There exist many approaches, which allow such an access on the authoring level. [5] introduces an in-scene immersive authoring tool, where authoring takes place in the running system. [9] describes an easy to use rapid VE development environment. Multimodal authoring approaches like [1, 4] further allow the composition of multimodal scenes. As we plan to use effects of other modalities, we adopted the concept of a single object keeping the information for all modalities from these approaches.

The Listen project [3] introduces manipulating auditive parameters in an audiovisual scene graph for restricted zones in the scene. This enables authors to create different auditive spaces in different parts of a real or virtual room. We use this concept to apply our effects to run-time definable spatial areas in the scene. As we intend to use semantically meaningful objects in our scenes and the objects can contain semantic information, we oriented on current findings of this area for structuring the scene and its semantic information. [6] and [7] present examples of semantically extended scenes.

Another closely related field is game authoring and in game effects. Most game engines support object and special effect access on the authoring level. Some game engines allow dynamic effect behavior for some effects via scripts. Game effects are generally not run-time changeable. Examples include the Unreal editor for the Unreal Engine (Epic Games), which enables effect types like full-screen post-processing, particle and audio effects. These effects are realized in separate

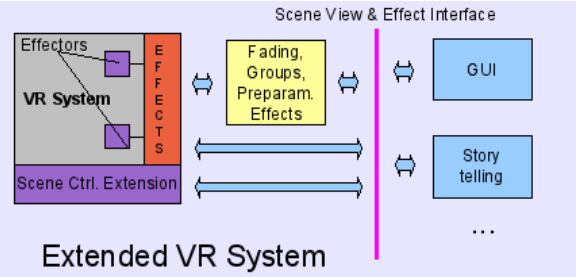


Figure 1: System overview

modules of the Unreal Engine. The effects can be accessed via the scripting language, but lack an uniform and intuitive access. The Source engine [8] uses a two step post-processing system for high dynamic range rendering and a color adjustment step. The color adjustments are made available for authors and are baked into game nodes for final game-play where they cannot be changed anymore. We found that today's game editors and engines do not provide for the generality of effects and run-time accessibility needed for our main purpose: to change the multimodal appearance and impact of the scene dynamically and uniformly.

[10] use non photo realistic rendering (NPR) techniques to create abstract views of 3D scenes. They are able to apply different NPR styles for different objects to emphasize one or more objects, which is comparable to our per object post-processing.

The introduced effects and effect mechanisms in Section 4 are not new and have been used in several games and VE's. Our contribution to using effects is the multi-pipe implementation, the mapping of technical to intuitive parameters, the per object post-processing effects, the general fading mechanism, and the flexible effect assignment and combination.

3 SYSTEM DESCRIPTION

Our System concept can be structured in three parts: The **scene control level** is the high level part, which lets moderators, engines, and authors easily modify the scene (middle and right part of Figure 1). The **effect configuration level** is a medium level, where effects and effect mappings can be defined ("EFFECTS" box in Figure 1). In the low level the effect types and the scene control with the per object effect manipulation are implemented (**implementation level**, see small squares and scene control extension in Figure 1). The scene control level is discussed in Sections 3.1 - 3.4 and the effect configuration level in Section 3.5. In Section 4 the effect types or effectors, as they are called in the following, are described.

The main goal of our system is to enable dynamic run-time assignable and changeable effects on a VE scene. On the scene control level there are three important questions that need to be answered when adding an effect to the scene. First, where do we want to apply the

effect to (*object/scene part selection*). Second, which effects are to applied (*effect selection*). Third, how and when do we want to apply them (*effect transition, effect timing*).

To choose where we want to assign the effects to, or which part of the scene should be affected by the effect, the scene has to be accessed in some way. This interface to the scene is called *scene control* in the following. On top of that scene control, we introduce methods to help the easy handling of manipulating complex scenes and effects (box in the upper center of Figure 1). The more important of those methods are fading, preparameterized effects, groups, and a preselection capability. These techniques allow fast and complex effect manipulation and define how and when to apply the effects, which is especially important for live performance and automated scene control. The following sections describe the introduced steps in more detail.

3.1 Scene and Effect control

For intuitive and run-time assignable effects an inexperienced user should be able to select a part of the scene and assign an effect from a list of implemented effects to them. However, for which parts of a scene is assigning effects reasonable? The first case of entities we identified, are semantically meaningful objects in the scene (like sky, buildings, and trees). Which objects are meaningful and which granularities to use, heavily depends on the scene and the application. A second case where applying effects is important in our opinion, are semantic spatial areas in the scene. Such areas are volumes, which do not have a direct visual representation. Examples for effects on areas are a cool scene appearance around a specific house, brightness on the beach, or a greenish look in a forest.

In our system, objects and areas can be structured in a tree hierarchy and each of these objects or areas can hold multiple effects. The implementation of this part has to be able to extract and manipulate objects from the scene, it has to provide areas, and it has to realize the connection between the effects and the objects (see Section 5.3).

After we know where to place an effect, we have to decide which specific effect is to be applied. The control over the effects should be equally high level. For choosing an effect, we only need to know what the effect does (name and description) and how we can influence its behavior (parameters). It is not important at that point whether the effect is realized via shaders, post-processing or manipulating OpenGL states. Our interface for selecting effects is an effect pool with descriptions of effects and parameters. All currently available effects in the system are in that pool and can be arbitrarily added to objects.

3.2 Preparameterized effects and groups

Most effects can be customized via parameters, for example red, green and blue parameters that adjust a simple RGB color balance effect. Because of the lack of time for adjusting the parameters online and the need to access some previously optimized parameters, predefinable settings are additionally required. In our system **preparameterized effects** allow users to store an effect with a specified set of parameter values. The preparameterized effects can be created anytime while experimenting with the parameters of an effect. They can be applied to objects like any other effect.

Our second method to enable an intuitive and quick effect handling are the following grouping mechanisms: **Object groups** allow us to add an effect to multiple objects at once. For example, say we want to modify the color of sky and water simultaneously. An object group holding the sky object and the water object is created and an effect is added to this group. **Effect groups** allow us to add multiple effects to an object or an object group. They make it possible to switch or fade all effect members on and off together. For order dependent effects, like post-processing, an effect order can be defined. **Multi-object effect groups** allow us to assign multiple objects with one or more different effects. They can be used for complex effect settings like to represent different weather situations. A dark rainy day multi-object effect group might combine dark sky, high waves, grey water, dull landscape (low saturation), and rain on all outdoor objects, which can be switched or faded as one. Object and multi-object effect groups are, because of their reference to specific objects, scene dependent. Effect groups and preparameterized effects can be shared between scenes.

3.3 Effect preselection

Not all effects are a good choice at all times or for all places and objects. Therefore, the number of available effects can be reduced per object, area, or by time constraints. This gives a live operator or an automated process the simplification that only effects that make sense at the current place and time are possible to select. Those restrictions and context dependent effects have to be prepared as an authoring step using the scene control, areas, and groups.

3.4 Effect intensity and fading

Being able to switch effects on and off at run-time for arbitrary objects is already a valuable improvement for manipulating a scene. Often, however, a subtle fade in, fade out, or fade between effects is required. When adding or removing effects at run-time, fading is in most cases preferable, as smooth changes are perceived as more natural. One example is slowly introducing a cold scene look by a darker and bluish appearance of the landscape and sky or the simulation of

Figure 2: First row: Effects in a marketplace VE scene (left to right): Fog, bleach bypass, bloom, rain. Second row: Island scene (left to right): a) different brightness adjustments for water, land, and sky, color balance on the whole scene, sunset sky; b) no landscape textures, random gray tone per face, edge enhance, rain; c) bloom on landscape, different color balance and brightness effects on landscape, water, and sky; d) daylight sky, adjusted landscape detail shader.

a cloud covering the sun by fading out a glow effect and fading in a low brightness effect. Another application of fading is, when different areas of the scene should have different appearances. When moving over the border, fading between the two appearances is required.

Implementing the fading capability involves being able to manipulate an effects intensity without altering the other effect parameters. The effect intensity defines the range of the full effect to some zero effect. Directly fading between two effects is only possible, if the two effects can exist simultaneously. For this kind of fading the intensity of the first effect is lowered and the intensity of the second effect is raised over a defined time interval. Fading is simple to realize for post-processing effects and some object shader effects, because they can easily be mixed with the original scene to an adjustable degree. For other effects the intensity mapping can be more complicated, but the generic fading capability for all effects exceeds the effort spent here.

3.5 Effect definition and mapping

The configuration layer in between the scene control interface and the implementation has two sub-layers: the **definition** of the effects to be used in the interface and the **mapping** of the parameters of the implementation to high level parameters. The mapping also introduces descriptions of effects and their parameters.

The effect definition holds resources like shaders and textures and combines them to form an effect. All necessary functions to easily create and configure effects are provided by the implementation layer (effectors). The effect definition part requires shader and script programming and is intended for effect artists.

The effect mapping can directly assign, combine, or modify internal parameters (e.g. uniforms) to form the final effects parameters. Descriptions, effect name, and parameter names to be seen in the final effect are supplied here. The mapping part requires only simple script programming and is, therefore, easily adjustable. It is intended to separate the technical formulated and parameterized effects from the final effects, which use meaningful parameters with appropriate ranges for their intuitive control. Effect definition and mapping build together a complete effect, which can be accessed from the effect pool.

4 EFFECTORS AND THEIR EFFECTS

Effectors are the implementations of effects or types of effects with the same implementation base. Effectors encapsulate the system internals needed for an effect type and they have to provide everything to assemble the effects in the effect definitions. The shader effector, for example, provides the interface for all shader based effects. Effectors can control the rendering pipeline, if necessary, and they are aware of restrictions of the system or restrictions regarding the coexistence with other effectors. In this section three effectors for a large number of possible visual effects are introduced. Examples of effects from these effectors can be seen in Figure 2. Effectors for ambient lighting, navigation effects, and auditive effects are currently under development.

4.1 Post-processing effector

Post-processing effects are a very powerful type of visual effects. They adjust the appearance of the scene with image processing techniques. Up to now, post-processing has mainly been used on the whole visible display area. In our implementation, it can also be applied on a per objects basis to better

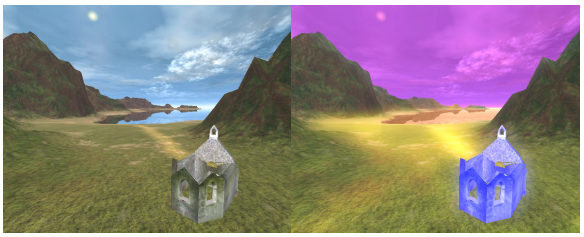


Figure 3: Scene without (left) and with (right) per object post-processing effects for sky, water, ground, and chapel

match our object based effect approach. Allowing different post-processing pipelines for sky, ground, and water surfaces already permits us to refashion the appearance of the scene more efficient than full screen post-processing. Figure 3 shows an example of per object post-processing. Post-processing effects of this kind have the advantage over shader effects that they are universally applicable, i.e. they can be added to any object in the scene and will have the intended effect, where shader effects often rely on specific geometry and vertex properties. The implementation is done via frame buffer objects and specialized shaders and is usable for multiple graphics pipes (see Section 5.6). The passes, shaders, uniforms, and effect parameters are defined and assembled to form the specific effects in the effect definition and mapping.

Our post-processor system is highly flexible. Each object or area can have several post-processing effects and each post-processing effect can incorporate several post-processing passes. Each pass has a post-processing shader, up to six input textures, an arbitrary number of uniforms and a result texture. Passes can be stacked and the results of different passes can be combined, i.e. the passes can be laid out in a directed graph structure, which allows for very complex effects. A simple two pass example is a bloom shader, which extracts all bright pixels in a first step and applies a blur shader to the extracted regions as a second step.

Some already implemented post-processing effects are color balance, bloom, blur, depth of field, edge detection, edge enhance, tone mapping, rain, and some sketch effects like hatching. An important group of post-processing effects, which can easily change the impact of an object, are color manipulation effects. We have implemented two approaches to modify the coloring of the scene. Our color grading effect assigns each color value a different color via a predefined color map. Combining simple post-processing effects like color balance, brightness, contrast, saturation, and gamma adjustment can produce similar but not as exact results to the mapped color grading approach. The advantage of this approach is the dynamic manipulation possibility of all components.

Realizing effects via post-processing has also some disadvantages. A problem with post-processing for stereo displays is that brought in content like a texture appears only on the screen plane. A rain effect, for example, which is believable on mono displays, can not be used for a stereo setting because of this effect. Our solution is to introduce a pipe dependent shift of the images to produce an eye offset. Using this, the seen plane can be moved before or behind the viewing plane, but it still appears as plane. In our rain post-processing effect five rain planes with different depth are used to simulate the rain in stereo. Each rain plane uses 5 differently sized versions of a rain texture which are moved with different speed over the screen using a post-processing vertex shader.

4.2 Shader effector

Object shader effects are common in most VR-systems and game engines. They are mostly used to realize specific surface materials and moving surfaces like water and clouds. Many effects which are usually implemented as normal shaders can be created via the per object post-processing effects in our setup. Whenever vertex based information is needed (position, normals, texture coordinates) a conventional shader has to be used. Being dependent on geometry, many shader effects are less generally applicable than post-processing effects. Because of that, shader effects are mainly used for the basic appearance of our scenes, which can be modified by the parameters of the shader effects.

The shader effector provides a means to simply add shaders and supply them with input textures and uniform variables. This makes it possible to easily integrate any existing GLSL shader in a few minutes. Our current implementation allows only one object shader effect per object. Most multi-pass shaders like non photo-realistic shader effects can be created by using one object shader and several post-processing shaders. Currently integrated shader effects are: animated water, sky, and grass, as well as a level of detail terrain shader and materials like plastic and glass.

4.3 GeoState effector

GeoStates encapsulate OpenGL graphics properties for the geometry (GeoSets) in OpenGL Performer style scene graphs. The GeoState of an object's geometry can be overridden with a GeoState effector. The effect definition and mapping defines which properties of the GeoStates are to be overridden and which property values should be applied instead. GeoState effectors can affect the following states: lighting, texture, fog, wireframe, highlighting, material, alpha function, transparency, decals, and light sources.

The fading of most GeoState effects is more problematic than for post-processing and shader effects. When exchanging textures, enabling highlighting, etc., there is no simple way to avoid a sudden change. One possibility, which we use for texture fading, has following

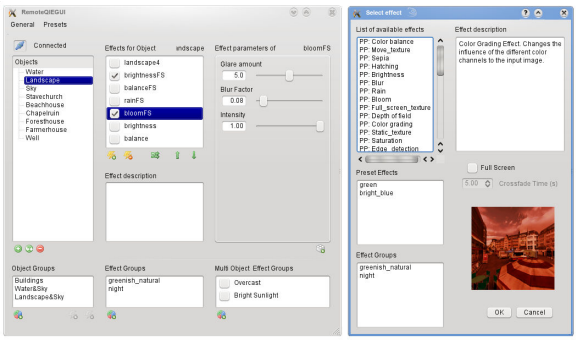


Figure 4: Scene Control GUI: Left: Object and effect control. Right: add a new effect.

process. We start from a single textured object without a shader. As a first step, we add a shader effect with two configurable textures, the one that was present and a new one. We fade from the first to the second texture via the shader effect's intensity. In the mean time a GeoState effect with the new texture is applied (hidden by the shader) and finally the shader effect is switched off again. We use this technique to fade between different sky textures for different times of day or weather conditions.

5 IMPLEMENTATION

This chapter gives an overview of the implementation of our system. The VR-system dependent parts of our implementation use Avango [12] which is based on OpenGL Performer [11]. Future adaptions for AvangoNG and VRJuggler (using OpenSceneGraph) are planned. Section 5.1 describes the implementation of the scene control interface and Section 5.2 the implementation of the effect control layer. Sections 5.4– 5.7 lay out the implementation of the effectors and necessary adjustments to the render pipeline.

5.1 Interface

The scene control interface is realized with XML-RPC. The implementations of objects, effects, and parameters define RPC functions for all functionalities of the interface. On the remote side, handles of the objects, effects, and parameters are used to access their interface functions. The modules for fading, groups, and preparameterized effects provide their functionalities also as an XML-RPC interface. External applications like storytelling or user state estimation can use the complete XML-RPC interface. The GUI which was implemented for moderators and authors also supports the complete interface inclusive fading and groups. The GUI is intended for controlling the scene during run-time and represents the light control board from our analogy in the introduction. A screen-shot of the GUI can be seen in Figure 4. The scene control GUI has three colums for displaying objects, effects, and effect parameters respectively. If an

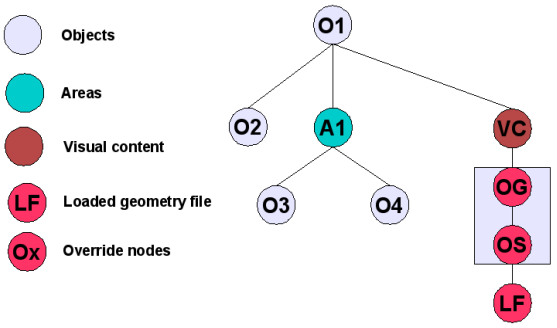


Figure 5: Extension of the scene graph

object is selected, its effects are displayed in the effects column. If an effect is selected or added, its parameters are displayed in the effect parameter section.

5.2 Effect definitions and mappings

The effect definitions are currently implemented in Scheme, which is the scripting language of Avango. We plan a reimplementation using Python to be more easily adaptable to other systems. To create an effect of a specific effector type, a C++ instance of this effector is created and initialized by loading the effect definition via its name. Each effect is located in a file system directory containing effect definition, effect mapping, descriptions, and resource files.

5.3 Scene control implementation

The main idea in our scene control implementation is to use the existing scene graph of a given VR-system and extend it to match our scene concept of semantic objects and effects per object. As scene graphs are extensible, the required nodes can be derived from existing nodes. The scene abstraction is mapped to the OpenGL Performer style scene graph as follows: Objects are inherited from transformation nodes (DCS) in the underlying scene graph. Each object has its visual contents as a special child (VC in Figure 5). Our implementation also allows for other content types, like auditive, haptic, or physics based content which can be rendered by the respective engines. The actual visual appearance of an object can be an arbitrarily deep sub scene graph or a loaded model (LF in Figure 5). Child objects and areas are also children of the transformation node (A1 and O2 in Figure 5). Other children than child objects and content are not allowed in the structure. Effects are not implemented as scene graph nodes, they are properties of the object node.

For applying shader and GeoState effects, special override nodes are placed between the VC node and the actual content (OG and OS in Figure 5). They allow to override the GeoState and shader settings of all underlying content geometry. The VC nodes also hold a post-processing ID to be used in object based post-processing (see Section 5.6).

5.4 GeoState effector

The GeoState effector is implemented using override nodes, which partly use the OpenGL Performer GeoState override functionality and in some cases manipulate OpenGL states directly. When a GeoState effect is activated, an override node is placed above the geometry of an object. This allows GeoState effects to change one or more states of this geometry.

5.5 Shader effector

The shader effector is implemented using a shader override group, which activates the shader for all underlying geometry. The shader uniform variables to manipulate the shader can be mapped directly or can be further processed to form effect parameters. Standard uniforms variables like time, user position, and light source positions are automatically provided for all shader effects.

Complex material properties like reflection and refraction may need additional render passes. An example is a water shader with water reflection and water refraction, which requires two additional render passes. In those cases, the scene or a part of the scene is rendered before the main render pass. For example, an upside down version of the scene is rendered for reflection which is stored in a texture. The shader effect can use this texture afterwards like any normal shader input texture.

5.6 Post-processing effector

The post-processing effector requires a more elaborate extension to the VR-system. The scene is rendered into a frame buffer sized texture using frame buffer objects (FBO) for each graphics pipe. A shader is then applied on a 2D rectangle holding this texture. The result is again rendered to a texture or, if the last post-processing effect has been reached, the final result is rendered in the frame buffer. In fullscreen mode, the post-processing shader is applied to all pixels of this texture. In the object based case, all objects with post-processing effects are assigned an ID. This number is used to mask the object in the stencil buffer during the rendering of the scene into the texture. The stencil/depth buffer is rendered in an additional texture using frame buffer objects, which can be accessed in the shaders. Only pixels with the according value in this texture are changed by the shaders. This way the shaders only affect the visual geometry of the masked objects.

In each shader pass the first two texture units are assigned to the color texture of the previous pass and the stencil/depth texture. The stencil/depth texture is only used for reading the depth and the object mask values during the post-processing. Six additional textures (assuming 8 texture units) can be freely used by the specific shaders. The number of simultaneously displayed per object post-processing effects is limited by the size of the stencil buffer. As a stencil buffer of 8 bit is available on most systems, this would limit the objects for holding separate post-processing effects in a scene to 255. To relax this limit we dynamically reassign the stencil values for the currently visible objects with post-processing effects. 255 visible objects with post-processing should suffice for most scenes and viewpoints.

5.7 Multi-pass rendering

Shader and post-processing effectors need to modify the render behavior of the VR-system. Each graphics pipe can have several pre-render passes, which render the scene with different settings and one main rendering pass. The pre-render passes are passed as FBO textures. These textures can be accessed by shader effects. The pre-render passes are only activated, if an effect is currently using them. The main render pass incorporates either the post-processing rendering process described earlier or a standard scene rendering. In the post-processing case, the main render pass renders the scene normally, then applies a post-processing pass per post-processing effect and object holding such an effect. The last post-processing pass writes to the frame buffer.

6 SYSTEM PERFORMANCE

Among several test scenes the system has been tested with three larger scenes. Two of them were conventional scenes adapted to the extended system. The adaption needed following steps: Splitting the scene into semantic parts to be used in the scene control, using these parts as visual content of our enhanced object nodes, and adding areas. The whole adaption process of the scenes has been managed in several hours.

An island scene (seen in the lower row of Figure 2) was designed for our system and uses advanced shader concepts and incorporates many areas. The main components are a $2.5km^2$ terrain mesh generated from a 512*512 point height-map and using a 4096*4096 pixel base texture rendered with Terragen, a $5km^2$ water plane, and a sky dome with a 2048*768 pixel base texture. The scene is currently populated with some buildings and billboard trees. The complete scene has approximately 450k polygons. For the landscape a LOD detail shader effect that uses textures for grass, sand, rock, and forest is used. The water is realized via a multi-pass watershader effect.

Our hardware setup is a dual Opteron HP XW9300 workstation with two synchronized Nvidia Quadro 4600 graphics boards. Our L-Shape display is driven by two Projectiondesign active stereo projectors, which are fed with two graphic pipes each.

	1Pipe	2Pipe	4Pipe
1. Basic island scene (Avango)	81fps	62fps	50fps
2. with our extensions	81fps	61fps	50fps
3. 2. + 2 render passes	65fps	35fps	25fps
4. 3. + water & land shaders	60fps	27fps	22fps
5. 4. + medium effect usage	58fps	25fps	20fps
6. 4. + heavy effect usage	55fps	22fps	16fps

Table 1: Performance for 1, 2, and 4 pipes.

Our measurements are taken for a single, a two, and a four pipe setup. A predefined path in the scene is used to generate an average frame rate. Table 1 shows the performance for different configurations from the bare scene to heavy effect usage. The heavy effect usage performance run includes 5 full screen post-processing effects, 15 post-processing effects on currently visible objects (landscape, sky, water), and several shader and GeoState effects. The medium effect usage run is done with 2 full screen and 5 currently visible object based post-processing effects.

The small difference of the first and second run show the low performance impact of our system extension. The high performance penalty for 4 pipes in run 3 to 6 is mainly caused by the two additional render passes (synchronization issues). To reduce this effect, we intend to introduce optimizations of the graphics pipes, like assigning processes to processors and sharing GL context for the pipes for the left and right eye, to reduce the texture load per PCIe interface. Nevertheless, the relatively small differences of run 4 to 6 show that high quality scenes and a large amount of effects are possible using a 4 pipe display.

7 CONCLUSION

We introduced a system, which is capable of dynamically manipulating a VE scene by adding effects for specific objects, areas, or the whole scene. The implemented system provides an intuitive interface allowing moderators, authors, or automated systems to modify the scene online using all available effects. It also provides an internal effect definition and mapping interface, which makes it possible to easily create new effects for already implemented effectors, like shaders or post-processing, and which maps technical variables to meaningful effect parameters. We introduced and discussed our implementation of the system, currently supporting Avango as a VR-system.

Our implemented visual effects have demonstrated the effectiveness and flexibility of the concept and our implementation. The full potential of the system for multimodal and varying content will show, when more modalities are supported and complex multimodal effect combinations can be generated. Effect performance tests with a complex scene showed that our implementation can enhance scenes with many real-time changeable effects without substantially limiting the performance.

Future work includes effects for the auditive and haptic modality, as well as the addition of several external effects like generating wind and ambient lighting. We are currently working on introducing a mapping of effects to moods they might produce. As a result of that, a target mood selection will cause appropriate effects and parameterizations to be used. As the impact of effects depends on scene context and users, this mapping is a big challenge. Integrating such a system with user assessment systems will make it possible to directly react with specific effects to the current state of the user.

REFERENCES

[1] M. Billinghurst, S. Baldis, L. Matheson, and M. Philips. 3d palette: A virtual reality content creation tool. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pages 155–156, Lausanne, Switzerland, 1997.

[2] S. Conrad, H. Krüger, and M. Haringer. Live tuning of virtual environments: The vr-tuner. In *Virtual environments 2004. 10th Eurographics Symposium on Virtual Enviroments*, pages 123–128, Grenoble, France, 2004.

[3] G. Eckel. Immersive audio-augmented environments - the listen project. In *Proceedings of the 5th International Conference on Information Visualization (IV2001)*, 2001.

[4] M. Green. Towards virtual environment authoring tools for content developers. In *VRST '03: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 117–123, New York, NY, USA, 2003.

[5] R. Holm, E. Stauder, R. Wagner, M. Priglinger, and J. Volkert. A combined immersive and desktop authoring tool for virtual environments. In *VR '02: Proceedings of the IEEE Virtual Reality Conference 2002*, page 93, Washington, DC, USA, 2002.

[6] E. Kalogerakis, S. Christodoulakis, and N. Moumoutzis. Coupling ontologies with graphics content for knowledge driven visualization. In *Proceedings of the IEEE Virtual Reality Conference 2006*, pages 43–50, 2006.

[7] M. E. Latoschik, P. Biermann, and I. Wachsmuth. Knowledge in the loop: Semantics representation for multimodal simulative environments. In *Smart Graphics*, volume 3638/2005 of *Lecture Notes in Computer Science*, pages 25–39. 2005.

[8] J. L. Mitchell, G. Mc Taggart, and C. Green. Shading in valve's source engine. In *SIGGRAPH Course on Advanced Real-Time Rendering in 3D Graphics and Games*, 2006.

[9] X. Qian, Z. Zhao, and R. Thorn. Rapid development of virtual environments a systematic approach for interactive design of 3d graphics. In *WSCG 2007, SHORT COMMUNICATIONS PROCEEDINGS I AND II*, pages 117–124, W Bohemia, Plzen, CZECH REPUBLIC, 2007.

[10] N. Redmond and J. Dingliana. A hybrid technique for creating meaningful abstractions of dynamic 3d scenes in real-time. In *WSCG 2008, FULL PAPERS*, pages 105–112, Univ W Bohemia, Plzen, CZECH REPUBLIC, 2008.

[11] J. Rohlf and J. Helman. Iris performer: a high performance multiprocessing toolkit for real-time 3d graphics. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 381–394, New York, NY, USA, 1994.

[12] H. Tramberend. Avocado: A distributed virtual reality framework. In *Procedings of IEEE Virtual Reality 1999 (IEEE VR 1999)*, pages 14–21, 1999.

Simulation of Massive Multibody Systems using GPU Parallel Computation

Alessandro Tasora
University of Parma
Dept. of Industrial Engineering
V.G.Usberti 181/A
43100, Parma, Italy
tasora@ied.unipr.it

Dan Negrut
University of Wisconsin–Madison
Dept. of Mechanical Engineering
1513 University Avenue
Madison, WI 53706, US
negrut@wisc.edu

Mihai Anitescu
Argonne National Laboratories
MCS division
9700 South Cass Avenue
Argonne, IL 60439, US
anitescu@mcs.anl.gov

Hammad Mazhar
University of Wisconsin–Madison
Dept. of Mechanical Engineering
1513 University Avenue
Madison, WI 53706, US
hmazhar@wisc.edu

Toby David Heyn
University of Wisconsin–Madison
Dept. of Mechanical Engineering
1513 University Avenue
Madison, WI 53706, US
heyn@wisc.edu

ABSTRACT

We describe an efficient method for the simulation of complex scenarios with millions of frictional contacts and mechanical constraints. To this end, the GPU processors of the modern graphic boards are used to solve the differential inclusion problem that represents the most challenging part of the multi–rigid–body problem. Thank to the massive parallelism offered by GPU boards, we are able to simulate sand, granular materials, soils and other complex physical scenarios with a large speedup respect to serial CPU–based algorithms.

Keywords

Constraints, dynamics, GPU, parallel computing, collision detection

1 INTRODUCTION

The simulation of the dynamics of multi-rigid-body systems is an useful tool in many areas, such as CAD, engineering, virtual reality, videogames and in computergraphics in general (for instance, when physical simulation is used for special effects in 3D movies).

Devices composed of rigid bodies interacting through frictional contacts and mechanical joints represent a numerical challenge because of the discontinuous nature of the motion; the dynamics is nonsmooth because of the discontinuous nature of noninterpenetration, collision, and adhesion constraints. Actually, the requirement that parts must be rigid increases the difficulty of the problem respect to the case of flexible parts such as in spring-based approaches.

Even mechanisms composed of few hundreds of parts and constraints may require lot of computational efforts; indeed, more complicated scenarios such as vehicles running on pebbles and sand such as in Fig. 1, soil and rock dynamics, flow and packing of granular materials, could require too long computational times. Results reported in [Mad07a] indicate that the most widely used commercial software for multibody dynamics runs into significant difficulties when handling simple problems involving hundreds of contact events, whereas cases with thousands of contacts become intractable. The method embraced in this work can solve efficiently problems with millions of contacts on a simple scalar CPU of the Pentium family, and an improved

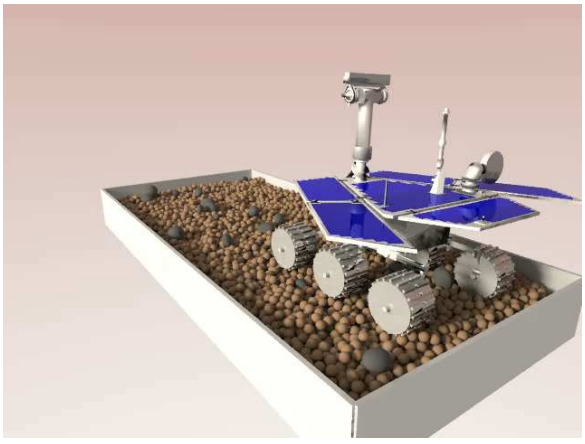


Figure 1: Simulation of a complex multi-rigid-body mechanism with contacts and joints.

performance can be obtained with the GPU version proposed herein, that can solve the contact dynamics with parallel computation.

Until recently, the massive computational power of parallel supercomputers has been available to a relatively small number of research groups, thus limiting the number of applications approached. This scenario is rapidly changing due to a trend set by general-purpose computing on the graphics processing unit (GPU). The CUDA libraries from NVIDIA allow to use the streaming microprocessors mounted in high-end graphics cards as general-purpose computing hardware. Presently, the raw computational power of these multiprocessors is measured in terms of Teraflops, that is hundreds of times the throughput of a modern scalar CPU.

Very few GPU projects are concerned with the dynamics of multibody systems and the two most significant are the Havok and the Ageia physics engines. Both are commercial and proprietary libraries used in the video-game industry. In this context, the goal of this work was to implement a general-purpose multibody solver on GPU multiprocessors backed by convergence results that guarantee the accuracy of the solution. Specifically, a parallel version was implemented of a numerical scheme presented in [Tas08a, Ani08a], which can robustly and efficiently approximate the bilaterally constrained dynamics of rigid bodies undergoing frictional contacts.

Unlike the so-called penalty or regularization methods, where the frictional interaction can be represented by a collection of stiff springs combined with damping elements that act at the interface of the two bodies, the approach embraced herein relies on a different mathematical framework. Specifically, the algorithms rely on time-stepping procedures producing weak solutions of the differential variational inequality (DVI) problem that describes the time evolution of rigid bodies with impact, contact, friction, and bilateral constraints. When compared to penalty-methods, the DVI approach has a greater algorithmic complexity, but avoids the small time steps that plague the former approach.

Early numerical methods based on DVI formulations can be traced back to [Mor83a, Lot82a, Mon93a], while the DVI formulation has been recently classified by differential index in [Pan03a]. Recent approaches based on time-stepping schemes have included both acceleration-force linear complementarity problem (LCP) approaches [Bar93a, Pan96a] and velocity-impulse LCP-based time-stepping methods [Ste96a, Ani97a, Ste00a]. Impulse-based methods, such as the one in [Ben07a], are becoming popular in the computer graphics field because of their robustness. The LCPs, obtained as a result of the introduction of inequalities in time-stepping schemes for DVI, coupled with a polyhedral approximation of the friction cone must be solved at each time step in order to determine the system state configuration as well as the Lagrange multipliers representing the reaction forces [Lot82a, Ste96a]. If the simulation entails a large number of contacts and rigid bodies, as is the case of granular materials, the computational burden of classical LCP solvers can become significant. Indeed, a well-known class of numerical methods for LCPs based on *simplex methods*, also known as *direct* or *pivoting* methods [Cot68a], may exhibit exponential worst-case complexity [Bar94a]. Moreover, the three-dimensional Coulomb friction case leads to a nonlinear complementarity problem (NCP): the use of a polyhedral approximation to transform the NCP into an LCP introduces unwanted anisotropy in friction cones [Ste96a, Ani97a].

In order to circumvent the limitations imposed by the use of classical LCP solvers and the limited accuracy associated with polyhedral approximations of the friction cone, a parallel fixed-point iteration method with projection on a convex set has been proposed, developed, and tested in [Ani08a]. The method is based on a time-stepping formulation that solves at every step a cone constrained optimization problem [Ani04a]. The time-stepping scheme has been proved to converge in a measure differential inclusion sense to the solution of the original continuous-time DVI. This paper illustrates how this problem can be solved in parallel by exploiting the parallel computational resources available on NVIDIA's GPU cards.

2 FORMULATION OF MULTIBODY DYNAMICS

The formulation of the equations of motion, that is the equations that govern the time evolution of a multibody system, is based on the so-called absolute, or Cartesian, representation of the attitude of each rigid body in the system.

The state of the system is denoted by the generalized positions $\mathbf{q} = \left[\mathbf{r}_1^T, \varepsilon_1^T, \ldots, \mathbf{r}_{n_b}^T, \varepsilon_{n_b}^T\right]^T \in \mathbb{R}^{7n_b}$ and their time derivatives $\dot{\mathbf{q}} = \left[\dot{\mathbf{r}}_1^T, \dot{\varepsilon}_1^T, \ldots, \dot{\mathbf{r}}_{n_b}^T, \dot{\varepsilon}_{n_b}^T\right]^T \in \mathbb{R}^{7n_b}$, where n_b is the number of bodies, $\mathbf{r}_j$ is the absolute position of the center of mass of the j-th body and the quaternions ε_j are used to represent rotation and to avoid singularities. Instead of using quaternion derivatives in $\dot{\mathbf{q}}$, it is more advantageous to work with angular velocities: the method described will use the vector of generalized velocities $\mathbf{v} = \left[\dot{\mathbf{r}}_1^T, \bar{\omega}_1^T, \ldots, \dot{\mathbf{r}}_{n_b}^T, \bar{\omega}_{n_b}^T\right]^T \in \mathbb{R}^{6n_b}$. Note that the generalized velocity can be easily obtained as $\dot{\mathbf{q}} = \mathbf{L}(\mathbf{q})\mathbf{v}$, where $\mathbf{L}$ is a linear mapping that transforms each $\bar{\omega}_i$ into the corresponding quaternion derivative $\dot{\varepsilon}_i$ by means of the linear algebra formula $\dot{\varepsilon}_i = \frac{1}{2}\mathbf{G}^T(\mathbf{q})\bar{\omega}_i$, with 3x4 matrix $\mathbf{G}(\mathbf{q})$ as defined in [Sha05a].

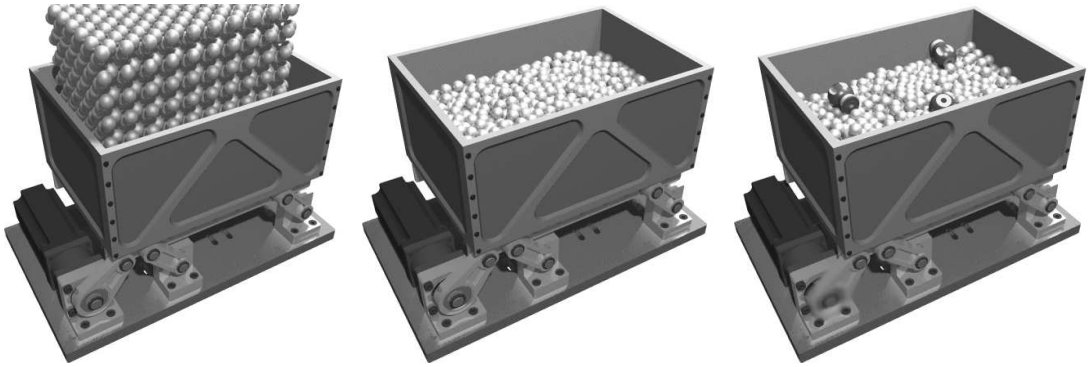


Figure 2: The proposed method can simulate the dynamics of devices with motors, joints and contacts, such as in the case of this size segregation machine that shakes thousands of steel spheres.

We denote by $\mathbf{f}(t,\mathbf{q},\mathbf{v})$ the set of applied, or external, generalized forces.

Bilateral constraints

Bilateral constraints represent kinematic pairs, for example spherical, prismatic or revolute joints, and can be expressed as algebraic equations constraining the relative position of two bodies. Assuming a set $\mathtt{B}$ of constraints is present in the system, they lead to the scalar equations $\Psi_i(\mathbf{q},t)=0, \quad i\in\mathtt{B}$. Assuming smoothness of constraint manifold, $\Psi_i(\mathbf{q},t)$ can be differentiated to obtain the Jacobian $\nabla_q\Psi_i=[\partial\Psi_i/\partial\mathbf{q}]^T$.

Constraints are consistent at velocity-level provided that $\nabla\Psi_i^T\mathbf{v}+\frac{\partial\Psi_i}{\partial t}=0$, where $\nabla\Psi_i^T=\nabla_q\Psi_i^T\mathbf{L}(\mathbf{q})$.

Contacts with friction

Given a large number of rigid bodies with different shapes, modern collision detection algorithms are able to find efficiently a set of contact points, that is points where a *gap function* $\Phi(\mathbf{q})$ can be defined for each pair of near-enough shape features. Where defined, such a gap function must satisfy the non-penetration condition $\Phi(\mathbf{q})\geq 0$ for all contact points.

Note that a signed distance function, differentiable at least up to some value of the interpenetration, can be easily defined if bodies are smooth and convex [Gue03a]. However, this is not always possible, for instance when dealing with concave or faceted shapes often used to represent parts of mechanical devices.

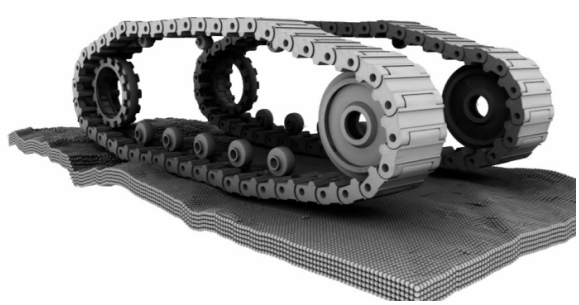


Figure 3: Simulation of a tracked vehicle on a granular soil: we used the GPU for both dynamics and collision detection between tracks, sprockets and pebbles.

When a contact i is active, that is $\Phi_i(\mathbf{q})=0$, a normal force and a tangential friction force act on each of the two bodies at the contact point. We use the classical Coulomb friction model to define these forces [Ani97a]. If the contact is not active, that is $\Phi_i(\mathbf{q})>0$, no contact forces can exist, and viceversa: this is the Signorini condition $\Phi_i(\mathbf{q})\geq 0$, $\widehat{\gamma}_{i,n}\geq 0$, $\Phi_i(\mathbf{q})\widehat{\gamma}_{i,n}=0$ that can be expressed using the complementarity notation [Ste96a]: $\Phi_i(\mathbf{q})\geq 0 \perp \widehat{\gamma}_{i,n}\geq 0$.

Given two bodies in contact A and B, let $\mathbf{n}_i$ be the normal at the contact pointing toward the exterior of the body of lower index, which by convention is considered to be body A. Let $\mathbf{u}_i$ and $\mathbf{w}_i$ be two vectors in the contact plane such that $\mathbf{n}_i,\mathbf{u}_i,\mathbf{w}_i\in\mathbb{R}^3$ are mutually orthonormal vectors.

The frictional contact force is impressed on the system by means of multipliers $\widehat{\gamma}_{i,n}\geq 0$, $\widehat{\gamma}_{i,u}$, and $\widehat{\gamma}_{i,w}$, which lead to the normal component of the force $\mathbf{F}_{i,N}=\widehat{\gamma}_{i,n}\mathbf{n}_i$ and the tangential component of the force $\mathbf{F}_{i,T}=\widehat{\gamma}_{i,u}\mathbf{u}_i+\widehat{\gamma}_{i,w}\mathbf{w}_i$.

The Coulomb model imposes the following nonlinear constraints:

$$\begin{aligned}
\widehat{\gamma}_{i,n} &\geq 0 \quad \perp \quad \Phi_i(\mathbf{q})\geq 0 \\
\mu_i\widehat{\gamma}_{i,n} &\geq \sqrt{\widehat{\gamma}_{i,u}^2+\widehat{\gamma}_{i,w}^2} \\
&\quad \langle\mathbf{F}_{i,T},\mathbf{v}_{i,T}\rangle=-||\mathbf{F}_{i,T}||\,||\mathbf{v}_{i,T}|| \\
&\quad ||\mathbf{v}_{i,T}||\left(\mu_i\widehat{\gamma}_{i,n}-\sqrt{\widehat{\gamma}_{i,u}^2+\widehat{\gamma}_{i,w}^2}\right)=0
\end{aligned}$$

where $\mathbf{v}_{i,T}$ is the relative tangential velocity.The constraint $\langle\mathbf{F}_{i,T},\mathbf{v}_{i,T}\rangle=-||\mathbf{F}_{i,T}||\,||\mathbf{v}_{i,T}||$ requires that the tangential force be opposite to the tangential velocity. Note that the friction force depends on the friction coefficient $\mu_i\in\mathbb{R}^+$.

An equivalent convenient way of expressing this constraint is by using the maximum dissipation principle:

$$(\widehat{\gamma}_{i,u},\widehat{\gamma}_{i,w})=\underset{\sqrt{\widehat{\gamma}_{i,u}^2+\widehat{\gamma}_{i,w}^2}\leq\mu_i\widehat{\gamma}_{i,n}}{\operatorname{argmin}} \mathbf{v}_{i,T}^T(\widehat{\gamma}_{i,u}\mathbf{u}_i+\widehat{\gamma}_{i,w}\mathbf{w}_i). \quad (1)$$

In fact, the the first-order necessary Karush-Kuhn-Tucker conditions for the minimization problem

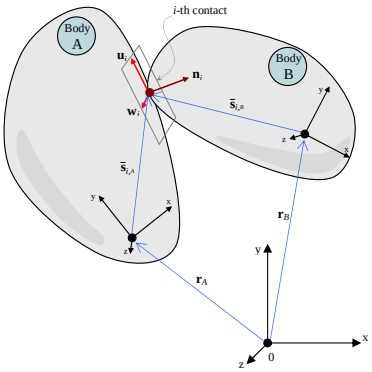


Figure 4: Contact i between two bodies $A,B \in \{1,2,\ldots,n_b\}$

(1) correspond to the Coulomb model above [Mor88a, Mon93a].

The complete model

Considering the effects of both the set $\mathtt{A}$ of frictional contacts and the set $\mathtt{B}$ of bilateral constraints, the time evolution of the dynamical system is governed by the following differential variational inequality (a differential problem with set-valued functions and complementarity constraints):

$$\begin{aligned}
\dot{\mathbf{q}} &= \mathbf{L}(\mathbf{q})\mathbf{v} \\
\mathbf{M}\dot{\mathbf{v}} &= \mathbf{f}(t,\mathbf{q},\mathbf{v}) + \sum_{i\in\mathtt{B}} \widehat{\gamma}_{i,b}\nabla\Psi_i + \\
&\quad + \sum_{i\in\mathtt{A}} \left(\widehat{\gamma}_{i,n}\mathbf{D}_{i,n} + \widehat{\gamma}_{i,u}\mathbf{D}_{i,u} + \widehat{\gamma}_{i,w}\mathbf{D}_{i,w}\right) \\
i\in\mathtt{B} &: \Psi_i(\mathbf{q},t) = 0 \\
i\in\mathtt{A} &: \widehat{\gamma}_{i,n} \geq 0 \perp \Phi_i(\mathbf{q}) \geq 0, \quad \text{and} \\
(\widehat{\gamma}_{i,u},\widehat{\gamma}_{i,w}) &= \operatorname*{argmin}_{\mu_i\widehat{\gamma}_{i,n}\geq\sqrt{\widehat{\gamma}_{i,u}^2+\widehat{\gamma}_{i,w}^2}} \mathbf{v}^T\left(\widehat{\gamma}_{i,u}\mathbf{D}_{i,u} + \widehat{\gamma}_{i,w}\mathbf{D}_{i,w}\right)
\end{aligned} \tag{2}$$

The tangent space generators $\mathbf{D}_i = [\mathbf{D}_{i,n}, \mathbf{D}_{i,u}, \mathbf{D}_{i,w}] \in \mathbb{R}^{6n_b\times 3}$ are sparse and are defined given a pair of contacting bodies A and B as:

$$\mathbf{D}_i^T = \begin{bmatrix} \mathbf{0} & \ldots & -\mathbf{A}_{i,p}^T & \mathbf{A}_{i,p}^T\mathbf{A}_A\tilde{\mathbf{s}}_{i,A} & \mathbf{0} & \ldots \\ \mathbf{0} & \ldots & \mathbf{A}_{i,p}^T & -\mathbf{A}_{i,p}^T\mathbf{A}_B\tilde{\mathbf{s}}_{i,B} & \mathbf{0} & \ldots \end{bmatrix} \tag{3}$$

where we use $\mathbf{A}_{i,p} = [\mathbf{n}_i, \mathbf{u}_i, \mathbf{w}_i]$ as the $\mathbb{R}^{3\times3}$ matrix of the local coordinates of the ith contact, and introduce the vectors $\bar{\mathbf{s}}_{i,A}$ and $\bar{\mathbf{s}}_{i,B}$ as contact point positions in body coordinates, see Fig. (4), with skew matrices $\tilde{\mathbf{s}}_{i,A}$ and $\tilde{\mathbf{s}}_{i,B}$.

3 THE TIME-STEPPING SCHEME

We formulate the dynamical problem in terms of measure differential inclusions [Ste00a], whose numerical solution can be obtained using the following time-stepping scheme based on the solution of a complementarity problem at each time step.

Given a position $\mathbf{q}^{(l)}$ and velocity $\mathbf{v}^{(l)}$ at the time-step $t^{(l)}$, the numerical solution is found at the new time-step $t^{(l+1)} = t^{(l)} + h$ by solving the following optimization problem with equilibrium constraints [Tas08a]:

$$\mathbf{M}(\mathbf{v}^{(l+1)} - \mathbf{v}^{(l)}) = h\mathbf{f}(t^{(l)},\mathbf{q}^{(l)},\mathbf{v}^{(l)}) + \sum_{i\in\mathtt{B}} \gamma_{i,b}\nabla\Psi_i + \\ + \sum_{i\in\mathtt{A}} \left(\gamma_{i,n}\mathbf{D}_{i,n} + \gamma_{i,u}\mathbf{D}_{i,u} + \gamma_{i,w}\mathbf{D}_{i,w}\right), \tag{4}$$

$$i\in\mathtt{B} : \quad \tfrac{1}{h}\Psi_i(\mathbf{q}^{(l)},t) + \nabla\Psi_i^T\mathbf{v}^{(l+1)} + \tfrac{\partial\Psi_i}{\partial t} = 0 \tag{5}$$

$$i\in\mathtt{A} : \quad 0 \leq \tfrac{1}{h}\Phi_i(\mathbf{q}^{(l)}) + \mathbf{D}_{i,n}^T\mathbf{v}^{(l+1)} \perp \gamma_n^i \geq 0, \tag{6}$$

$$(\gamma_{i,u},\gamma_{i,w}) = \operatorname*{argmin}_{\mu_i\gamma_{i,n}\geq\sqrt{\gamma_{i,u}^2+\gamma_{i,w}^2}} \mathbf{v}^T\left(\gamma_{i,u}\mathbf{D}_{i,u} + \gamma_{i,w}\mathbf{D}_{i,w}\right) \tag{7}$$

$$\mathbf{q}^{(l+1)} = \mathbf{q}^{(l)} + h\mathbf{L}(\mathbf{q}^{(l)})\mathbf{v}^{(l+1)}. \tag{8}$$

Here, γ_s represents the constraint impulse of a contact constraint, that is, $\gamma_s = h\widehat{\gamma}_s$, for $s = n,u,w$. The $\frac{1}{h}\Phi_i(\mathbf{q}^{(l)})$ term achieves constraint stabilization, and its effect is discussed in [Ani03a]. Similarly, the term $\frac{1}{h}\Phi_i(\mathbf{q}^{(l)})$ achieves stabilization for bilateral constraints. The scheme converges to the solution of a measure differential inclusion [Ani04a] when the step size $h \to 0$.

Several numerical methods can be used to solve (4)-(7) [Buc98a]. Our approach casts the problem as a monotone optimization problem by introducing a relaxation over the complementarity constraints, replacing Eq. (6) with $i\in\mathtt{A} : 0 \leq \frac{1}{h}\Phi_i(\mathbf{q}^{(l)}) + \mathbf{D}_{i,n}^T\mathbf{v}^{(l+1)} - \mu_i\sqrt{(\mathbf{v}^T\mathbf{D}_{i,u})^2 + (\mathbf{v}^T\mathbf{D}_{i,w})^2} \perp \gamma_n^i \geq 0$. The solution of the modified time-stepping scheme will approach the solution of the same measure differential inclusion for $h \to 0$ as the original scheme [Ani04a].

Previous work [Ani08a] showed that the modified scheme is a Cone Complementarity Problem (CCP), which can be solved efficiently by an iterative numerical method that rely on projected contractive maps. Omitting for brevity some of the details discussed in [Ani08a], introducing $\gamma_i = \{\gamma_{i,n},\gamma_{i,u},\gamma_{i,w}\}^T, i\in\mathtt{A}$, the algorithm makes use of the following vectors:

$$\tilde{\mathbf{k}} \equiv \mathbf{M}\mathbf{v}^{(l)} + h\mathbf{f}(t^{(l)},\mathbf{q}^{(l)},\mathbf{v}^{(l)}) \tag{9}$$

$$\mathbf{b}_i \equiv \left\{\tfrac{1}{h}\Phi_i(\mathbf{q}^{(l)}),0,0\right\}^T \quad i\in\mathtt{A}\ , \tag{10}$$

$$b_i \equiv \tfrac{1}{h}\Psi_i(\mathbf{q}^{(l)},t) + \tfrac{\partial\Psi_i}{\partial t}, \quad i\in\mathtt{B} \tag{11}$$

The solution, in terms of dual variables of the CCP (the multipliers), is obtained by iterating the following contraction maps until convergence:

$$\forall i\in\mathtt{A} : \quad \gamma_i^{r+1} = \Pi_{\Upsilon_i}\left[\gamma_i^r - \omega\eta_i\left(\mathbf{D}_i^T\mathbf{v}^r + \mathbf{b}_i\right)\right] \tag{12}$$

$$\forall i\in\mathtt{B} : \quad \gamma_i^{r+1} = \gamma_i^r - \omega\eta_i\left(\nabla\Psi_i^T\mathbf{v}^r + b_i\right) \tag{13}$$

At each iteration r, before repeating (12) and (13), also the primal variables (the velocities) are updated as:

$$\mathbf{v}^{r+1} = \mathbf{M}^{-1}\left(\sum_{z\in\mathtt{A}} \mathbf{D}_z\gamma_z^{r+1} + \sum_{z\in\mathtt{B}} \nabla\Psi_z\gamma_z^{r+1} + \tilde{\mathbf{k}}\right) \tag{14}$$

Note that the superscript $(l+1)$ was omitted for brevity.

The iterative process uses the metric projector $\Pi_{\Upsilon_i}(\cdot)$ [Tas08a], which is a non-expansive map $\Pi_{\Upsilon_i} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ acting on the triplet of multipliers associated with the *i*-th contact. Thus, if the multipliers fall into the friction cone, they are not modified; if they are in the polar cone, they are set to zero; in the remaining cases they are projected orthogonally onto the surface of the friction cone. The overrelaxation factor ω and η_i parameters are adjusted to control the convergence. Interested readers are referred to [Ani08a] for a proof of the convergence of this method.

The previous algorithm has been implemented on serial computing architectures and proved to be reliable and efficient. In the following the time-consuming part of the methodology, that is the CCP iteration, will be reformulated to take advantage of the parallel computing resources available on GPU boards.

4 PARALLEL SOLVER ON THE GPU

Modern GPU processors can execute thousands of threads in parallel, providing computating power in terms of Teraflops. These processors, usually devoted to the execution of pixel shading fragments for three dimensional visualization, can be also exploited for scientific computation thank to development environments such as CUDA from NVIDIA, that provides C++ functions to easily manage GPU data buffers and *kernels*, that is operations to executed in parallel on the data. The proposed algorithm fits well into the GPU multithreaded model because the computation can be split in multiple threads each acting on a single contact, or kinematic constraint, or rigid body depending on the stage of the computation.

Buffers for data structures

In the proposed approach, the data structures on the GPU are implemented as large arrays (*buffers*) to match the execution model associated with NVIDIA's CUDA. Specifically, threads are grouped in rectangular thread blocks, and thread blocks are arranged in rectangular grids. Four main buffers are used: the contacts buffer, the constraints buffer, the reduction buffer, and the bodies buffer.

Special care should be paid to minimize the memory overhead caused by repeated transfers of large data structures: we organized data structures in a way that minimizes the number of fetch and store operations and maximizes the arithmetic intensity of the kernel code, as recommended by the CUDA development guidelines.

The data structure for the contacts has been mapped into columns of four floats as shown in Fig. 5. Each contact will reference its two touching bodies through the two pointers B_A and B_B, in the fourth and seventh rows of the contact data structure.

There is no need to store the entire $\mathbf{D}_i$ matrix for the i^{th} contact because it has zero entries for most of its part, except for the two 12x3 blocks corresponding to the coordinates of the two bodies in contact. In fact, once the velocities of the two bodies $\dot{\mathbf{r}}_{A_i}$, ω_{A_i}, $\dot{\mathbf{r}}_{B_i}$ and ω_{B_i} have been fetched, the product $\mathbf{D}_i^T\mathbf{v}^r$ in Eq.(12) can be performed as

$$\mathbf{D}_i^T\mathbf{v}^r = \mathbf{D}_{i,v_A}^T\dot{\mathbf{r}}_{A_i} + \mathbf{D}_{i,\omega_A}^T\omega_{A_i} + \mathbf{D}_{i,v_B}^T\dot{\mathbf{r}}_{B_i} + \mathbf{D}_{i,\omega_B}^T\omega_{B_i} \tag{15}$$

with the adoption of the following 3x3 matrices

$$\begin{array}{lllllll} \mathbf{D}_{i,v_A}^T & = & -\mathbf{A}_{i,p}^T, & & \mathbf{D}_{i,\omega_A}^T & = & \mathbf{A}_{i,p}^T\mathbf{A}_A\tilde{\mathbf{s}}_{i,A} \\ \mathbf{D}_{i,v_B}^T & = & \mathbf{A}_{i,p}^T, & & \mathbf{D}_{i,\omega_B}^T & = & -\mathbf{A}_{i,p}^T\mathbf{A}_B\tilde{\mathbf{s}}_{i,B} \end{array} \tag{16}$$

Since $\mathbf{D}_{i,v_A}^T = -\mathbf{D}_{i,v_B}^T$, there is no need to store both matrices, so in each contact data structure only a matrix $\mathbf{D}_{i,v_{AB}}^T$ is stored, which is then used with opposite signs for each of the two bodies.

Also the velocity update vector $\Delta\mathbf{v}_i$, needed for the sum in Eq.(14) is sparse: it can be decomposed in small subvectors. Specifically, given the masses and the inertia tensors of the two bodies m_{A_i}, m_{B_i}, $\mathbf{J}_{A_i}$ and $\mathbf{J}_{B_i}$, the term $\Delta\mathbf{v}_i$ will be computed and stored in four parts as follows:

$$\begin{array}{ll} \Delta\dot{\mathbf{r}}_{A_i} = m_{A_i}^{-1}\mathbf{D}_{i,v_A}\Delta\gamma_i^{r+1}, & \Delta\omega_{A_i} = \mathbf{J}_{A_i}^{-1}\mathbf{D}_{i,\omega_A}\Delta\gamma_i^{r+1} \\ \Delta\dot{\mathbf{r}}_{B_i} = m_{B_i}^{-1}\mathbf{D}_{i,v_B}\Delta\gamma_i^{r+1}, & \Delta\omega_{B_i} = \mathbf{J}_{B_i}^{-1}\mathbf{D}_{i,\omega_B}\Delta\gamma_i^{r+1} \end{array} \tag{17}$$

Note that those four parts of the $\Delta\mathbf{v}_i$ terms are not stored in the *i*-th contact data structure or in the data structure of the two referenced bodies (because multiple contacts may refer the same body, hence they would overwrite the same memory position). These velocity updates are instead stored in the reduction buffer, which will be used to efficiently perform the summation in Eq.(14). This will be discussed shortly.

The constraints buffer, shown in Fig. 6, is based on a similar concept. Jacobians $\nabla\Psi_i$ of all scalar constraints are stored in a sparse format, each corresponding to four rows $\nabla\Psi_{i,v_A}$, $\nabla\Psi_{i,\omega_A}$, $\nabla\Psi_{i,v_B}$, $\nabla\Psi_{i,\omega_B}$. Therefore the product $\nabla\Psi_i^T\mathbf{v}^r$ in Eq.(13) can be performed as the scalar value $\nabla\Psi_i^T\mathbf{v}^r = \nabla\Psi_{i,v_A}^T\dot{\mathbf{r}}_{A_i} + \nabla\Psi_{i,\omega_A}^T\omega_{A_i} + \nabla\Psi_{i,v_B}^T\dot{\mathbf{r}}_{B_i} + \nabla\Psi_{i,\omega_B}^T\omega_{B_i}$. Also, the four parts of the sparse vector $\Delta\mathbf{v}_i$ can be computed and stored as

$$\begin{array}{ll} \Delta\dot{\mathbf{r}}_{A_i} = m_{A_i}^{-1}\nabla\Psi_{i,v_A}\Delta\gamma_i^{r+1}, & \Delta\omega_{A_i} = \mathbf{J}_{A_i}^{-1}\nabla\Psi_{i,\omega_A}\Delta\gamma_i^{r+1} \\ \Delta\dot{\mathbf{r}}_{B_i} = m_{B_i}^{-1}\nabla\Psi_{i,v_B}\Delta\gamma_i^{r+1}, & \Delta\omega_{B_i} = \mathbf{J}_{B_i}^{-1}\nabla\Psi_{i,\omega_B}\Delta\gamma_i^{r+1} \end{array} \tag{18}$$

Figure 7 shows that each body is represented by a data structure containing the state (velocity and position), the mass moments of inertia and mass values, and the external applied force $\mathbf{F}_j$ and torque $\mathbf{C}_j$. Note that to speed the iteration, it is advantageous to store the inverse of the mass and inertias rather than their original values, because the operation $\mathbf{M}^{-1}\mathbf{D}_i\Delta\gamma_i^{r+1}$ must be performed multiple times.

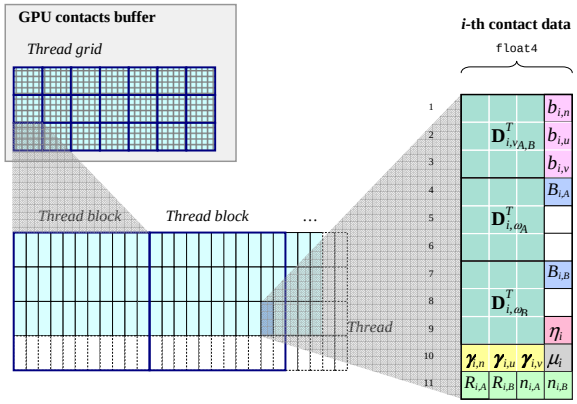


Figure 5: Grid of data structures for frictional contacts, in GPU memory.

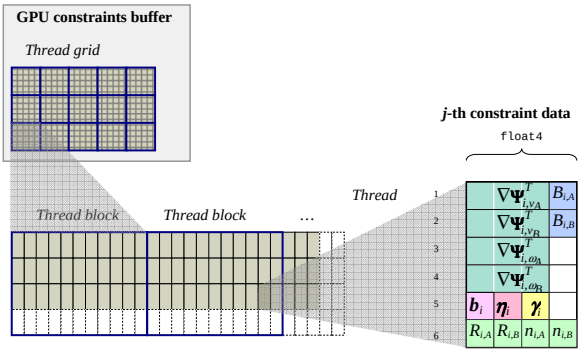


Figure 6: Grid of data structures for scalar constraints, in GPU memory.

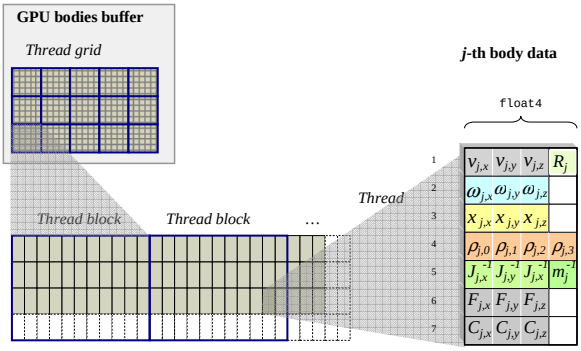


Figure 7: Grid of data structures for rigid bodies, in GPU memory.

The Parallel Algorithm

A parallelization of computations in Eq.(12) and Eq.(13) is easily implemented, by simply assigning one contact per thread (and, similarly, one constraint per thread). In fact the results of these computations would not overlap in memory, and it will never happen that two parallel threads need to write in the same memory location at the same time. These are the two most numerically-intensive steps of the CCP solver,

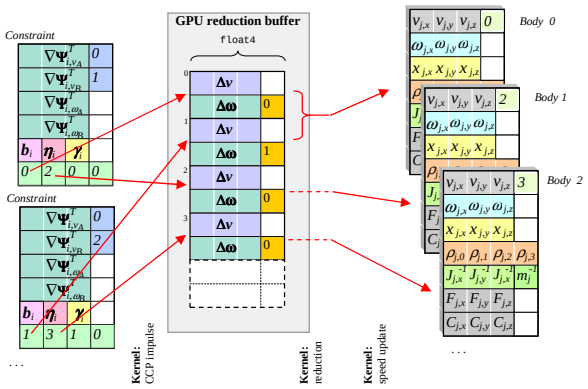


Figure 8: The reduction buffer avoids race conditions in parallel updates of the same body state.

called the **CCP contact iteration kernel** and the **CCP constraint iteration kernel**.

However, the sums in Eq.(14) cannot be performed with embarrassingly-parallel implementations: it may happen that two or more contacts need to add their velocity updates to the same rigid body. A possible approach to overcome this problem is presented in [Har07a], for a similar problem. We adopted an alternative method, with higher generality, based on the *parallel segmented scan* algorithm [Sen07a] that operates on an intermediate reduction buffer (Fig.8); this method sums the values in the buffer using a binary-tree approach that keeps the computational load well balanced among the many thread processors. In the example of Fig.8, the first constraint refers to bodies 0 and 1, the second to bodies 0 and 2; multiple updates to body 0 are then accumulated with parallel segmented reduction.

Since collision detection is the biggest computational overhead after the CCP solution, we also developed a GPU-based parallel code for collision detection, obtaining a 20x speedup factor when compared to the serial code of the Bullet library. The GPU collision code requires the use of multiple kernels and complex data structures that we cannot describe here because of limited space; details are available in [Maz09a].

The following pseudocode shows the sequence of main computational phases at each time step, for the most part executed as parallel kernels on the GPU.

Algorithm 1: Time Stepping using GPU.

1. (*GPU or host*) Perform collision detection between bodies, obtaining $n_{\mathbb{A}}$ possible contact points within a distance δ, as contact positions $s_{i,A}$, $s_{i,B}$ on the two touching surfaces, and normals $\mathbf{n}_i$.

2. (*Host, serial*) If needed, copy contact and body data structures from host memory to GPU buffers. Copy also constraint data (residuals b_i and jacobians) into the constraint buffer.

3. (*GPU, body-parallel*) **Force kernel**. For each body, compute forces $\mathbf{f}(t^{(l)}, \mathbf{q}^{(l)}, \mathbf{v}^{(l)})$, if any (example, gravity). Store these forces and torques into F_j and C_j.

4. (*GPU, contact-parallel*) **Contact preprocessing kernel**. For each contact, given contact normal and position, compute in-place the matrices $\mathbf{D}^T_{i,v_A}$, $\mathbf{D}^T_{i,\omega_A}$ and $\mathbf{D}^T_{i,\omega_B}$, then compute η_i and the contact residual $\mathbf{b}_i = \{\frac{1}{h}\Phi_i(\mathbf{q}), 0, 0\}^T$.

5. (*GPU, body-parallel*) **CCP force kernel**. For each body j, initialize body velocities: $\dot{\mathbf{r}}_j^{(l+1)} = h\, m_j^{-1}\mathbf{F}_j$ and $\omega_j^{(l+1)} = h\, \mathbf{J}_j^{-1}\mathbf{C}_j$.

6. (*GPU, contact-parallel*) **CCP contact iteration kernel**. For each contact *i*, do $\gamma_i^{r+1} = \lambda\ \Pi_{\Upsilon_i}\left(\gamma_i^r - \omega\eta_i\left(\mathbf{D}_i^T\mathbf{v}^r + \mathbf{b}_i\right)\right) + (1-\lambda)\gamma_i^r$. Note that $\mathbf{D}_i^T\mathbf{v}^r$ is evaluated with sparse data, using Eq. (15). Store $\Delta\gamma_i^{r+1} = \gamma_i^{r+1} - \gamma_i^r$ in contact buffer. Compute sparse updates to the velocities of the two connected bodies *A* and *B*, and store them in the $R_{i,A}$ and $R_{i,B}$ slots of the reduction buffer.

7. (*GPU, constraint-parallel*) **CCP constraint iteration kernel**. For each constraint *i*, do $\gamma_i^{r+1} = \lambda\ \left(\gamma_i^r - \omega\eta_i\left(\nabla\Psi_i^T\mathbf{v}^r + b_i\right)\right) + (1-\lambda)\gamma_i^r$. Store $\Delta\gamma_i^{r+1} = \gamma_i^{r+1} - \gamma_i^r$ in contact buffer. Compute sparse updates to the velocities of the two connected bodies *A* and *B*, and store them in the $R_{i,A}$ and $R_{i,B}$ slots of the reduction buffer.

8. (*GPU, reduction-slot-parallel*) **Segmented reduction kernel**. Sum all the $\Delta\dot{\mathbf{r}}_i$, $\Delta\omega_i$ terms belonging to the same body, in the reduction buffer.

9. (*GPU, body-parallel*) **Body velocity updates kernel**. For each j body, add the cumulative velocity updates which can be fetched from the reduction buffer, using the index R_j.

10. Repeat from step 6 until convergence or until number of CCP steps reached $r > r_{max}$.

11. (*GPU, body-parallel*) **Time integration kernel**. For each j body, perform time integration as $\mathbf{q}_j^{(l+1)} = \mathbf{q}_j^{(l)} + h\mathbf{L}(\mathbf{q}_j^{(l)})\mathbf{v}_j^{(l+1)}$

12. (*Host, serial*) If needed, copy body, contact and constraint data structures from GPU to host memory.

5 IMPLEMENTATION AND RESULTS

The GPU iterative solver and the GPU collision detection have been embedded in our C++ simulation software Chrono::Engine. We tested the GPU-based parallel method with benchmark problems and compared it, in terms of computing time, with the serial method.

Number of bodies	CPU CCP [s]	GPU CCP [s]	Speedup CCP	Speedup CD
16000	7.11	0.57	12.59	4.67
32000	16.01	1.00	16.07	6.14
64000	34.60	1.97	17.58	10.35
128000	76.82	4.55	16.90	21.71

Table 1: Stress test of the GPU CCD solver and GPU collision detection.

For the results of Tab.1, we simulated densely packed spheres that flow from a silo. The CPU is an Intel Xeon 2.66 GHz, the GPU is an NVIDIA Tesla C1060. The simulation time increases linearly with the number of bodies in the model. The GPU algorithm is at least one order of magnitude faster than the serial algorithm.

Other stress tests were performed with even larger amounts of spheres, such as in the benchmark of Fig.10. Similarly, the test of Fig.9 simulates one million of rigid bodies inside a tank being shaken horizontally (the amount of available RAM on a single GPU board limited us to go beyond that limit).

Using the proposed GPU method we are already able to simulate granular soil (pebbles, sand) under the tracks of a vehicle, see Fig.3, in fact our GPU collision detection code is able to handle nonconvex shapes by performing spherical decomposition. To simulate larger scenarios, with smaller grains of sand, future efforts will address the possibility of using domain decomposition, with clusters of multiple GPU boards on multiple host.

6 CONCLUSIONS

A parallel numerical method has been proposed for the simulation of multibody mechanical systems with frictional contacts and bilateral constraints. The parallel method is based on an iterative approach that falls within the mathematical framework of measure differential inclusions and is backed by a rigorous convergence analysis.

Results obtained with the proposed method demonstrate that the GPU version of the dynamics solver is about 20x faster than the CPU version. A similar speedup has been obtained for the collision detection.

7 ACKNOWLEDGEMENTS

We thank the NVIDIA corporation for sponsoring our research programs. Financial support for A.Tasora is provided in part by the Italian Ministry of University under the PRIN grant 2007Z7K4ZB. Mihai Anitescu was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

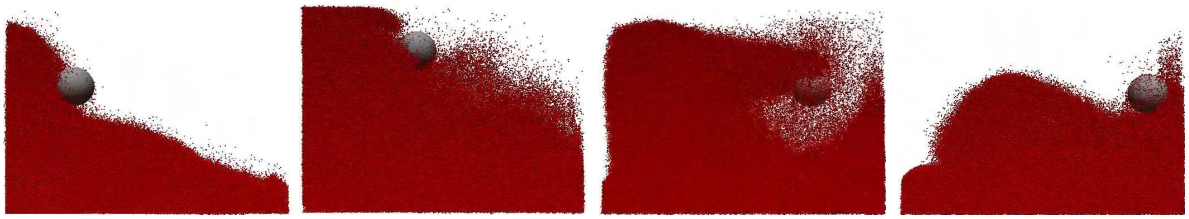


Figure 9: Benchmark with one million of rigid bodies with friction.

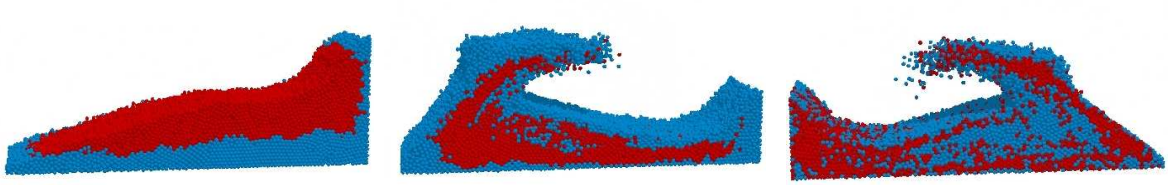


Figure 10: Benchmark: mixing of two granular materials.

REFERENCES

[Ani97a] Anitescu, M. and Potra, F.A. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14:231–247, 1997.

[Ani03a] Anitescu, M. and Hart, G.D. A constraint-stabilized time-stepping approach for rigid multibody dynamics with joints, contact and friction. *International Journal for Numerical Methods in Engineering*, 60(14):2335–2371, 2004.

[Ani04a] Anitescu, M. Optimization-based simulation of non-smooth rigid multibody dynamics. *Math. Program.*, 105(1):113–143, 2006.

[Ani08a] Anitescu, M. and Tasora, A. An iterative approach for cone complementarity problems for nonsmooth dynamics. *Computational Optimization and Applications*, 2008, in press.

[Bar93a] Baraff, D. Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica*, 10:292–352, 1993.

[Bar94a] Baraff, D. Fast contact force computation for nonpenetrating rigid bodies. In *Computer Graphics (Proceedings of SIGGRAPH)*, pages 23–34, 1994.

[Ben07a] Bender, J. Impulse-based dynamic simulation in linear time. *Journal of Computer Animation and Virtual Worlds*, 18(4,5):225–233, 2007

[Buc98a] Buck, M. and Schömer, E. Interactive Rigid Body Manipulation with Obstacle Contacts. In *6th WSCG Conf. in Central Europe on Computer Graphics and Visualization*, Plzen, 1998.

[Cot68a] Cottle, R.W. and Dantzig, G.B. Complementary pivot theory of mathematical programming. *Linear Algebra and Its Applications*, 1:103–125, 1968.

[Gil88a] Gilbert, E.G., Johnson, D.W. and Keerthi, S.S. A fast procedure for computing the distance between complex objectsin three-dimensional space. *Robotics and Automation, IEEE Journal of [see also IEEE Transactions on Robotics and Automation]*, 4(2):193–203, 1988.

[Gue03a] Guendelman, E., Bridson, R., and Fedkiw, R. Nonconvex rigid bodies with stacking. *ACM Trans. Graph.*, 22(3):871–878, 2003.

[Har07a] Harada, T. Real-time rigid body simulation on gpus. In Hubert Nguyen, editor, *GPU Gems 3*, chapter 23. Addison-Wesley, 2007.

[Lot82a] Lotstedt, P. Mechanical systems of rigid bodies subject to unilateral constraints. *SIAM Journal of Applied Mathematics*, 42(2):281–296, 1982.

[Mad07a] Madsen, J., Pechdimaljian, N. and Negrut, D. Penalty versus complementarity-based frictional contact of rigid bodies: A CPU time comparison. Technical Report TR-2007-05, SBEL, University of Wisconsin, Madison, 2007.

[Maz09a] Mazhar, H. GPU Collision Detection Using Spatial Subdivision With Applications In Contact Dynamics *Proceedings of ASME IDETC09*, San Diego, 2009.

[Mon93a] Monteiro Marques, M.D.P. *Differential Inclusions in Nonsmooth Mechanical Problems: Shocks and Dry Friction*, volume 9 of *Progress in Nonlinear Differential Equations and Their Applications*. Birkhäuser Verlag, Basel, Boston, Berlin, 1993.

[Mor83a] Moreau, Jean J. Standard inelastic shocks and the dynamics of unilateral constraints. In G. Del Piero and F. Macieri, editors, *Unilateral Problems in Structural Analysis*, pages 173–221, New York, CISM Courses and Lectures no. 288, Springer–Verlag, 1983.

[Mor88a] Moreau, Jean J. Unilateral contact and dry friction in finite freedom dynamics. In J. J. Moreau and P. D. Panagiotopoulos, editors, *Nonsmooth Mechanics and Applications*, pages 1–82, Berlin, Springer-Verlag, 1988.

[Pan96a] Pang, J.S. and Trinkle, J.C. Complementarity formulations and existence of solutions of dynamic multi-rigid-body contact problems with Coulomb friction. *Math. Program.*, 73(2):199–226, 1996.

[Pan03a] Pang, J.S. and Stewart, D.E. Differential variational inequalities. *Mathematical Programming*, 113(2):345–424, 2008.

[Sha05a] Shabana, A.A. *Dynamics of Multibody Systems*. Cambridge University Press, third edition, 2005.

[Ste96a] Stewart, D.E. and Trinkle, J.C. An implicit time-stepping scheme for rigid-body dynamics with inelastic collisions and Coulomb friction. *International Journal for Numerical Methods in Engineering*, 39:2673–2691, 1996.

[Ste98a] Stewart, D.E. Convergence of a time-stepping scheme for rigid body dynamics and resolution of Painleve's problems. *Arch. Rat. Mech. Anal.*, 145(3):215–260, 1998.

[Ste00a] Stewart, D.E. Rigid-body dynamics with friction and impact. *SIAM Review*, 42(1):3–39, 2000.

[Sen07a] Sengupta, S., Harris, M., Zhang, Y. and Owens, J.D. Scan Primitives for GPU Computing. *ACM Graphics Hardware*, 97–106, 2007.

[Tas08a] Tasora, A. A Fast NCP Solver for Large Rigid-Body Problems with Contacts, Friction, and Joints. In C. L. Bottasso Ed., *Computational Methods in Applied Sciences*, Springer, 12:45–55, 2008.

Simulating Atmospheric Pollution Weathering on Buildings

N. Mérillou	S. Mérillou	D. Ghazanfarpour	J.M. Dischler	E. Galin
XLIM / DMI	XLIM / DMI	XLIM / DMI	LSIIT,	LIRIS,
83 rue d'isle	83 rue d'isle	83 rue d'isle	Pôle API, Bd Séb. Brant	8, Bd Niels Bohr
87000 Limoges, France	87000 Limoges, France	87000 Limoges, France	67400 Illkirch, France	69622 Villeurbanne
nicolas.merillou@xlim.fr	stephane.merillou@unilim.fr	ghazanfarpour@unilim.fr	dischler@dpt-info.u-strasbg.fr	eric.galin@univ-lyon2.fr

ABSTRACT

Interactions between polluted atmosphere and materials lead to an early aging of numerous buildings and monuments. This weathering process leads to important changes in appearance, from color blackening to small-scale geometric alterations: a black crust grows onto parts of affected surfaces, depending on the geometry of the object as well as on its environment. In this paper, we present a method to simulate this very important weathering process. Our method is physically inspired and provides full control to designers, keeping plausible results. First, specific polluted zones are detected according to their real physical classification. Then, the modifications of aspect of each zone are computed. Our results demonstrate that our model matches well the observed behavior of real-world monuments and buildings affected by atmospheric pollution.

Keywords : weathering, natural phenomena, shading, texture.

1. INTRODUCTION

Nearly all real world objects are submitted to aggressive environmental conditions. They react to these conditions by changes in appearances, which is usually called weathering. To achieve high realism in image synthesis, weathering of materials must be simulated as it contributes critically to the visual richness of images [Dorsey2008]. A vast variety of techniques have been proposed so far, following two main tendencies: models accounting accurately for specific weathering processes (for example [Dorsey96]), and more generic models usually based on texture transfer or synthesis techniques [Lu2007]). When the aging processes to be simulated strongly depend on the geometry of affected objects, specific techniques generally give very good results [Merillou2008].

Because of the increase, in our society, of energy production and consumption due to industry and transport development, the effects of pollution on buildings and cultural monuments are increasingly studied in numerous scientific fields. Damaging of monuments due to pollution is worrying from both economical and cultural heritage points of view. Indeed, a lot of money is spent in cleaning and restoring buildings and monuments.

As outlined on figure 1, effects of polluted atmosphere on human constructions are visually very important. The goal of this paper is to propose a framework for visual simulation of atmospheric pollution. The problem of realistically reproducing the different weathering zones of a polluted object has not been specifically addressed so far in Computer Graphics. Our new algorithm main contributions are:

- a physically-inspired technique in order to provide plausible results;
- a local evaluation of the model providing a very good scalability to arbitrary scene complexity;
- an intuitive set of parameters to provide controllability for a designer.

Based on physics studies [Camuffo83], the effect of pollution onto surfaces is divided into two zones called by Camuffo et al. *black* and *white* zones. We propose to detect and render each without preprocessing step nor complex simulation. Moreover, our model is physically inspired: the way the surface is polluted is based on the physical processes described in [Lefevre2002].

The next section describes previous works concerning aging and weathering in computer graphics. Then in section 3, we review the numerous physical and chemical processes that lead to polluted surface appearance in order to provide a plausible model of the phenomenon. We describe the parameters that permit to control the effect. In section 4, we detail our method consisting in synthesizing on the fly specific textures linked to different polluted

zones. These textures are used to compute final aspect of the weathered object. We discuss obtained results in section 5. We finally conclude and present some possible future works.

Figure 1: *Photos of a real polluted object (left) and the same object after cleaning (right).*

2. RELATED WORK

In computer graphics, weathering phenomena are of great importance because of their major visual influence. Due to the large amount of different physical processes and their complexity, numerous studies have focused on specific phenomena. These techniques can be empirical or physically-based and can affect the entire rendering pipeline (geometry, reflectance properties, and colors). Becket and Badler [Becket90] have first handled surfaces imperfections by using a fractal based texture synthesis technique. Blinn [Blinn82] and Hsu [Hsu95] have proposed techniques to render surfaces covered by dust. Miller [Miller94] has provided accessibility shading algorithms permitting to render tarnished surfaces. Wong has proposed a geometry dependent method in [Wong97] to represent dust accumulation, patinas and peeling. In [Paquette01], authors modify objects geometry to handle impacts by different tools. Paint peeling and crackling have been investigated in [Paquette02]. In order to obtain more plausible results, some physically valid techniques have also been developed. Wet surfaces appearance has been studied in [Jensen99]. Corrosion (both patinas and destructive corrosion) has been investigated in [Dorsey96] and [Merillou2001]. A physically-based scratched surface model has been presented in [Bosch04]. A survey of weathering-related techniques has been presented in [Merillou2008].

Concerning building aging, specific studies have also been proposed. Dorsey [Dorsey96b] proposed a model to take into account dirtiness brought onto surfaces by flow processes. This technique permits to handle specific flow patterns on objects. Flows are only a small part of atmospheric pollution weathering. Weathering of stones has also been studied in [Dorsey99], including numerous weathering processes handled by a model of their chemical reactions. Efflorescence, a porosity related aging phenomenon, has been studied in [Shahidi2005] using solid texture synthesis. Moreover, these two last studies handle weathering problems as physical simulations, but do not take into account the localization of the processes. This point has been addressed in [Chen2005]. Authors have introduced the γ-ton map, obtained by a pre-processing step: it is built from the abstraction of considering aging particles interacting with objects. Several weathering processes can be localized by this way, but have then to be specifically handled in term of rendering and the γ-ton map has to be rebuild when geometry changes. In this paper, we focus our work on a physically inspired technique that does not need complete time-consuming simulations nor pre-processing step and still provides plausible results.

Several other techniques address the weathering problem. Replicating change in appearance by means of texture synthesis controlled by capture and transfer of weathering effects [Gu2006], [Lu2007] need acquisition devices that are not effective at the scale of real buildings. Atmospheric pollution weathering strongly depends on the object geometry itself, as well as on the surrounding geometry of other objects in the scene. Thus, texture transfer techniques seem inappropriate in our specific case.

3. ATMOSPHERIC POLLUTION

Among various kinds of building and monument degradations, atmospheric pollution is a major factor of change in appearance. Natural alteration processes are essentially due to weathering, i.e. thermal variation, humidity variation and bacteriological attack. However, all along the two previous centuries, the increase of energy production and consumption, due to transport and manufactures development as well as the use of new combustible (fuel, kind of fuels), has led to important sulphur emitting SO_2 under gaseous form and flying particles like soot. Thus, strong alteration of human constructions can be observed. Figure 2, presents a summary of the atmospheric pollution process consisting in three main steps.

Human activity produces numerous pollutants that fly into atmosphere such as SO_2 (that evolves to SO_3) and combustion particles. This is mainly observable in urban environments. These pollutants are transported into atmosphere and react with water (H_2O) to produce acid H_2SO_4 mixed with rain. This reaction is the first step of atmospheric pollution weathering as it creates acid rainfalls. In a second step, these acid rainfalls affect buildings and monuments and react with them. Building construction materials usually contain calcium

carbonates $CaCO_3$, that is able to react with water to create gypsum. This water mainly comes from rainfalls, but also from condensation due to fog for example [Delmonte97]. Gypsum that can be seen in our case as a grid that grows onto surfaces [Lefevre2002]. In the third and final step, dust and particles are trapped into the gypsum grid, creating a black crust that evolves from a few micrometers to a few centimeters.

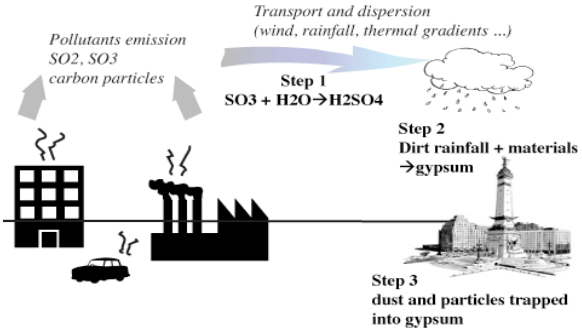


Figure 2: *Summarizing the atmospheric pollution process into three main steps. Step 1: pollutants react with water to create (step 2) acid dirt rainfall that reacts with materials to create a gypsum grid. In step 3, particles are trapped into the gypsum grid.*

It has been clearly established in physics [Camuffo83] that two behaviors have to be distinguished, due to different chemical reactions and inducing different appearances. This physical classification is based on two resulting appearances called *white* and *black* zones [Lefevre2002] (see figure 3).

White zones are directly submitted to strong rain impact and strong water streaming. These mechanical actions permit water to always clean up surfaces. Thus, in these areas, the material keeps its natural color. Each next rainfall evacuates particles deposited between two rainfalls, prohibiting gypsum grid creation.

Black zones: parts of objects that are not directly submitted to rainfalls or strong flows are not washed. However, the ambient moisture and weak flows permit the gypsum grid to develop itself without constraint. Blackening is due to the trapping of particles into the gypsum grid created on the objects external surface (see figure 2, step 3). The more the stone is porous, the more it traps moisture and the more the gypsum grid can develop itself.

4. MODELING AND RENDERING POLLUTED SURFACES

In order to propose a plausible modeling of atmospheric pollution of complete buildings, we need to perform two main steps. Firstly, differently affected zones have to be detected with a local method in order to avoid complex transport simulations or complete surface flow modeling. Secondly, the reflectance properties of each zone have to be modified. Figure 4 presents the pipeline of our method.

Figure 3: *Photo of a polluted object: white zones are washed by rainfalls or strong flows and remain protected from darkening, while the black zone is covered by pollution particles trapped into gypsum.*

Zone detection

Recall from section 3 that *white* zones are found on surfaces directly exposed to rainfall or in zones washed by strong flows. Thus we have to simulate the different zones according to their interactions with rainfalls and to predict the possible flows onto the surface. Instead of a physical simulation of flows or transports [Dorsey96b], [Chen2005], we propose a physically inspired simple and intuitive model that does not need any precomputing phase. Thus we only need to compute direct rain accessibility, where droplets direct impacts can wash the surface. This rain accessibility *rA* has then to be combined with an estimation of strong flows *rfE*, to fully characterize our zones.

4.1.1 Rain accessibility computation: rA

Our technique is a sort of directionally preferred ambient occlusion and is computed in a similar way as discussed in [Wong97]. Using ambient occlusion is a well-known trick in weathering simulation [Merillou08], and it can be easily adapted to simulate direct rain impacts by choosing plausible sampling directions. Over the time, rainfalls incident angles onto surfaces are guided by wind. Atmospheric pollution weathering is a long process: even if it starts as soon as a building or monument is exposed to pollutants, several years are often needed to observe important changes in appearance [Grieken98]. Thus, we propose to consider that rainfalls can come from every direction into a cone opened by angle α aligned with a direction vector *V*.

Figure 4: *Summary of the method, from left to right: Porosity controlled by a noise texture – Computing rain accessibility rA – Computing strong flows and runs-out rfE – Computing reflectance modifications for final image.*

Without any specific meteorological knowledge or artistic desire, we use classical values [Helming08] of rain incidence, leading in our model to $\alpha=20°$ with V the vertical world axis. If the artistic desire is to orient rainfalls in a preferential direction, or if we have such knowledge in a specific geographic location, these parameters can be easily and intuitively modified. For example a preferential wind direction can be easily modeled by modifying V. We finally define a *directional occlusion* as the occlusion inside the previously described cone.

Moreover, washing intensity is also due to a combination of flow and material porosity [Camuffo95]: porous stones absorb water by capillarity, creating a black crust even in rain-exposed areas. We empirically propose to use directly the porosity values presented in table 1 in order to modulate *rA*. Porosity is described as a percentage of surface covered by pores, thus:

$$rA = (1 - porosity) * (1 - directional\_occlusion)$$

The more the surface is porous, the more black crusts can develop. The porosity of building stones is described as a percentage of surface area covered by pores and evolves from *0%* (for perfectly dense materials) to more that *50%* (for very porous building materials). These values strongly depend on each material and are measured specifically in numerous studies, for example [Moropoulos09], [Sabbioni98], see table 1.

Material	Porosity
Carrara marble	0.6%
Verona red stone	8%
Portlandite mortars	18%
Limestone	0.5% to 23%
Clay bricks	27% to 47 %
Lime mortars	33% to 45%
Sandstone	1% to 30%

Table 1: *Common and specific building materials porosity examples.*

Moreover, this porosity value can exhibit strong variations, as shown on table 1, and even strong local variations on a same surface, as measured by [Weishauptova2004]. We account for these variations with a Perlin noise [Perlin85].

4.1.2 Possible flow estimator: rfE

Strong water run-off on *black* zones can also lead to washing and thus to a shift into a white area [Camuffo95]. Thus, we propose to compute an estimator of strong flow: *rfE*. Here, we do not need to compute the full flow onto the surface, we only have to detect if the considered surface point is in a geometric configuration that favors a strong flow or not. These strong flows can come from :

- *border flows:* geometric features ending above the considered point can lead to strong flows (as observable below the corner of a window for example, see figure 5);
- *valley flows:* vertically-oriented geometric valley-like features (observable on a statue drape for example, see figure 6);

In order to propose a local estimator, we sample directions from the considered point in the local tangent plane situated at a distance d just above the surface. This distance permits the designer to define the size of the geometric features to consider, according to each 3D model.

Border flows estimator *bfE*. To detect *border flows*, we sample directions oriented along the vertical axis inside an angular region φ. In our implementation, we use raycasting at a distance maximum t_{max}. Each of the N intersection distance t_i is stored for all the sample rays. We do want to detect strong variations in values of t_i to detect obstacle that can potentially provide strong flows. Thus, we compute the standard deviation of the t_i values, normalized by the distance t_{max}. We also use a correction factor that accounts for the slope of the considered surface: the less the surface is vertical, the less border flows can be strong. Moreover, This leads to:

$$bfE = \frac{\sqrt{\frac{1}{N}\sum (t_i - t_{mean})^2}}{t_{\max}} \sin\left(\angle(N, vertical)\right)$$

Figure 5 shows the method and a result obtained on the front of a building. The parameters φ and t_{max} permit to control the desired shape of the run-outs.

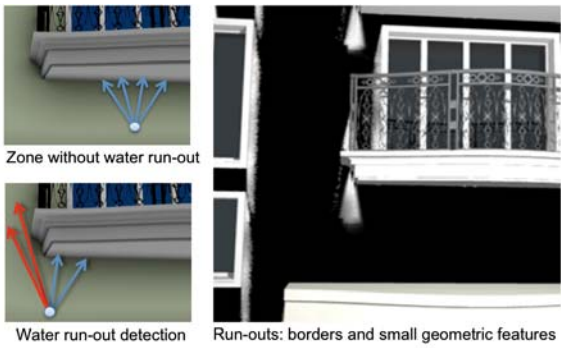


Figure 5: *Border flows detection by sampling the geometry in the local tangent plane of the surface.*

Valley flows estimator *vfE*. We use the same sampling than the previous one, but as we want to detect valleys, we create direction pairs by generating one sample direction inside a horizontal region (angle θ), and coupling it with its opposite direction in the local tangent plane (as illustrated on figure 6). Again with raycasting at a maximum distance t'_{max} we increase the number of hits h only when the two rays of a pair both intersect any geometry. The *valley occlusion* is defined as h/n_{rays}. Such sampling can also detect pits into the geometry, but as we are interested only in strong flows, we modulate *vfE* in the same way as *bfE*:

$$vfE = \frac{h}{n_{rays}} \sin\left(\angle(N, vertical)\right)$$

Figure 6 presents examples of our valley-flows detection method on the *Athena* statue.

In order to keep a physically inspired model, we also have to account for the influence of porosity: the more the surface is porous, the less the flows are strong. Thus, we define the possible flow estimator *rfE* as:

$$rfE = (1 - porosity)(bfE + vfE)$$

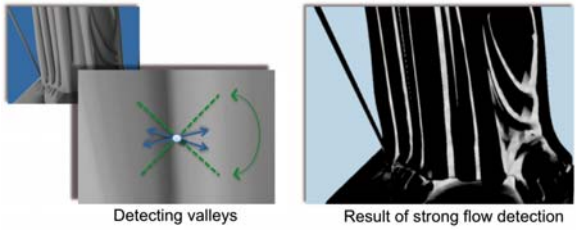


Figure 6: *Valley flows local detection: pairs of sampling directions inside an horizontal region permit to localize valley-like geometric features.*

Due to the phenomenology of atmospheric pollution weathering (see section 3), flows are only to be considered in *black* zones (previously detected by *rA*): *white* zones are already washed out. Note also that the main drawback of our proposed technique is that we cannot detect flows originating from other points of the object. The porous nature of building materials limits this problem (porous materials absorb water by capillarity) and permits to keep a local evaluation of pollution weathering. Finally, combining *rA* and *rfE* at each renderer point gives us the local pollution rate P_R.

$$P_R = \begin{cases} 1 & if \quad rA + rfE > 1 \\ rA + rfE & otherwhise \end{cases}$$

Rendering: reflectance and surface alteration

As outlined in section 3, one of the consequences of atmospheric pollution is the soiling of building materials. In [Watt08] such visual nuisance can be measured as the reflectance contrast of a darkened area by comparison to the reflectance of the base surface. Thus, several models have been developed, based on experimental data, for predicting the rate of soiling. They postulates that the rate at which uncovered area becomes covered is proportional to the size of the uncovered area:

$$R = R_0\left(\frac{A}{A_0}\right) + R_p\left(\frac{A - A_0}{A_0}\right)$$

With: R the object base reflectance, R_0 the reflectance from unaffected surface, R_p the reflectance from covering particles, A_0 the total area of surface that receives depositing particles and A the uncovered area of surface which receives depositing particles.

In atmospheric pollution weathering, the model for area coverage is usually an exponential function of exposure time [Pio98]:

$$A = A_0 \exp(-Kt)$$

With: K the soiling constant directly proportional to the polluted particles concentration C_p: $K=\lambda C_p$ (in physics, λ is the dose-response constant).

This leads to the evolution of reflectance as a function of exposure time:

$$R = \left(R_0 - R_p\right)\exp\left(-\lambda C_p t\right) + R_p$$

Since the concentration of pollutants strongly depends on various parameters such as the classification of the location (rural, urban, industrial...), we keep it as a user driven parameter that is directly related to the polluting rate. From our phenomenological point of view, we use the previously computed rate P_R as the evolution of polluted particles concentration: on washed parts, such concentration is minimum and on black areas, it is at its maximum. During the rendering step, this process progressively modulates the color of the darkened areas:

$$R = \left(R_0 - R_p\right)\exp\left(-P_R\, t\right) + R_p$$

R is then used as the base material reflectance in the designer chosen BRDF. Thus it can easily be ported to any real-time shading language (entry maps such as rain accessibility and flows can be pre-computed). Note that with black crust mainly composed of black carbon particles $R_p=0$.

As denoted in [Camuffo95], the black crust is dendritic, leading to a very rough irregular aspect. The thickness of this crust above affected surfaces is a few millimeters and depends on the porosity of the underlying material. Thus, to empirically account for this phenomenon, we simply add a rough bump mapping to the black crust rendering to account for such effect (we use a Fractional Brownian Motion perturbation). The depth of this bump map is modulated by the porosity of the material: the more the material is porous, the more the black crust is thick and dendritic.

5. RESULTS

The model presented in this paper has been implemented in our own realistic rendering engine. Note that as the pollution textures are computed *on the fly*, our method does not increase the renderer memory requirements. Figure 7 shows a clean and a polluted bunny with a thick bump mapped black crust. The top of the polluted bunny is submitted to rain fall and stays clean, while its bottom is polluted. Figure 8 shows the detection of flows on the geometric features of the column as well as detection of flows into corners. Figure 9 shows the progressive weathering of Athena due to atmospheric pollution. Different weathering rates are observable due to rain accessibility as well as strong flows. This example exhibits a strong porosity, reducing the *white* zone. Note that a homogeneous darkening of the surface cannot match the expected behavior.

Figure 7: *Pollution with a design choice of a thick bumped black crust.*

Figure 10 exhibits pollution evolution on a building and a zoom on specific patterns that shows the importance of zone detection in the atmospheric pollution weathering simulation. Rendering time of the clean building is 140s with our renderer and the polluted building is rendered in 200s (with 9 samples for *rA* and 9 samples for *rfE*).

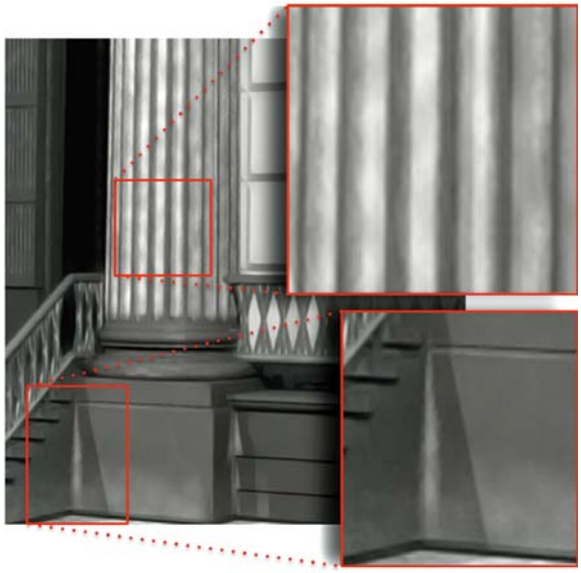


Figure 8: *Importance of flow detection in the patterns observable on polluted objects.*

Finally, figure 11 shows the ease of use and scalability of our model. As previously discussed, it relies on a local evaluation and does not need any iteration nor preprocessing step to choose the pollution level. The cost of our technique depends on the occlusion-like implementation. In our Monte-Carlo rendering engine, we rely on raycasting for such feature. Unpolluted building on figure 10 is computed in 49 seconds. The polluted building is computed in 75 seconds. This is due to the number of sample taken for zone detection (we use 16 samples for *rA* and 16 samples for *rfE)*.

6. CONCLUSION AND FUTURE WORKS

In this paper, we have presented a method that permits to simulate, on a physically-inspired basis, the major effect of atmospheric pollution weathering. This specific weathering process is visually very important and contributes to improving realism of rendered images. Beyond this visual importance, this simulation can be useful in several other fields such as architecture or cultural heritage preservation as our tool permits to detect weathered zones (zones to protect with high priority). We are actually porting our method to GPU (OpenCL / CUDA) for interactive performance. Moreover, the physical analysis presented here makes our phenomenological approach simple and intuitive. As future works, we aim to improve our model in order to take into account other materials: metals and glasses are also affected by atmospheric pollution. We are also actively studying de-weathering possibilities as treatments to perform on real photos to virtually clean real objects.

Figure 9: *Athena statue perfectly clean (left) and progressively affected by atmospheric pollution weathering.*

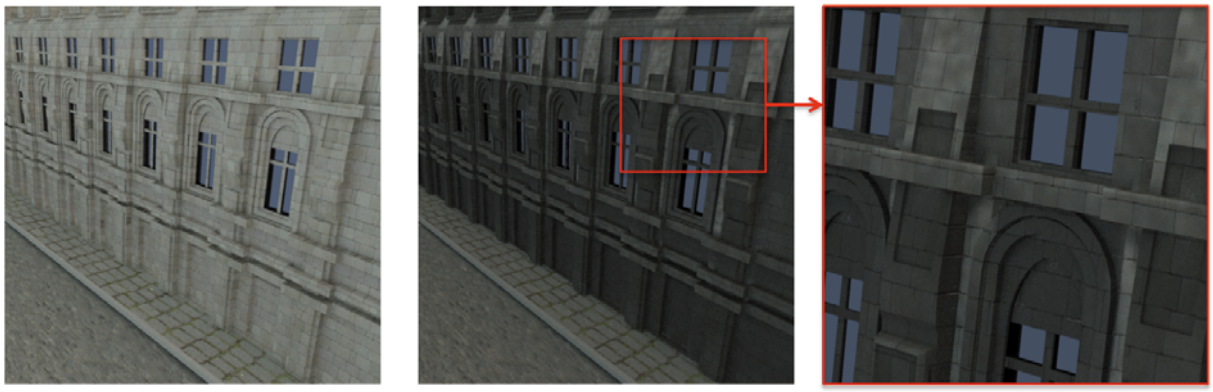


Figure 10: *A zoom on different patterns automatically detected on the front of a building (the left image is the clean building, for reference). Rain accessibility, porosity variations and strong flows are clearly observable.*

7. AKNOWLEDGEMENTS

This work is supported by the French National Agency for Research (ANR) under agreement ANR-06-MDCA-004-01.

Figure 11: *Our model is scalable and can render complex buildings and streets without complex physical simulation or preprocessing step.*

8. BIBLIOGRAPHY

[Becket90] BECKET, W., BADLER, N. I.: Imperfection for Realistic Image synthesis. Journal of Visualization and Computer Animation, 1, 1 (august 1990), 26-32

[Blinn82] BLINN, J. F.: Light Reflection Functions for Simulation of Clouds and Dusty Surfaces. In Computer Graphics (Proceedings of SIGGRAPH 82). (1982), vol. 16, pp. 21-29.

[Bosch04] BOSCH, C., PUEYO, X., MERILLOU, S., GHAZANFARPOUR, D.: A Physically-Based Model for Rendering Realistic Scratches. Computer Graphics Forum (Proceedings of Eurographics '04), 23, 3 2004), 361-370

[Camuffo83] CAMUFFO, D., MONTE, M. D., SABBIONI, C.: Origin and Growth Mechanism of the Sulfated Crust on Urban Limestone. Water, Air, and Soil Pollution, 19, 1983), 351-359

[Camuffo95] CAMUFFO, D.: Physical Weathering Of Stones. The Science of Total Environment, 167, 1995), 1-14

[Chen2005] Chen Y, Xia L, Wong TT, Tong X, Bao H, Guo B, et al. Visual simulation of weathering by g-ton tracing. ACM Transactions on Graphics 2005;24(3):1127–33.

[Delmonte97] MONTE, M. D., ROSSI, P.: Fog and Gyp- sum Crytals on Building Materials. Atmospheric Environ- ment, 31, 11 1997), 1637-1646

[Dorsey96] DORSEY, J., HANRAHAN, P.: Modeling and Rendering of Metallic Patinas. In ACM SIGGRAPH. (1996), pp. 387 - 396.

[Dorsey96b] DORSEY, J., PEDERSEN, H. K., HANRA- HAN, P.: Flow and Changes in Appearance. In ACM SIG- GRAPH. (1996), pp. 411 - 420.

[Dorsey99] DORSEY, J., EDELMAN, A., JENSEN, H. W., LEGAKIS, J., PEDERSEN, H. K.: Modeling and Ren- dering of Weathered Stone. In ACM SIGGRAPH. (1999), pp. 225 - 234.

[Dorsey2008] DORSEY, J., RUSHMEIER, H., SILLION, F.: Digital Modeling of Material Appearance. Morgan Kaufmann/Elsevier. 2008.

[Grieken98] GRIEKEN, R. V., DELALIEUX, F., GY- SELS, K.: Cultural heritage and the environment Pure and Applied Chemistry 70 12 1998), 2327-2331

[Gu2006] Gu J, Tu C, Ramamoorthi R, Belhumeur P, Matusik W, Nayar SK.Time-varying surface appearance: acquisition, modeling, and rendering. Boston: ACM SIGGRAPH; 2006.

[Helming08] HELMING, K.: Wind Speed Effects On Rain Erosivity. Geomorphology, 94, 3-4 2008), 290-313

[Hsu95] HSU, S., WONG, T.: Simulating Dust Accumu- lation. IEEE Computer Graphics and Applications, 15, 1 1995), 18-22

[Jensen99] JENSEN, H. W., LEGAKIS, J., DORSEY, J.: Rendering of Wet Material. In Rendering Techniques '99. (1999), pp. 273-282.

[Lefevre2002] LEFEVRE, R. A., AUSSET, P.: Atmospheric Pollution and Building Materials: Stone and Glass. Geological Society Special Publication : Natural Stones, Weathering Phenomena, Conservation Strategies and Case Studies, 205, 2002), 329-345

[Lu2007] LU, J., GEORGHIADES, A. S., GLASER, A., WU, H., WEI, L. Y., GUO, B., DORSEY, J., RUSHMEIER, H.: Context-Aware Textures. ACM Transactions on Graphics, 6, 1 (January 2007)

[Merillou2001] MERILLOU, S., DISCHLER, J. M., GHAZANFARPOUR, D.: Corrosion: simulating and ren- dering. In Graphics interface 2001. (2001).

[Merillou2008] MERILLOU, S., GHAZANFARPOUR, D.: A survey of aging and weathering phenomena in computer graphics. Comput. Graph., 32, 2 2008), 159-174

[Miller94] MILLER, G.: Efficient algorithms for local and global accessibility shading. In ACM SIGGRAPH. (1994), pp. 319 - 326.

[Moropoulos09] MOROPOULOS, A., POLIKRETI, K.: Principal Component Analysis in Monuments Conservation: Three Application examples. Journal of Cultural Heritage, 10, 2009), 73-81

[Paquette01] PAQUETTE, E., POULIN, P., DRET- TAKIS, G.: Surface aging by impacts. In Graphics interface 2001. (2001).

[Paquette02] PAQUETTE, E., POULIN, P., DRET- TAKIS, G.: The simulation of Paint Cracking and Peeling. In Graphics Interface 2002. (2002), pp. 59-68.

[Pio98] A.PIO, C., RAMOS, M. M., DUARTE, A. C.: Atmospheric Aerosol and Soiling of External Surfaces in an Urban Environment. Atmospheric Environment, 32, 11 1998), 1979-1989

[Perlin85] PERLIN K. An image synthesizer. In SIGGRAPH '85 conference proceedings, San Francisco, CA, July 22–26 1985, 1985, p. 287–96.

[Sabbioni98] SABBIONI, C., ZAPPIA, G., GHEDINI, N., GOBBI, G., FAVONI, O.: Black Crust On Ancient Mortars. Atmospheric Environment, 32, 2 1998), 215–223

[Shahidi2005] SHAHIDI, S., MERILLOU, S., GHAZ- ANFARPOUR, D.: Phenomenological simulation of Efflo- rescence in brick constructions. In Eurographics Workshop on Natural Phenomena. (2005).

[Watt08] WATT, J., JARRETTA, D., HAMILTONA, R.: Dose Response Functions for the Soiling of Heritage Ma- terials due to Air Pollution Exposure. Science of the Total Environment, 400, 1-3 2008), 415-424

[Weishauptova2004] WEISHAUPTOVA Z. PRIKRYL R. Porosimetric studies in rocks: methods and application for weathered building stones. Dimension Stones 2004, Taylor & Francis Group, London.

[Wong97] WONG, T. T., NG, W. Y., HENG, P. A.: A Geometry Dependent Texture Generation Framework for Simulating Surface Imperfections. In Proceedings of Eurographics Workshop on Rendering. (1997), pp. 139 - 150

A Narrow Band Level Set Method for Surface Extraction from Unstructured Point-based Volume Data

Paul Rosenthal Vladimir Molchanov Lars Linsen

Jacobs University, Bremen, Germany

{p.rosenthal, v.molchanov, l.linsen}@jacobs-university.de

ABSTRACT

Level-set methods have become a valuable and well-established field of visualization over the last decades. Different implementations addressing different design goals and different data types exist. In particular, level sets can be used to extract isosurfaces from scalar volume data that fulfill certain smoothness criteria. Recently, such an approach has been generalized to operate on unstructured point-based volume data, where data points are not arranged on a regular grid nor are they connected in form of a mesh. Utilizing this new development, one can avoid an interpolation to a regular grid which inevitably introduces interpolation errors. However, the global processing of the level-set function can be slow when dealing with unstructured point-based volume data sets containing several million data points.

We propose an improved level-set approach that performs the process of the level-set function locally. As for isosurface extraction we are only interested in the zero level set, values are only updated in regions close to the zero level set. In each iteration of the level-set process, the zero level set is extracted using direct isosurface extraction from unstructured point-based volume data and a narrow band around the zero level set is constructed. The band consists of two parts: an inner and an outer band. The inner band contains all data points within a small area around the zero level set. These points are updated when executing the level set step. The outer band encloses the inner band providing all those neighbors of the points of the inner band that are necessary to approximate gradients and mean curvature. Neighborhood information is obtained using an efficient *k*d-tree scheme, gradients and mean curvature are estimated using a four-dimensional least-squares fitting approach.

Comparing ourselves to the global approach, we demonstrate that this local level-set approach for unstructured point-based volume data achieves a significant speed-up of one order of magnitude for data sets in the range of several million data points with equivalent quality and robustness.

Keywords: Level sets, unstructured point-based volume data, isosurface extraction.

1 INTRODUCTION

Many modern technologies and methods generate volume data that is not gridded anymore. The data points can have an arbitrary distribution without any connectivity. A major group of such data stem from numerical simulations of natural phenomena. They are carried out on unstructured point-based data to gain a high flexibility and provide new insights by additional degrees of freedom. Many physical simulations are, for instance, carried out as smoothed particle hydrodynamics simulations with millions of particles. Such simulations allow for the reproduction of complex natural phenomena by not only simulating the evolution of data at the sample points but also simulating the flow of the sample points under respective forces. Hence, data points move over time, change their positions and neighborhoods, and are distributed with a highly varying density. Such a flexible data structure saves computation time when using high sample density only in those regions, where action takes place.

Level-set methods have a large variety of applications and, in particular, have entered the fields of image processing and scientific visualization for extracting features from scalar data. We are mainly interested in the aspect of segmenting scalar volume data. Many algorithms and approaches with different modifications of the main level-set idea exist. Most of the algorithms address a specific problem or a specific type of data. Typically, the algorithms operate on hexahedral cells and a given initial level-set function is modified to explicitly or implicitly minimize a given energy functional.

Recently, we have presented an approach generalizing the basic idea of level sets to unstructured point-based volume data [19]. It operates directly on unstructured data, i. e., we do not resample the data over a structured grid nor are we generating global or local polyhedrizations. By doing so, resampling errors are avoided, which are inevitably introduced when resampling unstructured data over a regular structured grid using scattered data interpolation techniques.

The level-set method involves the evolution of a function using an iterative numerical integration scheme. At each iteration step, the approach typically induces complex calculations and the evaluation of partial differential equations at each sample location and each iteration step. When dealing with large data sets and small

time-integration steps, this can lead to enormous computation times until convergence of the level-set process. The calculations for unstructured point-based data are yet more complex than those in the gridded case, slowing down the process even more. To overcome this problem, narrow-band methods have been proposed in recent years. The main idea is to update the level-set function only in a narrow band of interest and neglect all other sample points, saving a major number of computations. Narrow-band methods have turned out to be very useful for fast yet correct level-set computations on gridded data. To our knowledge, no algorithm exists yet that combines the accurate computations of level-sets with the efficiency of a narrow-band localization directly applied to unstructured point-based volume data. We propose such a method for smooth isosurface extraction based on a fast local level-set approach.

We initialize the level-set function at the sample point positions and extract the zero level set. This zero level set is being transformed into the desired surface by application of the iterative level-set process to the level-set function. Therefore, we can restrict the process to a narrow band of sample points around the current zero level set.

The main ideas of our local level-set method and the structure of the whole visualization pipeline are introduced in Section 3. The creation of the narrow band and the update of the level-set function in the narrow band are explained in Sections 4 and 5, respectively.

The actual level-set process applied to the narrow band and the required approximation of gradient and mean curvature for unstructured point-based volume data using a four-dimensional least-squares method is described in Section 6. Finally, results and their discussion, including the analysis of computation times and detailed comparisons, are provided in Section 7.

2 RELATED WORK

The fast visualization of large unstructured point-based volume data sets is a challenging task, even more when dealing with highly varying sample point densities. The generation of a polyhedrization [4] of the unstructured data points can be very slow for large data sets and is thus not practicable. Hence, the most common way of dealing with unstructured point-based volume data is to resample to a structured grid using scattered data interpolation techniques [7]. Subsequently, a large variety of well-known methods like isosurface extraction, region-growing methods, and level-set methods can be applied to gridded data to generate the desired visualizations. Level-set methods, in particular, are almost without exception executed on regular hexahedral grids what facilitates discrete derivative computations.

Unfortunately, such resampling steps to a regular grid always introduce inaccuracies, which heavily depend on the resolution of the grid and the point density of the data set. In our case, when dealing with data sets with highly varying point density, the interpolation error can be enormous when using regular grids that fit today's commodity hardware memory constraints. Adaptive grids can reduce the error, but the more adaptive it gets the more complicated the processing becomes and we are looking into data sets with a difference of three orders of magnitude in point density. This raises the desire to directly operate on unstructured point-based volume data.

The original idea of level sets is to implicitly represent a surface as the solution of an equation with respect to an underlying scalar field. The (dynamic) level-set methods go back to Sethian and Osher [15, 16, 21], who first described the evolution of such a closed hypersurface by application of PDEs to the underlying scalar field.

Due to their flexibility and capability level-set methods have developed rapidly in the recent years. Many different approaches exist and the range of application areas is wide. Breen et al. [2] presented a general framework for level-set segmentation of a large variety of regular data sets. Museth et al. [14] use a level-set method to segment non-uniform data sets, where non-uniform data denotes unions of multiple regular data sets with non-uniform resolutions. Enright et al. [5] apply a level-set approach to an octree-based adaptive mesh. Many other approaches, e.g. [22], exist that target a huge variety of level-set segmentation tasks on different structured data sets.

In terms of unstructured data, the particle level-set methods [8] use free particles during the level-set computations, but still need an underlying structured grid to compute the motion of the particles. We recently presented the first approach, directly processing level sets on unstructured point-based volume without any grid calculation or reconstruction of the scalar field [19]. We initialize the level-set function only at sample positions and carry out the whole level-set evolution at these locations. One drawback of the method is the relatively low speed when compared to state-of-the-art level-set methods for gridded data.

The largest gain in computation speed for level-set methods was achieved with the introduction of local level-set methods by Adalsteinsson and Sethian [1]. These methods, also called narrow-band methods, have been developed rapidly in the last years [12, 17] and have also been transformed to work on today's fast and parallel graphics hardware [10]. The main idea of these narrow-band approaches is to carry out the level-set process only in a small region around the level set of interest. Similar to active contours [9, 13], this requires the knowledge about the location of the zero level set. However, contrary to active contours, where the zero level set is explicitly deformed until it converges to the desired surface, the main idea of just deforming

the level-set function is maintained in the narrow-band methods. This allows for easy changes of the topology of the zero level set.

To our knowledge, there exists no method which combines the accuracy of directly applying level sets to unstructured data and the fast level-set computation using a narrow band. We propose such an approach that directly operates on the unstructured point-based data and does not need to reconstruct any scalar field at any positions other than the sample points.

3 GENERAL APPROACH

Let $f : D \to \mathbb{R}$ be a volumetric scalar field with bounded domain $D \subset \mathbb{R}^3$ given at a set of unstructured sample points $\mathbf{x}_i \in D$. Our goal is to efficiently extract a smooth isosurface $\Gamma_{\text{iso}} \subset D$ with respect to a given isovalue f_{iso}. This task should be carried out utilizing a narrow-band level-set approach which only operates on the sample points. The level-set approach provides a handle to control the smoothness of the resulting surface. The visualization pipeline consists of three main phases: the initialization, the level-set evolution, and the rendering. Its flow chart is depicted in Figure 1.

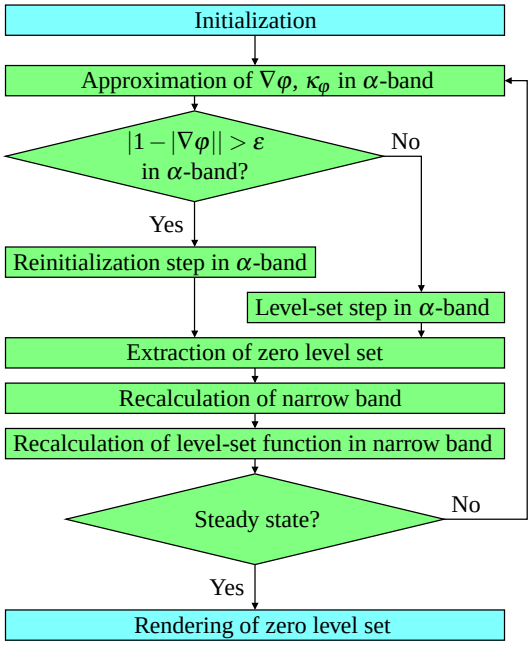


Figure 1: Illustration of the whole visualization pipeline. In the initialization phase, the level-set function and the narrow band are initialized. Next, the level-set function and the narrow band are consecutively updated following Equation (2) until the level-set function reaches steady state. Finally, the zero level set is rendered as resulting smooth isosurface.

In the initialization phase, we build a data structure that stores and handles neighborhood information, to efficiently compute level-set function properties, extract isopoints, and generate the narrow band. This is done using a three-dimensional *k*d-tree. In a preprocessing step, we compute the n nearest neighbors for each sample. Here, $n = 26$ is chosen, inspired by the regular case on a structured equidistant hexahedral grid, where every sample $\mathbf{x}$ has 26 nearest neighbors in the L_∞-metric.

Afterwards, the level-set function $\varphi : D \to \mathbb{R}$ is initialized for every sample of the data set as a signed-distance function to a sphere. The points of the zero level set are extracted using direct isosurface extraction [18], which results in a point cloud representation of the extracted surface. Around the extracted isosurface, a narrow band of sample points consisting of two layers is generated. The inner layer or α-band consists of all sample points, which have a smaller distance to the isosurface than a given band width d_α. The outer layer or β-band consists of all sample points, which are not lying in the α-band but are needed to compute gradients or mean curvature at α-band points. The actual level-set process will be performed only within the α-band.

In the iterative processing phase, first the gradient $\nabla\varphi$ of the level-set function and the mean curvature κ_φ are approximated for the sample points in the α-band. The approximation is computed using a four-dimensional least-squares approach. If the norms of the level-set function gradients exceed a given threshold, a reinitialization step is applied to the level-set function with respect to the special Eikonal equation

$$\frac{\partial \varphi}{\partial t} = \text{sign}(\varphi)(1 - |\nabla\varphi|) \,. \tag{1}$$

The reinitialization step brings the level-set function again close to a signed-distance function, which assures good numerical behavior of the whole process. If no reinitialization is necessary, a level-set step is performed in the α-band with respect to the equation

$$\frac{\partial \varphi}{\partial t} = \big((1-\lambda)(f - f_{\text{iso}} - \varphi) + \lambda\,\kappa_\varphi\big)\,|\nabla\varphi| \,, \tag{2}$$

which models a weighted combination of hyperbolic normal advection [15] and mean curvature flow [6] with smoothness parameter $\lambda \in [0,1]$. Since we use an explicit Euler time discretization for updating the level-set function at the sample points, the time steps are bounded by the Courant-Friedrichs-Lewy (CFL) condition [3] to permit numerical stability. In particular, this condition also assures that the zero level set cannot leave the α-band in a single iteration.

After updating the function values in the α-band, the new zero level set is extracted and the narrow band has to be updated. Since we only update the function values of the points in the α-band, the points that have been added to the α-band after the level-set step do not hold the correct level-set function values. Consequently, the values have to be recomputed with respect to the new zero level set. This is done by interpolating between computed level-set function values and signed-distance

function values in the α-band. Similarly, the points in the β-band have not been updated and may hold wrong level-set function values. These points are updated by assigning signed-distance function values to the new zero level set.

The local level-set process is executed until the level-set function reaches steady state, i. e., until the function values do not change more than a given threshold from one time step to the subsequent one. After convergence, the zero level set has the desired properties of Γ_{iso} and is rendered using splat-based ray tracing [11].

4 NARROW BAND CREATION

Since the movement of the zero level set during the level-set iteration is bounded in each step by the CFL-condition, it is possible to restrict the level-set process to a band around the zero level set without loosing flexibility or distorting the result. However, the level-set process at a sample point requires the approximation of level-set function derivatives in this point and, therefore, the level-set function values of its nearest neighbors. Restricting level-set update and derivative approximation to points within the band around the zero level set would, consequently, lead to oscillations at the border of the band, since derivatives there are less precisely approximated than near the zero level set.

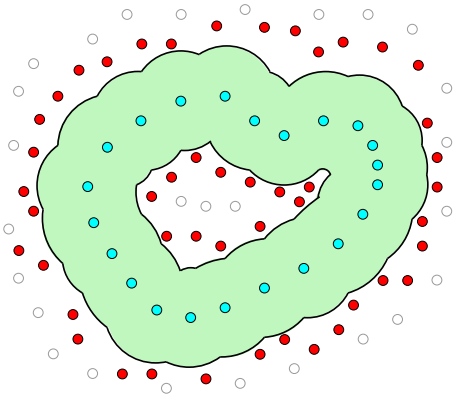


Figure 2: Construction of the two-layer band around the zero level set (colored blue). The zero level set is extracted in form of a point cloud representation. Then, all sample points with distance to the zero level set less than d_α are marked as belonging to the inner layer of the band (green). Thereafter, all additional sample points needed for the gradient computations within the level-set process are marked as belonging to the outer layer of the band (red points).

We chose, instead, to use a band consisting of two nested layers. We want to use the notation α-band and β-band for the two layers. The construction of these layers is illustrated in Figure 2. The α-band describes the volume, where the actual level-set process is performed. All sample points having a smaller distance to the zero level set than the width d_α of the α-band are marked as members of the α-band. As the zero level set is extracted in a point cloud representation, the distance of a sample point to the zero level set is defined as the distance to the nearest point of the zero level set. For each sample point the distance is efficiently obtained by a nearest-neighbor query to the *k*d-tree of the points of the zero level set.

The size of d_α depends on the distribution of the sample points, i.e., is data-dependent. To capture the entire area surrounding the zero level set, d_α has to be at least greater than the maximum distance of neighboring points within the point cloud representation of the zero level set. In fact, it should be even twice as large to permit a reasonable recalculation of the level-set function values. We discuss this in more detail in Section 5. In all our experiments, it was sufficient to choose one global value for d_α for the whole local level-set process, even for data sets with highly varying point densities. Hence, d_α was chosen depending on the sample distribution of the data set and independent of the zero level sets.

For the processing of the points in the α-band with respect to Equations (1) and (2), sample points outside the α-band are needed to approximate level-set function derivatives. Having established the α-band, all additional sample points that are required for derivative approximation of the points belonging to the α-band are marked as members of the β-band, which merely acts as support for the α-band processing.

5 NARROW BAND UPDATE

In the initialization phase, the level-set function is set globally to a radial signed-distance function. Hence, the level-set function values of points in the initial narrow band are directly given. However, after executing one local level-set step in the α-band, extracting the new zero level set, and updating the narrow band, the sample points may have changed their band membership. Consequently, some sample points may have been added to the α-band. For these new members of the α-band, the level-set function values are outdated.

Moreover, function values of all the points in the β-band have not been updated in the level-set step, but their new values are required for the subsequent level-set iteration. Hence, additional updates of the level-set function values at some sample points of the narrow band are likely to be required in each iteration step. An illustration of the idea is given in Figure 3.

For points with distance smaller than $\frac{d_\alpha}{2}$ to the zero level set, we can be sure that the level-set function values have been updated in the level-set iteration step. This is due to the fact that the level-set function value update is executed with a step size that is bounded by the CFL condition. Hence, no adjustment is necessary.

Sample points in the narrow band with distance to the zero level set larger than $\frac{d_\alpha}{2}$ (red points in Figure 3) may have not been updated or may even not have been in the narrow band at all in the preceding iteration step.

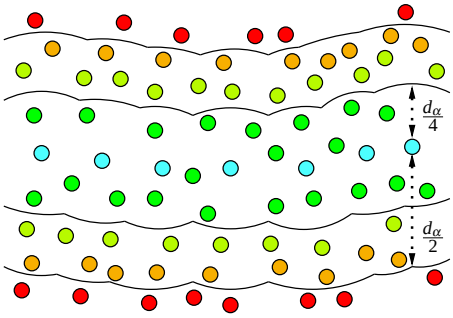


Figure 3: Updating the level-set function on the narrow band with size d_α: Points (green) with minimum distance to the zero level set points (blue) smaller than $\frac{d_\alpha}{4}$ have correctly been updated in the level-set iteration step. Points (red) in the narrow band with distance to the zero level set greater than $\frac{d_\alpha}{2}$ might have not be included in the computations of the last level-set iteration step. We assign their level-set function value to the signed distance to the zero level set points. For all points in the α-band lying in between, the new level-set function value is interpolated between the old level-set function value and their signed distance to the zero level set points.

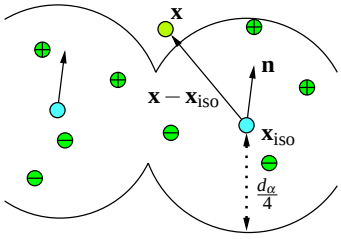


Figure 4: Derivation of correct sign for signed-distance function generation at sample points with distance to the zero level set greater than $\frac{d_\alpha}{4}$. Each point of the zero level set $\mathbf{x}_{\text{iso}}$ holds a surface normal $\mathbf{n}$, inducing an orientation of the zero level set and therewith inducing a well-defined sign for all points with distance greater than $\frac{d_\alpha}{4}$.

Hence, the level-set function value needs to be recomputed as illustrated in Figure 4. Since the level-set function is always kept near a signed-distance function it is feasible to reset the level-set function values in these points to a signed-distance function. Let $\mathbf{x}$ be such a sample point that needs to be updated. Then, we make use of the - already computed - distance d to the nearest zero level set point $\mathbf{x}_{\text{iso}}$. The sign for the signed-distance function is directly induced by the surface normal of the nearest zero level set point as illustrated in Figure 4. Altogether, we set the level-set function value for $\mathbf{x}$ to

$$\varphi_{\mathbf{x}} := \text{sign}\left(\langle \mathbf{n}, \mathbf{x}-\mathbf{x}_{\text{iso}} \rangle\right) |\mathbf{x}-\mathbf{x}_{\text{iso}}| \ .$$

Although the level-set function is kept near a signed-distance function at points with distance $d \leq \frac{d_\alpha}{2}$ to the zero level set and is explicitly been set to a signed-distance function at all the other points of the narrow band, there might occur a discontinuity at distance $d = \frac{d_\alpha}{2}$ to the zero level set. This can be avoided by interpolating between level-set function value and constructed signed-distance function value in a transition zone. We chose this transition zone to be the distance range $d \in \left[\frac{d_\alpha}{4}, \frac{d_\alpha}{2}\right]$ to the zero level set. For a continuous transition, we apply an interpolation using the monotone polynomial

$$k : [0,1] \to [0,1] \ , \quad k(t) := 6t^5 - 15t^4 + 10t^3 \ .$$

Consequently, for each sample point $\mathbf{x}$ with distance $d \in \left[\frac{d_\alpha}{4}, \frac{d_\alpha}{2}\right]$ to the zero level set, we recalculate its level-set function value by

$$\varphi_{\mathbf{x}} := \varphi_{\mathbf{x}} \cdot \left(1 - k\left(\frac{4d - d_\alpha}{d_\alpha}\right)\right) + d \cdot k\left(\frac{4d - d_\alpha}{d_\alpha}\right) \ ,$$

which produces a C^2-continuous blending between both functions.

In summary, keeping the level-set values for points with distance $d \in \left[0, \frac{d_\alpha}{4}\right)$, blending between level-set function and signed-distance function for points with distance $d \in \left[\frac{d_\alpha}{4}, \frac{d_\alpha}{2}\right]$, and setting all other points in the narrow band to a signed-distance function results in a continuous recalculation of the level-set function in the narrow band.

6 LOCAL LEVEL-SET PROCESS

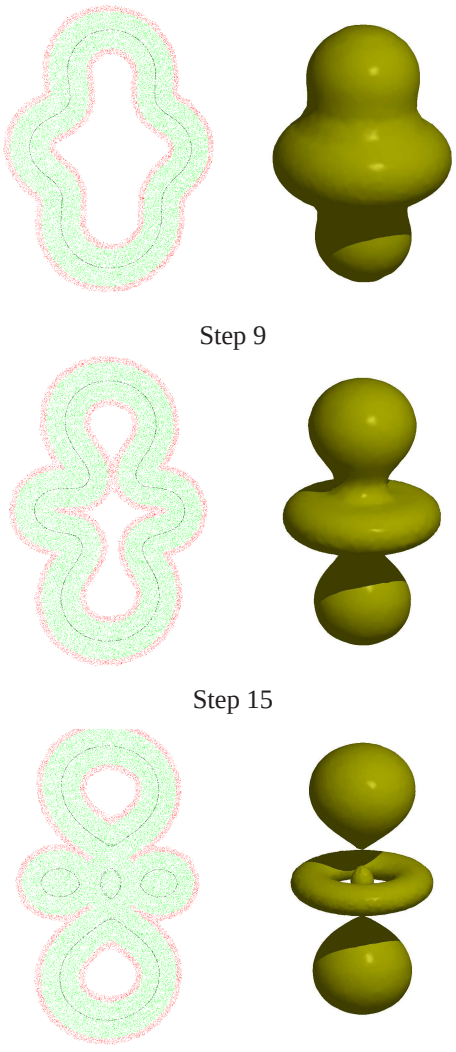
To process the level-set function according to hyperbolic normal advection and mean curvature flow as modeled in Equation (2), we apply the methods already used in the global case [19] to estimate the level-set function gradient $\nabla\varphi$ and mean curvature κ_φ in each sample point in the α-band of each time step. Therefore, only nearest-neighbor computations for the sample points are used. Furthermore, no information about the scalar field other than at the sample points in the narrow band is needed.

Since the quality of the level-set process significantly degrades if the level-set function φ is not close to a signed-distance function of the zero level set, the level-set function is initialized as a signed-distance function and the function values of points in the β-band get reset to represent a signed-distance function to the zero level set. However, the level-set process cannot maintain this property. To keep the level-set function near a signed-distance function, it is reinitialized using a PDE-based approach solving the special Eikonal equation (1),when the norms of the gradients of the function exceed a certain threshold.

Since the level-set processes following Equations (1) and (2) are performed using an explicit time discretization of order one, the used time step has to be chosen carefully to keep the zero level set always in the α-band and fulfill the CFL-condition. In fact, meeting the CFL-condition is also sufficient to keep the zero level set within the α-band during computations, as already observed by Peng et al. [17].

7 RESULTS AND DISCUSSION

The presented approach was applied to a variety of data sets to verify our method and evaluate it in terms of accuracy and speed. For performance analysis we applied it to an unstructured point-based volume data set with eight million randomly distributed samples. The data set was generated by resampling the regular Hydrogen data set of size $128 \times 128 \times 128$ to the random points. An illustration of the evolution process for this data set is shown in Figure 5.



Step 9

Step 15

Step 24

Figure 5: Evolution of the zero level set when applying the local level-set method with $\lambda = 0.01$ to the Hydrogen data set with eight million sample points. For each time step, a splat-based ray tracing of the zero level set is shown on the right-hand side. On the left-hand side, a point rendering of a slab of the data set is shown illustrating the narrow band. Extracted surface points of the zero level set are colored black, sample points in the α-band are colored green, and sample points in the β-band are colored red. Sample points not belonging to the narrow band are not rendered.

The same data set was used to analyze the performance of our method in terms of speed. The data set was downsampled to different sizes for investigating the scalability of our approach. All computation times were measured on a single 2.66 GHz XEON processor.

The runtime analysis for the preprocessing phase is given in Table 1. We observe that most of the time is required for the nearest neighbor computation. However, the data to which we typically apply our method stem from numerical simulations, where nearest neighbors are known such that this pre-processing step can be dropped.

samples	*k*d-tree gen.	NN calc.	*k*d-neig. calc.
2M	4 sec	111 sec	4.1 sec
4M	9 sec	239 sec	10.1 sec
8M	19 sec	541 sec	21.9 sec

Table 1: Computation times for the preprocessing of the Hydrogen data set with different sample quantities, including the generation of the *k*d-tree, calculation of 26 nearest neighbors (only required if not given), and the calculation of the *k*d-tree neighbors needed for zero level-set surface extraction.

The computation times for one level-set iteration step analyzed with respect to the Hydrogen data set are shown in Table 2. It comprises a summary of the results including the number of samples, the number of extracted zero level set points, and the calculation times of all computation steps, i. e., for the zero level set extraction, for the update of the narrow band (including narrow band function value updates), and for the level-set process.

samples	points	point ex.	band up.	ls step
2M	18k	1.9 sec	1.8 sec	1.5 sec
4M	29k	2.6 sec	2.4 sec	2.8 sec
8M	47k	4.1 sec	3.9 sec	5.2 sec

Table 2: Computation times for one level-set iteration step of the Hydrogen data set with different numbers of sample points. The number of extracted zero level set points as well as the computation times for zero level set extraction are given. Moreover, the computation times for the narrow band update and the calculation of one level-set step are given.

The whole local level-set process for extracting a smooth isosurface from the Hydrogen data set with eight million sample points and given nearest neighbors needed 24 steps and was performed in 6 minutes. For the four million version of the data set, the overall computation time for the entire level-set approach including pre-computations dropped to 84 seconds. This is a significant speed-up in comparison to the time of 13 minutes for the global level-set process [19] for the four million Hydrogen data set.[1] When comparing the computation times per iteration (including computation of zero level set, narrow band, and level-set function

[1] These numbers assume that the nearest neighbors are known.

update), the 7.8 seconds of our presented local level-set method are one order of magnitude faster than the 67.8 seconds for the global level-set method [19]. Still, our local level-set method produces equivalent results in terms of quality and correctness.

Next, we have analyzed the speed of the iterative phase of our approach with respect to the number of points in the α-band. We applied the method to a re-sampled unstructured point-based data set of eight million randomly distributed samples, generated from the regular Engine data set of size $256 \times 256 \times 128$. A rendering of the zero level set after convergence is shown in Figure 6.

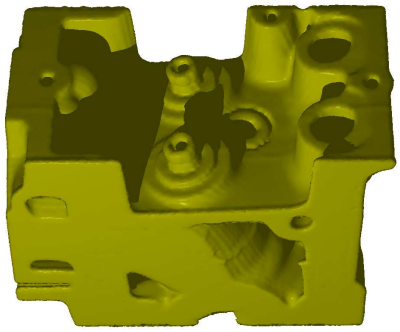


Figure 6: Splat-based ray tracing of the zero level set extracted after convergence of the local level-set method with $\lambda = 0.01$. The method was applied to the Engine data set with eight million unstructured sample points. (Data set courtesy of General Electric.)

Just by looking at the rendering, the complex topology and geometry of the zero level set is obvious. Consequently, the number of isopoints steadily increases during the local level-set evolution as well as the number of sample points in the narrow band. A comparison of the computation speed of our local method with different number of extracted zero level set points is given in Table 3. Note that the number of points in the α-band is increasing sub-linearly with the number of extracted zero level set points, since several surface parts get very close to each other and their neighboring regions share data points in the band.

ls points	α-points	point ex.	band u.	ls step
20k	130k	2.1 sec	2.0 sec	3.5 sec
49k	287k	4.3 sec	3.8 sec	6.3 sec
83k	404k	7.6 sec	6.9 sec	8.5 sec

Table 3: Computation times for the iteration phase of the Engine data set at different steps of the level-set process with different numbers of zero level set points. For each step, the number of extracted zero level set points, the number of points in the α-band, and all computation times within an iteration, i. e., for extracting the zero level set, updating the narrow band, and performing one level-set step, are given.

Finally, we applied our local level-set method to unstructured point-based volume data from a real scientific application. The data set was provided by astrophysical particle simulations [20] of Stephan Rosswog, Jacobs University, Bremen, Germany. In the simulation, a set of particles representing a White Dwarf is passing a black hole and is torn apart by the strong gravity. The data set represents one time step of this simulation and consists of 500k sample points. The density of the points in space is varying in two orders of magnitude, directly showing that scattered data interpolation techniques would not be applicable due to the introduction of enormous errors.

We applied our local level-set method to this practical data set to compare the results of the local and the global method. Both methods were applied with the same initial signed-distance function, the same isovalue, and the same smoothness parameter. Renderings of the obtained zero level sets after convergence of both methods are shown in Figure 7. The presented local level-set method required only 6 minutes of computation time (including all computations) until reaching steady state. In comparison, the global level-set process using actually asynchronous time integration required 68 minutes. In terms of quality, there are no significant differences visible between the resulting surfaces. We have measured the distance between the points of the two extracted zero level sets. The maximum deviation between the surfaces was always of the same order as the stopping criterion for the level-set process. This results from the linear interpolation of surface points from the quasi-linear level-set function.

All experiments have shown, that the proposed local level-set method is significantly faster than the global method but equivalent in terms of quality. The narrow-band method is also able to process objects with complex topology and is applicable to real-world data.

8 CONCLUSION

We have presented a local level-set method that combines the accuracy of directly applying level sets to unstructured data and the fast level-set computation using narrow bands. We directly operate on the unstructured data and do not need to process data at any positions other than the sample points, effectively avoiding interpolation errors. For each iteration, we extract the zero level set, build a narrow band of sample points around it and process the level-set function only in the narrow band with respect to a combination of normal advection with mean curvature flow. The final zero level set is visualized using point-based rendering.

Our method is capable of robustly processing data sets with several million sample points and highly varying point density. For all investigated examples we achieved a speed-up of one order of magnitude compared to the global level-set method. Nevertheless, the proposed method achieves the same results in terms of quality.

Figure 7: Comparison of global and local level sets for the White Dwarf data set with 500k unstructured sample points with highly varying point density. Both methods have been applied to the same initial signed-distance function using the same isovalue and the same smoothness parameter $\lambda = 0.1$. The zero level set obtained by the global level-set method with asynchronous time integration is shown on the left-hand side. In comparison, the smooth isosurface generated by our local level-set method, shown on the right-hand side, exhibits no significant difference in terms of quality, but was generated more than ten times faster.

ACKNOWLEDGMENTS

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under project grant LI-1530/6-1.

REFERENCES

[1] David Adalsteinsson and James A. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118(2):269–277, 1995.

[2] David Breen, Ross Whitaker, Ken Museth, and Leonid Zhukov. Level set segmentation of biological volume datasets. In Jasjit S. Suri, David L. Wilson, and Swamy Laxminarayan, editors, *Handbook of Biomedical Image Analysis, Volume I: Segmentation Models, Part A*, pages 415–478, New York, NY, USA, 2005. Kluwer.

[3] Richard Courant, Kurt Otto Friedrichs, and Hans Lewy. On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, 11(2):215–234, 1967.

[4] Herbet Edelsbrunner and Nimish R. Shah. Incremental topological flipping works for regular triangulations. In *SCG '92: Proceedings of the eighth annual symposium on Computational geometry*, pages 43–52, New York, NY, USA, 1992. ACM.

[5] Douglas Enright, Frank Losasso, and Ronald Fedkiw. A fast and accurate semi-lagrangian particle level set method. *Computers and Structures*, 83(6–7):479–490, 2004.

[6] Lawrence C. Evans and Joel Spruck. Motion of level sets by mean curvature I. *Journal of Differential Geometry*, 33(3):635–681, 1991.

[7] Richard Franke and Gregory M. Nielson. Scattered data interpolation: A tutorial and survey. In Hans Hagen and Dieter Roller, editors, *Geometric Modeling: Methods and Applications*, pages 131–160. Springer, New York, NY, USA, 1991.

[8] Simone E. Hieber and Petros Koumoutsakos. A lagrangian particle level set method. *Journal of Computational Physics*, 210(1):342–367, 2005.

[9] Hon Pong Ho, Yunmei Chen, Huafeng Liu, and Pengcheng Shi. Level set active contours on unstructured point cloud. In *CVPR '05: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 690–697, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[10] Aaron E. Lefohn, Joe M. Kniss, Charles D. Hansen, and Ross T. Whitaker. A streaming narrow-band algorithm: Interactive computation and visualization of level sets. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):422–433, 2004.

[11] Lars Linsen, Karsten Müller, and Paul Rosenthal. Splat-based ray tracing of point clouds. *Journal of WSCG*, 15:51–58, 2007.

[12] Chohong Min. Local level set method in high dimension and codimension. *Journal of Computational Physics*, 200(1):368–382, 2004.

[13] Bryan S. Morse, Weiming Liu, Terry S. Yoo, and Kalpathi Subramanian. Active contours using a constraint-based implicit representation. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 285–292, Washington, DC, USA, 2005. IEEE Computer Society.

[14] Ken Museth, David E. Breen, Leonid Zhukov, and Ross T. Whitaker. Level set segmentation from multiple non-uniform volume datasets. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 179–186, Washington, DC, USA, 2002. IEEE Computer Society.

[15] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*. Springer, New York, NY, USA, 2003.

[16] Stanley. Osher and James A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[17] Danping Peng, Barry Merriman, Stanley Osher, Hongkai Zhao, and Myungjoo Kang. A PDE-based fast local level set method. *Journal of Computational Physics*, 155(2):410–438, 1999.

[18] Paul Rosenthal and Lars Linsen. Direct isosurface extraction from scattered volume data. In Beatriz Sousa Santos, Thomas Ertl, and Kenneth I. Joy, editors, *EuroVis06: Proceedings of the Eurographics/IEEE-VGTC Symposium on Visualization*, pages 99–106, Aire-la-Ville, Switzerland, 2006. Eurographics Association.

[19] Paul Rosenthal and Lars Linsen. Smooth surface extraction from unstructured point-based volume data using PDEs. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1531–1546, 2008.

[20] Stephan Rosswog, Enrico Ramirez-Ruiz, William Raphael Hix, and Marius Dan. Simulating black hole white dwarf encounters. *Computer Physics Communications*, 179(1–3):184–189, 2008.

[21] James A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, Cambridge, UK, 2nd edition, 1999.

[22] Mark Sussman and Emad Fatemi. An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow. *SIAM Journal on Scientific Computing*, 20(4):1165–1191, 1999.

3D Facial Animation for Mobile Devices

José Mario De Martino and Tatiane Silvia Leite
Department of Computer Engineering and Industrial Automation
School of Electrical and Computer Engineering
University of Campinas
P.O. Box 6101 - ZIP 13083-970 Campinas - SP - Brazil
martino@dca.fee.unicamp.br, tatiane.leite@gmail.com

ABSTRACT

This article presents the implementation of a 3D facial animation system for mobile devices. Due to the large processing and memory requirements for this type of application, its use on mobile devices was not possible until recently. Currently, however, with the increasing development of powerful hardware and with the spread of cellular telephony, 3D applications for these devices have become even more promising. The system presented in this article was implemented based on a previous one designed for desktop platforms. Both systems are driven by two kinds of input. The first one is an audio file containing the speech to be presented, and the second is a timed phonetic transcription specifying the phonemes of the utterance and their associated time spans. With the input data, the system converts phonemes into phonetic context dependent visemes that handle perseveratory and anticipatory coarticulation effects. The visemes are then used to modify the 3D polygonal mesh of the fa ce keeping the synchronization with the audio. The initial evaluation of the systems indicates that, despite the limitations of mobile platform, the implementation of video telephony using 3D facial animation is viable in modern mobile devices.

Keywords: facial animation, mobile devices, visemes.

1 INTRODUCTION

Facial animation is a crucial technology for the development of embodied virtual characters capable of reproducing the communication style, based on speech, facial expressions and gesture, which humans are well familiar with. This technology refers to the techniques for specifying and controlling a synthetic face to reproduce facial expressions and the visual articulatory movements associated with the production of the speech. Unlike the cartoon-like animations, which do not represent all the movements of the face during a speech, admit exaggeration, and usually are not concerned with reality, computer facial animation usually strives for realism.

Depending on the application, the virtual characters created by the computer facial animation can play such diverse roles as instructors, assistants, avatars, presenters or sale representatives. Not so long ago this kind of software was only possible to be implemented in rendering servers or workstation with powerful processor-memory architecture. The large requirements of real-time processing were impediments for implementing this technology in other platforms, specially on mobile ones. But thanks to the advances in microelectronics and the spread of mobile telecommunication, modern mobile devices have the powerful hardware required for this type of application. Additionally the features demanded by users currently have changed significantly, which stimulate the development of innovative applications such as facial animation for mobile devices. In particular, the deployment of facial animation on mobile phones is a promising option for mobile video telephony due to the poten tial reduction of data transmission. In this scenario, instead of video transmission, it is only necessary to transmit a few parameters specifying the speech and associated facial expressions.

The system implemented seeks to perform the animation of the human face in close agreement with the acoustic signal of speech. For the generation of facial animation in sync and in harmony with the speech, it is imperative to reproduce the articulatory movements associated with all phonemes of the language. For that, in addition to identifying the postures of the articulatory gestures associated with the phonemes, it is also necessary to have the representation of the transitions between these postures considering the coarticulation effects.

The coarticulation effects are expressed by the changing of the articulatory pattern of a particular phoneme according to the influence of the articulation of an adjacent phoneme or a close one in the chain of sound production. The effects of coarticulation make, for instance, the "b" of the word "bat" visually different from the "b" of the word "boot". In the latter case, the articulatory movement necessary for the production of the sound associated with the "oo" significantly influences the visible aspects of the articulation of the "b". The coarticulation can be classified as perseveratory or anticipatory. In perseveratory coarticulation, the articulation of a speech segment is influenced by the ar-

ticulatory movement of a predecessor segment in the chain that composes the phonetic speech. In anticipatory coarticulation, the articulatory movement of the segment is influenced by the articulation of a successor segment.

One of the most important issues of any speech synchronized facial animation system is the establishment of proper parameters for the visible articulatory representation, which is also known as visemes. The system proposed in this article is based on a set of visemes that contemplates the effects of adjacent perseveratory and anticipatory coarticulation for the Brazilian Portuguese language [1].

This paper is organized as follows: Section 2 shows a review of the state of the art in facial animation for mobile devices; Section 3 presents the system architecture; Section 4 provides an overview of the set of visemes; Section 5 explains the facial movements changes occurred during the speech; Section 6 discusses the influence of the movement in the vertexes around the mouth; Section 7 presents the strategy to model de visible articulatory movements based on the articulatory targets of the visemes; Section 8 shows a preliminary evaluation of the implemented system; and finally in Section 9 the conclusion is presented.

2 RELATED WORKS

Many different approaches for facial animation designed to run on desktop computers have been proposed in the literature. However, there are fewer published works that refer exclusively to facial animation for mobile platforms.

The main articles in this area have used the MPEG-4 standard [4], also characterized as geometric parametrization, to perform facial animation. Next, it is summarized the most important works based on this standard.

The system presented in [8] describes an approach to enable facial animation framework applications for the Web and mobile platforms. In the work [6], the facial animation is developed to achieve the implementation of a system called "LiveMail: Personalized Avatars for Mobile Entertainment". In this system, the speech and the appropriate facial animation are created automatically by speech synthesis. The facial animation created can be sent over the network as 3D animated messages or as short videos in MMS. The work in [7] describes a parametric 3D facial animation capable of representing and animating faces in a photo-realistic manner.

All mentioned research efforts are based on the MPEG-4 standard for facial animation. In this standard, a face is defined by 84 FPs (Feature Points), which are controlled by 68 FAPs (Facial Action Parameters). Among all 68 FAPs, 66 are considered low level. These FAPs are based on the study of minimal facial actions and are related to muscle actions. They represent a complete set of basic facial actions, and therefore allow the representation of facial expressions more naturally.Basically each low-level FAP is used to define the displacement of an FP in a particular direction. The FPs (and their associated FAPs) are arranged in ten different groups. The FAPs can be efficiently compressed and included in a Face and Body Animation (FBA) low-bit rate streams. Any MPEG-4 compliant facial animation system can decode and interpret FBA bitstreams, and a synthetic animated face can be seen.

The realistic articulatory facial animation in the MPEG4 standard presents difficulties. The low-level FAPs, despite their names suggest an articulatory interpretation, are purely geometric parameters. For example, the FAP *Open-Jaw* operates only in the FP *Bottom of the chin* not influencing, for example, the FPs *Bottom of the lower teeth* and *Top of the lower teeth*, which should move with the jaw in an articulatory interpretation. Additionally, the low-level FAPs do not take into account articulatory gestures as, among others, the mouth opening and the height of the mouth and lip protrusion. The mapping of these gestures to a set of FAPs is not obvious, and is not defined in the standard. To overcome these difficulties, the standard defines a set of high-level FAPs to represent visemes. Unfortunately the 14 high-level visemes that has been defined have the serious limitation of only contemplating, if so, the English language. Furthermore, the standard leaves to the developer the decision of how these visemes deform the face. And finally, it is possible, although not recommended, to change the face using the high-level and low-level parameters simultaneously, which can lead to unpredictable results.

The work [5] presents the system called InCA, which is a distributed personal assistant conversational agent. The front-end runs on a PDA and uses facial animation and natural speech to interact with the user and provide services such as e-mail, calendar, weather reports, among other services, using the network to overcome the PDA computational limitations. Its facial animation considers input data very similar to what it used in work presented in this paper. InCA also needs a playlist with phonemes and the corresponding timing information and the audio speech. However, InCA uses only 18 mouth positions which are generated before the speech production. This way, each phoneme produced is mapped into one of these mouth positions.

Differently from the above mentioned works, the approach presented in this paper uses phonetic context dependent visemes that cope with perseverative and anticipatory coarticulation. Additionally, the movements of the temporomandibular joint and the lip tissue are modeled from a set of visemes established by the analysis of a Brazilian Portuguese linguistic corpus. Al-

though the corpus is restricted to Brazilian Portuguese, the methodology is general enough to be applied to other languages.

3 SYSTEM ARCHITECTURE

The implemented system can be represented by the schematic diagram in Figure 1. The core of the system, shown in the darker area, consists of two modules: Phoneme-Viseme Converter and Facial Animation. The Phoneme-Viseme module is responsible for converting the timed phonetic description in a sequence of visemes. The Facial Animation module uses the sequence of visemes generated by the Phoneme-Viseme converter to control the movements of the virtual face. The main task of the facial animation system is to reproduce the articulatory movements described by the visemes and, through the required calculations, to generate the sequence of images that make up the animation.

For each viseme a sequence of images is generated and shown on the screen of the mobile device. As soon the images are generated from Facial Animation module, they are reproduced on the screen synchronized with the audio, using the functionality of the Mobile Media API for this management [10].

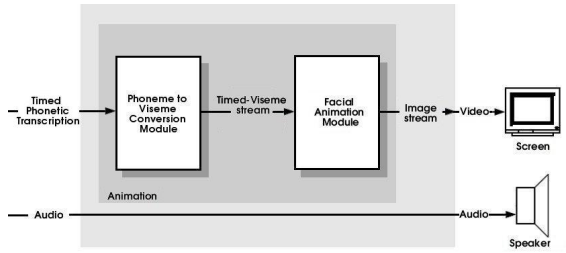


Figure 1: Diagram of animation system implemented.

The first step of the system is the input of the timed phonetic transcription and the corresponding conversion of the phonetic segments into visemes. This timed phonetic transcription contains the sequence of phonetic segments that compose a particular utterance, as well as the duration of each segment. From that description, the phonetic segments that compose the speech are separated into groups of visemes, and at the end of the process a sequence of visemes, identified and established for the Brazilian Portuguese [1] is obtained. Such visemes will give the necessary information for the creation of different images of the virtual head through its parameters.

After the identification of the visemes in the speech, the system makes the changes corresponding to these geometric visemes, modifying the polygonal mesh of the 3D virtual face.

4 PHONEME TO VISEME CONVERSION

For the characterization of a set of visemes for the Brazilian Portuguese, the article [1] identifies the visible patterns of articulatory movement during the production of the phonetic segments of the language in different phonetic contexts.

The identified visemes are expressed by a parametric model that incorporates the articulatory targets established for the segments and the instant of reaching these targets within the range of sound production, still contemplating the effects of coarticulation between adjacent segments in the chain of speech production.

The context-based visemes expressed with the symbols of the International Phonetic Alphabet [3] are shown in Tables 1 and 2.

Viseme	Context
<p$_1$>	[pi] [pa] [ipɪ] [ipɐ] [ipʊ̈] [apɪ] [apɐ] [apʊ̈] [upɐ]
<p$_2$>	[pu] [upɪ] [upʊ̈]
<f$_1$>	[fi] [fa] [ifɪ] [ifɐ] [ifʊ̈] [afɪ] [afɐ]
<f$_2$>	[fu] [afʊ̈] [ufɪ] [ufɐ] [ufʊ̈]
<t$_1$>	[ti] [tu] [itɪ] [itɐ] [itʊ̈] [atɪ] [atʊ̈] [utɪ] [utɐ] [utʊ̈]
<t$_2$>	[ta] [atɐ]
<s$_1$>	[si] [sa] [isɪ] [isɐ] [asɪ] [asɐ]
<s$_2$>	[su] [isʊ̈] [asʊ̈] [usɪ] [usɐ] [usʊ̈]
<l$_1$>	[li] [ilɪ] [alʊ̈] [ulɪ] [ulɐ]
<l$_2$>	[la] [ilɐ] [alɪ] [alɐ]
<l$_3$>	[lu]
<l$_4$>	[ilʊ̈] [ulʊ̈]
<ʃ$_1$>	[ʃi] [ʃa] [iʃɪ] [iʃɐ] [iʃʊ̈] [aʃɪ] [aʃɐ] [aʃʊ̈] [uʃɪ] [uʃɐ]
<ʃ$_2$>	[ʃu] [uʃʊ̈]
<ʎ$_1$>	[ʎi] [ʎa] [iʎɪ] [iʎɐ] [aʎɪ] [aʎɐ]
<ʎ$_2$>	[ʎu] [uʎɪ] [uʎɐ]
<ʎ$_3$>	[iʎʊ̈] [aʎʊ̈] [uʎʊ̈]
<k$_1$>	[ki] [ikɪ] [ikɐ] [akɪ] [ukɪ] [ukɐ]
<k$_2$>	[ka] [akɐ]
<k$_3$>	[ku] [ikʊ̈] [akʊ̈] [ukʊ̈]
<ɾ$_1$>	[ɣi] [ɣa] [iɾɪ] iɾɐ] [aɾɪ] [aɾɐ] [uɾɐ]
<ɾ$_2$>	[ɣʊ̈] [iɾʊ̈ [aɾʊ̈] [uɾɪ] [uɾʊ̈]

Table 1: Consonantal visemes and their phonetic contexts.

Viseme	Context
<i$_1$>	all contexts but [tit] and [ʃiʃ]
<i$_2$>	[tit] [ʃiʃ]
<a>	all contexts
<u>	all contexts
<ɪ>	all contexts
<ɐ>	all contexts
<ʊ̈>	all contexts

Table 2: Vocalic visemes and their phonetic contexts.

It is important to remember that although the system has been developed for the parameters determined for

the Brazilian Portuguese, it can be adapted to other languages.

5 FACIAL MOVEMENTS

The geometric transformations applied to the 3D head geometric model are of two kinds: rigid body transformations and deforming transformations. Rigid body transformations are associated with the articulatory movement that allows the protrusion or retraction (moving forward and backward) and elevation or depression (moving up and down) of the jaw, working only with the lower lip vertexes, and with the face below the lower lip, as shown in Figure 2.

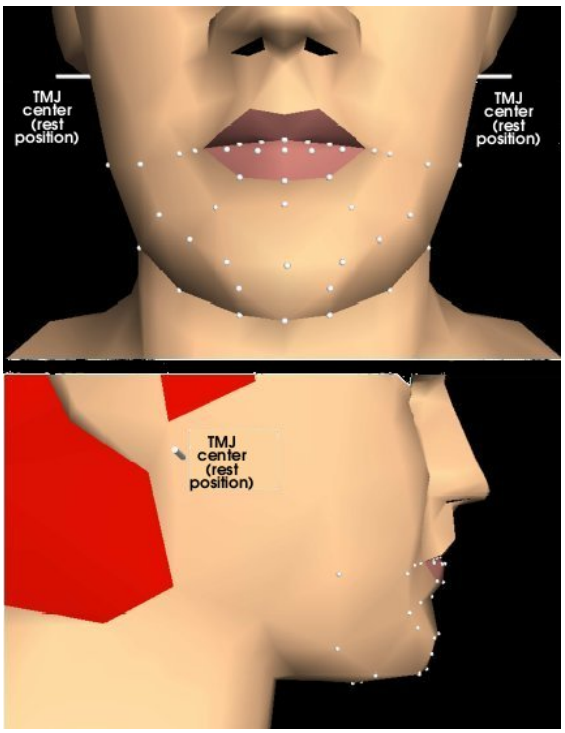


Figure 2: Vertexes transformed because of the articulatory movements.

The deforming transformations are associated with the voluntary movements of the face tissue, which includes the region of the lips and the region around them that characterize the production of the speech segments.

The facial deformations are represented by the trajectories of the fiduciary points $P1$, $P2$ and $P3$, located in the upper lip, in the corner of the mouth and in the lower lip, respectively, as seen in Figure 3. However, only the $P3$ point suffers significant influence from the jaw movement, showing that this is the point that suffers greater influence considering both movements: voluntary movement of the lip and the movements of the jaw. The $P1$ and $P2$ points don't suffer significant influence from the movements of the jaw and their behavior are defined only by the voluntary movements of the lip. The trajectories of the fiduciary points were captured from a real speaker as reported in [1].

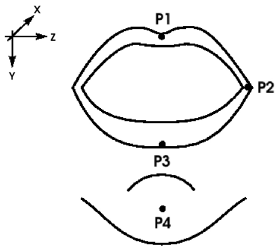


Figure 3: Fiduciary Points.

The rigid body transformations assume that, during speech production, the TMJ (temporomandibular joint) executes both rotation movement and translation movement in sagittal plane). The rotation movement is performed around the center of the TMJ, but this center can also suffer translation due to the sliding of the articular disc, shown in Figure 4. Thus, the trajectory of the fiduciary P4 point located in the chin of the speaker, represents the motion of the jaw and allows to estimate the components of TMJ rotation and translation.

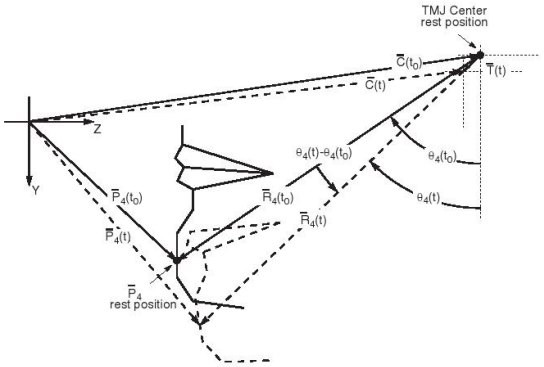


Figure 4: Temporomandibular joint behavior.

5.1 Temporomandibular joint behavior

The rotation movement of the jaw is done around the TMJ center that undergoes translation due to movement of the articular disc. Assuming that the trajectory from the fiduciary P4 point, in Figure 4, reflects the movement of the jaw, it is possible to estimate the components of rotation and TMJ translation.

The angle of rotation can be given by the equation 1.

$$\theta_4(t) = arctan\left(\frac{z_4(t) - z_C(t_0)}{y_4(t) - y_C(t_0)}\right) \quad (1)$$

where $y_4(t)$ and $z_4(t)$ are the coordinates y and z of the P4 trajectory at instant t, and $y_C(t_0)$ and $z_C(t_0)$ are the coordinates x and y of the TMJ center at instant t_0.

It is also possible to estimate the r_4 radius, considering the position of rest, presented in equation 2.

$$r_4 = \sqrt{[y_4(t_0) - y_C(t_0)]^2 + [z_4(t)0) - z_C(t_0)]^2} \quad (2)$$

And the components of TMJ translation are given by the equations 3 and 4.

$$y_T(t) = y_4(t) - y_C(t_0) - r_4 cos(\theta_4(t)) \quad (3)$$

$$z_T(t) = z_4(t) - z_C(t_0) - r_4 sin(\theta_4(t)) \quad (4)$$

5.2 Lower lip behavior

The fiduciary $P3$ point, in Figure 3, represents the movement of the lower lip. Its trajectory may be separated into two components. The first is referent to the jaw movement that affects the opening of the mouth, thus changing the position of the lower lip. Therefore, the angle of $P3$ rotation is given by the equation 5.

$$\theta_3(t0) = arctan\left(\frac{z_3(t0) - z_C(t_0)}{y_4(t) - y_C(t_0)}\right) \quad (5)$$

And the r_3 radius is given by equation 6.

$$r_3 = \sqrt{[y_3(t_0) - y_C(t_0)]^2 + [z_3(t)0) - z_C(t_0)]^2} \quad (6)$$

The second component is referent to the voluntary movement of the lip tissue. This movement is necessary to produce some specific articulatory postures, as protrusion and lip extension. The equations 7 and 8 represent the behavior of the lower lip.

$$y_L(t) = y_3(t) - y_4(t) - r_3 cos(\theta_3(t_0) + \theta_4(t) - \theta_4(t_0)) - r_4 cos(\theta_4(t)) \quad (7)$$

$$z_L(t) = z_3(t) - z_4(t) - r_3 sin(\theta_3(t_0) + \theta_4(t) - \theta_4(t_0)) - r_4 sin(\theta_4(t)) \quad (8)$$

5.3 Upper lip behavior

It is assumed that the behavior of the upper lip does not have significant influence over the handling of the temporomandibular joint, so their movements are defined exclusively by the voluntary movement of the upper lip tissue in order to produce the sounds of speech. Therefore, the fiduciary $P1$ point is responsible for defining the behavior of the upper lip.

6 REGIONS OF INFLUENCE

To make the geometric changes concerning the voluntary movement of facial tissue around the mouth, the original system provides a model to express the visible movement of this tissue during the talk.

This model provides a spheroidal region of influence around the mouth, inspired in the elliptical formation of the orbicular muscle of the mouth [9]. The region of influence was split into two parts:

- Upper Region: influenced by the behavior of the upper lip;
- Lower Region: influenced by the behavior of the lower lip.

In the system developed for PC, each one of these regions are two concentric spheroids, illustrated in Figure 5, which presents the region that influences the upper lip behavior. And considering these spheroids, the distance is calculated from each vertex to the $P1$ and $P2$ points and from these calculations the movement factor is set, which determines how much each vertex is influenced by the changes.

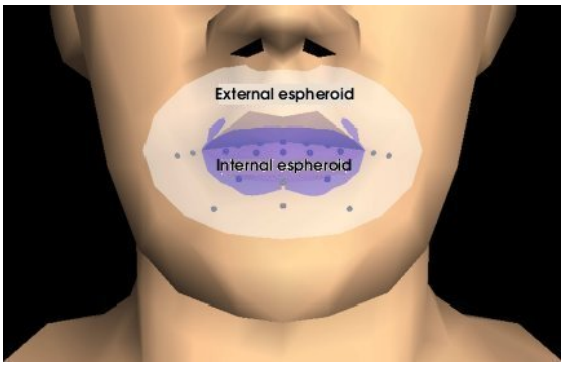


Figure 5: Upper region of influence and associated vertices.

But in the developed system, the values of these factors were pre-established, so they are not calculated during the execution of the program. It was designed to avoid an overburdened processing. And from the values of these factors are calculated the new positions of the vertexes, and in sequence performed the geometric transformations.

7 INTERPOLATION

The interpolation curve adopted for the representation of the movement was the cubic parametric curve of Hermite [2]. This choice was taken because it is important that the curve of geometric interpolation preserves the continuity between the various segments of a chain of visemes / phonemes and guarantees that the time derivative equals to zero in the instants when the articulatory target is reached.

Thus, knowing the values of displacements in x, y and z directions associated with the articulatory targets of a viseme and the instants of reaching these targets, it is possible to approximate the trajectory of the fiduciary points in a chain of phonemes by the interpolation of the articulatory target displacements.

Equation 9 presents the parametric model adopted.

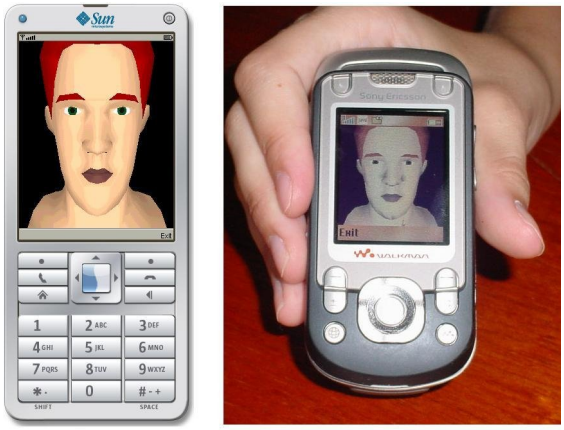
$$\begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} = \begin{bmatrix} I_x & F_x \\ I_y & F_y \\ I_z & F_z \end{bmatrix} \begin{bmatrix} 2 & -3 & 1 \\ -2 & 3 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ 1 \end{bmatrix} 0 \leq t \leq 1 \quad (9)$$

The parameters of the equation define:

- $x(t)$, $y(t)$ and $z(t)$: coordinates of the fiduciary point;
- I_x, I_y and I_z are the shifts in x, y, and z directions from the fiduciray point in the articulatory target of the initial viseme;
- F_x, F_y and F_z are the shifts in x, y, and z directions from the fiduciary point in the articulatory target of the final viseme;
- and t is an independent variable, normalized in relation to the time interval between two viseme targets.

8 PRELIMINARY EVALUATION

Figure 6 shows the snapshots of the implemented 3D facial animation system running on simulator and on a mobile phone.



(a) (b)

Figure 6: (a) in the WTK (Wireless Toolkit) simulator (b) in the device W600i Sony Ericson, which supports the 3D API.

The implemented system was evaluated and had some of its characteristics analyzed, such as the size of the resulting JAR file of the generated code, the necessary memory for the execution and a survey of the most relevant Java methods to the performance of the system through the time of execution and execution frequency of these methods. All these characteristics were raised using the well-known WirelessToolkit 2.5 simulator.

The first feature evaluated was the size of the JAR package, which when is generated contains all the files needed to run the application on the mobile device. The JAR file contains the M3G file with the virtual head, the audio file (WAV) with the speeches, the phonetic transcription timed for the speeches and the bytecodes (Java files) needed to execute the application in the Java Virtual Machine. At the end of the development, the JAR file had approximately 300 KBytes, which is therefore a viable size even for mobile phones with limited memory capacity.

The second feature evaluated was the size of dynamic memory (heap) needed for the execution. This plays an even more important role, because if we do not have enough memory to create the objects, the application will not be executed properly. The size of heap memory used in this application is approximately 2 MB, indicating that the current mobile devices will not have problems on executing the application, because they usually have a significantly greater heap memory.

Finally, the execution frequency of methods and the duration of these executions were analyzed. The results obtained met the expected results; approximately 78% of the processing time refers to the core image painting module (TalkingHeadCanvas.paint). This module contains the methods that are most called during the execution of the application. These methods refer to the updating schedule, calculations of rotation and translation and updating of images on the screen. The other 22% of the execution time is related to the other methods, such as M3G file loading, and also the WAV file loading, among others.

To test the implementation in real devices, it was downloaded into a Sony Ericsson mobile phone, model W600i, which is shown in Figure 6. Not only this Sony Ericsson model supports this type of application, but several models of the largest mobile phone manufacturers also support this type of application, such as Motorola, Nokia, LG and others. These newer models are able to run the implemented application, as well as have a reasonable processing power, an appropriate screen size, dynamic memory equal to or above 2 MB and support the Mobile 3D Graphics API (JSR-184).

9 CONCLUSIONS

Even with all the memory and processing limitations in a mobile device, 3D applications have shown to be increasingly promising for embedded systems. The system presented in this paper shows a solution of a specific facial animation for these devices. The main concern is beyond the processing power of the device; it refers primarily to the small display that probably could prevent the visualization of the speech of the talking head. In this work it was possible to see that despite the small size of the screen, the 3D presentation could be done in a satisfactory way, becoming a major attraction since the facial animation is a technology that can help to enable video telephony services at low rates of transmission, which is a common situation in cellular networks. The facial animation loaded on a mobile device needs only the transmission of phonetic timing information, or equivalent information for the visual presentation of the interlocutor, thereby preventing the transmission of a large volume of data associated with the video transmission.

Among the various aspects that make the production of a realistic facial animation a complex achievement,

the work has focused on the treatment of visible articulatory movements, taking into account the concept of visemes as well as the effects of articulation which are applied to them. that they suffer. In spite of the fact that the system was developed to Brazilian Portuguese speeches, it can easily be adapted for other languages.

REFERENCES

[1] Jose Mario De Martino, Leo Pini Magalhães, and Fabio Violaro. Facial animation based on context-dependent visemes. *Computers & Graphics*, 30(6):971–980, 2006.

[2] J. D. Foley, A. van Dam, S. K. Feiner, and Hughes J. F. *Computer Graphics Principles and Practice*. Addison-Wesley Publishing, 2 edition, 1990.

[3] IPA. *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*. International Phonetic Association, International Phonetic Association - Cambridge, 1999.

[4] ISO/IEC. *Information technology - Coding of audio-visual objects - Parte 2: Visual*. ISO/IEC 14496-2:2001(E), Geneva, Switzerland, December 2001. International Standard.

[5] M. W. Kadous and C. Sammut. *InCA: A Mobile Conversational Agent*. Springer, 2004.

[6] Miran Mosmondor, Tomislav Kosutic, and Igor S. Pandzic. Livemail: personalized avatars for mobile entertainment. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 15–23, New York, NY, USA, 2005. ACM.

[7] P. Omedas, F. Berrizbeitia, G. Szijártó, B. Kiss, and B. Takács. Model-based facial animation for mobile communication. *1st Ibero - American Symposium on Computer Graphics (SIACG)*, 2002.

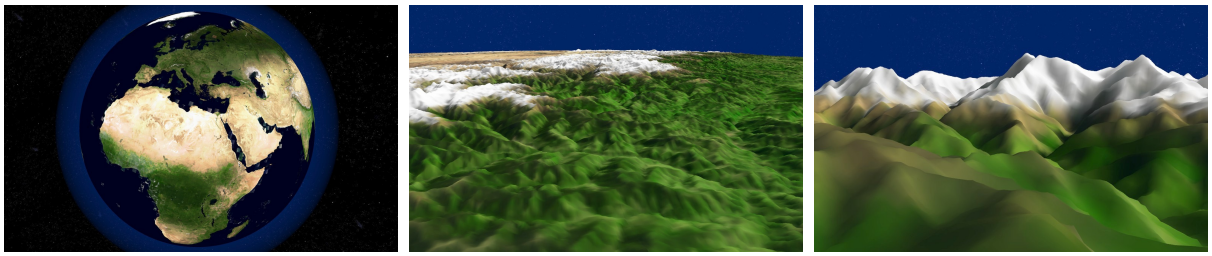
[8] Igor S. Pandzic. Facial animation framework for the web and mobile platforms. In *Web3D '02: Proceedings of the seventh international conference on 3D Web technology*, pages 27–34, New York, NY, USA, 2002. ACM.

[9] F. I. Parke and K. Waters. *Computer Facial Animation*. AK Peters, 1996.

[10] Sun Microsystem. *Java Specification Request 135: Mobile Media API*. `http://jcp.org/en/jsr/detail?id=135` (Last access: 05/14/2008).

Adaptive Real-Time Rendering of Planetary Terrains

Raphaël Lerbour Jean-Eudes Marvie Pascal Gautron
{raphael.lerbour, jean-eudes.marvie, pascal.gautron}@thomson.net
Thomson Corporate Research
1, avenue de Belle Fontaine, CS 17616
65576 Cesson-Sevigne, France



ABSTRACT

As virtual worlds applications get more and more demanding in terms of world complexity and rendering quality, rendering virtual terrains and planets in real-time introduces many challenges. In this paper, we provide a full-featured solution for rendering of arbitrary large terrain datasets in the context of client-server streaming. Our solution automatically adapts to arbitrary network bandwidths and client capabilities, ranging from high-end computers to mobile devices. We address the problem of rendering highly complex terrain databases comprising several hundred gigabytes of data, which therefore cannot be entirely loaded in memory nor rendered in real-time. The contributions of this paper solve important issues for high quality terrain rendering: adaptive texture mapping, inexpensive removal of geometry cracks and support of planetary terrains.

Keywords: Planetary terrain, adaptive rendering, adaptive streaming, generic data structure, level of detail.

1 INTRODUCTION

Virtual 3D worlds become more and more omnipresent in many applications, ranging from video games to cinema and virtual training. As time gets by, the virtual worlds applications get more and more demanding in terms of world complexity and rendering quality. Rendering virtual terrains and planets in real-time introduces many challenges, in particular for data representation, streaming and high quality rendering. We address the problem of rendering highly complex terrain databases comprising several hundred gigabytes of data, which therefore cannot be entirely loaded in memory nor rendered in real-time. Building upon a generic solution for terrain streaming, the contributions of this paper solve important issues for high quality terrain rendering: adaptive texture mapping, removal of geometry cracks and support of planetary terrains.

A virtual terrain is first described by geometry, which provides the relief information. However, the photometry embeds essential clues to the terrain structure and contents: a high quality terrain texture provides information on the ground type, vegetation, climate... We define a dual decomposition of the photometry information that matches the representation of the terrain and allows for high-performance, independent refinement of the geometry and photometry. Consecutive levels of details are filtered for further quality improvement.

While the use of multi-resolution eases the streaming and rendering of large terrains, artifacts or *cracks* tend to appear at the edge of adjacent terrain patches which use different levels of detail. We introduce a simple and secure method for stitching together adjacent patches regardless of their current level of detail.

Virtual terrains are generally considered planar, yielding a straightforward management of elevation data. However, our method supports arbitrarily large terrain datasets, and is hence able to render an entire planet with high detail. We address the representation of planetary terrains in which the elevation is relative to a reference ellipsoid instead of a plane. Based on a gnomonic projection scheme, we devise a crack-free uniform-angle sampling method for high quality projected elevation data. Our approach reduces the size of the dataset while preserving geometry and photometry

details.

Current graphics hardware rely on depth buffering for determining the polygons visible through each pixel of the image. At the planetary scale, the precision of the depth buffer may be challenged, yielding flickering and crawling artifacts. We solve this issue by adjusting the clipping planes of the view frustum so that the rendered planet surface. This increases the depth precision by using the entire depth representation range, while clipping parts of the planet located behind the horizon.

This paper is organized as follows: Section 2 describes relevant previous work in the field of adaptive rendering of large terrains. Section 3 presents the generic adaptive streaming solution in which our contributions are integrated. The next sections introduce our contributions: Section 4 presents our method for adaptively mapping high quality texture maps. In Section 5, we devise our method for patch stitching across different levels of details. Section 6 addresses the rendering of planetary terrains and adaptive clipping.

2 RELATED WORK

Adaptively streaming and rendering huge terrain maps requires using specifically adapted data structures and algorithms. An extensive survey of techniques for rendering large terrains have been recently published by Pajarola *et al.* [PG07]. This section describes previous work particularly related to this paper.

To manage those maps all the way from a server hard disk to a client rendering system, we use an existing generic solution based on a hierarchy of square blocks with levels of detail [LMG09]. Section 3 briefly describes this solution.

The *Clipmap* [TMJ98] is a powerful solution for huge texture support in 3D hardware rendering with no constraint due to geometry. *Geometry Clipmaps* [LH04] extend this concept to geometry and photometry information. While this method provides high performance and quality, the level of detail selection is performed in circular zones all around the viewer, without accounting for the actual visual importance of each terrain block. Furthermore, such circular refinement scheme may introduce a performance hit when streaming the terrain over a low-bandwidth network.

However, the generic solution we use for adaptive streaming and rendering can handle any kind of map: like many other works [CGG+03, WMD+04, Hwa05], we link two identical hierarchical structures respectively representing textures and elevation.

To avoid cracks between adjacent blocks when rendering a hierarchy of geometric terrain blocks, one can use triangular blocks with fixed resolution on their edges [Lev02, CGG+03, Hwa05]. This needs no explicit handling but prevents using levels of detail in blocks. *Geomipmaps* [dB00] use square blocks with levels of detail like us, and fix cracks by skipping samples on the edges of high definition blocks when they are not used by their neighbors. There is no multi-resolution block hierarchy so neighborhood management is simple, but this is not adapted for large terrains. Finally, one can solve cracks in a hierarchy of square blocks with no neighborhood knowledge by selecting LODs at block edges instead of centers [LKES07]. Block centers are then rendered using additional sample masks to internally stitch the LODs of the edges. Unfortunately, this requires all the data of the terrain to be available so that both sides of an edge render at the selected LOD.

Planets in *Google Earth*, *NASA World Wind* and other solutions [CH06] use the equirectangular cylindrical or *plate carrée* projection, the traditional standard for digital representations of planets. This projection causes large sampling distortions for non-equatorial regions: in particular, polar areas look very stretched when rendered and much redundant data are stored and rendered. Using a cube to represent the planet like [CGG+03] offers a significant improvement. However this solution does not use regularly sampled blocks: all samples need barycentric coordinates to compute their 3D position from those of the corners of the block. We prefer using a regular map projection and directly obtain positions with no additional information or interpolation.

3 GENERIC STREAMING SOLUTION

The generic solution underlying our work adapts to the speed of the network and the rendering performance of the client so any database can be used in any conditions [LMG09] (Figure 1). This section recalls the base data structure and the algorithms of [LMG09].

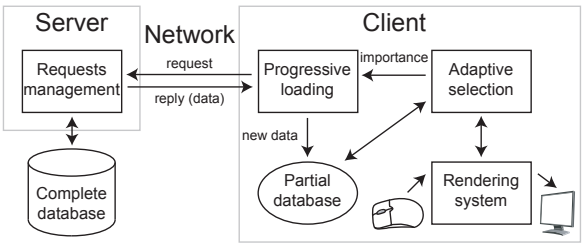


Figure 1: Architecture for adaptive streaming and rendering of terrain data. The user guides rendering on the client. Selected available data are rendered while missing data are requested and fetched from the server.

3.1 Data structure

The data structure, presented in Figure 2, consists in subdividing the sample map into a uniform tree of blocks, then organizing the samples of these blocks into a succession of levels of detail (LODs) of increasing resolution. Successive blocks and LODs share their data to avoid redundancy: new samples are spatially interleaved between the previous ones. In addition, a block only needs the data of its first LOD to be ren-

dered, allowing the other LODs to be loaded progressively.

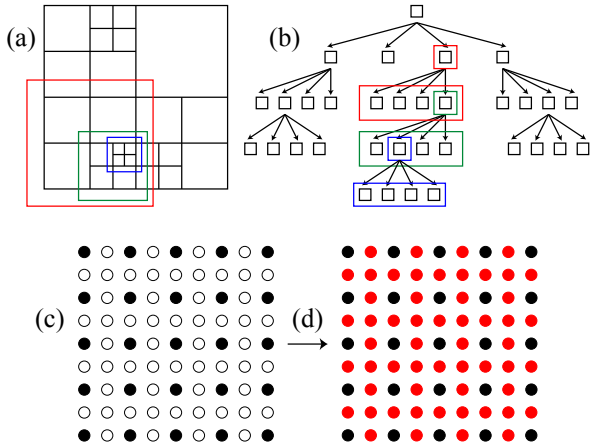


Figure 2: Tree of blocks with levels of detail. **a)** Successive subdivisions of the terrain map on the client side (red, green, blue). **b)** The corresponding incomplete tree. **c)** First LOD of a 9×9 block, only solid black samples are used **d)** Red samples (second LOD) are interleaved between black ones (first LOD).

3.2 Adaptive streaming

The complete tree of blocks is first stored in a single file on the server's hard disk. The data for any client request are contiguous and their position is obtained directly.

The client then progressively loads data from the server. A measure of importance guides the order in which data requests are transmitted. To optimize the relevance of loaded data and prevent overloading the network, the request queue is continuously updated.

An incomplete tree of blocks is explicitly stored on the client and constantly updated with asynchronous split and merge operations. When a block is created during a split, it allocates a single 2D array of samples in memory and obtains its first LOD from its parent. It can then load and interleave new LODs in previously unused array positions.

3.3 Adaptive rendering

The last part of the solution is the selection of data to render on the client. This part first culls invisible blocks, then chooses an LOD for each block using a measure of importance. This measure depends on a general quality factor that adapts to the rendering speed. When a desired LOD is unavailable, this triggers a new data request or tree update operation and an available LOD is used instead. Once an LOD is chosen for rendering, a precomputed mask extracts its samples from the array of the block as in [PM05].

4 ADAPTIVE PHOTOMETRY

Texture maps convey much detail about the terrain such as the local climate, ground type and vegetation. Such information tends to contain high frequencies, and therefore the texture maps are usually larger than elevation maps. Just like the terrain geometry, this potentially huge amount of data requires an adaptive management of the levels of detail. We propose to extend the geometry management scheme presented in Section 3 by introducing a dual multi-resolution structure for texture management. Geometry and photometry are kept coherent by linking their respective LOD trees.

4.1 Linked trees

Textures are mapped onto the terrain geometry to provide extra details on the appearance of the ground. As the texture and elevation LOD trees may have different size and depth, we use distinct trees and maintain a link between the two incomplete trees on the client so that each elevation block uses an adapted texture block. In practice, each block stores a link to the block of the other tree which covers the same terrain area, if any.

Classical texture rendering usually requires mapping one texture onto each elevation block. Our approach relaxes this constraint: when rendering an elevation block, we use its link to the texture tree to find the exact texture block correspondence. In available, the texture block is used directly. Otherwise, we recursively query the parents of the elevation block until a valid linked texture block is found.

We also introduce two constraints on split and merge operations to avoid creating texture blocks that cannot be used due to the lack of corresponding geometry data. First, we avoid splitting a texture block if the corresponding elevation block does not have any children on which the refined texture could be mapped. That is, a texture block cannot split if the linked elevation block is a leaf of the tree.

The second constraint prevents an elevation block from merging if its linked texture block has children. That is, an elevation block cannot merge if the linked texture block is not a leaf.

Let us recall that our solution aims at streaming terrain data through heterogeneous networks. Therefore, as described in [LMG09], the split and merge requests are queued and performed asynchronously with a potentially important delay due to network latency. Consequently, we enforce the constraints not only when triggering such operations, but also just before the actual execution of the operations.

4.2 Implementation Details

This section details specific implementation aspects of our work for further performance.

The split and merge operations are performed according to a measure of importance which depends on the on-screen coverage of the block. Therefore, we evaluate the importance for geometry blocks, and reuse it to refine their linked texture blocks.

A terrain block is defined by a triangle mesh, which requires texture coordinates for texture mapping. In our approach, the elevation tree may be deeper than the texture tree: a texture block may be mapped onto many terrain blocks. In this case the texture coordinates must be adapted to obtain a coherent appearance. Such adaptation can be performed inexpensively by transforming a default texture coordinate set using OpenGL texture matrices.

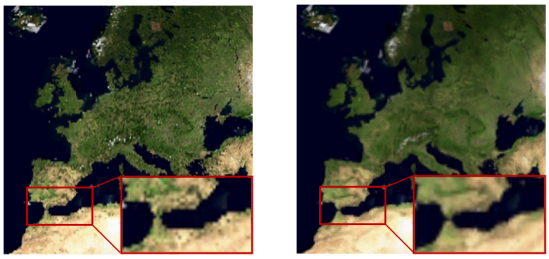
Texture LODs are implemented as mipmaps for 3D rendering: when we load a new LOD, we add a mipmap to the texture to increase quality. To enforce mipmapping even when only the first LOD of a block is available, we also create lower resolution mipmaps during splits.

We also take advantage of programmable graphics hardware to loosen the two constraints of Section 4.1. In those constraints, the refinement level of the texture tree is limited to avoid mapping several texture blocks onto one geometry block. Using the support of multiple textures in the fragment shader, a single geometry block can be rendered using several texture blocks.

4.3 Filtering

Although nearest-point subsampling is often acceptable for elevation maps, aliasing artifacts arise using this technique for texture maps. We therefore choose to use filtering when building higher levels of the texture tree during server file creation. We may use any image filtering method as long as it produces regularly sampled LODs. Wavelets come into mind, but we prefer using a scheme that is very fast to decode so it can be used with low performance clients.

We integrated the *Progressive Texture Map* (PTM) texture filtering method [MB03] in our solution (Figure 3). This multi-resolution scheme perfectly matches the LOD data structure used in our method. Furthermore, it uses simple and fast bilinear interpolation, and stores small delta values within each LOD to losslessly reconstruct samples using the previous LOD instead of redundantly transmitting all the samples. This adds a 8.3% overhead in LOD size compared to point sampling, but still saves 18.7% compared to redundant LOD streaming.



(a) Point sampling (b) PTM filtering

Figure 3: Visual impact of texture filtering.

When using texture filtering, different LODs of a block use different values for a given sample. This prevents us from using a single sample array for each block. We therefore store each LOD of the block in a separate array. During split operations, we split all of them instead of creating lower resolution mipmaps.

5 FIXING GEOMETRY CRACKS

Rendering multi-resolution terrains leads to the use of several levels of detail depending on the importance of each block. In terms of geometry, adjacent blocks may have different resolutions and hence cannot be perfectly stitched, yielding vertical gaps or "cracks" (Figure 4).

This section details a simple yet efficient method for avoiding such artifacts using *edge strip masks* to stitch together a set of geometry blocks with heterogeneous refinement levels.

(a) Rendering with cracks (b) Block stitching

Figure 4: Our method avoids cracks by stitching blocks with different resolutions.

5.1 Edge Strip Masks

As explained in Section 3.3, we use triangle strip masks to extract the samples of an LOD from the common sample array of a block. As adjacent blocks may have different resolutions, we aim at adjusting the triangle strip masks so that the edges of adjacent blocks share the same set of vertices.

We first propose to subdivide a block into five parts: inner block samples and four sets of edge samples. We keep the base idea of strip masks [PM05] in the inner block for managing the main level of detail of the block. On the block edges we create triangle strips called *edge strip masks*, which stitch the block to the level used in the adjacent block (Figure 5).

For each edge, *edge strip masks* systematically reduce the resolution of the block with the higher definition LOD, skipping vertices not used by the other block. Otherwise, we could need data that are not loaded yet. Note that only one large strip is used for blocks that do not need stitching.

Note that the *edge strip masks* undergo the same process as regular strip masks: each strip mask is cached within the graphics memory and reused at need. Therefore our crack removal technique does not introduce any processing nor rendering overhead.

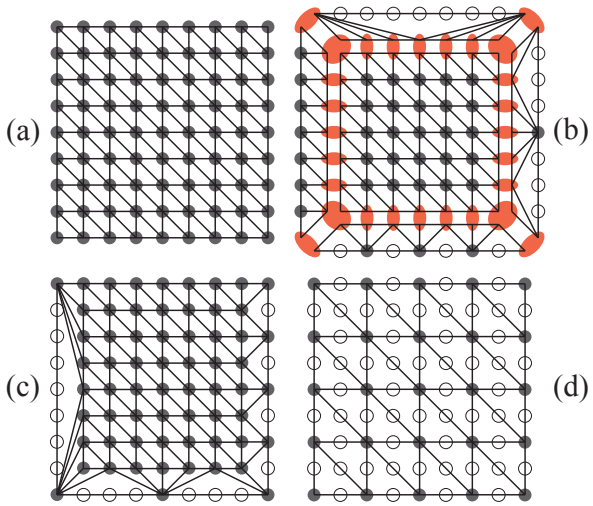


Figure 5: *Edge strip masks* for block edges. **a)** Block using its last LOD. **b)** Block using its last LOD, with *edge strip masks* on its edges to adapt to its neighbors based on the LOD difference. Left: no difference, bottom: one level, right: two levels, top: three levels. Red samples appear stretched for legibility. **c)** Mirror of (b), with the strips stitched as when rendered. **d)** Block using its next to last LOD.

5.2 Neighborhood management

Deciding which *strip edge* to use on the edges of a block requires knowledge of the resolution of its neighbors. To this end, we maintain a table of one neighbor per block edge, possibly with a different resolution. To ensure coherency and robustness, we do not update the neighbors list when a neighbor that is on the same tree level splits: this would lead to more than one neighbor per edge. In that case, we know that this neighbor block has equal or higher resolution. As the purpose of *edge strip masks* is the reduction of resolution across edges, we simply use the higher resolution *edge strip mask*. Potential reduction will be performed by the higher resolution neighbor blocks.

Note that our aim is the removal of cracks at the edges of the blocks. If an edge is not visible, we simply use the default *strip mask* as no adaptation is needed.

5.3 Split and merge constraints

In some cases, we cannot stitch the common edge of two neighbor blocks. This happens when the lowest-definition LOD of the block on the lower tree level has a higher definition than the current LOD of the other block (see Figure 6). This occurs especially often when using a block resolution not in $2^n + 1$ form. To enforce the robustness and reliability of the method, we add constraints of the split and merge operations to limit the maximum difference of tree levels between neighbors based on their resolution.

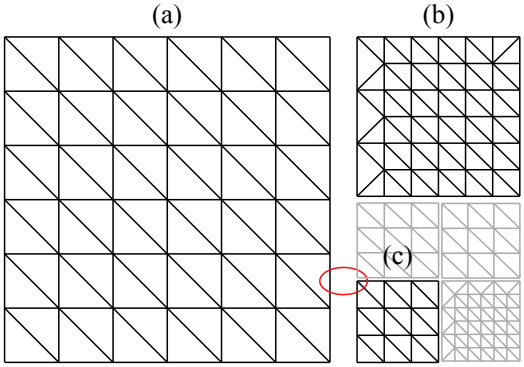


Figure 6: Non-stitchable blocks: **a)** A 7×7 block using its last LOD. **b)** A neighbor block located one tree level lower: it adapts on its left edge. **c)** A neighbor block located two tree levels lower. This block cannot adapt to block (a) as its upper-left sample (in red) is not shared by this neighbor.

6 PLANETARY TERRAINS

The previous Sections introduce our method for streaming and rendering highly detailed terrains. In this Section we propose an extension to planets by mapping a virtual terrain onto a planet-shaped ellipsoid.

6.1 Map projection

The representation of planetary information such as the 3D relief onto a 2D elevation map requires a step of projection, which can be done in numerous ways. Planets are generally modeled using a sphere mapped onto a rectangle using a cylindrical projection. However, this projection induces sampling issues: areas around the poles get far more samples than those around the equator.

Instead, we map the sphere onto the faces of a bounding cube. We use the gnomonic projection to project points from the surface of the sphere onto the corresponding tangent face (Figure 7). The main reason behind this choice is a more uniform sampling than simple cylindrical projection as well as fast 3D reconstruction. After this projection we build a tree of blocks for each side of the cube, its root block covering the entire side.

Elevation values are relative not to the surface of the cube but to the reference surface of the planet defined by a given datum (for instance, the WGS84 standard is used for the Earth). Using such a datum allows to handle spherical coordinates but reconstruct a more accurate ellipsoidal model of the planet. To render the planet correctly we first project the samples back into spherical coordinates. Then, for each sample of a block, we use the gnomonic projection to get the normalized direction of the corresponding point on the surface from the center of the planet. We then apply the datum formula and add the de-quantified elevation value of the sample to get its 3D coordinates in a planet-centric coordinate system for rendering.

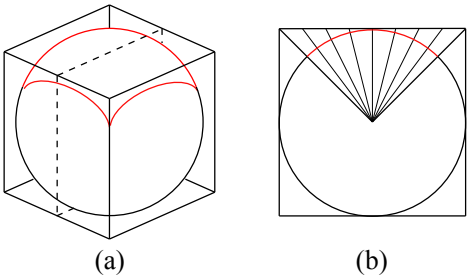


Figure 7: Mapping of a spherical planet onto a cube with the gnomonic projection. **a)** The top part of the sphere (in red) projects onto the top side of the cube. **b)** 2D cut of the cube along the dotted plane in (a). The gnomonic projection uniformly samples the cube side by mapping points from the surface of the sphere onto the plane of projection based on their direction from the sphere center.

6.2 Adjusted Gnomonic Sampling

Projecting the planet onto a cube using the gnomonic projection offers a more uniform sampling than simple cylindrical projection. However, like any map projection there are still sampling inconsistencies: the solid angle covered by a sample is approximately 75% smaller around the corners of the face than around the center of projection. To overcome this problem, we introduce a simple adjustment to the gnomonic projection for improved sampling quality.

Instead of sampling the plane of projection, we propose to sample the map directly in spherical coordinates with steps expressed in terms of angles (Figure 8). Using a planet-centric cartesian coordinate system whose axes are aligned with the edges of the cube, we define two leading axes for each face of the cube, with the third one passing through the center of projection of this face. We can then perform independent 2D rotations around both leading axes in $[\frac{-\pi}{4}, \frac{\pi}{4}]$ using a single angle step to obtain a 2D sampling of the surface of the planet that can be projected onto the face of the cube. The samples are then more evenly distributed on the surface of the planet: the solid angle covered by a sample is only 33% smaller around the corners of the face than around the center of projection, yielding a gain both in terms of compactness and quality.

When projected using the gnomonic projection, our samples do not map evenly onto the projection plane. We note that, in regard to a given leading axis, the 1D coordinate of the projected sample is the tangent value of the angle formed by the sample, the center of the planet and the center of projection (see Figure 8(b)). Therefore our solution allows us to store sample values within a simple uniform 2D map. The coordinates of any sample on the gnomonic plane in $[-1,1]$ can then be obtained by computing the tangent of the two angles. Another advantage of this method is its straightforward integration within our crack removal solution.

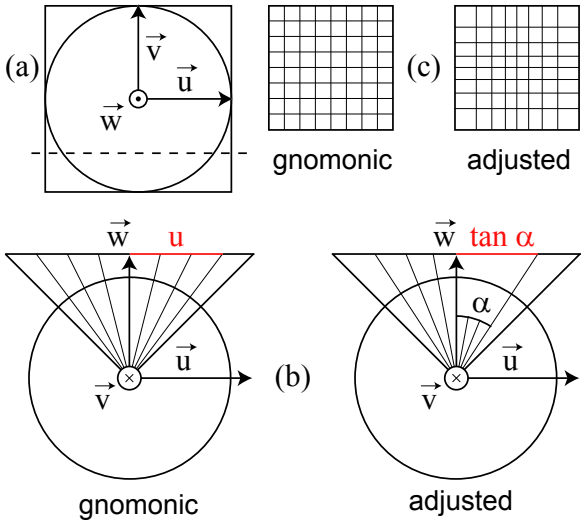


Figure 8: Adjusted gnomonic sampling. **a)** Top view of the cube. We use vectors $\vec{u}$ and $\vec{v}$ to parameterize the top face. **b)** 2D side cuts of the cube along the dotted line from (a). Gnomonic projection: the line of projection is uniformly sampled by translating along $\vec{u}$. Adjusted projection: the surface of the circle is uniformly sampled by rotating around $\vec{v}$. **c)** Top views of the face with projected samples. Gnomonic: the face is a regular 2D map. Adjusted: we use the tangent of the angles to get the coordinates of a sample in the plane of projection.

6.3 Geometry cracks between maps

Our adjusted gnomonic sampling scheme allows for high quality terrain sampling of each side of the cube. However, as the sides of the cube correspond to different maps, classical crack artifacts may appear when using heterogeneous resolutions across edges. This section explains how our projection can benefit from our crack removal solution (Section 5).

The core of the method is based on numbering and orienting the faces of the cube in a way that facilitates neighborhood management: that is, we want to ensure that blocks are ordered the same way on both sides of a given edge. This would allow us to use the same system as for neighbors that are part of the same map. However, neighbor maps cannot have the same $\vec{u}$ or $\vec{v}$ axis on their common edge in all cases because they are the sides of a cube. Figure 9 presents how we organize the sides of the cube to match the cube edge vectors, and Table 1 gives the neighborhood correspondences between sides.

6.4 Rendering and culling improvement

Graphics hardware use *near* and *far* clipping planes to bound the depth of rendered geometry, and we also take these planes into account when performing view frustum culling. In the case of a planet, which is a very large body, we want the *far* plane to be far enough to ensure

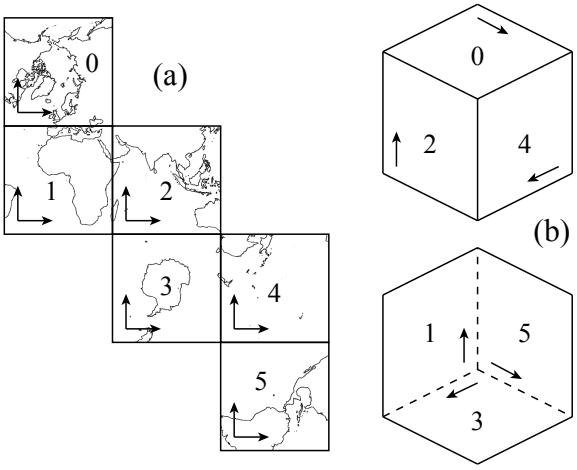


Figure 9: Orientation and numbering of the faces of the cube. **a)** Pattern of the cube with continents of the Earth for reference. Horizontal axes: $\vec{u}$, vertical axes: $\vec{v}$. **b)** Side view of the cube (top: visible side, bottom: hidden side). $\vec{v}$ axes are placed near the "left" edge of the face ($u = 0$).

Face #	Edge	Neighbor #	Neighbor edge
Even	Left	+5	Right
	Top	+4	
	Right	+2	Top
	Bottom	+1	
Odd	Left	+4	Bottom
	Top	+5	
	Right	+1	Left
	Bottom	+2	

Table 1: Neighborhood correspondences between the faces of the cube. Left, top, right and bottom respectively refer to edges where $u = 0$, $v = 1$, $u = 1$ and $v = 0$. Neighbor face number is obtained by adding the given value to the current face number, modulo six.

that no visible part of the planet is abusively ignored. However, using an arbitrary large *near-far* difference make poor use of the depth buffer rendering precision and causes flickering problems. Moreover, when placing the *far* plane behind the planet, all the surface that is visually culled by the planet itself (the grayed area in Figure 10) is uselessly rendered. We avoid those issues by adapting both planes to the position of the viewpoint. Clipping planes are orthogonal to the viewing direction and are defined by their distance to the viewpoint. Since we manipulate only distances and the planet can be approximated using a sphere, we may work in the plane defined by the center of the planet, the viewpoint position and the viewpoint direction vector.

The *far* distance corresponds to the horizon and is computed as Figure 10 explains. Equations 1 present how we compute the desired distance $\|VK\|$. Using this method, the rendered surface gets smaller as the viewpoint gets closer to the planet: the adaptive solution can

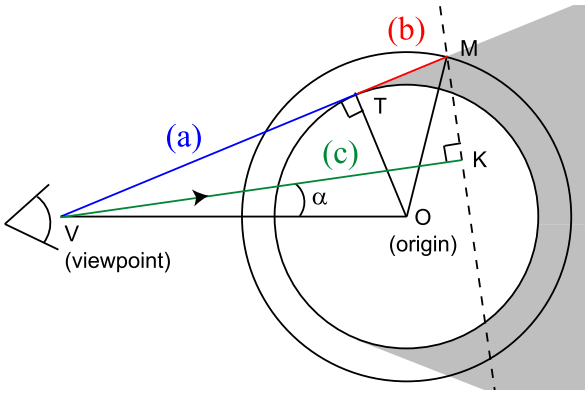


Figure 10: Horizon culling: we compute the distance $\|VK\|$ to the far plane. KM is the 2D projection of this plane, anything behind it is not rendered. The grayed area is visually occluded by the planet itself. **a)** We first compute $\|VT\|$, based on the minimum planet radius $\|OT\|$ and the distance $\|VO\|$ between the viewpoint V and the center of the planet O. **b)** We then add the constant $\|TM\|$ based on the minimum and maximum planet radii $\|OT\|$ and $\|OM\|$. **c)** We then project $\overrightarrow{VM}$ onto the viewing direction to obtain $\|VK\|$.

then use more data to improve the quality of visible and important areas.

$$\begin{aligned} \|VK\| &= \|VM\| \times \cos\widehat{MVK} \qquad (1) \\ &= \|VM\| \times \cos(\widehat{TVO} - \alpha) \\ &= (\|VT\| + \|TM\|) \times \\ &\quad \left(\frac{\|VT\|}{\|VO\|}\cos\alpha + \frac{\|OT\|}{\|VO\|}\sin\alpha\right) \qquad (2) \end{aligned}$$

We compute the *near* distance in a much simpler manner: the minimum distance to the planet is $\|VO\| - \|OM\|$. We tune this value using the field of view to avoid culling parts of the planet on the screen corners.

7 RESULTS

We tested the proposed methods on large global datasets available online: CGIAR-CSI SRTM for elevation with 90m definition, and Unearthed Outdoors TrueMarble^{TM}for color with 250m definition. Those maps use the *plate carrée* projection: we reprojected the data before creating our terrain files (Table 2).

The re-projected datasets are 25% smaller than the source due to the more uniform sampling that filters out most of the redundant data. In addition, we extended the server file format to support variable size LODs and added simple and fast LOD compression using Zlib to save disk and network bandwidth, leaving decompression on the client side.

Once we built the files on the server, we connected a client equipped with a 3.2GHz Core 2 Duo and a GeForce 9800GTX for a real-time 3D interactive Earth walkthrough using a screen resolution of 1680×1050

Input dataset		Projection		File creation	
Name	Size	Time	Size	Time	Size
SRTM	174G	9h	127G	4h50	15G
TrueMarble	42G	1h40	31G	40m	7G

Table 2: Preprocessing results. Re-projection uses about the same definition as the source maps. Without texture filtering, we obtained a 6GB file in 30 minutes for TrueMarble. The SRTM dataset is actually 58GB large but provides incomplete data which we filled with zeroes, yielding a global virtual 174GB input. Computed files contain a 11-level quad-tree of 53×53 blocks for each cube side for SRTM, and a 8-level quad-tree of 168×168 blocks per face for TrueMarble. We use two LODs per block to get uniform block update times in performance tests.

pixels and a network bandwidth of 1Mbps as shown on first page. With a 2 million polygons budget per frame, the terrain renders at 39.3 frames per second with fixed cracks. This is a small 4% decrease from rendering with cracks due to the additional triangle strips. About 5 to 10% of the rendering time is consumed by the generic solution, the rest being used for rendering.

We also tested the speed of the block update operations that occur when a new LOD is received. With texture filtering, creating a 168×168 LOD takes 0.89ms: this is only 6% more than without filtering. Using our planet projection adjustment, reconstruction of the new 3D vertices of a 53×53 LOD from elevation samples takes 0.57ms, compared to 0.33ms with plain gnomonic projection and 0.12ms when simply elevating samples from a plane. This is an important increase but those times are still negligible compared to network latency, even on lower performance clients. In addition, update operations do not directly interfere with rendering smoothness because they run in a separate thread.

8 CONCLUSION

This paper provides a full-featured solution for real-time rendering of arbitrarily large terrain datasets within a client-server context over the internet. We proposed solutions to critical issues of high quality terrain rendering: adaptive texture mapping, removal of geometry cracks and support of planetary terrains.

Our adaptive photometry scheme introduces a dual multi-resolution data structure for high definition texture representation. Our solution yields high performance storage, streaming and rendering.

Artifacts known as "cracks" tend to appear in many multi-resolution terrain rendering methods. We introduce *edge strip masks*, an inexpensive and robust method for cracks removal based on data masks.

As our technique is able to manage fully-detailed, planet-sized terrains, we propose an adjusted gnomonic sampling scheme for storing and rendering planetary terrains accounting for the actual planet shape. We also propose specific improvements for rendering virtual planets on graphics hardware.

REFERENCES

[CGG+03] Paolo Cignoni, Fabio Ganovelli, Enrico Gobbetti, Fabio Marton, Federico Ponchio, and Roberto Scopigno. Planet-sized batched dynamic adaptive meshes (P-BDAM). In *Proceedings of IEEE VIS*, pages 147–155, 2003.

[CH06] Malte Clasen and Hans-Christian Hege. Terrain rendering using spherical clipmaps. In *Proceedings of EuroVis*, pages 91–98, 2006.

[dB00] Willem H. de Boer. Fast terrain rendering using geometrical mipmapping. In *Unpublished and only available at http://www.flipcode.com/articles/article_geomipmaps.pdf*, 2000.

[Hwa05] Lok M. Hwa. Real-time optimal adaptation for planetary geometry and texture: 4-8 tile hierarchies. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):355–368, 2005.

[Lev02] Joshua Levenberg. Fast view-dependent level-of-detail rendering using cached geometry. In *Proceedings of VIS*, pages 259–266, 2002.

[LH04] Frank Losasso and Hugues Hoppe. Geometry clipmaps: terrain rendering using nested regular grids. *ACM Transactions on Graphics*, 23(3):769–776, 2004.

[LKES07] Yotam Livny, Zvi Kogan, and Jihad El-Sana. Seamless patches for GPU-based terrain rendering. In *Proceeding of WSCG*, pages 201–208, 2007.

[LMG09] Raphael Lerbour, Jean-Eudes Marvie, and pascal Gautron. Adaptive streaming and rendering of large terrains: A generic solution. In *Proceedings of WSCG*, 2009.

[MB03] Jean Eudes Marvie and Kadi Bouatouch. Remote rendering of massively textured 3D scenes through progressive texture maps. In *Proceedings of IASTED*, pages 756–761, 2003.

[PG07] Renato Pajarola and Enrico Gobbetti. Survey on semi-regular multiresolution models for interactive terrain rendering. *The Visual Computer*, 2007.

[PM05] Joachim Pouderoux and Jean-Eudes Marvie. Adaptive streaming and rendering of large terrains using strip masks. In *Proceedings of GRAPHITE*, pages 299–306, 2005.

[TMJ98] Christopher C. Tanner, Christopher J. Migdal, and Michael T. Jones. The clipmap: a virtual mipmap. In *Proceedings of SIGGRAPH*, pages 151–158, 1998.

[WMD+04] Roland Wahl, Manuel Massing, Patrick Degener, Michael Guthe, and Reinhard Klein. Scalable compression and rendering of textured terrain data. In *Proceedings of WSCG*, pages 521–528, 2004.

Simultaneous Incremental Reconstruction of Object Geometry and Appearance for Interactive 3-D Model Acquisition

Hannes Dohrn Hannes Stadler Marco Winter Günther Greiner

University of Erlangen-Nuremberg
Chair of Computer Science 9 (Computer Graphics)
Am Wolfsmantel 33
91058 Erlangen, Germany

{sihadohr, sihastad}@i9.informatik.uni-erlangen.de,
{marco.winter, guenther.greiner}@informatik.uni-erlangen.de

ABSTRACT

While the creation of three-dimensional models from real-life objects is a commonly applied process, the reconstruction of complete datasets from such objects is still a delicate task. In this paper, a simple yet powerful framework is proposed that is able to reconstruct both geometry and appearance of a given object interactively. Working in an incremental mode of operation, it enables the user to reconstruct a given scene with full visual feedback during the progress of the reconstruction process. For geometry reconstruction, an existing surface reconstruction algorithm has been investigated and adjusted to the needs of the framework. Furthermore, a hardware-accelerated surface light field algorithm has been integrated into the framework that performs appearance reconstruction of the object.

The target application of our framework is the reconstruction of real-life objects using mobile acquisition devices. We demonstrate the performance and usefulness of our framework by reconstructing models from previously acquired datasets of real-life objects. Furthermore, we provide results of experiments run in our own model acquisition setup.

Keywords

Image-Based, Acquisition, Light fields, Modeling, Reconstruction, Interaction

1. INTRODUCTION

Today, the creation of three-dimensional models from real-life objects is a commonly applied process. Typical applications include environment modeling, rapid prototyping, virtual museums or the digitalization of scene objects in movie industry. However, the reconstruction of a complete dataset from a real-life object that is ready for rendering or further processing still is a delicate task.

Various methods have been developed to support this process. Structure-from-motion or stereo algorithms are used to reconstruct geometric information using color images of a given scene. Furthermore, hardware devices have been developed that allow for the direct acquisition of partial geometry information of objects, and software algorithms are successively applied to reconstruct the complete surface from this data. While matured, these approaches generally produce imperfect results in the case of insufficient data. Therefore, it is preferable to reconstruct 3-D models incrementally, i. e. to consecutively add new input data to an existing partial reconstruction for further refinement. By visualizing the current results the user has the ability to actively control the reconstruction process. However, this requires the applied reconstruction and visualization methods to work at interactive frame rates.

Today, first hand-held multimodal input devices are available that facilitate the simultaneous real-time acquisition of both color and range images of a scene [BH04, LHWL07]. The resulting huge amount of object information provided by these devices requires processing frameworks that are able to handle this data efficiently for interactive reconstruction. Furthermore, the final reconstruction of the captured object should be of high visual and geometric quality.

In this paper, a framework is presented that addresses the aforementioned issues and allows for simultaneous incremental reconstruction of both geometric and visual properties of a given object. As input it expects color and range images of an object as well as corresponding calibration data. For reconstructing the geometry part of the object, an existing surface reconstruction method has been investigated and modified to be applicable for incremental reconstruction. For optimal reconstruction of the model's visual appearance, a GPU-accelerated online implementation

of surface light fields is utilized that is parametrized on the currently reconstructed geometry. Both parts of the framework have been optimized for online reconstruction, thus allowing the interactive visualization of the current status of the reconstructed object.

This paper is organized as follows: previous work related to our paper is summarized in Section 2. Afterwards, Section 3 gives a brief overview of our proposed reconstruction framework. In Section 4, the used surface reconstruction algorithm and its modifications are described. Section 5 then describes the applied surface light field implementation. The results and conclusion of our work are shown in Sections 6 and 7, respectively.

2. RELATED WORK

The acquisition and reconstruction of 3-D models from real-life objects is a heavily investigated research area. For example, different approaches are known that reconstruct surfaces from unordered point clouds or range images. These may be distinguished as *parametric surface methods* [TL94, CL96] that make use of the known topology of the input data, *implicit methods* [HDD+92, OBA+03] which apply implicit distance functions that represent the surfaces, or *Delaunay-based methods* [EM94, DG03]. However, most of these approaches handle surface reconstruction as an offline process only. In the area of incremental surface reconstruction, only few publications are known. Curless and Levoy [CL96] incrementally build a distance volume from range images representing the final surface, however, the successive surface extraction step still is done offline. Turk and Levoy [TL94] directly reconstruct the surface using the provided triangulated range images, which allows for online visualization of the currently reconstructed geometry. Two different frameworks are proposed in [RHHL02] and [BH04] that facilitate the online reconstruction of real-life objects at interactive rates. However, texturing of the reconstructed object is only done in a separate offline step. None of the described approaches supports the incremental reconstruction of visual information of the object.

For reconstructing the visual appearance of real-life objects, *light field* and *lumigraph* models are the preferred choice: by utilizing the provided image information, they are able to synthesize new views of a given object, while also preserving highly view-dependent visual effects (e. g. mirroring, transparency or sub-surface scattering). Different approaches exist [LH96, GGSC96, SVSG01, BBM+01, CBCG02] that differ in the actual light field parametrization used, and whether geometric information of the object is applied to support the synthesizing process. In the latter case, reconstruction results usually are improved significantly, provided that the geometry is of sufficient quality. While some of these approaches are suitable for incremental reconstruction [BBM+01, CHLG05], this only restricts to the visual part. The underlying geometry still has to be provided by using other methods.

3. THE RECONSTRUCTION FRAMEWORK

Our proposed framework aims to provide a complete pipeline for simultaneous incremental reconstruction of geometry and appearance of a given object. Therefore, it makes use of established algorithms which have been extended and adjusted to our needs.

As a basis for the surface reconstruction part, the ridge-based approach by Süßmuth et al. [SG07] has been chosen because of its initial properties. The method aims at providing a very robust reconstruction algorithm that delivers high quality results even in the presence of strong noise, which is a vital property for reconstructing range images coming from scanning devices. Additional post-processing steps have been implemented that complement the main algorithm, like mesh decimation and topology cleaning.

For the appearance reconstruction part, the surface light field model (SLF) [CBCG02] has been chosen as the basic method. Among all light field approaches, it provides the most realistic visual results. Furthermore, efforts have been made to realize an online version for reconstructing surface light fields [CHLG05], which is a major advantage of this approach for our purposes.

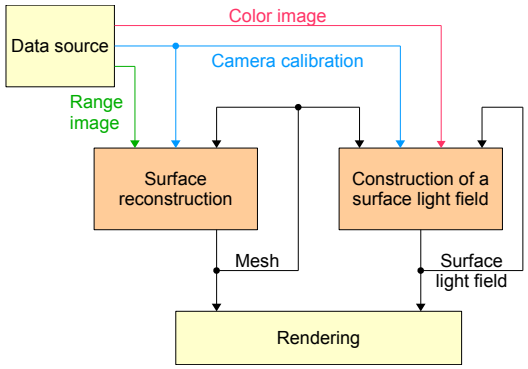


Figure 1: The complete pipeline of our reconstruction framework.

The whole reconstruction process is depicted in Figure 1. Here, a data source provides image information about the object, consisting of color and range images as well as camera calibration data. The latter provides information about the camera's extrinsic and intrinsic parameters. The current range image and calibration data is fed into the surface reconstruction part of our framework, where it is processed for extending the existing reconstructed geometry of the object. The extended geometry is reused in the successive surface

reconstruction steps, and is also provided to the light field reconstruction part. Here, together with the current color image and camera calibration data, the existing light field is extended using the new provided information. The progress of the object reconstruction process is visualized at any time using the rendering part of the framework.

4. INCREMENTAL SURFACE RECONSTRUCTION

In this section, the implemented surface reconstruction approach is described briefly, and its modifications for incremental mode of operation are explained. For more detailed information about the original reconstruction process, we would like to address the interested reader to the original paper.

4.1. Overview of the Ridge-based Approach

The ridge-based surface reconstruction approach by Süßmuth et al. assumes that a sample point p does not necessarily interpolate the surface but is rather the center of a Gaussian probability distribution $G_p(x)$ describing its location. By summing up all probability distributions defined by the points from a given point cloud, one receives a global probability distribution $f(x)$ that describes the potential location of the surface.

The algorithm can be subdivided into two main steps. In the first step, the Gaussian kernels centered at the sampled points are summed up and analyzed at the nodes of a regular, discrete grid. The analysis encompasses the computation of the density function's gradient $\nabla f(x)$ and the direction of maximum curvature $K_{max}(f)(x)$, where the latter is calculated by eigenanalysis of the Hessian matrix $H(f)(x)$. In the second step, the scalar product $\langle \nabla f(x), K_{max}(f)(x) \rangle$ is used in a modified marching-cubes [LC87] algorithm in order to identify ridges which are then interpreted as the surface.

Figure 2 illustrates the different stages of the algorithm for the 2-D case. As can be seen in Figure 2d, many "spurious" ridges were extracted beside the actual sine curve. In order to avoid extraction of those ridges, Süßmuth et al. assume that the node with highest density always lies on the main ridge corresponding to the actual surface. Hence they only accept the surface which evolves from this node.

Finally, to lower computational effort, the Gaussion kernel is cut off in regions where its contribution to the density function, gradient and curvature is becoming insignificant. If the value of the Gaussian kernel drops below an user-specified threshold at a grid node, its calculation and analysis is omitted at the respective node. Thus, each grid node receives contributions only from those points which lie within the cut-off radius around that node.

4.2. The Incremental Ridge-based Approach

The original approach was designed to work on a complete point cloud, however, the accumulation of the density function as well as its analysis can be done easily in an incremental way. The Gaussian kernel $G_p(x)$ for point p and its gradient $\nabla G_p(x)$ are evaluated at each grid node n, if the respective point lies within the node's cut-off radius. The density function $f(x)$ and its gradient $\nabla f(x)$ are then evaluated at each node n, by summing up the contributions of all points P_n within the respective node's cut-off radius. For a node n, the evaluation of $f(x)$ would be done like this: $f_n = \sum_{p \in P_n} G_p(\|p-n\|)$. The gradient ∇f_n and the Hessian matrix $H(f_n)$ are evaluated accordingly.

This way, new point clouds can be integrated into the existing sampling of function $f(x)$ and its analysis by simply resuming the summation process. Once all new points have contributed to the density function $f(x)$, its gradient and its Hessian matrix, the eigenanalysis of $H(f)(x)$ is performed and the scalar product $\langle \nabla f(x), K_{max}(f)(x) \rangle$ is evaluated at every updated grid node n.

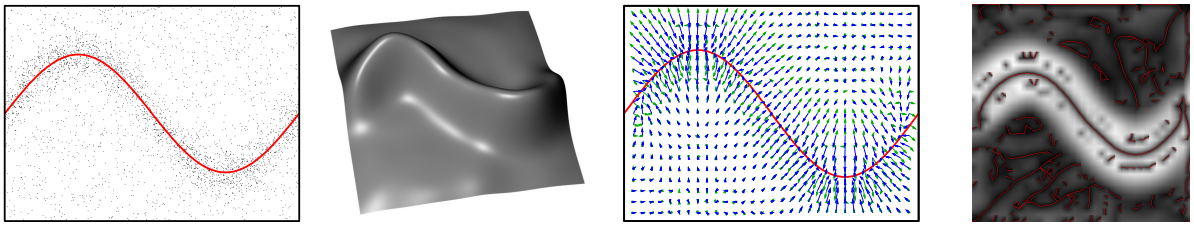
After the discrete grid has been updated, new surface parts are extracted using a modified marching-cubes algorithm. Here, tracing the surface from the node of highest density is not applicable any more: Using an incremental and interactive approach, we need to be able to reconstruct the surface in different parts of the captured object simultaneously, as multiple unconnected parts may occur which all belong to the actual surface.

Therefore, the extraction algorithm considers multiple nodes as starting points for tracing the surface and does so only for nodes where the density has reached a certain user-specified threshold. To further prevent spurious ridges from being extracted, only surface parts with a user-specified minimum number of reconstructed triangles will be accepted for addition to the object's mesh.

Once a new surface part is added to the mesh, all grid cells that contributed to this part are locked, thus preventing the marching-cubes algorithm from considering these cells in upcoming iterations of the reconstruction process. Furthermore, all nodes adjacent to locked cells are also locked, meaning that these nodes' density, gradient and maximum curvature will not be updated any more. This way we make sure that updates will not displace the zero-crossing, which otherwise leads to discontinuities when neighboring cells also extract their part of the surface. Figure 3 illustrates this process.

4.3. Mesh Improvement Techniques

After new mesh parts have been added to the reconstruction in each iteration, we apply different mesh



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

Figure 2: (a) A noisy sampling of a sine curve. (b) The density function $f(x)$. (c) The gradient (blue) and maximum curvature direction (green) of the density function. (d) The absolute scalar product as gray value image and the curves (both wanted and spurious) extracted by the marching cubes algorithm.

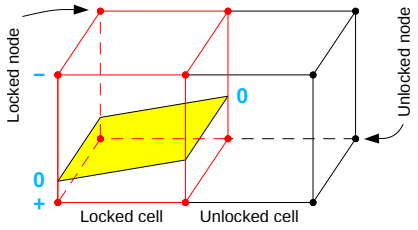


Figure 3: Example for surface extraction inside the discrete grid. Once a surface (yellow) has been extracted within one of the cells, the cell itself is locked as well as all adjacent nodes (red).

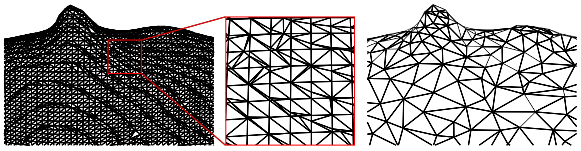


Figure 4: Left: A mesh produced by the marching-cubes algorithm. Middle: An enlarged section from the mesh. Right: The same mesh after improvement.

improvement techniques in order to prepare the mesh for use with surface light field construction. This is necessary for the ridge-based approach, as the used marching-cubes algorithm generates meshes that usually consist of very small triangles, where many of them are showing poor aspect ratios. Furthermore, it is advisable to reduce the number of reconstructed triangles in order to improve reconstruction performance in general. To handle the mentioned issues, the following tasks are performed:

- Small holes are filled using a simplified version of Liepa's method [Lie03], which only triangulates holes, omitting the steps of simplification and fairing.
- Mesh simplification is performed by edge contraction using quadric error metrics as presented by Garland and Heckbert [GH97]. Non-edge contractions are not performed, thus keeping separated parts of the mesh unconnected.
- A simplified version of edge relaxation via mesh edge flipping as proposed by Dyer et al. [DZM07] is applied. The steps of remeshing and mesh decimation are omitted.

Figure 4 shows an example mesh before and after application of the described improvement methods. In order to avoid discontinuities, these steps are not performed at the boundary of the current surface reconstruction. Therefore, the boundary keeps its typical cube-like shape caused by the surface extraction algorithm until new mesh information has been reconstructed. Figure 5 shows the typical topology of a reconstructed mesh at its interior and its boundary, respectively.

5. ONLINE CONSTRUCTION OF SURFACE LIGHT FIELDS

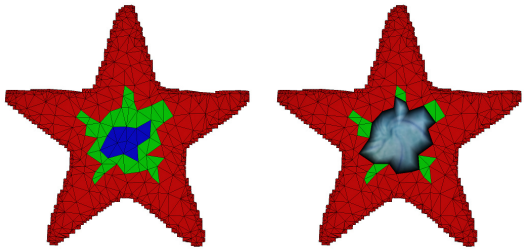
A surface light field approximates the 4-D function $f(r,s,\theta,\phi)$ which describes the outgoing radiance for every point (r,s) on an object's surface into every viewing direction (θ,ϕ). For efficient storage and hardware rendering, Chen et al. [CBCG02] proposed a decomposition of the 4-D light field function into a sum of products of lower-dimensional functions:

$$f(r,s,\theta,\phi) \approx \sum_{k=1}^{K} g_k(r,s)h_k(\theta,\phi)$$

Significant data compression is achieved by limiting the sum to K terms, where K usually lies between 3 and 6 when using an offline principal component analysis (PCA) as decomposition algorithm. Both functions g_k and h_k, which are exclusively parametrized on the surface or the viewing direction respectively, can be stored on graphics hardware as texture maps. Using the vertex-centered light field partitioning scheme introduced by Chen et al., new views of the object can then be rendered at interactive frame rates using programmable graphics hardware.
However, the PCA algorithm used to compress and decompose the 4-D light field function is an expensive offline method that is not suited for online construction of a light field dataset, as it requires a complete recomputation of the decomposition when adding new image

information. To this end, Coombe et al. [CHLG05] replaced the offline PCA with an incremental singular value decomposition (Online-SVD), thus making the online construction of surface light fields feasible.

For our framework, we implemented an optimized version of the online construction algorithm from [CHLG05]. In our implementation, the tasks of image resampling and visibility testing were redesigned as GPU algorithms to further speed up the online construction process. Furthermore, the visibility test of the original approach has been modified. Since a complete geometry mesh of the surface is not available for the visibility tests, the current range image is used instead as a representation for the object's geometry.



(a) (b)

Figure 5: A partial reconstruction of the front of the *star* dataset from the OpenLF toolset [Too09]. The colors indicate the different areas. The right image shows the parts of the mesh for which a light field was already constructed.

The two tasks of surface reconstruction and construction of the surface light field can be executed in an alternating way, such that new light field information may immediately be added to recently reconstructed mesh parts (Figure 1). However, the interaction of both tasks is complicated by the fact that mesh reconstruction and mesh improvement must never change those parts of the surface for which light field information already has been constructed. Therefore, the object's mesh is organized in three areas which are depicted in Figure 5a. Modifications of the mesh, including deletion of faces or vertices, are only possible at the red "border" faces which are near to the mesh boundary where new faces will connect in future iterations. The middle area of green faces consists of so-called "fixed" faces. Neither their vertices nor their topology may change anymore, but since they are adjacent to border faces their vertices' normals may be subject to change. Finally, the blue faces at the center are the "normal-fixed" faces whose vertex normals are fixed as well. In our implementation, light field information is only stored at the triangle fans that contain at least one blue face, as shown in Figure 5b.

6. RESULTS

The proposed framework has been extensively tested against the freely available datasets of the OpenLF toolset [Too09], which contain images, geometry and calibration data from acquired real-life objects. The available information is highly accurate, and therefore well suited as ground-truth data to test our framework's performance and accuracy, without introducing additional errors from various pre-processing steps.

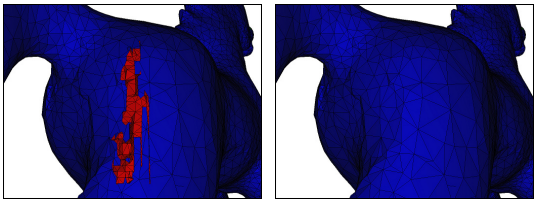
However, the images provided by the OpenLF datasets are photographs taken at discrete camera positions, and the provided geometry is given as a complete mesh instead of several range images. To simulate data coming from a moving hand-held acquisition device that is suited for our framework, a modified light field viewer has been implemented that constructs surface light fields from OpenLF datasets that are freely accessible by the user. Color and range images are generated by the viewer by sampling the currently rendered image and depth information, and are provided via network to our framework for further processing.

Using this setup, the simultaneous reconstruction of geometry and appearance of the OpenLF datasets has been performed. An example for the visual feedback of the reconstruction progress can be seen in Figure 8. A visual comparison between the "original" model and the one reconstructed by our framework is shown in Figure 6. Here, it can be seen that our framework is able to successfully reconstruct the provided models with high geometric and visual accuracy. However, the SLF reconstruction tends to "smooth" visual highlights, which is also a typical result of the original online-SLF approach.

(a) (b)

Figure 6: Reconstruction of the *star* dataset: the original model as rendered by our SLF viewer is shown in (a), the model reconstructed by our framework is shown in (b).

Due to the nature of the applied surface reconstruction algorithm, different artifacts may occur during reconstruction. One example is the erroneous extraction of spurious ridges, as shown in Figure 7a. These artifacts cannot be avoided in all cases, however, if identified, they can be successfully discarded by removing all but the largest connected submesh (Figure 7b). Also, the reconstruction algorithm may not be able to close all mesh holes. These reconstruction artifacts are mainly caused by noisy input data and imperfect choice of user-specified reconstruction parameters. As the optimal parameters for a given object cannot be determined analytically, most of them have to be found by trial-and-error.



(a) (b)

Figure 7: (a) Extraction of spurious ridges at the *horse* dataset. (b) The same mesh after final cleanup.

Tables 1 and 2 show average timings for the integration of new data into an existing reconstruction. The values for "slf update" show the timings of the complete surface light field update process, while the remaining values show timings of different parts of the surface reconstruction and the different improvement algorithms. As can be seen in these tables, the reconstruction speed highly depends on the resolution of the depth images. When using higher resolutions, the most time-consuming part of the reconstruction is the update of the density function, which is also due to the resolution of the function's grid. Using lower resolutions, the grid's density may be decreased accordingly, and so the impact of this step vanishes. It can also be noticed that, due to our GPU optimizations, the surface light field update process only takes a fraction of the whole update time. Eventually, it can be seen that the incremental reconstruction process can be performed at interactive frame rates when processing the mentioned datasets.

Using our framework, we also started first experiments with the acquisition of own datasets. As input device, we utilized the 2-D/3-D time-of-flight camera described in [LHWL07], which has been calibrated using the OpenCV library [Lib09]. The provided input data was pre-processed in a simple way in order to reduce noise of the depth information before it was delivered to our framework. Using this setup, we were able to reconstruct information of a silicon liver model, which is shown in Figure 9. Despite the pre-processing, the incoming depth information still exhibited a significant

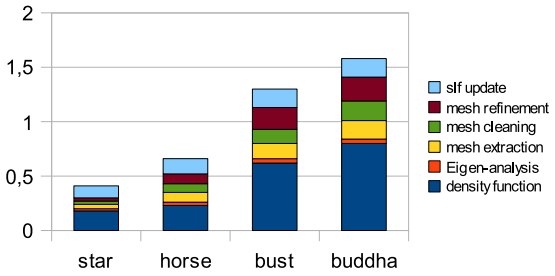


Table 1: Performance of the reconstruction framework for depth map size 176×144 (timings in s)

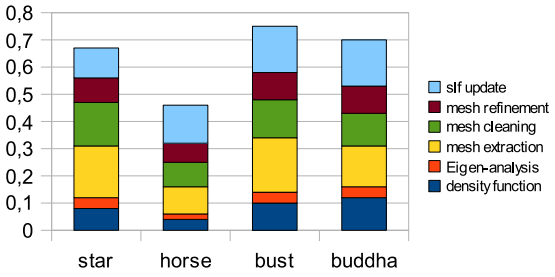


Table 2: Performance of the reconstruction framework for depth map size 64×48 (timings in s)

amount of noise. Nevertheless, our modified reconstruction method successfully recreated all parts of the model that were seen by the camera – it can be recognized that holes or gaps in the reconstruction mainly appear in those areas where no information was provided to the framework. Note that the visual reconstruction of the model looks less complete because of our introduced update policy for the surface light field.

7. CONCLUSION AND FUTURE WORK

We have presented a novel framework for the incremental reconstruction of digital models from real-life objects. The framework features an enhanced incremental approach for interactive surface reconstruction, as well as an online surface light field technique for high-quality reconstruction of both geometry and appearance information.

The surface reconstruction approach has shown good results for data with a moderate level of noise. However, as the noise intensity increases, there is also a gain in the probability for reconstruction artifacts like spurious ridges. Therefore, efficient pre-processing algorithms that smooth data and remove outliers will be necessary to significantly improve the overall quality of reconstructions from "real-life" acquisition devices.

Due to our update policy of the surface light field, the surface reconstruction part of our framework finishes long before the construction of the light field reaches sufficient quality. While the progress of the surface reconstruction is obvious to the user, the quality of the light field is hard to determine and calls out for special

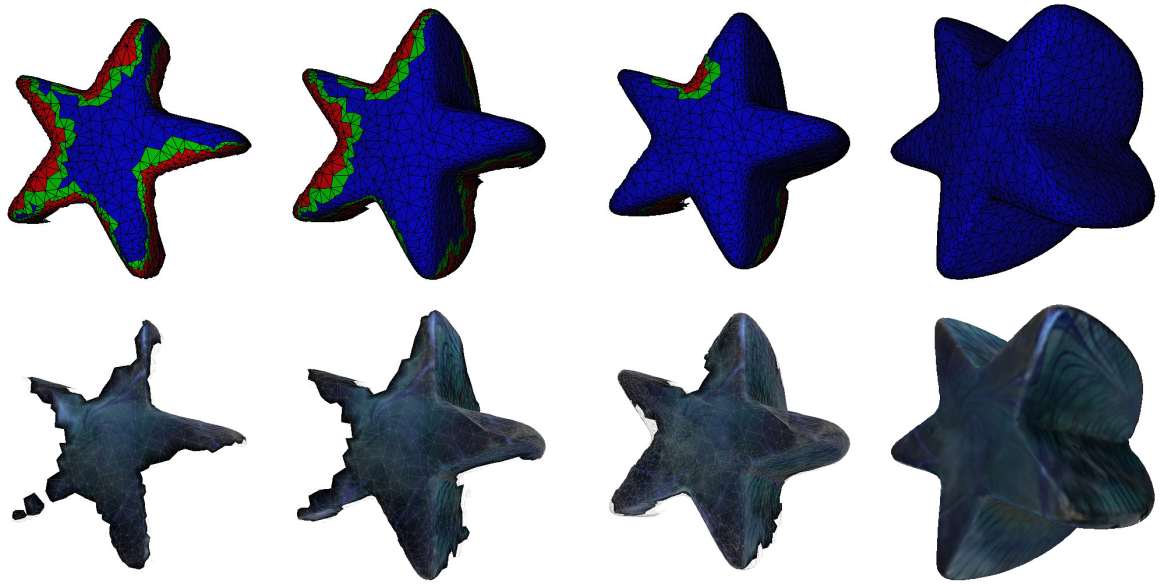
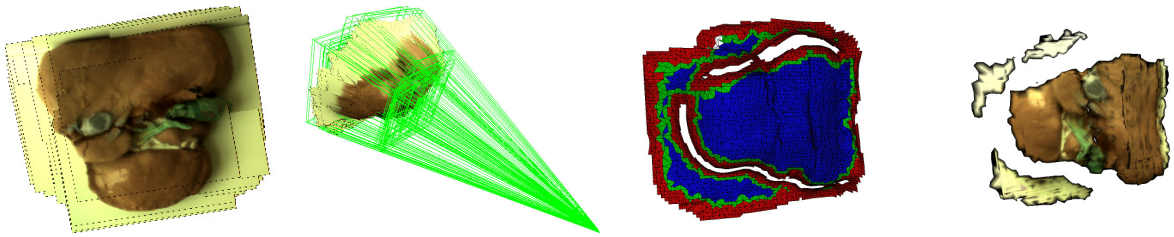


Figure 8: Different iterations of a reconstruction of the *star* dataset from the OpenLF library.



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

Figure 9: Partial reconstruction of a liver model. Figures (a) and (b) show the source images and calibration parameters, while the reconstructed geometry and appearance is shown in figures (c) and (d), respectively.

visualization. Coombe et al. propose a quality heuristic which directs the user to areas where more data is needed. However, this quality measure does not indicate the viewing direction for which the approximation still lacks information. Therefore, we are interested in investigating different approaches for feedback visualization to further improve acquisition time and final model quality.

Our first experiments with the acquisition and processing of "real" data using a 2-D/3-D time-of-flight camera have been quite promising. For improved reconstruction results, we plan to integrate better pose information of the camera in our acquisition process.

ACKNOWLEDGEMENTS

We would like to thank Jochen Süßmuth for making his implementation available to us.

REFERENCES

[BBM+01] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 425–432, New York, NY, USA, 2001. ACM.

[BH04] T. Bodenmueller and G. Hirzinger. Online surface reconstruction from unorganized 3d-points for the dlr hand-guided scanner system. In *Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium (3DPVT)*, pages 285–292, Washington, DC, USA, 2004. IEEE Computer Society.

[CBCG02] Wei-Chao Chen, Jean-Yves Bouguet, Michael H. Chu, and Radek Grzeszczuk. Light field mapping: Efficient representation and hardware rendering of surface light fields. In *ACM Transactions on Graphics*, pages 447–456, 2002.

[CHLG05] Greg Coombe, Chad Hantak, Anselmo Lastra, and Radek Grzeszczuk. Online construction of surface light fields. In

Proc. Eurographics Symposium on Rendering, 2005.

[CL96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 303–312, New York, NY, USA, 1996. ACM.

[DG03] Tamal K. Dey and Samrat Goswami. Tight cocone: a water-tight surface reconstructor. In *Proceedings of the eighth ACM symposium on Solid modeling and applications (SMA)*, pages 127–134, New York, NY, USA, 2003. ACM.

[DZM07] Ramsay Dyer, Hao Zhang, and Torsten Möller. Delaunay mesh construction. In *Proceedings of the fifth Eurographics symposium on Geometry processing (SGP)*, pages 273–282, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[EM94] Herbert Edelsbrunner and Ernst P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 43–54, New York, NY, USA, 1996. ACM.

[GH97] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 209–216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[HDD+92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 71–78, New York, NY, USA, 1992. ACM.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, 21(4):163–169, 1987.

[LH96] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 31–42, New York, NY, USA, 1996. ACM.

[LHWL07] O. Lottner, K. Hartmann, W. Weihs, and O Loffeld. Image registration and calibration aspects for a new 2d / 3d camera. In *EOS Conference on Frontiers in Electronic Imaging*, pages 80–81. European Optical Society, 2007.

[Lib09] Open Computer Vision Library. http://sourceforge.net/projects/opencvlibrary/, 2009.

[Lie03] Peter Liepa. Filling holes in meshes. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing (SGP)*, pages 200–205, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[OBA+03] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. *ACM Transactions on Graphics*, 22:463–470, 2003.

[RHHL02] Szymon Rusinkiewicz, Olaf Hall-Holt, and Marc Levoy. Real-time 3d model acquisition. *ACM Transactions on Graphics*, 21(3):438–446, 2002.

[SG07] Jochen Süßmuth and Günther Greiner. Ridge based curve and surface reconstruction. In *Proceedings of the fifth Eurographics symposium on Geometry processing (SGP)*, pages 243–251, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[SVSG01] Hartmut Schirmacher, Christian Vogelgsang, Hans-Peter Seidel, and Günther Greiner. Efficient free form light field rendering. In *Proceedings of the Vision, Modeling and Visualization Conference (VMV)*, pages 249–256. Aka GmbH, 2001.

[TL94] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 311–318, New York, NY, USA, 1994. ACM.

[Too09] OpenLF Light Field Mapping Toolset. http://sourceforge.net/projects/openlf/, 2009.

Efficient Non-Linear Editing of Large Point Clouds

Fabian Aiteanu Patrick Degener Reinhard Klein

Universität Bonn
Institut für Informatik II - Computergraphik
D-53117 Bonn, Germany
{aiteanu, degener, rk}@cs.uni-bonn.de

ABSTRACT

Editing 3D models is often performed on triangular meshes. We generalize editing operations based on differential coordinates to work on point clouds without explicit connectivity information. This allows a point cloud to be interpreted as a surface or volumetric body upon which physically plausible deformations can be applied. Our multiresolution approach allows for a real-time editing experience of large point clouds with 1M points without any offline processing. We tested our method on a range of synthetic and real world data sets acquired by laser scanner. All of them were interactively editable and produced intuitive deformation results within few minutes of editing.

Keywords: Point clouds, non-linear editing, real-time, differential coordinates, subsampling.

1 INTRODUCTION

Virtual 3D objects are a common asset for movies and games. In former years those were predominantly created manually, but in later years it has become more common to scan real objects and then edit and modify their virtual counterparts. For some application areas, e.g. structured light scanning, data is available only as point clouds, though editing 3D models is predominantly performed on triangular meshes. Despite ongoing efforts in the area of surface reconstruction, triangulating point sets remains computationally demanding and is error-prone in the presence of noise and outliers. Additionally, point clouds are usually sampled much denser than meshes and thus contain amounts of data, which are several times larger than for comparable meshes. This in turn can negatively influence the interactivity of editing operations.

To address this problem specialized editing approaches for point sampled models have been proposed. However, to handle the larger complexity of point clouds, these methods resort to relatively simple deformation models like transformation interpolation or linear models based on differential coordinates. As recently analysed by Sorkine and Botsch [SB09], a common problem of these simple deformation models is a counterintuitive deformation behaviour. For deformation of triangle meshes this problem is well known, and subsequently non-linear deformation models have been developed that do not show these problems. The higher intuitivity though comes at the cost of a significantly higher computational effort.

Space deformations are a different approach towards editing operations. They can conceptually be calculated quickly, but lack the ability to respond to the structure of the edited model. Points which are far apart in geodesic distance can be close in space and thus be influenced by editing operations unintentionally. To overcome this limitation, cage-based techniques try to separate such geodesically distant regions from each other, but the user is required to create or at least adjust the cage manually. In contrast, our solution does not require any manual intervention.

We present a novel method to enable direct real-time editing of point clouds without the need for prior

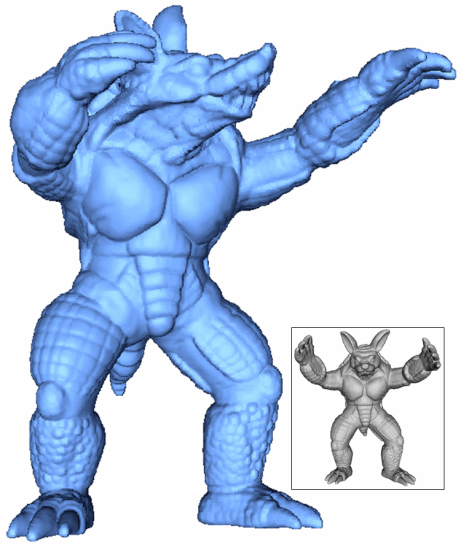


Figure 1: Armadillo model after 3 editing steps: head has been twisted by 45°, left arm twisted by 90°, right arm bent by 30°.

meshing. Our method shows intuitive, physically plausible deformation behaviour and does not require manual preprocessing like cage construction. It is based on geodesic distances only and avoids the problems of space deformation based approaches described above. Using our method, large point sampled geometry (Figure 1) can be edited as a whole at interactive frame rates.

In summary our contribution is this:

- We present an approach for editing point clouds while preserving local surface details. Due to an underlying non-linear deformation model, the obtained deformations are physically plausible, even for large handle transformations.
- Our algorithm does not pose any specific assumptions upon the structure of the point cloud being edited. The input requires solely the vertex positions in 3D space, but no explicit connectivity information. The point cloud may be irregularly sampled or contain outliers. It is even possible to handle point sampled volumes.
- As point clouds are usually sampled much denser than comparable meshes, we present a means of performing the deformation calculations on a coarse scale and transfer the results to the fine scale representation of the point cloud at interactive frame rates. To the best of our knowledge, no other system has been presented which could perform similar non-linear deformations without offline processing.

1.1 Related Work

As mentioned in the introduction, several methods exist for editing meshed input data, but which are not directly applicable to point clouds. In the following we concentrate on methods targeting point clouds.

The challenge of editing point clouds interactively has been pursued in a number of papers. In an early work of Pauly et al. [PKKG03] they deform point clouds in real-time using linear interpolations between the handles. Miao et al. [MFXP08] use a more sophisticated method with differential coordinates. Like all linear methods they both produce counterintuitive results for large deformations.

Especially for editing large models the required storage can exceed the available main memory, or the computation time can become prohibitively high. It is thus often necessary to use a simplified model upon which the editing is performed. Wand et al. [WBB+07] describe a method to visualize data sets with a size of several gigabytes in a multi-resolution out-of-core method. The direct editing operations are limited to simple translations or deletions of vertices. More complex operations can only be performed in offline computations. Boubekeur et al. [BSS07] use a streaming method to perform an offline simplification of a large mesh into a smaller point cloud, which can then be edited interactively. A second offline streaming step applies the modifications to the original mesh.

Wicke et al. [WSG05] use the concept of thin shells to construct a network of so-called fibres on the surface of a point cloud, which are then used to model and calculate possible deformations. The fibres provide a mesh representation at a coarse scale, and the deformations are later applied to the detailed representation. As both of the methods of Boubekeur and Wicke require time-consuming steps before and after the user performs the actual editing, the user has to wait for the post-editing steps to complete in order to view the final result. We deem this unsuitable for ad-hoc editing, because any corrections or further editing steps require repeating the whole procedure.

In our method, the actual editing of the point cloud is performed on a differential representation of its surface. This has recently gained much attention in the context of mesh editing. Sorkine and Botsch [SB09] compare gradient-based methods and Laplacian-based methods as the predominant methods used for differential representation-based deformations. They identify two classes: linear methods can provoke a counterintuitive behaviour for deformations, since they cannot handle both rotations and translations of local frames simultaneously, resulting in artifacts of different kinds. Non-linear methods [BPGK06, SA07] solve these issues, but are time-consuming to compute. The mesh editing framework presented by Paries et al. [PDK07] represents local frames using quaternions. By enforcing additional frame constraints they produce intuitive results for translations, rotations and scaling. Their non-linear approach is computationally demanding and thus applicable only to medium sized meshes. We generalize their approach to be applicable to large point clouds.

Space deformations (see e.g. [HSL+06, XZY+07, BPWG07, AOW+08]) are a different approach towards editing operations. They do not directly manipulate the points' positions, but warp the space and thus indirectly manipulate the points. The space deformations can conceptually be calculated quickly, but they lack the ability to respond to the structure of the edited model. Points which are far apart in geodesic distance can be close in space and thus be influenced by editing operations unintentionally. To overcome this limitation, Huang et al. [HSL+06] used a control mesh enclosing the model to separate such geodesically dis-

tant regions from each other. In their approach the user is required to define the control mesh. In contrast, our solution does not require any manual intervention.

The multiscale representation introduced by Pauly et al. [PKG06] and extended by Duranleau et al. [DBP08] encodes the displacement of points between the representations at different scales. It allows to edit the representation for a specific detail level using a space warping function and then apply the deformation to the other detail levels. This enhances interactivity and controllability of the result, but still has the mentioned limitations of space deformations.

Meshless methods, of which space deformations are one specialization, provide further aspects. The phyxels proposed by Müller et al. [MKN+04] are used to fill a volume and preserve it during editing operations. The small amount of phyxels required to fill a volume, allow for physically plausible deformations which can be calculated quickly. However, as Müller et al. already mention, such volumetric approaches are not applicable to point clouds representing a surface.

1.2 Overview

First we present a brief explanation of the mesh editing framework of Paries et al. [PDK07] in Section 2 as it constitutes the basis upon which our work is built. Our generalization to point clouds follows, including the concepts developed to use the mesh editing framework without the need for meshing the input data. In Section 3 we explain our multiresolution method, which allows editing point cloud data sets which are considerably larger than comparable meshes. Although the amount of data points may be up to two scales of magnitude higher, our parallel GPU implementation still provides several frames per second during editing operations. We conclude this paper with an overview of the results (Section 4) and our planned extensions for the future (Section 5).

2 EDITING

2.1 Mesh Editing

As the underlying model for a mesh Paries et al. [PDK07] employ a differential representation of local surfaces. For each vertex in the region of interest $x_i \in R$ they define an orthonormal local frame $F_i = (t_i^1, t_i^2, N_i)$ with right hand orientation. F_i can be interpreted as a rotation matrix and thus be expressed in terms of a unit quaternion q_i.

For each pair of adjacent vertices (i, j) in the mesh M the difference between the quaternions is calculated as $q_i^j = \bar{q}_i \cdot q_j$. During editing and user interaction the mesh surface can be reconstructed from the differential representation and the equation

$$q_i \cdot q_i^j = q_j \tag{1}$$

which is a linear system. Fixing a single unit quaternion q_{i_0} is sufficient for getting a unique solution (except for its rotation), by iteratively solving the remaining equations.

Reconstructing the mesh solely based on the local frames and ignoring the geometry can lead to unintuitive results, because translations are not accounted for. To overcome this, Paries et al. added constraints which impede a change of coordinates of 1-ring neighbors in the local frames F_i. They formulate the constraints as

$$q_i(1, c_i^j)^t \bar{q}_i = (1, x_j - x_i)^t \tag{2}$$

where $c_i^j := F_i^{-1}(x_j - x_i)$ are the local coordinates of x_j in frame F_i.

User input consists of a set of handle vertices H which specify additional frame constraints q_k^{const} and positional constraints x_k^{const}. Given those two sets of constraints, the frame differences q_i^j and the local coordinates c_i^j, the constrained reconstruction problem is to find a set of local frames q_l and absolute vertex coordinates x_l for all vertices within the region of interest R that satisfy Equations (1), (2) and

$$q_k = q_k^{const} \qquad x_k = x_k^{const} \qquad \forall k \in H. \tag{3}$$

As the resulting equation system is overconstrained, Paries et al. solve it in least squares sense using a non-linear optimization procedure. They alleviate the complexity issue by parting the original non-linear problem into several linear equation systems of which the LU-matrices remain constant throughout one editing step. Thus after specifying a region of interest and editing handles, a precomputation step factorizes the original matrices in parallel on the GPU. During the actual editing the linear equation systems are solved on the CPU with backsubstitution, which can be performed at several frames per second even for medium sized meshes with up to 100k vertices.

2.2 Editing Point Clouds

The intention to seek for a generalization of the editing operations from mesh data structures to arbitrary point clouds, comes from the observation that in practice raw data from range scanning devices often is available only in form of 3D vertex coordinates. Depending on the device, color information may also be available, but it plays only a minor role for editing operations.

Though advanced triangulation techniques like MLS surface approximation exist, they are usually dependant upon a specific structure of the point cloud, e.g. a closed surface. Margins, outliers, or in general arbitrarily distributed points pose severe problems. To be more robust, most methods have the undesired effect of smoothing the surface represented by the point cloud, thus degrading the quality of the data set. Furthermore, the computation time for the triangulation becomes notably high for large data sets.

Although the editing framework of Paries et al. relies upon mesh data structures to ensure connectivity of the vertices and to find a vertex's 1-ring, the basic equation systems are formulated without the need for an explicit mesh. They do though require the definition of a local neighborhood in order to calculate the local frames. The simplest choice for a local neighborhood, which comprises each vertex's nearest neighbors in 3D space, is sufficient for our needs and very fast to calculate.

In our approach we use the k nearest neighbors of each vertex, which are computed in a pre-processing step. To determine the nearest neighbors, we use an octree to partition the point cloud, which can be calculated in $O(n \log n)$ time. The choice of k was experimentally determined, and may theoretically be any $k \geq 2$, since we need at least two neighboring points to define the local frame for x_i. In practice, choosing k too small, like e.g. $k = 2$ leads to line-shaped disconnected components, which are not treatable well as a surface by the later algorithm steps. On the other hand, choosing k too large linearly slows down all calculations which iterate the nearest neighbors. We found $k = 5$ or $k = 6$ to provide a good tradeoff between speed and stability, and it is also the average number of neighbors in a regular triangulation.

Using only the initially found nearest neighbors for each vertex can lead to non-symmetric relations, especially if the input point cloud is irregularly sampled. This can prevent a complete traversation of the graph induced by the nearest neighbors, and split it into several seemingly disconnected components. Although removing the non-symmetric neighbors reduces the amount of data to process, it may lead to more fragmentation. Our choice of inserting the missing links to create bi-directional nearest neighbors relations counteracts the possible fragmentation. At the same time it ties outliers to the main connected component(s). The problem with auxiliary connected components which do not include user-defined constraints is that they render Equation system (1)-(3) underdetermined, and thus provide no stable solution. During the preprocessing step these unconstrained components are detected and removed from the editable part of the model.

2.3 Area and Volume Preservation

Volume preservation is a key feature used in mesh editing and has also been applied to point cloud editing [MKN+04]. It is however not unambiguous how to define this property, since a point cloud does not possess an inherent volume, as opposed to a closed mesh. This is especially the case for point clouds originating from single-image 3D capturing devices, which can only capture one side of any given object at a time. As our method strives to preserve the local neighborhood of each point, the volume is only an affected property, not one which is calculated or optimized.

In particular, if a surface is stretched using handles on opposite sides, the surface is also enlarged in perpendicular direction to preserve the local shape of each point's 1-ring. This is a rather counterintuitive behaviour, since most materials in nature exhibit the opposite behaviour of shrinking in perpendicular direction when stretched in the other direction (e.g. rubber and metal). Nevertheless, auxetic materials exist, which do enlarge in perpendicular direction (e.g. special foams), and others which do not change their perpendicular size at all (e.g. cork). Our method can be parameterized to exert any one of these elasticity behaviours by scaling the local frames q_i in each iteration before solving the equation systems, as detailed in [PDK07].

3 SAMPLING

The matrix factorization step performed during precomputation is necessary to reduce the calculation time during the actual editing of a model. We employed the SuperLU library [DEG+99] which is suited well to decompose the sparse matrices. Although the matrix size grows squarely with the number of handle vertices, the sparsity leads to a computation effort which grows below quadratic. Nevertheless, from 100k vertices onwards this requires several minutes of precomputation and eventually the matrix size grows too large to be handled in main memory.

Since complex operations on large data sets are today still limited in their speed by the available computation power, we devised a multiresolution scheme similar to Wicke et al. [WSG05]. The time-consuming solving of the Equation systems (1)-(3) is performed on a coarse-scale subsample of the input point cloud, and the fine-scale representation is then interpolated. Pauly et al. [PGK02] survey different point based simplification methods. For our subsampling we chose a method from the category of hierarchical clustering methods, since they adapt well to point clouds with varying point sampling densities. To generate the subsamples, we use an octree partitioning the original

point cloud. Taking the sample points from each leaf of the octree allows us to handle arbitrary point clouds, not only those representing implicit surfaces. At the same time, areas of the original point clouds which have a high sampling density will receive a higher amount of sample points, thus preserving local details.

Calculating the nearest neighbors for the sample points can be performed with the octree that was already calculated. For some models like the dragon model this can yield undesired connections between points (Figure 2b), which cannot be considered adjacent in the original, but become adjacent in the sampled point cloud due to the lower point density. In the work presented by Xu et al. [XZY+07] the user is required to manually create a control mesh which prevents such connections between originally unconnected parts. Our connected sample points are in effect similar to the control mesh used by Xu et al., but we sought for an alternative working without manual intervention. The idea is to find the nearest neighbors in a geodesic sense instead of the euclidean distance. As the geodesic distance conceptually requires a surface to be applicable, using the nearest neighbors relation of the original point cloud satisfies this need, but still makes it applicable to non-surface-like point clouds. With a breadth first search (Algorithm 1) for each point in the original point cloud, the geodesic nearest neighbors can be found in $O(n \cdot r)$ time, where n is the number of original points and r is the original-to-sample number ratio. The result can be seen in Figure 2c.

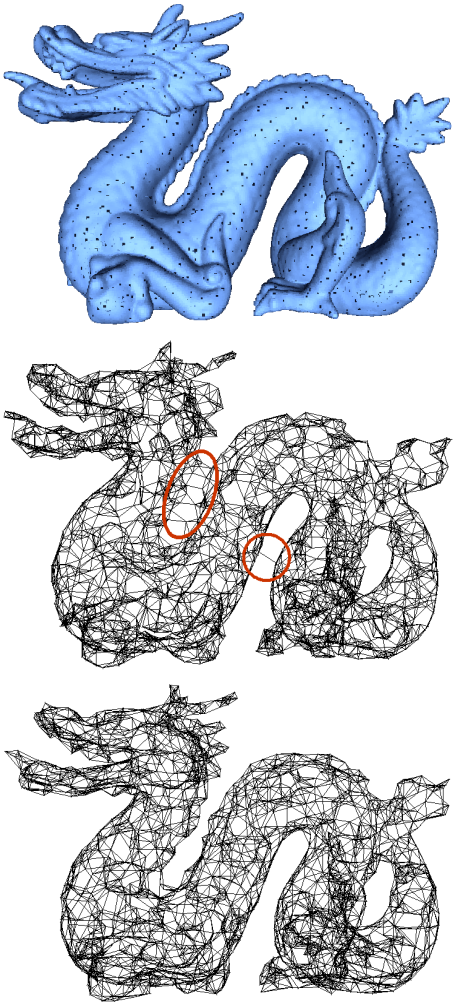


Figure 2: a) Original point cloud with sample points drawn in black. b) Sample points connected using nearest neighbors. Wrong connections between originally separated parts are created due to proximity. c) Calculating nearest neighbors on the original point cloud with a breadth first search prevents wrong connections.

After each iterative solving step for the basic equations, the new position p_i' and normal n_i' of each original point is interpolated from the translations and rotations of its nearest sample points:

$$p_i' = \sum_{s \in neighbors(i)} w_i^s \cdot (q_s \cdot p_i^s + p_s) \tag{4}$$

$$n_i' = \sum_{s \in neighbors(i)} w_i^s \cdot (q_s \cdot n_i^0) \tag{5}$$

with

$$w_i^s = \frac{1}{|p_i^s|^2} \Big/ \sum_{s \in neighbors(i)} \frac{1}{|p_i^s|^2} \tag{6}$$

where $p_i^s = p_i^0 - p_s^0$ is the coordinate of point p_i in the local coordinate system of p_s, and q_s is the quaternion describing the rotation of s from its original to its new orientation. The weights w_i^s account for a smaller influence of neighbors which have a greater distance.

We implemented this interpolation method on the CPU first, but the involved quaternion multiplications proved to be a limiting factor to the achievable

Input: point i_0; number of sample neighbors to find k; Samples S; Original nearest neighbors relation N
init queue $q \Leftarrow i_0$
init geodesic neighbors $G(i_0) = \emptyset$
repeat
  $j = q.dequeue()$
  for all $n \in N(j)$ **do**
    if n not visited **then**
      $q.enqueue(n)$
      mark n as visited
  if $j \in S$ **then**
    add j to $G(i_0)$
until $|G(i_0)| \geq k$ or $q = \emptyset$

Algorithm 1: Geodesic nearest sample neighbors for a point

Model	# points	# sample points	Subsampling	Precomputation	Equation solving	Upsampling
Bunny	35k	100	4.0	0.1	0.001	0.003
Bunny	35k	1000	0.5	2.0	0.019	0.004
Armadillo	172k	200	101.3	0.1	0.002	0.016
Armadillo	172k	5000	3.2	18.3	0.098	0.018
Dragon	437k	200	390.2	0.1	0.002	0.037
Dragon	437k	5000	13.7	23.3	0.116	0.040
Plain	1000k	200	1923.0	0.1	0.002	0.083

Table 1: Computation times in seconds on a 2.4GHz CPU and a NVidia GeForce 8800 GTX GPU. Subsampling and precomputation are executed once, while equation solving and upsampling occur each frame.

speed, since multiplying two quaternions includes 16 floating point multiplications, and multiplying a quaternion with a 3D vector includes 27 floating point multiplications. The sheer amount of operations required to perform the interpolations for large data sets with over 1M points does not allow for a true real-time editing experience using this approach, as the frame rates can drop to 1-2 frames per second. The implementation of the upsampling step in CUDA for parallel computation on the GPU is straightforward, since p_i' and n_i' can be computed independently for each point on the available GPU processors. Our current implementation running on an off-the-shelf graphics card (NVidia GeForce 8800 GTX) performs on average 5-8 times faster than the CPU version. During the editing operation, for 1M points we measured 8 frames per second for the GPU implementation, while the CPU variant yielded only 1.4 frames per second.

4 RESULTS

We tested our method on a variety of point clouds, including both artificial and scanned models. All of the models have been used without manual preprocessing. Table 1 gives an overview of the computation time for setup and each iteration step during the editing phase. Usually the model converges to its final shape during 3-5 iterations.

As we have explained before, the equation solving time consumption grows slower than $O(n^2)$ where n is the number of sample points, while the upsampling step requires only linear time depending on the number of points to be interpolated.

The visual quality of the result is largely independant of the number of sample points, unless it falls below a certain threshold depending on the complexity of the model. This means e.g. for the Stanford Bunny (Figure 3), which does not have a complex structure, that the resulting point positions have very small deviations when calculated with 100, 1,000 or 10,000 sample points. Only with fewer than about 100 sample points the upsampling step produces artifacts for large deformations (Figure 4). For the Dragon model (Figure 5), which is more complex due to its winding body, sampling artifacts can be observed for less than 500 points (Figure 4). Those artifacts could be circumvented by a more sophisticated interpolation scheme which does take into account not only the nearest neighboring sample points, but a selection of sample points which are evenly distributed on the surface around the point to be interpolated. A too low sample density does however defeat the purpose of our nonlinear editing method by reducing it to the linear interpolation.

One minor drawback of our implementation using quaternions is the inability to perform handle rotations of more than 360° within one editing step. This is due to the normalization of quaternions which we perform for stability reasons when solving the equation systems. In practice however this is not of concern, as it is possible to perform several editing steps with smaller rotations to achieve the desired result.

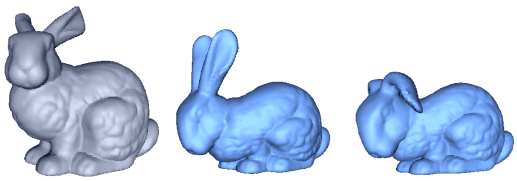


Figure 3: Bunny model with 2 editing steps.

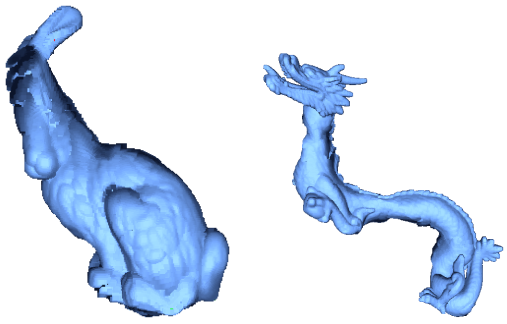


Figure 4: Bunny and Dragon models edited with only 50 sample points each. Sampling artifacts can be observed in the head area of the Bunny and neck of the Dragon.

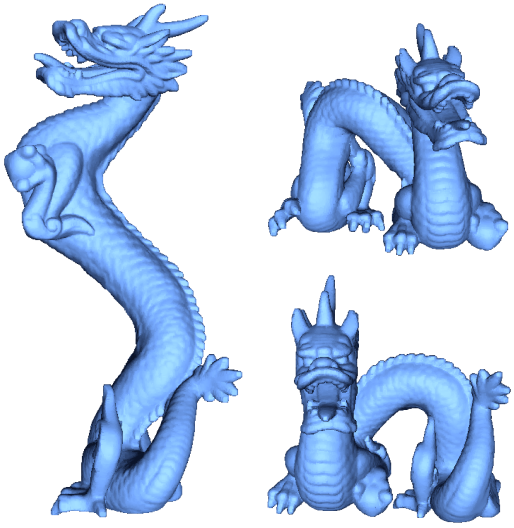


Figure 5: Dragon model after rotation and translation of the head. Session time including subsampling and precomputation for each result: 1 minute.

5 CONCLUSIONS AND FUTURE WORK

We have presented a novel method to interactively edit large point clouds. Using a non-linear deformation model allows to produce physically plausible deformations even for large modifications. The multiresolution approach faithfully handles the coarse scale deformations while the model details are preserved. Our parallel implementation provides the speed necessary for real-time editing scenarios, shortening the time required to produce a desired result. For models which are larger than the available main memory, our method could be extended with a further sampling level, for which the upsampling step would be performed offline. Our method does not require any manual preconditioning steps to create a control mesh, and is thus suited for direct editing of arbitrary point clouds.

For the future we plan to incorporate the deformation features into a model reconstruction framework which can work on a number of scanner-acquired time-varying point clouds.

REFERENCES

[AOW+08] B. Adams, M. Ovsjanikov, M. Wand, H. Seidel, and L. Guibas. Meshless modeling of deformable shapes and their motion. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 77–86, Dublin, Ireland, 2008. ACM/Eurographics, Eurographics Association.

[BPGK06] M. Botsch, M. Pauly, M. Gross, and L. Kobbelt. Primo: coupled prisms for intuitive surface modeling. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 11–20. Eurographics Association, 2006.

[BPWG07] M. Botsch, M. Pauly, M. Wicke, and M. Gross. Adaptive space deformations based on rigid cells. *Computer Graphics Forum*, 26(3):339–347, 2007.

[BSS07] T. Boubekeur, O. Sorkine, and C. Schlick. Simod: Making freeform deformation size-insensitive. In *IEEE/Eurographics Symposium on Point-Based Graphics 2007*, September 2007.

[DBP08] F. Duranleau, P. Beaudoin, and P. Poulin. Multiresolution point-set surfaces. In *GI '08: Proceedings of graphics interface 2008*, pages 211–218, Toronto, Ont., Canada, Canada, 2008. Canadian Information Processing Society.

[DEG+99] J. Demmel, S. Eisenstat, J. Gilbert, X. Li, and J. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Analysis and Applications*, 20(3):720–755, 1999.

[HSL+06] J. Huang, X. Shi, X. Liu, K. Zhou, L. Wei, S. Teng, H. Bao, B. Guo, and H. Shum. Subspace gradient domain mesh deformation. *ACM Trans. Graph.*, 25(3):1126–1134, 2006.

[MFXP08] Y. Miao, J. Feng, C. Xiao, and Q. Peng. High frequency geometric detail manipulation and editing for point-sampled surfaces. *Visual Computer*, 24(2):125–138, 2008.

[MKN+04] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 141–151, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[PDK07] N. Paries, P. Degener, and R. Klein. Simple and efficient mesh editing with consistent local frames. Technical Report CG-2007-3, Universität Bonn, July 2007.

[PGK02] M. Pauly, M. Gross, and L. Kobbelt. Efficient simplification of point-sampled surfaces. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 163–170, Washington, DC, USA, 2002. IEEE Computer Society.

[PKG06] M. Pauly, L. Kobbelt, and M. Gross. Point-based multiscale surface representation. *ACM Trans. Graph.*, 25(2):177–193, 2006.

[PKKG03] M. Pauly, R. Keiser, L. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. *ACM Trans. Graph.*, 22(3):641–650, 2003.

[SA07] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 109–116, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[SB09] O. Sorkine and M. Botsch. Tutorial: Interactive shape modeling and deformation. In *Eurographics*, 2009.

[WBB+07] M. Wand, A. Berner, M. Bokeloh, A. Fleck, M. Hoffmann, P. Jenke, B. Maier, D. Staneker, and A. Schilling. Interactive editing of large point clouds. In Baoquan Chen, Matthias Zwicker, Mario Botsch, and Renato Pajarola, editors, *Symposium on Point-Based Graphics 2007 : Eurographics / IEEE VGTC Symposium Proceedings*, pages 37–46, Prague, Czech Republik, 2007. Eurographics Association.

[WSG05] M. Wicke, D. Steinemann, and M. Gross. Efficient animation of point-sampled thin shells. *Computer Graphics Forum*, 24(3):667–676, 2005.

[XZY+07] W. Xu, K. Zhou, Y. Yu, Q. Tan, Q. Peng, and B. Guo. Gradient domain editing of deforming mesh sequences. *ACM Trans. Graph.*, 26(3):84, 2007.

A Facade Tracking System for Outdoor Augmented Reality

José F. Martins[1,2] Jorge A. Silva[1,3] A. Augusto de Sousa[1,4]
[1]FEUP
[2]ISMAI, [3]INEB, [4]INESC Porto
R. Dr. Roberto Frias
4200-465 Porto, Portugal
{jfmm, jsilva, aas}@fe.up.pt

ABSTRACT
We describe a real-time facade tracking system that uses, as setup information, only two images of a facade, captured on the moment. No more previous information is needed, such as a facade 3D model, dimensions or aspect ratio. Feature points and their local descriptors are extracted from that pair of images and used during the detection and tracking of the facade. Additionally, parallax and topological information is also used in order to increase the overall robustness of the tracking process. Experiments show that the system can detect and track a wide variety of facades, including those that are not entirely planar, partially occluded or have few distinguishable visual landmarks. The reliance on on-the-spot information, alone, makes this system useful for Outdoor Augmented Reality applications, in an Anywhere Augmentation urban context.

Keywords
Outdoor Augmented Reality, Anywhere Augmentation, Facade Tracking, Computer Vision.

1. INTRODUCTION
The main purpose of Augmented Reality (AR) is to add, in real-time, virtual objects to real world images in such way that they appear to naturally belong to that world [Bar01].

Adding virtual objects to a real image is only visually convincing if they are perfectly registered with the real world. In order to do that, it is necessary to render them from a virtual camera that, ideally, should have the same pose as the real one. There are several methods that use a GPS, a gyroscope, an accelerometer or other sensors to determine the camera pose [Rol01]. However, the methods that appear to have the greatest potential are those that use captured images. One way to determine the camera pose is to find a set of, at least, four matches between points in an image of a reference plane, taken with a known camera pose (usually, a frontal one) and points from the same plane in a image whose camera pose is to be determined [Lep05].

The reference plane must be continuously tracked. For this purpose, Lepetit et al describe several methods.

In indoor scenes, the reference plane is usually an easily identifiable synthesized pattern, as happens with the well known ARToolkit [Kat99, Art09].

In outdoor scenes, the use of synthesized patterns is not feasible for several reasons, namely, the need for a large pattern and the probable difficult positioning of the pattern in the surrounding environment. However, in an outdoor urban environment, the facades of the buildings can act as a suitable reference pattern.

In the AR domain, some important contributions to facade tracking have been made during the current decade [Sim02, Jia04, Rei06, Xu08]. Generally, they can use a building facade as a reference pattern for achieving the registration between the virtual objects and the real world scenario. The affordability and overall good image quality of off-the-shelf digital cameras make the use of captured images a very common approach [Sim02, Xu08].

However, using only captured images as input data has some drawbacks: the captured image can be blurred by jerky or fast camera movements and the reference pattern may become partially visible or not visible at all because of occlusion. These drawbacks make facade tracking very difficult or even impossible to achieve. To overcome them, some systems combine captured images with other types of data, taken from sensors like GPS and/or inertial sensors [Jia04, Rei06, Rei07].

The first step of a facade tracking system is to detect the facade in the captured frames. This will allow the AR system to do the camera pose initialization (CPI). After that, most systems track the facade, based on what happened on previous frames, until the tracking fails, when the CPI must be redone. However, several systems cannot solve the CPI problem automatically; some demand for user interaction [Sim02, Jia04] while others need to know the initial camera position in the real world [Rei06, Xu08].

When the tracking fails, the system must be able to detect the facade again. Although most systems can perform this task automatically, some of them need the camera pose to be similar to the last known pose (prior to tracking failure) [Sim02, Jia04] or to one of the previous known poses [Rei06].

To achieve a more robust tracking, some systems require a 3D model of the facade. It can be a wire-frame model [Jia04], or a more elaborated textured one [Rei06]. Unfortunately, creating the model is not an easy task and may take considerable time. Additionally, since it must be created before the tracking phase, anywhere augmentation is an impossible goal for these systems. Such a goal is possible for systems that only need on-the-spot information to start tracking [Sim02, Xu08]. This information is limited to a single image of the facade or just a few ones.

The determination of the camera pose can be based on the matching of feature points, as in [Sim02, Xu08], edges [Jia04] or a combination of the above [Rei06]. Reference image based systems try to match feature points in the captured image with those previously detected in the reference image, while 3D model based systems try to match edges in the image with the rendered 3D model edges.

During the augmentation phase, reference image based systems, like [Sim02, Xu08], usually operate at relatively long distances from the tracked facade (distances superior to the overall dimensions of the facade).

This paper presents a tracking system that can detect and track a building facade in a video stream, provided that the facade has a dominant planar surface and that two images of it, previously taken from different viewpoints, are available. This system overcomes some of the limitations of other systems exclusively based on captured images, namely: it only needs information gathered on-the-spot; the CPI problem is solved without user intervention; when the tracking fails, the camera pose does not need to be similar to a previously known one, for the facade to be detected again; and, finally, the camera does not need to be far from the facade.

The paper is organized as follows: section 2 presents a brief overview of the system; sections 3 and 4 describe the most significant steps of its implementation; section 5 presents the results of the system evaluation and, finally, section 6 enumerates the most important conclusions retrieved from this work.

2. OVERVIEW OF THE SYSTEM

The proposed tracking system has two operating phases: a setup phase and a working phase. In the setup phase, the system acquires and prepares all the necessary information for the next phase, namely a set of feature points, coplanar with the dominant plane of the facade, and their topological relations.

In the working phase, the system will try to detect the facade in a captured image, by matching the feature points detected in the video sequence images with those detected in the setup phase, by using feature descriptors and topological information. If the detection is successful, it will start tracking the facade, by using a conventional sparse optical flow technique. When the tracking fails, the system reverts to the detection phase and the process is repeated.

To detect the facade in an image, it would be convenient to have a reference image, in a frontal pose. In most cases, this pose is impossible to obtain. In addition, since most facades are not planar, a single image, even in a frontal pose, may be insufficient for a correct detection of the facade. Instead, using two reference images of the facade, taken from different positions, will help to identify the coplanar points that, in the proposed way for determining the camera pose, are used in the matching process.

The matching process between one of the reference images of a facade and an arbitrary image of the same facade, captured with an unknown pose, is not a simple task, for various reasons: first, because they have different poses; then, because facades rarely are truly planar; finally, because the two images can be significantly different due to lighting variations, reflections, occlusions or even changes of the facade visual appearance (e.g. when a window blind is closed, between the captures).

Several matching methods are based on feature point extraction from images. Feature point detectors, like SIFT [Low03], SURF [Bay06], FAST [Ros06] or FIRST [Bas08] have been used because of their robustness to changes in scale, view point, luminance and even to partial occlusions. For each extracted feature point there is an associated local descriptor. The comparison of the descriptors is the basis of the matching process. However, since there are many repeated visual landmarks in a facade (e.g. windows), a straightforward comparison would, most likely, produce a large number of incorrect matching pairs. Therefore, additional processing is needed to remove those false matches. In the proposed system,

topological constraints are used for this purpose. Similar constraints have been used by other authors [Fer03] to help solving the matching problem between images acquired from different poses.

In the following, the two above mentioned phases are described.

3. SETUP PHASE

The setup phase has the following steps: (a) capture of two facade reference images; (b) delineation of the facade region of interest (ROI); (c) detection of feature points; (d) identification of all the feature points that belong to the facade dominant plane and (e) topological characterization of the feature points.

The information resulting from this phase is: the facade ROI in both reference images; the detected feature points that belong to the facade dominant plane, Π (a plane coplanar with a major part of the facade surface), and the characterization of the feature points. This characterization is achieved in two ways: by a local descriptor that will be used in the matching process and by two types of topological information, associated with each feature point: the collinearity information which tells if it is collinear with other feature points, making up a "line" of feature points and the sidedness information which tells on which side of each of these "lines" the point lies.

Reference Images and ROI

Two reference images of the facade are needed. These images, I_L and I_R, must be taken from different positions and preferably from opposite sides of the facade (Figure 1). This will increase the parallax effect which will help identifying the feature points that do not belong to the dominant plane, Π.

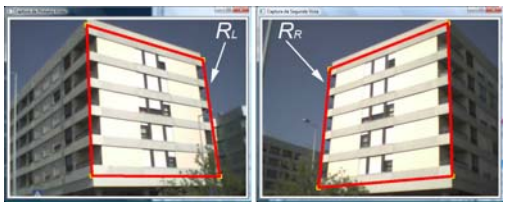


Figure 1. ROIs delineation on I_L and I_R.

The facade ROIs, R_L and R_R, must be delineated on I_L and I_R (Figure 1). This is the only step in the whole process requiring human intervention. This step is necessary to restrict the feature point detection to R_L and R_R and also to identify Π. R_L and R_R are manually delineated by the user, through the selection of the corners of the quadrilaterals, on I_L and I_R, corresponding to a rectangle in the facade. This rectangle does not need to visually exist; only some of its edges and corners are needed, as a visual aid for the user. However, it must be coplanar with Π and cover a major part of the facade (ideally, all of it).

Feature Point Detection

To detect the facade in an image, I_U, captured with an arbitrary pose, it is necessary to match feature points between I_U and the reference images (I_L and I_R). In I_L and I_R, feature detection is restricted to R_L and R_R.

Feature points are detected using the SURF algorithm [Bay06], which is one of the fastest available and has a great potential for AR applications. This algorithm returns the coordinates of each detected point, its feature strength (Hessian value), size, orientation and Laplacian value, as well as a descriptor of its neighborhood in the image. This descriptor is an array of 128 elements that describes the distribution of Haar-wavelet responses within the feature point neighborhood and has the important properties of being invariant to rotation, scale and luminance. These properties help the matching process to produce a higher rate of correct matches between different poses. The detected SURF point sets in R_L and R_R will be named, respectively, S_L and S_R. Only points with a Hessian value greater or equal to 500 are retained. For a better matching performance, both sets are filtered, in order to assure that all the remaining features have a minimum size and a minimum distance between each other.

Identification of Points Belonging to Π

In the working phase, the facade will be detected through the matching of two sets of coplanar points. Therefore, only points belonging to Π should, ideally, be used; all the other points should be removed from S_L and S_R.

The identification of the points non-coplanar with Π has two important steps. In the first one, R_L and R_R are transformed into frontal pose (Figure 2), F_L and F_R. In order to rectify both R_L and R_R, the true dimensions of these regions, or at least their aspect ratio, must be known. Usually, those dimensions are unknown, but the aspect ratio of a rectangle can be calculated from a non-frontal pose image of itself, through the use of the image vanishing points, as described in [Cip99].

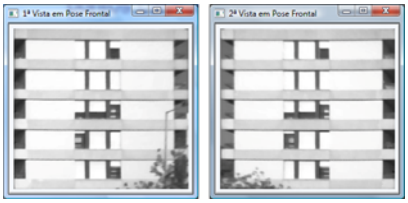


Figure 2. ROIs in a frontal pose (F_L and F_R).

In the second step, S_L and S_R are projected into a frontal pose. After this, the feature points, in R_L and R_R, which belong to Π can be easily identified, since their positions should be the same in F_L and F_R. However, it is also possible that a facade landmark

can generate a feature point in one of the reference images, but not in both of them.

So, in order to identify the largest possible number of points belonging to Π, another approach, that does not require a feature point to be detected in both reference images, was adopted instead. Each feature point, in either R_L or R_R, is projected into both F_L and F_R. The neighborhoods of each projected point, in F_L and F_R, are compared using normalized cross correlation and if they are similar enough (correlation factor superior or equal to 85%) then the point is identified as belonging to Π. A feature point resulting from a landmark, like a lamppost, that is away from Π, will usually have different neighborhoods, in F_L and F_R (Figure 3), being identified as non-coplanar with Π.

Figure 3. Detection of feature points that do not belong to Π.

Topological Characterization of the Feature Points

Facades frequently have a large number of very similar visual landmarks. This contributes to a large rate of false matches. A possible solution for this problem would be to use only singular feature points (points whose descriptor is unique). However, in the case of a facade, this solution would probably reduce the number of useful points to a few.

In this particular case, the topological characterization of the points is a better solution to make each point more easily distinguishable from the others. In many facades, visual landmarks are usually concentrated along horizontal/vertical "lines". In both F_L and F_R, these "lines" can be easily identified, since they are approximately coincident with the rows/columns of the image. Rows/columns of F_L and F_R that have at least 12 feature points within a threshold distance, T_1=5 pixels (Figure 4) are retained as horizontal/vertical "lines". All feature points that do not belong to any retained "line" are removed.

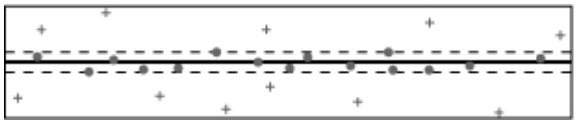


Figure 4. Identification of feature points concentrated near an image row, in F_L or F_R.

The topological characterization process associates the following two types of information with each point, in F_L and F_R: collinearity and sidedness. Collinearity identifies the horizontal and/or vertical "line(s)" of near-collinear feature points that the point belongs to. Sidedness identifies the side where the point lies relatively to each other "line".

4. WORKING PHASE

During the working phase, an image sequence is acquired and processed. This phase is divided into two subphases: detection and tracking. In order to start tracking, the facade must be detected in one of the incoming frames. If the detection is successful, the facade is tracked in the following frames until the tracking fails. Then, the process is repeated.

Facade Detection

The detection subphase consists of the following steps: (a) capture of a frame, I_U; (b) detection of feature points; (c) matching between the feature points detected in I_U and those detected in I_L and I_R; (d) elimination of false matches and (e) calculation of the homography between the matched feature points. Using this homography, it is possible to delineate the facade ROI in the captured frame and determine the camera pose.

4.1.1 Feature Point Detection and Matching

The first step is to capture a frame I_U in which the facade will be either detected or tracked, whatever the case may be. The camera pose and the existence of the facade in I_U are both unknown. Facade detection requires a matching process between the SURF feature points detected in I_U and those detected in I_L and I_R (during the setup phase). In I_U, feature point detection is applied to the whole image. The resulting set of detected points in I_U will be named S_U.

In order to maximize the number of correct matching pairs, two matching processes, $m(S_U,S_L)$ and $m(S_U,S_R)$, are applied to the feature sets S_U, S_L and S_R. To accelerate the matching process, each set is divided into two subsets, based on the Laplacian signs, thus avoiding unnecessary comparisons between local descriptors of points with different signs.

The Best-Bin-First (BBF) method [Bei97] is used for matching. This method uses local feature descriptors in order to find, for each point of the first image, the best match (nearest neighbor), in the second image. A k-d tree [Fri77] is used to store the local descriptors of the points from one of these sets. The nearest neighbor of a given descriptor, on the other set, can be found, very efficiently, by searching only a relatively small part of this tree. The returned nearest neighbor is considered valid, if the Euclidean distance between both descriptors is inferior to certain threshold (a value of 3 was used in our experiments).

Two sets of matching pairs, $P_L=m(S_U,S_L)$ and $P_R=m(S_U,S_R)$, result from this matching process. If neither P_L nor P_R has enough pairs (at least 20), the detection is considered unsuccessful. Otherwise the

set with the largest number of matching pairs, named P_S, is selected.

4.1.2 Elimination of False Matches

P_S usually has some incorrect pairs whose number can be affected by several factors: images acquired with different camera poses; luminance variation; the used local descriptor; the matching method; and the visual content of both images. In order to obtain a more robust facade detection and a precise camera pose, it is imperative to reduce, as much as possible, the number of incorrect matches.

A common method to solve this problem [Har03] uses the RANSAC algorithm [Fis81] to calculate, in a robust form, the homography that relates two images of the same plane. This method starts by randomly selecting a set of matching pairs. This set is used to calculate a homography that will be used to project each point of the first image into the second one. The distance between the projected point and its matched pair is calculated. If the distance is too large, the pair is labeled as an incorrect one (outlier). This process is repeated until the number of correct matching pairs (inliers) is acceptable or a maximum number of iterations is reached. If a minimum number of inliers is not achieved, the facade detection is declared as failed. Usually, this method produces good results even in the presence of a large number of outliers. However, in a facade, its straightforward use may generate poor results.

Usually, inliers display a common and uniform behavior that set them apart from outliers, which show a behavior of a random nature. In the case of a facade, outliers may display a common and uniform behavior, for example, when feature points belonging to one building floor match with similar points from another floor. If the number of matches is substantial, RANSAC may wrongly assume that they are inliers.

The proposed solution for outlier elimination is based on the application of the collinearity and sidedness constraints to the selected set of matching pairs (P_S).

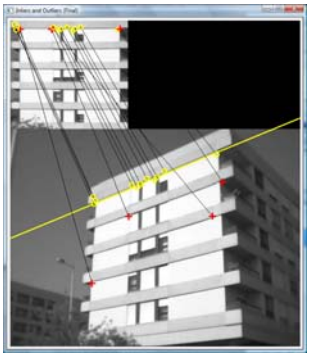


Figure 5. Extracted horizontal line with some outliers (represented by the symbol "+").

The collinearity constraint states that coplanar landmarks that are collinear in F_L or F_R should also be collinear in I_U. Feature points that do not verify this constraint are most likely outliers and must be removed from P_S (Figure 5).

This constraint is applied in the following way: a "line" of feature points in S_L or S_R is taken; these feature points are matched with points in S_U (the resulting subset is named L_U); if L_U has a minimum number (8) of matched points, a line is fitted to these points, using RANSAC; the resulting fitted line is considered valid if it fits a minimum number of points in L_U (40%). The points in L_U that do not belong to the line are marked as outliers and removed.

In order to fit a line to the points in L_U, it is necessary to define a threshold, T_1, for the distance between each point and the line, beyond which the point will be labeled as an outlier. Since the size of the facade in I_U depends on its distance to the camera, T_1 must be updated by a scaling factor; this factor is calculated as the ratio between two distances d_1 (measured between the two horizontal/vertical lines in F_R or F_L, having the largest sets of matched points in P_S) and d_2 (measured between the corresponding lines in I_U) as shown in Figure 6.

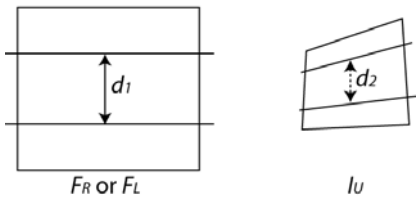


Figure 6. Distances d_1 and d_2 used to update the threshold T_1.

After the application of the collinearity constraint, some incorrect matches may still remain. To remove them, the sidedness constraint is applied.

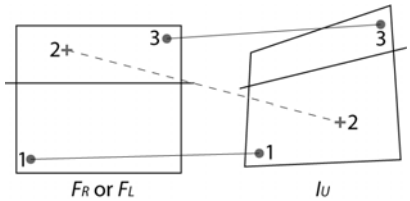


Figure 7. Verification of the sidedness constraint (point 2 violates the constraint).

Let us consider a line fitted to L_U and the corresponding line in F_L or F_R; a point A, in I_U, not belonging to L_U; the point B, in F_L or F_R, matched with A. The sidedness constraint states that A and B must be on the same side of the line fitted to L_U and the corresponding line in F_L or F_R (Figure 7). Ideally, a single mismatch would be sufficient to mark a point as an outlier. However, since there are occasionally some poorly fitted lines, a single mismatch may be insuffi-

cient to assume that conclusion. To overcome this problem, for each matched point, the number of times that sides match and mismatch, in both images, relatively to each fitted line, are totaled. If the first total is inferior to the double of the second one, then the point is marked as an outlier and removed.

4.1.3 Facade Delineation

If the number of matches remaining in P_S is above a threshold (8 pairs), the facade is declared as being present in I_U and the last step of the facade detection consists of delineating the contour of the facade in I_U. In order to delineate the facade on I_U and to enable the AR system to perform the CPI, a homography between the feature points of the reference image (I_L or I_R) and those of I_U is calculated, using RANSAC, and refined, using the Levenberg-Marquardt method [Har03]. Using the final homography, the corner points of the facade ROI (R_L or R_R), in the reference image are projected into I_U, thus delineating the ROI in this image. As a final validation step, the ROI shape must also pass a convexity test. Finally, the homography can also be used by the AR system to determine the initial camera pose.

Facade Tracking

The tracking subphase consists of the following steps: (a) capture of the next frame, I_{U+1}; (b) tracking of a selected set of feature points via sparse optical flow; (c) replacement of the lost tracked points and positional correction of the ones with high tracking error, if feasible, and (d) calculation of the homography between the feature points tracked in I_{U+1} and those detected previously in I_L or I_R..

4.2.1 Feature Point Tracking

After the facade has been successfully detected in the current frame, the system will try to track it in the next one. To do that, a number of feature points must be selected for tracking.

The selected points will be tracked by sparse optical flow between two consecutive frames (I_U and I_{U+1}). The optical flow implementation is based on the Bouguet sparse variant [Bou99] of the Lucas-Kanade optical flow algorithm [Luc81]. This implementation demands that the tracked points are good features to track which, in this case, means that they must be corner points. Unfortunately, not all SURF points are corner points so, during the setup phase, all the SURF feature points, coincident or near (within the feature size) to a Harris corner point [Har88], were labeled as good features to track. The position, in F_L or F_R, of the neighboring Harris corner point relative to the SURF point is stored, since the first one will be the point effectively tracked.

The subset P_K of matching pairs with good features to track is extracted from the current set of matching pairs P_S. From P_K, the system will select the points that are near the corners and to the center of the ROI. The maximum number of tracked points depends on the performance of the hardware platform.

The optical flow algorithm is applied to all the neighboring Harris Corner points of the selected SURF points in the current captured frame (I_U) and the next one (I_{U+1}). The algorithm returns the new estimated position for the tracked points in I_{U+1}.

4.2.2 Feature Point Replacement or Correction

Some points can be lost or have a high tracking error. The system tries to replace the lost points with new ones and correct the position of the points that have a high tracking error. In order to do both, the SURF detector is applied to a very small region centered on the estimated point position in I_{u+1}. The local descriptors of the detected points are matched with the single local descriptor of the replacement point or the point with high tracking error and if the match is successful, the point is replaced or its position is corrected, whatever the case may be.

If the number of remaining points is greater than 8, the homography is calculated by RANSAC, making it possible to delineate the ROI, and the tracking process will continue. Additionally, the AR system will be able to determine the camera pose in I_{U+1}. Otherwise, if the number of remaining points is insufficient, the tracking process is considered as failing and the system will try to detect the facade in the next frames, until the detection is successful.

5. SYSTEM EVALUATION

Since the proposed method depends on a large number of parameters, it would not be possible to evaluate its performance for each parameter combination. So, the parameters were selected through several experimental tests related to the particular result desired for each one. The global parameter setting was evaluated through a set of experiments described below.

The evaluation of the system was divided into two parts for evaluating separately the detection subphase only and the whole working phase. The evaluation was done on a computer running at 2.4 GHz using images with a 640×480 resolution.

In the first part, the system was forced to detect a facade in image sequences of four different buildings (Figure 8). These facades have some properties that can make the detection difficult, namely: landmarks non-coplanar with the dominant facade plane (all the facades, particularly no. 2), rounded lines (no. 4), small occlusions from trees, lampposts and traffic signs (nos. 1, 2 & 4). The used reference images were, for each sequence: 1.7 and 1.11, 2.2 and 2.9, 3.1 and 3.2, and, 4.5 and 4.8.

In all the four sequences, the facade was delineated with an average reprojection error inferior to 2 pixels, respectively: 1.98 pixels, 1.83 pixels, 1.16 pixels and 1.45 pixels. The reprojection error in each image was calculated as the average distance between the four reprojected corners and the real ones, these being manually selected.

The detection was robust even when the facade had a large number of visually similar landmarks (facades no.1 & no. 2) or non-coplanar ones (no. 2) and in the presence of small occlusions (nos. 1, 2 & 4). In spite of the large number of round visual landmarks and the existence of few horizontal/vertical lines on the facade the robustness of the detection (no. 4) did not decrease. The system was also able to detect a facade from viewpoints at larger distances than those used to capture the reference images (no. 3). As expected, the system is unable to robustly detect facades that do not have a dominant planar surface and those that have few visual landmarks or large occlusions.

In the second part, a captured video was used to evaluate the working phase (Figure 9). The building has a large planar surface with some non-coplanar landmarks (the balconies) and there is also a small occlusion (the tree). The video sequence has 200 frames. The facade was detected in the first frame and tracked, without failure, until the last frame. The two images in Figure 1 were used as reference images.

In order to evaluate the camera pose determination, an augmented wireframe box was registered to the tracked facade. By visual inspection of the augmented video, it is possible to conclude that the augmented box was satisfactorily registered with the facade.

In the setup phase, the identification of points belonging to the facade dominant plane took on average 334 ms to complete and the topological characterization was executed with an average time of 356 ms. During the tracking subphase, it was possible to achieve a real-time frame rate (superior to 25 fps). However, during the detection subphase, the frame rate dropped to a value between 3 to 10 fps (depending on the visual content of the frame). The drop on the frame rate was caused mainly by the feature point detection and by the two matching processes. On the other hand, the elimination of false matches by the application of the topological constraints took a median time of 52 ms to complete.

Using a smaller local descriptor (with 64 elements) and a faster but more limited version of SURF, the U-SURF [Bay06], it is possible to boost the detection frame rate. Unfortunately, in this case, the detection phase generates a higher reprojection error that has a negative impact in the overall robustness of the tracking system.

Many other factors should be considered for a more complete evaluation of the system, such as: the number of coplanar landmarks available, the camera pose of the reference images, the accuracy of the selection of the ROIs corners, the occlusions and lighting variations in the captured images, etc. However, such an evaluation is beyond the scope of this paper.

6. CONCLUSION

This paper has presented a tracking system that can detect and track a facade in a video sequence. The main contribution of this work is the use of parallax and topological information, namely the collinearity and sidedness constraints, to increase the overall robustness of the facade detection.

This tracking system can be very useful for an Anywhere Augmentation AR system, operating in an urban environment. To start operating, it only needs two reference images of a facade, taken on-the-spot.

Several types of facades can be detected and tracked, provided that they have a dominant planar surface and enough visual landmarks. Facades having many landmarks that are non-coplanar with its dominant plane or a large number of visually similar landmarks and those suffering from small occlusions were successfully handled.

Future work will be focused on two main developments: first the tracking of several facades and, as a generalization, the tracking of a building considered as an inter-related group of facades; second, the use of all the detected feature points in a facade, including the, presently discarded, non-coplanar ones. Additionally, a more complete and precise evaluation of the proposed tracking system will be done.

7. REFERENCES

[Art09] <http://www.hitl.washington.edu/artoolkit>

[Bar01] Barfield, W. and Caudell, T. Basic Concepts in Wearable Computers and Augmented Reality. Fundamentals of Wearable Computers and Augmented Reality, LEA Pub., pp.3-26, 2001.

[Bas08] Bastos, R. and Dias, J.M.S. Automatic Camera Pose Initialization, using Scale, Rotation and Luminance Invariant Natural Feature Tracking. WSCG' 08, pp.97-104, 2008.

[Bay06] Bay, H., Tuytelaars, T. and Gool, L.V. SURF: Speeded Up Robust Features. 9th ECCV, pp.404-417, 2006.

[Bei97] Beis, J.S. and Lowe, D.G.. Shape Indexing Using Approximate Nearest-Neighbor Search in High-Dimensional Spaces. IEEE CVPR, pp.1000-1006, 1997.

[Bou99] Bouguet, J.-Y., Pyramidal Implementation of the Lucas Kanade Feature Tracker Description

of the Algorithm, Intel Corp., Microprocessor Research Labs., 1999.

[Cip99] Cipolla, R., Drummond T. and Robertson, D. Camera Calibration from Vanishing Points in Images of Architectural Scenes. BMVC, pp.382-391, 1999.

[Fer03] Ferrari, V., Tuytelaars, T and Van Gool, L. Wide-baseline Multiple-view Correspondences. IEEE CVPR, pp.718-725, vol.1, 2003.

[Fis81] Fischler, M.A. and Bolles, R.C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. ACM 24, pp.381-395, 1981.

[Fri77] Friedman, J., Bentley J. and Finkel, R. An Algorithm for Finding Best Matches in Logarithmic Expected Time. ACM Trans. Math. Software, vol.3, is.3, pp.209-226, 1977.

[Har88] Harris, C. and Stephens, M. A Combined Corner and Edge Detector. 4th Alvey Vision Conference, pp.147-151, 1988.

[Har03] Hartley, R. and Zisserman, A. Multiple View Geometry in Computer Vision (Second Edition). Cambridge University Press, 2003.

[Jia04] Jiang, B., Neuman U. and You, S. A Robust Tracking System for Outdoor Augmented Reality. IEEE VR, pp.27-31, 2004.

[Kat99] Kato, H. and Billinghurst, M. Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. IWAR, pp.85-94, 1999.

[Lep05] Lepetit, V. and Fua, P. Monocular Model-based 3D Tracking of Rigid Objects: A Survey. Foundations and Trends® in Computer Graphics and Vision. Vol. 1, No.1, pp.1-89, 2005.

[Low03] Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. IJCV, pp.91-110, 2003.

[Luc81] Lucas, B. and Kanade, T. An Iterative Image Registration Technique with an Application to Stereo Vision, 7th IJCAI, pp.674-679, 1981.

[Rei06] Reitmayr, G. and Drummond, T.W. Going out: Robust Model-based Tracking for Outdoor Augmented Reality. ISMAR'06, pp.109-118, 2006.

[Rei07] Reitmayr, G. and Drummond, T.W. Initialization for Visual Tracking in Urban Environments. ISMAR'07, pp.161-172, 2007.

[Rol01] Rolland, J.P., Davis, L.D. and Baillot, Y. A Survey of Tracking Technology for Virtual Environments. Fundamentals of Wearable Computers and Augmented Reality, LEA Publishers, pp.67-112, 2001.

[Ros06] Rosten, E. and Drummond, T. Machine learning for high-speed corner detection. 9th ECCV, pp.430-443, 2006.

[Sim02] Simon, G. and Berger, M-O. Reconstructing While Registering: a Novel Approach for Markerless Augmented Reality. ISMAR'02, pp.285-293, 2002.

[Xu08] Xu, K., Chia, K.W. and Cheok, A.D. Real-time Camera Tracking for Marker-less and Unprepared Augmented Reality Environments. Image and Vision Computing, vol.26, is.5, pp.673-689, 2008.

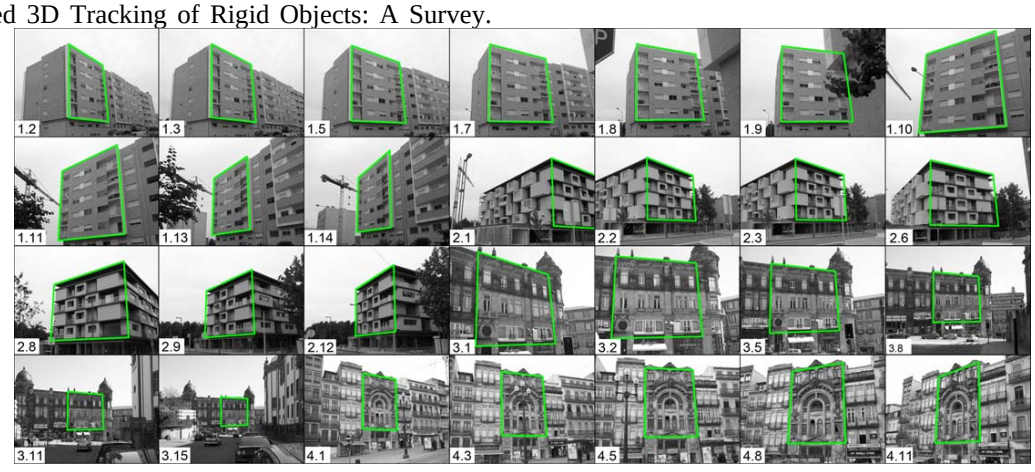


Figure 8. Samples of some of the image sequences used for the evaluation of the detection subphase.

Figure 9. Frames of the augmented video of a facade used in the evaluation of the working phase.

3D non-rigid point cloud based surface registration based on mean shift

Thomas Fabry Dirk Smeets Dirk Vandermeulen Paul Suetens
K.U.Leuven, Faculty of Engineering, Department of Electrical Engineering,
Center for Processing Speech and Images, Medical Imaging Research Center
thomas.fabry@uz.kuleuven.be

ABSTRACT

In this article, we present a new 3D non-rigid surface registration algorithm for unstructured point clouds. The algorithm has very low data requirements and can easily be adapted to use additional information other than vertex positions such as texture information or prior knowledge about local deformations. It is robust to noise, outliers and to some extent to missing data. The mathematical theory is based on probability density estimation. Furthermore, we use the mean shift formula for fast computation. The algorithm is able to use different regularisation models for the deformation. Quantitative and qualitative experiments are conducted on artificial surfaces and on real 3D face data.

Keywords: 3D non-rigid Registration, Mean Shift, Probability Density Estimation

1 INTRODUCTION

1.1 3D shape registration

Registration, defined as finding a mapping between two (3D) images or other geometric data structures, is an important computer vision research topic with many applications, such as 2D automatic panorama stitching [7] or object detection in 2D camera surveillance [2]. In medical imaging, registration is of key importance to combine the complementary properties of different imaging modalities like CT (Computed Tomography) and MRI (Magnetic Resonance Imaging) [18], for comparing medical images of the same person but on different time points (where tumors might have grown or shrunk, contrast has been injected [16]...), or for the alignment of intra-operative surfaces to pre-operative images.

The subfield of registration we are concerned with in this paper is 3D surface registration: finding a mapping between two 3D shapes, in our case represented as point clouds. As 3D capturing devices become less expensive and prove to be of value for face recognition, registration of 3D surfaces also becomes important. It is a problem of considerable interest in, e.g., computer graphics, 3D shape aquisition and reconstruction and statistical shape analysis in computer vision and medical imaging. For instance, the morphable statistical face model of Blanz and Vetter [6] is a method based on 3D face scans. For this method, all the faces have to be represented by one topologically consistent mesh configuration, meaning that all faces have to be represented by a mesh with a constant number of vertices and triangles. As this cannot be done at capturing time, a 3D surface registration is needed.

1.2 Related work

Most of the 3D registration literature covers rigid registration, in which only translations and rotations can be dealt with (a transformation is called rigid if the Euclidean distance between any two points remains constant). The most popular example is the Iterative Closest Point (ICP) algorithm of Besl and McKay [5], which has shown to be very successful and for which many variants have been developed. In a nutshell, ICP is a two-step iterative algorithm. In step one, the closest points between two matches are sought, and in the second step, the rigid transformation parameters that best account for the ensemble of these correspondences are calculated, after which this transformation is applied. These two substeps are repeated until convergence.

The problem of non-rigid registration, in which more general deformations besides translations and rotations are admissible, is far more difficult. Recently, a number of 3D non-rigid registration algorithms have been presented in the literature. Most of these algorithms are based on the ICP-algorithm. One example is the optimal step non-rigid ICP algorithm of Amberg et al. [1] in which the ICP framework is extended to non-rigid registration, while keeping the advantageous properties of the original ICP (Iterative Closest Point) algorithm [5]. It is robust to missing data, but as it is based on the ICP framework, it is not expected to be robust to outliers.

Another popular non-rigid registration algorithm related to ICP is Robust Point Matching introduced by Chui and Rangarayan in [9]. RPM is also a two-step iterative algorithm. The first step is the calculation of

a soft-assign (fuzzy) correspondence matrix including outlier handling. In the second step, the transformation that parameterizes the non-rigid registration is updated by solving a least-squares problem including a specific non-rigid deformation parameterization (more specifically, the *Thin Plate Spline* deformation model is chosen). The iterations are governed by an annealing scheme where the annealing parameter controls the fuzziness of the correspondences computed in the first step.

In the same paper ([9]), also another algorithm is developed and used as baseline: a non-rigid *Thin Plate Spline* (TPS) ICP algorithm. Here, at each iteration, *hard* point correspondences are calculated and then TPS transformation parameters are optimized. The hard point correspondences are found using the nearest neighbor heuristic. To handle outliers, points too far away are not considered. The second step is the calculation of the transformation parameters for the TPS transformation by minimizing

$$E_{TPS}(f) = \sum_i ||y_i - f(x_i)||^2 + \lambda \iint \left[(\frac{\partial^2 f}{\partial x^2})^2 + 2(\frac{\partial^2 f}{\partial x \partial y})^2 + (\frac{\partial^2 f}{\partial y^2})^2 \right] \mathrm{d}x\mathrm{d}y, \quad (1)$$

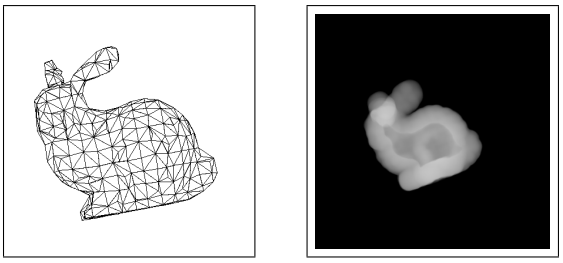
with $f(x_i) = x_i \cdot d + \phi(x_i) \cdot w$ and $\phi(x_i) = ||x - x_i||^2 log||x - x_i||$ the TPS kernel, d a matrix representing the affine transformation and w a warping coefficient matrix representing the non-affine transformation. The parameter λ is set using an annealing procedure.

An algorithm that is not based on ICP but bears close resemblance to ours is the simultaneous non-rigid registration method for multiple point sets for atlas construction of Wang et al. [19]. Here, the 3D model point clouds that need to be registered are represented by a Probability Density Estimate (PDE), which is also a key feature in our algorithm. The algorithm of Wang et al. proceeds by quantifying the distance between these PDE's using an information theoretic measure, which is optimized over the parameters of a deformation model.

2 MEAN SHIFT REGISTRATION

2.1 Probability Density Estimation and Mean Shift

In the majority of 3D surface processing algorithms, 3D objects and surfaces are represented as a mesh: a bunch of vertices (3D points) on the object's surface connected by polygons, mostly triangles. An example of a mesh is shown in figure 1(a). But, as has already been mentioned, we will, in this work, represent 3D surfaces as a Probability Density Estimate (PDE) based on a discrete point cloud version of the surface. These points



(a) (b)

Figure 1: The Stanford bunny represented as (a) a mesh; (b) a volume rendered Probability Density Estimate.

are most readily available and are the most basic representation of surfaces, and are sometimes the only available information. By representing the 3D surface as a PDE, every point in 3D space has a probability of lying on the surface of the 3D object we are working with. This statistical way of representing 3D objects will – to a certain amount – give our algorithm the ability to deal with missing data and outliers. An example of a 3D PDE can be seen in figure 1(b).

We construct the PDE of the 3D surface by using the Kernel Density Estimation (KDE) technique. The PDE is thus constructed in the following way:

$$\hat{p}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x_i}), \quad (2)$$

where $\hat{p}$ is the PDE itself, $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$ are the 3D points of the discretized surface, and $K(\mathbf{x})$ is a scalar function satisfying

$$\int_{\mathbb{R}^3} \mathbf{x}\mathbf{x}^T K(\mathbf{x}) \mathrm{d}\mathbf{x} = c_K I, \quad (3)$$

$$\int_{\mathbb{R}^3} \mathbf{x} K(\mathbf{x}) \mathrm{d}\mathbf{x} = 0, \quad (4)$$

$$\lim_{\|\mathbf{x}\| \to \infty} \|\mathbf{x}\|^3 K(\mathbf{x}) = 0, \quad (5)$$

$$\int_{\mathbb{R}^3} K(\mathbf{x}) \mathrm{d}\mathbf{x} = 1. \quad (6)$$

Here, $\|\cdot\|$ is the Euclidean norm, and c_K is a constant. We follow the method of Comaniciu and Meer [10] for Kernel Density Estimation and are thus only interested in the special class of radially symmetric kernels satisfying $K(\mathbf{x}) = c_{k,d} k(\|\mathbf{x}\|^2)$ with $c_{k,d}$ a normalization constant to make $K(\mathbf{x})$ integrate to one (d stands for the dimensionality of the problem). $k(\cdot)$ is called the profile of the kernel. Using a kernel with bandwidth h, the kernel density estimate is:

$$\hat{p}(\mathbf{x}) = \frac{c_{k,d}}{nh^d} \sum_{i=1}^{n} k(||\frac{\mathbf{x} - \mathbf{x_i}}{h}||^2). \quad (7)$$

If we define $g(\mathbf{x}) = -k'(\mathbf{x})$, we can write the density gradient estimate as:

$$\hat{\nabla} p_(\mathbf{x}) = \frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} \mathbf{x_i} g(||\frac{\mathbf{x}-\mathbf{x_i}}{h}||^2) \right] \cdot \left[\frac{\sum_{i=1}^{n} \mathbf{x_i} g(||\frac{\mathbf{x}-\mathbf{x_i}}{h}||^2)}{\sum_{i=1}^{n} g(||\frac{\mathbf{x}-\mathbf{x_i}}{h}||^2)} - \mathbf{x} \right] \quad (8)$$

The last factor is called the *mean shift*:

$$m_h(\mathbf{x}) = \frac{\sum_{i=1}^{n} \mathbf{x_i} g(||\frac{\mathbf{x}-\mathbf{x_i}}{h}||^2)}{\sum_{i=1}^{n} g(||\frac{\mathbf{x}-\mathbf{x_i}}{h}||^2)} - \mathbf{x}. \quad (9)$$

Because this can be written as

$$m_h(\mathbf{x}) = \frac{1}{2} h^2 c \frac{\hat{\nabla} p_{h,K}(\mathbf{x})}{\hat{p}_{h,G}(\mathbf{x})}, \quad (10)$$

it can be seen that, at location $\mathbf{x}$, the mean shift is proportional to the density gradient estimate with kernel K, normalized with the density estimate computed with kernel G, and thus always points at the direction of maximum increase in density.

Because of the normalization, the mean shift can be thought of as an adaptive gradient estimation for use in gradient ascent methods. The mean shift is adaptive in the way that it is bigger in regions with low density, and smaller in regions with high density. This is what we will need for our registration algorithm.

In this paper, we will use the Gaussian kernel function $G(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}} \exp(-\frac{1}{2}\|\frac{\mathbf{x}}{h}\|^2)$, leading to:

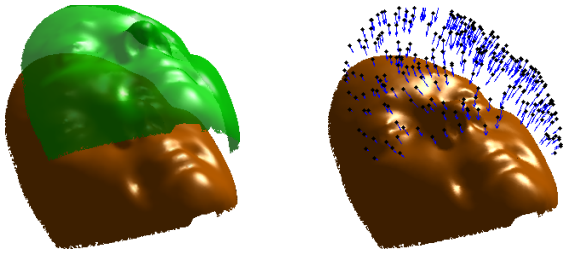
$$m_{h,G}(\mathbf{x}) = \frac{\sum_{i=1}^{n} \mathbf{x_i} \exp(||\frac{\mathbf{x}-\mathbf{x_i}}{h}||^2)}{\sum_{i=1}^{n} \exp(||\frac{\mathbf{x}-\mathbf{x_i}}{h}||^2)} - \mathbf{x}. \quad (11)$$

It should be noticed that this theory and the associated algorithms are also valid for other kernel functions.

2.2 Rigid mean shift registration

If the PDE of interest represents a 3D surface, the mean shift vector will always point in the direction that needs to be followed to get closer to the surface. This is exactly what we need for a registration algorithm. Also, as can be seen from equation (11), the mean shift vector in every point $\mathbf{x}$ is not only dependent on the nearest point $\mathbf{x_i}$ in the (target) point cloud, but every point in the (target) point cloud, weighted with its distance to point $\mathbf{x}$, is taken into account. This will assure the robustness of our method against noise and outliers.

As a start, the two objects that need to be registered are assumed to lie in the same 3D Euclidean space. From now on, we will talk about the *target surface*, the static surface to which the other surface, the moving *floating surface*, has to be registered. We can now for every point $\mathbf{x}$ in the floating point cloud compute the mean shift vector $m_{h,G}(\mathbf{x})$ in the PDE of the target surface by using equation (11). This is illustrated in figure 2.



(a) (b)

Figure 2: (a) The floating and the target surface in the same Euclidean space; (b) the mean shift vectors of the target surface, evaluated in the points of the floating surface (subsampled for clarity).

After the calculation of the mean shift vectors, we need to transform these vectors into one rigid transformation for the whole 3D model. We do this by calculating the optimal rigid transformation matrix that approximates these vectors in a least squares sense and applying it. This is the same procedure as used in the traditional ICP algorithm.

These two steps, the calculation of the mean shift vectors and the computation of the optimal rigid transformation matrix, are then iterated, until convergence.

The algorithm can thus be summarized as follows:

```
REPEAT
 - Calculate mean-shift vectors at the
    points of the floating surface.
 - Calculate the optimal rigid
    transformation matrix and
    apply it.
UNTIL convergence
```

An example of a Mean Shift Rigid Registration between two different facial surfaces can be seen in figure 3 .

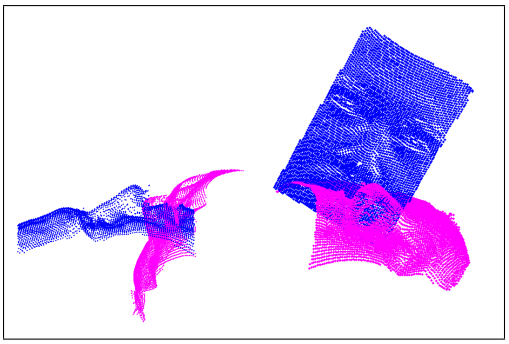
2.3 Non-rigid mean shift registration

The classical non-rigid registration algorithm scheme is a two-step scheme very similar to the rigid registration: first, compute a displacement vector for every point in the floating surface, and then optimize the parameters of a non-rigid transformation model with relation to these displacement vectors.

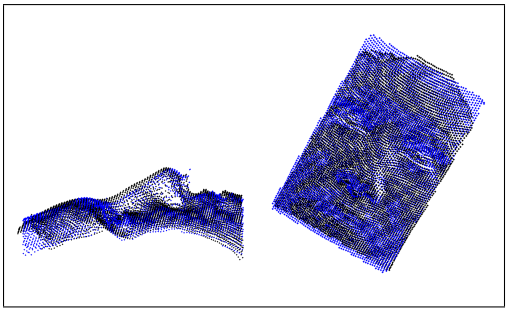
We on the other hand use the mean shift vectors as local estimates of a non-parametric deformation, and in such a way we don't need any explicit deformation model . These displacement vectors are then regularized using smoothing or quasi-interpolation and scaling before they are applied to the points of the floating surface.

For the regularization, we implemented two different strategies, but of course there are a plethora of other possibilities.

Radial Basis Function regularization The first strategy is *quasi-interpolation with radial basis functions*



(a) before registration



(b) after registration

Figure 3: A Sample Mean Shift Rigid Registration of two different faces.

(RBFs). A radial basis function interpolant $s(\mathbf{x})$ is a function of the form:

$$s(\mathbf{x}) = \sum_{i=1}^{n} \mu_i \Phi(||\mathbf{x} - \mathbf{x_i}||) - p(\mathbf{x}), \qquad (12)$$

where $\mathbf{x_i}$ are the interpolation data points, $\Phi(\mathbf{x})$ are radially symmetric basis functions, $p(\mathbf{x})$ is a low degree polynomial and μ_i are the unknown RBF coefficients. For interpolation, the RBF coefficients μ_i are computed assuring

$$s(\mathbf{x_i}) = m_i, \quad i = 1, \ldots, N, \qquad (13)$$

with m_i the values at the data points $\mathbf{x_i}$. For quasi-interpolation, the RBF coefficients can be determined by minimizing

$$\rho||s||^2 + \frac{1}{N}\sum_{i=1}^{N}(s(\mathbf{x_i}) - m_i)^2, \qquad (14)$$

where $||s||^2$ denotes the smoothing penalty defined by

$$\|s\|^2 = \int_{\mathbb{R}^2} ((\frac{\partial^2 s(\mathbf{x})}{\partial x^2})^2 + (\frac{\partial^2 s(\mathbf{x})}{\partial y^2})^2 + (\frac{\partial^2 s(\mathbf{x})}{\partial z^2})^2 + 2(\frac{\partial^2 s(\mathbf{x})}{\partial x \partial y})^2 + 2(\frac{\partial^2 s(\mathbf{x})}{\partial x \partial z})^2 + 2(\frac{\partial^2 s(\mathbf{x})}{\partial y \partial z})^2)\mathrm{d}\mathbf{x} \qquad (15)$$

and ρ is a smoothing constant: the smaller ρ, the more the interpolant is passes through the data points $\mathbf{x_i}$, and vice versa. This approach is known as *spline smoothing* [8]. For the regularization of the mean shift vectors using radial basis functions, we make use of three radial basis function interpolants s_x, s_y, s_z: one for each component dimension of the mean shift vector. This means that we have to compute three RBF interpolants;

$$s(\mathbf{x}) = \begin{cases} s_x(\mathbf{x}) = \sum_{i=1}^{n} \mu_{i,x} \Phi(||\mathbf{x} - \mathbf{x_i}||) - p_x(\mathbf{x}) \\ s_y(\mathbf{x}) = \sum_{i=1}^{n} \mu_{i,y} \Phi(||\mathbf{x} - \mathbf{x_i}||) - p_y(\mathbf{x}) \\ s_z(\mathbf{x}) = \sum_{i=1}^{n} \mu_{i,z} \Phi(||\mathbf{x} - \mathbf{x_i}||) - p_z(\mathbf{x}) \end{cases}, \qquad (16)$$

with constraints

$$s(\mathbf{x_i}) = \mathbf{m_{h,i}} = \begin{cases} m_{h,i,x} \\ m_{h,i,y} \\ m_{h,i,z} \end{cases}, \qquad (17)$$

where $\mathbf{m_{h,i}}$ are the mean shift vectors from equation (11). The regularized mean shift vectors are then computed as

$$\tilde{\mathbf{m}}_{\mathbf{h,i}}(\mathbf{x_i}) = \lambda \cdot (s_x(\mathbf{x_i}), s_y(\mathbf{x_i}), s_z(\mathbf{x_i})), \qquad (18)$$

with $\mathbf{x_i}$ the points of the floating point cloud and λ a scaling parameter.

Gaussian smoothing as regularization Another possibility for regularizing the mean shift vectors is using Gaussian smoothing. Because the interpolation points do not lie on a regular grid, we perform Gaussian smoothing using the Gauss transform. Here, the regularized mean shift vectors are computed using (18), with $\mathbf{x_i}$ the points of the floating point cloud, and

$$s(\mathbf{x}) = \begin{cases} \sum_{i=1}^{N} m_{h,i,x} \exp(||\frac{\mathbf{x}-\mathbf{x_i}}{h_s}||^2) \\ \sum_{i=1}^{N} m_{h,i,y} \exp(||\frac{\mathbf{x}-\mathbf{x_i}}{h_s}||^2) \\ \sum_{i=1}^{N} m_{h,i,z} \exp(||\frac{\mathbf{x}-\mathbf{x_i}}{h_s}||^2) \end{cases}. \qquad (19)$$

Here, h_s is the smoothing parameter. The regularized mean shift vector at point $\mathbf{x}$ is thus computed as an inverse distance weighted sum of the mean shifts in all points of the point cloud.

Algorithm The non-rigid registration algorithm can thus be summarized as follows:

```
REPEAT
 - Calculate (weighted) mean-shift
   vectors at the points of the
   floating surface.
 - Regularize and scale the
   displacement vectors.
 - Move the points of the floating
   surface using the computed
   displacement vectors.
UNTIL convergence
```

2.4 Implementation

The algorithm was implemented in MATLAB. Both the mean shift computation and the Gaussian smoothing are essentially a series of Gauss transforms:

$$\sum_{i=1}^{N} q_i \exp(\| \frac{\mathbf{x_j} - \mathbf{x_i}}{h} \|^2), \forall j = 1 \ldots M \qquad (20)$$

When solved naively, and $M = N$ the Gauss transform's complexity is quadratic ($\mathcal{O}(N^2)$). The Fast Gauss Transform (FGT) of Greengard and Strain [14] reduced the complexity of the Gauss Transform to linear ($\mathcal{O}(N)$). The Improved Fast Gauss Transform (IFGT) of Yang et al. [20] has further increased the speed of the Gauss transform, mainly for high dimensional problems.

In another class of fast implementations, the Gauss Transform is computed using a space-partitioning data structure for organizing the point clouds. These methods are often referred to as *kd-tree* methods (short for *k*-dimensional tree methods) [13, 15].

The FIGTree algorithm of Morariu et al. [17] is an implementation that combines various fast algorithms, including a IFGT and *kd*-tree algorithm, automatically selecting the best parameters for each problem. As such, the speed of our registration algorithms can benefit from using this package.

The RBF calculation is of a similar complexity. Conventional methods for RBF calculation, using 3D biharmonic (thin-plate) splines are $\mathcal{O}(N^3)$. Hierarchical and fast multipole methods [3] can reduce this complexity to $\mathcal{O}(N \log N)$. These fast results can for instance be obtained using the FastRBF Toolbox [12].

To increase the robustness of our algorithm, the iterations are done in an annealing or multiscale way. In optimization, annealing is a way of iterative improvement of the cost function by letting a temperature parameter decrease in a controlled way. In our algorithm, the Gauss transform's kernel bandwidth parameter for the mean shift computation is large in the beginning and is decreased over the iterations.

3 EXPERIMENTS

In this section, we present some results for our registration algorithm. In this paper, we will focus on the non-rigid registration. For results of the rigid registration, we refer to [11].

3.1 Proof of concept

To proof the concept of the mean shift based non-rigid registration algorithm, we make use of meshes of superquadric surfaces. Deformed meshes were then created by non-rigidly transforming them using a thin plate spline deformation field and resampling the deformed meshes. Afterwards, the original configuration was estimated by applying the mean shift non-rigid registration algorithm from the deformed (floating) surface to the original (static) surface. To make the challenge fair, since the surfaces were deformed using a TPS deformation field, we used Gaussian smoothing as regularizer.

landmark number	landmark name
1	nose tip
2	left inner eye corner
3	right inner eye corner
4	left outer eye corner
5	right outer eye corner
6	left mouth corner
7	right mouth corner

Table 1: The landmarks used in the registration of intra-subject face scans experiment.

The results of these registrations as illustrated in figure 4, and prove qualitatively that the new non-rigid registration algorithm can lead to good registrations. The non-rigid registration took less then 2 minutes on one cluster node with a dual-core AMD Opteron 2220 processor.

3.2 Registration of intra-subject face scans with face expressions

To test the ability of our algorithm to deal with real-world data, we make use of the Binghamton University 3D Facial Expression (BU-3DFE) Database [21]. This is a database containing 100 subjects, with for each subject 25 scans for different face expressions. In this experiment, we tried to register the neutral face scan of one person to eight of the expression face scans, on which seven anatomical landmarks were manually indicated. A set of face scans with different expressions are visualised in figure 5. To construct reliable statistical models, it is required that the landmark locations of one face instance are mapped as closeby as possible to the landmark locations of the target face, when registered. The result of the non-rigid registration can then be validated by the statistical distribution of the Euclidean distances of the landmarks on the registered face to the real landmark locations:

$$d = \sqrt{(l_{o,x} - l_{r,x}) + (l_{o,y} - l_{r,x}) + (l_{o,z} - l_{r,x})}, \qquad (21)$$

where $l_o = l_{o,x}, l_{o,y}, l_{o,z}$ are the landmark locations before registration and $l_r = l_{r,x}, l_{r,y}, l_{r,z}$ are the landmark locations after registration . This is what can be found in figure 6. For comparison, also results of the non-rigid TPS-ICP registration described in the introduction are included.

The landmark numbers are explained in table 1. The experimental results show that the mean shift non-rigid registration algorithm outperforms the TPS-ICP algorithm, especially in regions where large expression-induced shape variations are to be expected, at the mouth corners. For these experiments, we tuned the parameters of both algorithms as well as possible.

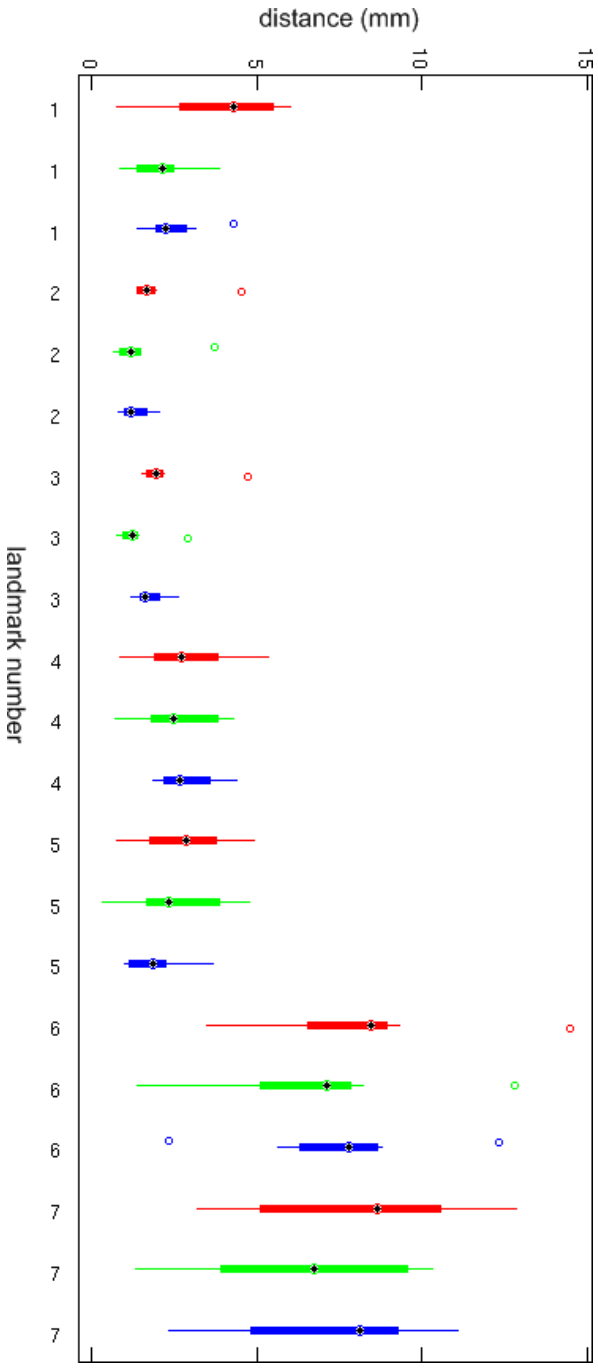


Figure 6: Registration of intra-subject face scans with face expressions. The red bars (1,4,7,10,13,16 & 19) represent the results after only rigid registration, green bars (2,5,8,11,14,17 & 20) indicate results after mean shift non-rigid registration, and blue bars (3,6,9,12,15,18 & 21) represent results after non-rigid ICP registration. (Color version online.)

4 CONCLUSIONS AND FUTURE WORK

We developed a novel non-rigid point cloud based surface registration algorithm based on mean shift. The algorithm shows to be able to perform good non-rigid registrations. Quantitative experiments prove the power of our algorithm.

The algorithm's theory is stated in 3D, and we make only use of the 3D point locations of the discretisized surface representation. The framework can however be expanded to N dimensions. A straightforward application that can benefit from a 4D registration is recognition for 3D facial gestures, which can be seen as a 4D surface [4]. We however plan to use the extendibility to N dimensions for improving the performance of the 3D registration. In its current implementation, the algorithm is expected to be more suitable for expressionless inter-person registration than for intra-person registration where it has to deal with face expressions. We want to tackle this problem by building a N dimensional space consisting of 3 Euclidean shape space dimensions and additional dimensions containing a.o. texture and curvature information.

Furthermore, we plan to review variable kernel methods, to deal with unevenly sampled surfaces and the non-uniform nature of face surfaces. A broad validation of the algorithm on a 3D face database is also planned.

ACKNOWLEDGEMENTS

This work is supported by the Flemish Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT Vlaanderen), the Research Programme of the Fund for Scientific Research - Flanders (Belgium) (FWO) and the Research Fund K.U.Leuven.

REFERENCES

[1] B. Amberg, S. Romdhani, and T. Vetter. Optimal step nonrigid ICP algorithms for surface registration. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007.

[2] C. Anagnostopoulos, T. Alexandropoulos, S. Boutas, V. Loumos, and E. Kayafas. A template-guided approach to vehicle surveillance and access control. In *Advanced Video and Signal Based Surveillance, 2005. AVSS 2005. IEEE Conference on*, pages 534–539, Sept. 2005.

[3] R.K. Beatson and L. Greengard. A short course on fast multipole methods. *Wavelets, multilevel methods and elliptic PDEs*, pages 1–37, 1997.

[4] L. Benedikt, D. Cosker, P.L. Rosin, and D. Marshall. 3D facial gestures in biometrics: from feasibility study to application. In *Biometrics: Theory, Applications and Systems, 2008. BTAS 2008. 2nd IEEE International Conference on*, pages 1–6, 29 2008-Oct. 1 2008.

[5] P. J. Besl and H. D. Mckay. A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256, 1992.

[6] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive*

techniques, pages 187–194, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[7] Matthew Brown and David G. Lowe. Automatic panoramic image stitching using invariant features. *Journal International Journal of Computer Vision*, 74(1):59–73.

[8] J. C. Carr, R. K. Beatson, B. C. McCallum, W. R. Fright, T. J. McLennan, and T. J. Mitchell. Smooth surface reconstruction from noisy range data. In *ACM GRAPHITE*, pages 119–126, Melbourne, Australia, February 2003.

[9] Haili Chui and Anand Rangarajan. A new point matching algorithm for non-rigid registration. *Comput. Vis. Image Underst.*, 89(2-3):114–141, 2003.

[10] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, May 2002.

[11] Thomas Fabry, Dirk Vandermeulen, and Paul Suetens. 3D face recognition using point cloud kernel correlation. In *BTAS '08: Proceedings of the IEEE Second International Conference on Biometrics Theory, Applications and Systems*, Arlington, Virginia, USA, September 2008.

[12] FarField Technology. *FastRBF MATLAB Toolbox Manual*, August 2004.

[13] A.G. Gray and A.W. Moore. Nonparametric density estimation: Toward computational tractability. In *SIAM Data Mining*, 2003.

[14] L. Greengard. The fast Gauss transform. *SIAM J. Sci. Stat. Comput.*, 12(1):79–94, 1991.

[15] Dongryeol Lee, Alexander Gray, and Andrew Moore. Dual-tree fast gauss transforms. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 747–754. MIT Press, Cambridge, MA, 2006.

[16] Dirk Loeckx, Walter Coudyzer, Frederik Maes, Dirk Vandermeulen, Guido Wilms, Guy Marchal, and Paul Suetens. Nonrigid registration for subtraction CT angiography applied to the carotids and cranial arteries. *Academic Radiology*, 14(12):1562 – 1576, 2007.

[17] Vlad I. Morariu, Balaji Vasan Srinivasan, Vikas C. Raykar, Ramani Duraiswami, and Larry S. Davis. Automatic online tuning for fast gaussian summation. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[18] Pieter Slagmolen, Dirk Loeckx, Sarah Roels, Xavier Geets, Frederik Maes, Karin Haustermans, and Paul Suetens. Nonrigid registration of multitemporal CT and MR images for radiotherapy treatment planning. In *Biomedical Image Registration*, volume 4057/2006, pages 297–305. Publisher Springer Berlin / Heidelberg, 2006.

[19] F. Wang, B.C. Vemuri, A. Rangarajan, and S.J. Eisenschenk. Simultaneous nonrigid registration of multiple point sets and atlas construction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(11):2011–2022, Nov. 2008.

[20] C. Yang, R. Duraiswami, N.A. Gumerov, and L. Davis. Improved fast gauss transform and efficient kernel density estimation. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 664–671 vol.1, Oct. 2003.

[21] Lijun Yin, Xiaozhou Wei, Yi Sun, Jun Wang, and Matthew J. Rosato. A 3d facial expression database for facial behavior research. In *FGR '06: Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition*, pages 211–216, Washington, DC, USA, 2006. IEEE Computer Society.

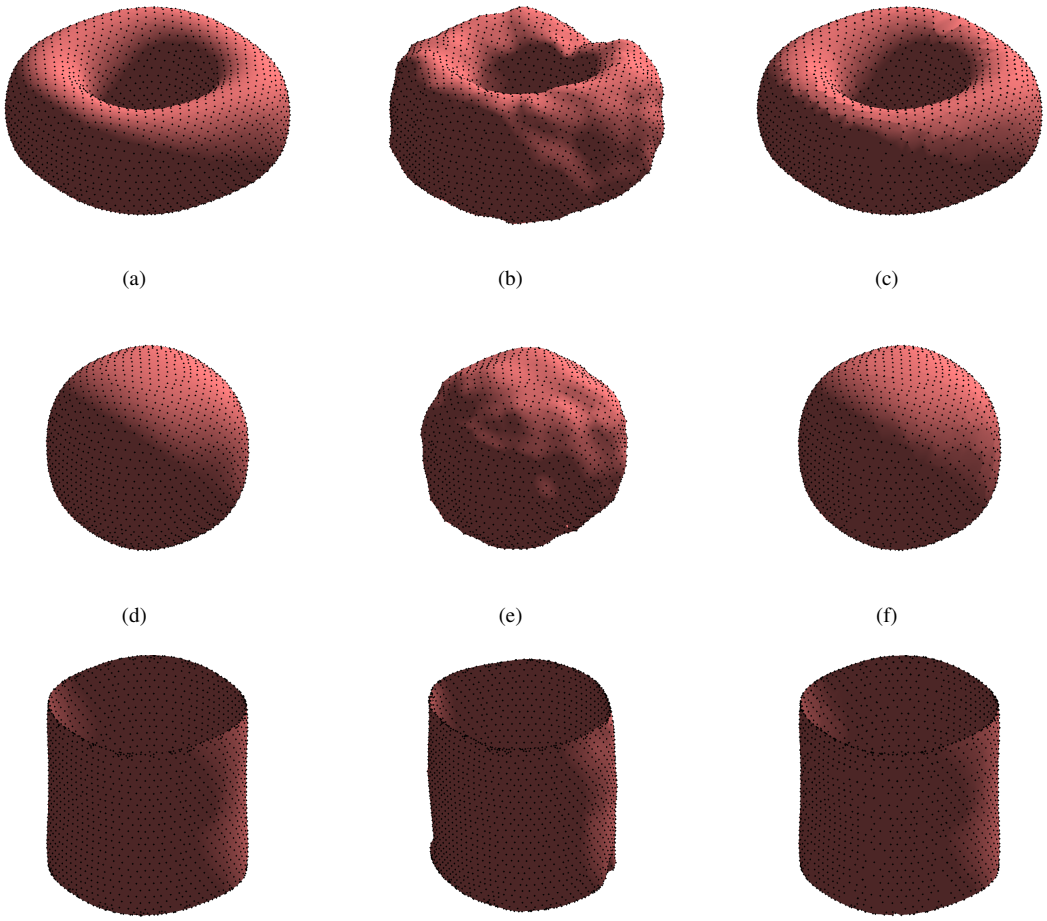


Figure 4: Results of the mean shift non-rigid registration algorithm. (a), (d) and (g) are the original surfaces. (b), (e) and (h) are deformed, noisy version hereof. These were registered to the originals, and the result is shown in (c), (f) and (g). Only the point clouds were used, but for clarity of visualization, the point clouds are displayed as shaded meshes with black dots on the point locations.

Figure 5: Nine face surfaces captured from the same person but with different facial expressions.

Real-Time Gaze Tracking with a Consumer-Grade Video Camera

Andrea Keil
TU Braunschweig, Germany
andrea.keil@cg.cs.tu-bs.de

Georgia Albuquerque
TU Braunschweig, Germany
g.albuquerque@cg.cs.tu-bs.de

Kai Berger
TU Braunschweig, Germany
k.berger@cg.cs.tu-bs.de

Marcus Andreas Magnor
TU Braunschweig, Germany
m.magnor@cg.cs.tu-bs.de

ABSTRACT

Eye gaze can be a rich source of information to identify particular interests of human users. Eye gaze tracking has been largely used in different research areas in the last years, as for example in psychology, visual system design and to leverage the user interaction with computer systems. In this paper, we present an IR-based gaze tracking framework that can be easily coupled to common user applications and allows for real-time gaze estimation. Compared to other gaze tracking systems, our system uses only affordable consumer-grade hardware and still achieves fair accuracy. To evaluate the usability of our gaze tracking system, we performed a user study with persons of different genders and ethnicities.

Keywords: Eye tracking, gaze tracking, human-computer interaction, interactivity, application.

1 INTRODUCTION

For us humans, the eyes may be the most important source of information. Our gaze is the primary sense for experiencing our environment. Recent improvements in computer and image processing technologies have allowed our gaze to influence the environment and become an important branch of Human-Computer Interaction (HCI) via Eye and Gaze Tracking. Selecting icons simply by looking at them is a pleasant idea. This can be continued up to the complete controlling of a computer just by looking at the objects of interest instead of selecting them with the mouse. Moreover, this application can be necessary for handicapped people that could write words on a monitor keyboard by looking at the single characters and achieve interactivity with their environment.

As our gaze reflects our attention and our interests, gaze tracking can be used for perceptive tasks as well. What is the customer of the newly developed software application looking at immediately? Can he find all the functionality he needs, and where is he searching for them? When we look at homepages or online-shops or advertisement, what element is drawing our attention in the first place? All those questions may be answered when gaze can be tracked.

WSCG'2010, February 1 – February 4, 2010
Plzen, Czech Republic.

In this paper, we propose an IR-based gaze tracking system for user applications working in real time with consumer-grade hardware. Gaze tracking systems that use head-mounted devices are usually more accurate and allow free head movement compared to non-intrusive systems (that make no contact with the subject), but such systems are frequently uncomfortable and unnatural to the user. Therefore, we chose developing a non-intrusive gaze tracking system with a fixed camera. Our system consists of a simple setup and does not require expensive cameras.

In the following, we will give a short overview of the related work (see Section 2) and discuss the advantages and the liabilities of our system. In Section 3, our gaze detection pipeline is introduced and each step explained in detail. An application example is presented in Section 3.5 and in Section 4 we present a simple user study that was performed to evaluate the system usability. Finally, we conclude the paper in Section 5 with a short summary and possible future work.

2 RELATED WORK

The most popular technique for gaze tracking is the use of a single camera and IR light. The invisible light produces the corneal reflection depicting a strong reference point. Research efforts in this area are focused on simple calibration, accuracy and non-intrusive everyday execution.

Nahlaoui et al. [ANJKS05] developed a gaze tracking system using two IR point light sources for two reference points. Their field of application is the therapy of visually handicapped people. Zu et al. [ZFJ02] used two IR LED rings on a panel that can be turned on and

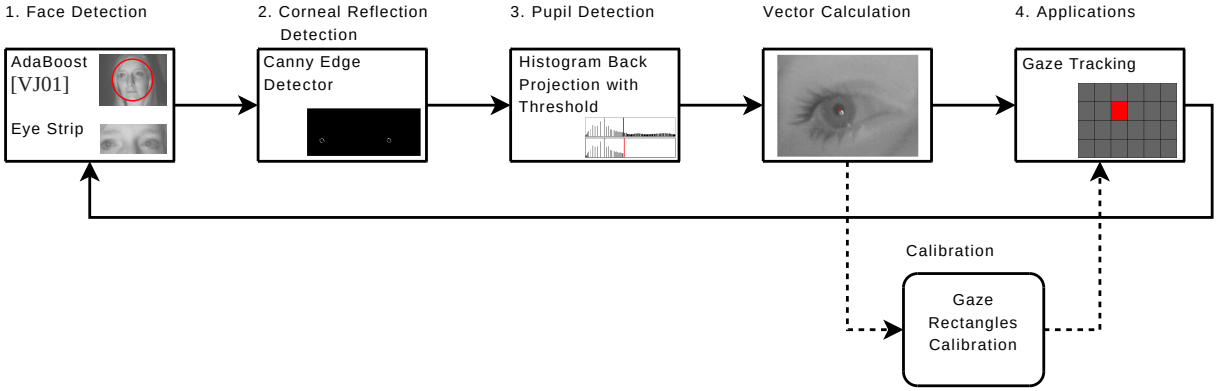


Figure 1: The pipeline of our system.

off alternately in order to receive bright and dark pupil illumination images in the even and odd field of the interlaced image. Other approaches apply a cornea model in order to estimate the gaze [LP07] or use face detection and exploit symmetry to find the gaze direction [MSWB04].

As the IR light shines directly toward the eye, glasses can cause bright extra reflections disturbing or preventing the detection of the corneal reflection. Ebisawa and Ohno et al. [Ebi98, OMY02] developed gaze tracking system that tackles glasses reflections.

A large group of similar approaches is the 3D approach for gaze tracking. They use a stereo camera system for 3D eye localization [OMK03, SL04, SEO96] and derive the 3D center of the corneal curvature in world coordinates. Other works [OM04, PCG+03] additionally use a steerable IR camera on a pan-tilt unit that follows the user so that free head movement is allowed.

Another possibility that allows free head movement is a head-mounted device that is always focused on the user's eye [LWP05, BP04]. Li et al. actually introduced a new pupil detection method called *Starburst* that requires a high eye image resolution, however. This is usually not available when using a standard web-cam.

Morimoto et al. [MAF02] introduced a novel approach for free head gaze tracking, that even works without calibration. This approach is based on the theory of spherical mirrors and the Gullstrand's eye model [Gul09].

In contrast to those proposed methods, our system does not use head-mounted devices because we aimed to develop a non-intrusive gaze tracking system. This decision also supports our goal to create a consumer-grade gaze tracking system that doesn't need fancy additional devices.

We do not use stereo cameras or cameras on a pan-tilt unit as we aimed for low-cost equipment. Instead, we use a single web-cam that is placed on a tripod near the user. The IR LEDs can be put anywhere near the monitor facing towards the user. This guarantees a very easy setup routine.

Furthermore, we apply two dimensional calibration and detection methods instead of 3D approaches where stereo cameras are necessary.

3 GAZE DETECTION PIPELINE

Fig. 1 depicts our **gaze detection pipeline** consisting of four main steps. As input, the pipeline receives an image of an ordinary web-cam adapted with an IR-filter. The first step in the pipeline (Step 1) in Section 3.1 reduces the region of interest performing a face detection in the input image, and in the detected face region, an eye strip is cut out so that further computations only occur in this strip. Step 2 in Section 3.2 is the detection of the corneal reflection, a small white reflection point that is produced by an IR light point source. In Step 3 in Section 3.3, the pupil of the human eye is searched in the direct neighborhood of the corneal reflection. From this two points, the *gaze vector* can be determined and the system is calibrated. Finally, the gaze vector is applied to the screen and the gaze can be estimated as Step 4 in Section 3.5.

3.1 Face Detection

Figure 2: A detected face via AdaBoost [VJ01].

In Step 1, a face is detected in the given input image. We use AdaBoost for this task, the adaptive boosting algorithm based on Haar-like features proposed in [VJ01] and extended in [LM02]. The in OpenCV implemented cascade was taken for our face detection. An example for the detection can be seen in Fig. 2.

Figure 3: The cut out eye strip from the detected face.

According to traditional rules of proportion, dividing the human face into four equal-sized rectangles, 1×4, we can determine that the eyes shall be positioned in the second division. From this division, the eye strip, Fig. 3, is generated. These four divisions are computed individually by the face extent given in each face detection step.

3.2 Corneal Reflection Detection

The *corneal reflection* is the first Purkinje image that is visible when IR light shines toward the human eye. It appears as a bright white spot in the adjacency of the pupil. There are four Purkinje images that describe where in the eye an object is reflected. The first Purkinje image is the reflection from the outer surface of the cornea, the second from the inner cornea's surface. The third Purkinje image is the reflection from the outer surface of the lens and the fourth from the outer lens' surface.

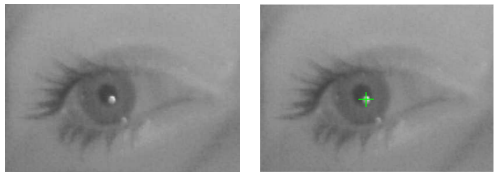
In Step 2, our system uses the Canny edge detector. It applies two thresholds to the eye strip image so that only a small number of white pixels remain in the eye strip image, Fig. 4. They mark edges of the corneal reflection. For the center of the corneal reflection, the centroid of those white pixels is taken, Fig. 5(b).

Figure 4: The output of the Canny edge detector, accentuated. The two corneal reflections produce white pixels in this image.

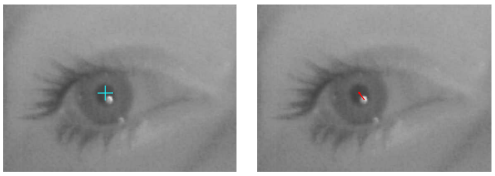
3.3 Pupil Detection

In Step 3, we assume that the pupil is near the corneal reflection. Thus, we search in the neighborhood of the earlier detected point. This is done via histogram back projection and thresholding the histogram. An input image for this method can be seen in Fig 6(a). The corresponding histogram is shown in Fig. 6(c). This histogram is thresholded with a low value and the darkest pixels are kept, Fig. 6(d). The threshold value was determined experimentally.

Via histogram back projection, only the pixels corresponding to the thresholded histogram are back projected into a new image. With this method, the darkest pixels from the input image that derive from the pupil are marked in the new image. As we assume the pupil in the adjacency of the corneal reflection, only pupil pixels are considered, Fig 6(b). Again, the centroid of these marked pixels is taken as the pupil center, Fig. 5(c).



(a) (b)



(c) (d)

Figure 5: Fig 5(a) shows the human eye reflecting IR light as the corneal reflection. In Fig 5(b) and Fig 5(c), a detected corneal reflection and pupil can be seen. A resulting gaze vector is shown in Fig 5(d).

3.4 Calibration Method

Gaze Vector

A gaze vector $\vec{v}_{input_i}$ with $i \in \mathbb{N}$ is computed between the corneal reflection's center and the pupil's center. The corneal reflection center is defined as the origin and the pupil center is defined as the end. If we assume that the human cornea is a perfect sphere and the user's head remains still, the position of the corneal reflection will not change when the user moves his or her eyes. But as the pupil position changes, the relative position between corneal reflection center and pupil center changes, too. This relative position is represented by the gaze vector that can be used to estimate the user's gaze.

An example of a gaze vector can be seen in Fig. 5(d). The calibration method works as follows: The user has to look at the four corners of the monitor. These four calibration vectors $\vec{v}_{calib_1}, \vec{v}_{calib_2}, \vec{v}_{calib_3}$ and $\vec{v}_{calib_4}$, are saved, where:

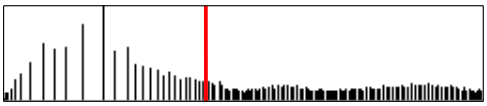
$$\vec{v}_{calib_i} = \begin{pmatrix} c_{i_x} \\ c_{i_y} \end{pmatrix}, 1 \leq i \leq 4, \qquad (1)$$

with $\vec{v}_{calib_1}$ being the topleft calibration vector, $\vec{v}_{calib_2}$ being the topright calibration vector, $\vec{v}_{calib_3}$ being the bottomleft calibration vector and $\vec{v}_{calib_4}$ being the bottomright calibration vector.

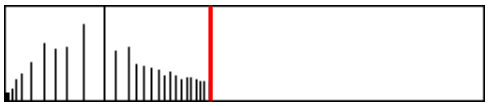
Although we are working on consumer-grade equipment that provides no high definition resolution, a short

(a) The input image.

(b) The back projected image based on the thresholded histogram.



(c) The corresponding histogram to the input image. The red line indicates the threshold value.



(d) The thresholded histogram. The red line indicates the threshold value.

Figure 6: The histogram back projection. The input image (Fig 6(a)) and its histogram (Fig 6(c)). It is thresholded and the resulting histogram (Fig. 6(d)) is back projected as a new image (Fig. 6(b)).

gaze vector can still be used in order to achieve fair accuracy. Therefore, a system of *gaze rectangles* is introduced that divides the screen in a given number of rectangles. This is defined as $A[col, row]$.

The rectangle width $rect_{width}$ and height $rect_{height}$ is computed according to the smallest span of x and y coordinates, respectively, as follows:

$$rect_{width} = \begin{cases} \dfrac{|c_{1_x} - c_{2_x}|}{columns}, & |c_{1_x} - c_{2_x}| < |c_{3_x} - c_{4_x}| \\ \dfrac{|c_{3_x} - c_{4_x}|}{columns}, & otherwise, \end{cases} \tag{2}$$

$$rect_{height} = \begin{cases} \dfrac{|c_{1_y} - c_{3_y}|}{rows}, & |c_{1_y} - c_{3_y}| < |c_{2_y} - c_{4_y}| \\ \dfrac{|c_{2_y} - c_{4_y}|}{rows}, & otherwise, \end{cases} \tag{3}$$

with *columns* being the total number of columns to display and *rows* the total number of rows to display. Both values can be determined by the user before starting the system. $columns \times rows$ defines the system's resolution.

3.5 Applications

In Step 4, the computed gaze vector (Section 3.4) is used for the gaze tracking application. The system will highlight only the gaze rectangle the user looks at. In order to determine which gaze rectangle has to be highlighted, the origin of each input vector is translated into the image origin in the upper left of the screen. This is achieved by subtracting the top left calibration vector $\vec{v}_{calib_1}$ from the input vector $\vec{v}_{input_i}$, Fig. 7. In this way, we have all input vectors' origins in the upper left of the screen. This results in new vectors $\vec{v}_{new_i}$.

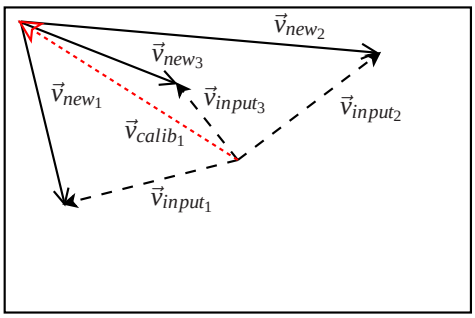


Figure 7: The calibration system that translates every input vector's origin into the upper left corner.

In order to determine the gaze rectangle the user is looking at, defined as $A[col, row]$, Equation 4 and 5 are computed.

$$col = \frac{|input_{i_x} - c_{1_x}|}{rect_{width}}, \tag{4}$$

$$row = \frac{|input_{i_y} - c_{1_y}|}{rect_{height}}, \tag{5}$$

with $i \in \mathbb{N}$, $input_{i_x}$ being the x component of the i-th input vector and $input_{i_y}$ the y component, respectively.

4 RESULTS

Our gaze detection system was developed in a Linux PC with an AMD Athlon 64 X2 Dual Core Processor 4600+. We used the Philips web-cam SPC 900NC with a resolution of 640×480 at a frame rate of 15 fps. It has an IR-sensitive CCD chip and was modified with an IR filter. The camera has a F/2.2 lens with an angular aperture of 55° and a focal length of $4.5mm$. The system is implemented in C++ and the OpenCV library.

In our experiments, a 6×4 gaze rectangle grid is displayed on the screen so that the system has a resolution of 24 gaze regions. This can be seen in Fig. 8.

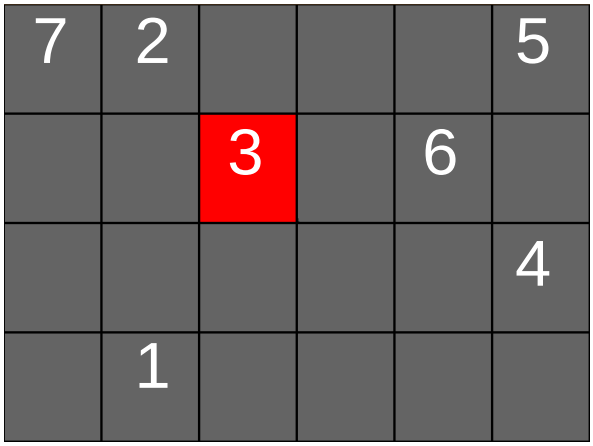


Figure 8: **User Study:** Displayed are seven target rectangles that the user study participants were asked to look at for five seconds. The graph in Fig. 9 shows the mean percentage of hits per target rectangle.

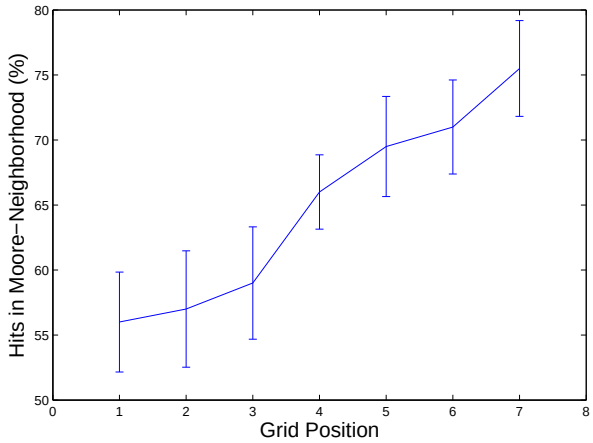


Figure 9: **User Study:** The graph depicts the results of the conducted user study with ten participants. For each of the seven target rectangles (see Fig. 8), the mean percentage of correct hits in the Moore neighborhood is plotted including its variance.

User Study

To evaluate the accuracy of our gaze tracking system, a user study was conducted with ten participants of different gender and different origins. Although neither of them wore glasses, two wore contact lenses. The contestants sat down comfortably in front of the test system. They were asked to place their heads in the camera's field of view and hold their heads still for the test phase. The camera was placed on a tripod in front of the participants and a little to the left. The IR LEDs were positioned under the monitor facing the user. The calibration was performed by looking at the four corner points (Section 3.4). A test phase consisted of seven target gaze rectangles which the contestants were asked to look at. Per rectangle, their eye motion was recorded for five seconds. The target rectangles are shown in Fig. 8 and were located as follows: Two of them were corner gaze rectangles, number 5 and 7, two were located in the center of the screen, number 3 and 6, and three in the periphery, number 1, 2 and 4. These seven target rectangles provide well suited showcase results.

The graph in Fig. 9 depicts the average results of the proposed system. For each rectangle position in the grid, the mean percentage of correct hits in the Moore neighborhood of the target rectangle during the recording time is plotted. The vertical bars depict the variance of the contestants' results for each grid position.

Further results showed that every single targeted gaze rectangle was hit at least once.

Infrared Safety

We are working on images that are taken under IR light that has a wavelength of $780nm$ up to $1400nm$. Therefore, an IRED array is used that consists of two IREDs with a wavelength of $880nm$ and a radiant intensity of $160mW/sr$ each, Fig. 10.

This results in an irradiance to the eye of $0.088mW/cm^2$ at a distance of $60cm$. At a distance of $30cm$, the irradiance is $0.355mW/cm^2$. In [SW80], $10mW/cm^2$ are given as a safe amount of IR light under chronic exposure. In [Lam77], the maximum permissible exposure for 16 minutes up to 8 hours is $0.717mW/cm^2$ using our IRED array.

Both values are well below both recommended safety levels.

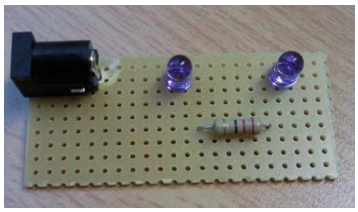


Figure 10: The IRED array we used for our system.

5 CONCLUSION AND FUTURE WORK

We proposed an IR-based gaze tracking system for user applications working in real time with consumer-grade hardware. The user study showed that it achieves fair accuracy. The system uses an easy calibration method that requires only four calibration points.

In future work, the corneal reflection and pupil detection should be improved for a more robust detection. This could be done with adaptive thresholding and more precise detection methods, e.g. ellipse fitting. The next step would be to refine the 6×4 grid if the vector had a much higher resolution. Therefore, we are planning to try other cameras for example with a higher resolution or a better image quality. As the system does not allow head movements, another goal in future work would be to use the face location provided by the face detection in order to compensate head movements.

The calibration method requiring four calibration points could be simplified so that only two calibration points were necessary, namely two opposing corners of the monitor. The gaze rectangle properties could still be computed correctly with information of only two corners. Another calibration method assuming a linear correlation between input gaze vector and output gazing point could be applied as well. This method would require three calibration points for a 2×2 scale and rotation matrix and a two dimensional offset vector.

REFERENCES

[ANJKS05] M.Y. Al Nahlaoui, K. Jostschulte, R. Kays, and J. Schmitz. Kostenguenstiges Eye-Trackingverfahren fuer die ambulante Sehtherapie. In *6. Wuerzburger Medizintechnik Kongress*, pages 1–6, 2005. In German.

[BP04] J.S. Babcock and J.B. Pelz. Building a lightweight eye-tracking headgear. In *Proceedings of the 2004 symposium on Eye tracking research & applications*, page 114. ACM, 2004.

[Ebi98] Y. Ebisawa. Improved video-based eye-gaze detection method. *IEEE Transactions on Instrumentation and Measurement*, 47(4):948–955, 1998.

[Gul09] A. Gullstrand. Appendix II. The optical system of the eye, von Helmholtz H. *Handbuch der Physiologischen Optik*, pages 350–358, 1909.

[Lam77] D. Lamarre. Development of criteria and test methods for eye and face protective devices. Technical report, 1977.

[LM02] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *IEEE ICIP*, volume 1, pages 900–903. Citeseer, 2002.

[LP07] E.C. Lee and K.R. Park. A study on eye gaze estimation method based on cornea model of human eye. *Lecture Notes in Computer Science*, 4418:307, 2007.

[LWP05] D. Li, D. Winfield, and D.J. Parkhurst. Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. In *Proceedings of the IEEE Vision for Human-Computer Interaction Workshop at CVPR*, pages 1–8, 2005.

[MAF02] C.H. Morimoto, A. Amir, and M. Flickner. Free head motion eye gaze tracking without calibration. In *Conference on Human Factors in Computing Systems*, pages 586–587. ACM New York, NY, USA, 2002.

[MSWB04] J.J. Magee, M.R. Scott, B.N. Waber, and M. Betke. Eyekeys: A real-time vision interface based on gaze detection from a low-grade video camera. In *IEEE Workshop on Real-Time Vision for Human-Computer Interaction (RTV4HCI)*, pages 1–8. Citeseer, 2004.

[OM04] T. Ohno and N. Mukawa. A free-head, simple calibration, gaze tracking system that enables gaze-based interaction. In *Proceedings of the 2004 symposium on Eye tracking research & applications*, pages 115–122. ACM New York, NY, USA, 2004.

[OMK03] T. Ohno, N. Mukawa, and S. Kawato. Just blink your eyes: a head-free gaze tracking system. In *Conference on Human Factors in Computing Systems*, pages 950–957. ACM New York, NY, USA, 2003.

[OMY02] T. Ohno, N. Mukawa, and A. Yoshikawa. FreeGaze: a gaze tracking system for everyday gaze interaction. In *Proceedings of the 2002 symposium on Eye tracking research & applications*, pages 125–132. ACM New York, NY, USA, 2002.

[PCG+03] A. Perez, ML Cordoba, A. Garcia, R. Mendez, ML Munoz, JL Pedraza, and F. Sanchez. A precise eye-gaze detection and tracking system. In *11th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, pages 1–4. Citeseer, 2003.

[SEO96] A. Sugioka, Y. Ebisawa, and M. Ohtani. Noncontact video-based eye-gaze detection method allowing large head displacements. In *IEEE Int. Conf. on Medicine and Biology Society*, pages 526–528, 1996.

[SL04] S.W. Shih and J. Liu. A novel approach to 3-D gaze tracking using stereo cameras. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(1):234–245, 2004.

[SW80] D.H. Sliney and M. Wolbarsht. *Safety with lasers and other optical sources: a comprehensive handbook*, page 147. Plenum Press New York, 1980.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. Computer Vision and Pattern Recognition, 2001. CVPR 2001. In *Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages 8–14, 2001.

[ZFJ02] Z. Zhu, K. Fujimura, and Q. Ji. Real-time eye detection and tracking under various light conditions. In *Proceedings of the 2002 symposium on Eye tracking research & applications*, pages 139–144. ACM New York, NY, USA, 2002.

Heuristic Convergence Rate Improvements of the Projected Gauss–Seidel Method for Frictional Contact Problems.

Morten Poulsen
Department of Computer Science,
University of Copenhagen, Denmark
mrtn@diku.dk

Sarah Niebe
Department of Computer Science,
University of Copenhagen, Denmark
niebe@diku.dk

Kenny Erleben
Department of Computer Science,
University of Copenhagen, Denmark
kenny@diku.dk

ABSTRACT

In interactive physical simulation, contact forces are applied to prevent rigid bodies from penetrating and control slipping between bodies. Accurate contact force determination is a computationally hard problem. Thus, in practice one trades accuracy for performance. The result is visual artifacts such as viscous or damped contact response. In this paper, we present heuristics for improving performance for solving contact force problems in interactive rigid body simulation. We formulate the contact force problem as a nonlinear complementarity problem, and discretize the problem using a splitting method and a minimum map reformulation. The resulting model is called the Projected Gauss–Seidel method. Quantitative research results are presented and can be used as a taxonomy for selecting a suitable heuristic when using the Projected Gauss–Seidel method.

Keywords: Nonlinear Complementarity Problem, Contact Forces, Convergence Rate, Projected Gauss–Seidel.

1 SLOW CONVERGENCE RATES

Most open source software for interactive real time rigid body simulation uses the Projected Gauss–Seidel (PGS) method for computing contact forces. This includes the two most popular open source simulators Bullet and Open Dynamics Engine. However, the PGS method is not always satisfactory as it suffers from two major problems: a linear convergence rate [4] and inaccurate friction forces in stacks [9]. The linear convergence rate results in viscous motion at contacts, as well as loss of high frequency effects. The viscous appearance results in a delayed contact response which reduces plausibility [15]. Improving convergence might lead to increased animation quality and higher fidelity. The prospect of improving a state-of-the-art method – the PGS method – has motivated this study of heuristics, aimed at improving convergence.

With this paper, we present a rigorous novel mathematical derivation of the PGS method as well as experience gained from quantitative research results. The results can be used as a taxonomy for selecting a suitable heuristic when using the PGS method.

WSCG'2010, February 1 – February 4, 2010
Plzen, Czech Republic.

2 PREVIOUS WORK

Rigid body simulation was introduced to the graphics community in the late 1980's [8, 12], using penalty based and impulse based approaches to describe physical interactions. Penalty based simulation is not easily adopted to different simulations. Mirtich [11] presented an extended and improved impulse based formulation, however stacking was a problem and it suffered from creeping. These problems has since been rectified [7]. Constraint based simulation [2] has received much attention as an alternative to penalty based and impulse based simulation. Constraint based simulation can be divided into two groups: maximal coordinate and minimal coordinate methods [6]. The focus of this paper is maximal coordinate methods, which are dominated by complementarity formulations. Alternatives to complementarity formulations are based on kinetic energy [10] and motion space [16]. However, the former solves a more general problem and is not attractive for performance reasons, the latter does not include frictional forces.

Complementarity formulations are either acceleration based formulations [20] or velocity based formulations [19]. Acceleration based formulations can not handle collisions [1], in addition they suffer from indeterminacy and inconsistency [18]. Velocity based formulations suffer from none of these problems, for this reason we use a velocity based formulation for this paper. The approach we present here is based on a reformulation of the frictional problem as a nonlinear complementarity problem. This results in a slightly inaccurate model with relatively few variables to solve for. This makes it advantageous in interactive simulations from a performance viewpoint.

The state-of-the-art method for solving complementarity formulations in interactive rigid body simulation, is the projected Gauss–Seidel method. To our knowledge, no mathematical derivation of the PGS method

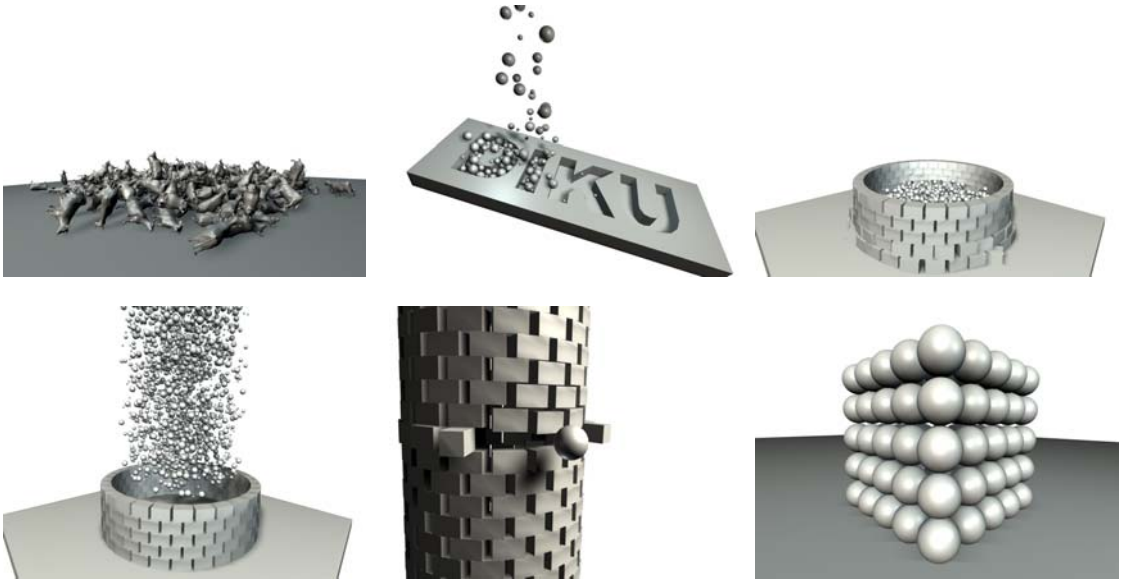


Figure 1: Renderings of selected setups from the test data sets used to examine the performance achieved by using different heuristics for the projected Gauss–Seidel method. The setups varies from 100-10000 interacting rigid bodies in varying degrees of structured configurations.

for interactive rigid body simulation has been presented in the Computer Graphics literature, nor has any studies on its convergence behavior or rates been published.

3 THE NONLINEAR COMPLEMENTARITY PROBLEM FORMULATION

The frictional contact force problem can be stated as a linear complementarity problem (LCP) [19]. However, a slightly different formulation is used in interactive physical simulations, we will derive this formulation. Without loss of generality, we will only consider a single contact point. The focus of this paper is on the contact force model, so the time stepping scheme and matrix layouts are based on the velocity-based formulation in [5]. We have the Newton–Euler equations,

$$\mathbf{M}\mathbf{u} - \mathbf{J}_n^T\lambda_n - \mathbf{J}_t^T\lambda_t = \mathbf{F}, \tag{1}$$

where $\mathbf{J}_n$ is the Jacobian corresponding to normal constraints and $\mathbf{J}_t$ is the Jacobian corresponding to the tangential contact impulses. $\mathbf{M}$ is the generalized mass matrix and $\mathbf{u}$ is the generalized velocity vector. We wish to solve for $\mathbf{u}$ in order to compute a position update. For clarity and readability we have, without loss of generality, abstracted the discretization details within the Lagrange multipliers λ_n, λ_t and generalized external impulses $\mathbf{F}$. Since the contact plane is two dimensional, we span this plane by two orthogonal unit vectors, t_1 and t_2. Any vector in this plane can be written as a linear combination of these two vectors. Thus, $\mathbf{J}_t$ has only two rows corresponding to the two directions. From (1) we can obtain the generalized velocities,

$$\mathbf{u} = \mathbf{M}^{-1}\mathbf{F} + \mathbf{M}^{-1}\mathbf{J}_n^T\lambda_n + \mathbf{M}^{-1}\mathbf{J}_t^T\lambda_t. \tag{2}$$

Let the Lagrange multipliers $\lambda = \begin{bmatrix}\lambda_n & \lambda_t^T\end{bmatrix}^T$ and contact Jacobian $\mathbf{J} = \begin{bmatrix}\mathbf{J}_n & \mathbf{J}_t\end{bmatrix}^T$, then we write the relative contact velocities $\mathbf{y} = \begin{bmatrix}y_n & \mathbf{y}_t^T\end{bmatrix}^T$ such that,

$$\mathbf{y} = \mathbf{J}\mathbf{u} = \underbrace{\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T}_{\mathbf{A}}\lambda + \underbrace{\mathbf{J}\mathbf{M}^{-1}\mathbf{F}}_{\mathbf{b}}. \tag{3}$$

To compute the frictional component of the contact impulse, we need a model of friction. We base our model on Coulomb's friction law. In one dimension, Coulomb's friction law can be written as [2],

$$y < 0 \Rightarrow \lambda_t = \mu\lambda_n, \tag{4a}$$
$$y > 0 \Rightarrow \lambda_t = -\mu\lambda_n, \tag{4b}$$
$$y = 0 \Rightarrow -\mu\lambda_n \leq \lambda_t \leq \mu\lambda_n. \tag{4c}$$

For the full contact problem, we split $\mathbf{y}$ into positive and negative components,

$$\mathbf{y} = \mathbf{y}^+ - \mathbf{y}^-, \tag{5}$$

where

$$\mathbf{y}^+ \geq 0, \quad \mathbf{y}^- \geq 0 \quad \text{and} \quad \left(\mathbf{y}^+\right)^T\left(\mathbf{y}^-\right) = 0. \tag{6}$$

For a frictional contact problem, we define the bounds $-l_t(\lambda) = u_t(\lambda) = \mu\lambda_n$ and for normal impulse $l_n(\lambda) = 0$ and $u_n(\lambda) = \infty$. Combining the bounds with (4), (5) and (6), we reach the final nonlinear complementarity problem (NCP) formulation,

$$\mathbf{y}^+ - \mathbf{y}^- = \mathbf{A}\lambda + \mathbf{b}, \tag{7a}$$
$$\mathbf{y}^+ \geq 0, \tag{7b}$$
$$\mathbf{y}^- \geq 0, \tag{7c}$$
$$u(\lambda) - \lambda \geq 0, \tag{7d}$$
$$\lambda - l(\lambda) \geq 0, \tag{7e}$$
$$\left(\mathbf{y}^+\right)^T (\lambda - l(\lambda)) = 0, \tag{7f}$$
$$\left(\mathbf{y}^-\right)^T (u(\lambda) - \lambda) = 0, \tag{7g}$$
$$\left(\mathbf{y}^+\right)^T \left(\mathbf{y}^-\right) = 0, \tag{7h}$$

where $l(\lambda) = [l_n(\lambda) \quad \mathbf{l}_t(\lambda)]^T$ and $u(\lambda) = [u_n(\lambda) \quad \mathbf{u}_t(\lambda)]^T$. The advantage of the NCP formulation is a much lower memory footprint than the LCP formulation. The disadvantage is solving the friction problem as two decoupled one dimensional Coulomb friction models.

4 THE PROJECTED GAUSS–SEIDEL METHOD

The following is a derivation of the PGS method for solving the frictional contact force problem, stated as the NCP (7). Using a minimum map reformulation, the i^{th} component of (7) can be written as

$$(\mathbf{A}\lambda + \mathbf{b})_i = \mathbf{y}_i^+ - \mathbf{y}_i^-, \tag{8a}$$
$$\min(\lambda_i - l_i, \mathbf{y}_i^+) = 0, \tag{8b}$$
$$\min(u_i - \lambda_i, \mathbf{y}_i^-) = 0. \tag{8c}$$

where $l_i = l_i(\lambda)$ and $u_i = u_i(\lambda)$. Note, when $\mathbf{y}_i^- > 0$ we have $\mathbf{y}_i^+ = 0$ which in turn means that $\lambda_i - l_i \geq 0$. In this case, (8b) is equivalent to

$$\min(\lambda_i - l_i, \mathbf{y}_i^+ - \mathbf{y}_i^-) = -(\mathbf{y}^-)_i. \tag{9}$$

If $\mathbf{y}_i^- = 0$ then $\lambda_i - l_i = 0$ and complementarity constraint (8b) is trivially satisfied. Substituting (9) for $\mathbf{y}_i^-$ in (8c) yields,

$$\min(u_i - \lambda_i, \max(l_i - \lambda_i, -(\mathbf{y}^+ - \mathbf{y}^-)_i)) = 0. \tag{10}$$

This is a more compact reformulation than (7) and eliminates the need for auxiliary variables $\mathbf{y}^+$ and $\mathbf{y}^-$. By adding λ_i we get a fixed point formulation

$$\min(u_i, \max(l_i, \lambda_i - (\mathbf{A}\lambda + \mathbf{b})_i)) = \lambda_i. \tag{11}$$

We introduce the splitting $\mathbf{A} = \mathbf{M} - \mathbf{N}$ and an iteration index k. Then we define $\mathbf{c}^k = \mathbf{b} - \mathbf{N}\lambda^k$, $l^k = l(\lambda^k)$ and $u^k = u(\lambda^k)$. Using this we have

$$\min(u_i^k, \max(l_i^k, (\lambda^{k+1} - \mathbf{M}\lambda^{k+1} - \mathbf{c}^k)_i)) = \lambda_i^{k+1}. \tag{12}$$

When $\lim_{k\to\infty} \lambda^k = \lambda^*$ then (12) is equivalent to (7). Next we perform a case-by-case analysis. Three cases are possible,

$$(\lambda^{k+1} - \mathbf{M}\lambda^{k+1} - \mathbf{c}^k)_i < l_i \Rightarrow \lambda_i^{k+1} = l_i, \tag{13a}$$
$$(\lambda^{k+1} - \mathbf{M}\lambda^{k+1} - \mathbf{c}^k)_i > u_i \Rightarrow \lambda_i^{k+1} = u_i, \tag{13b}$$
$$l_i \leq (\lambda^{k+1} - \mathbf{M}\lambda^{k+1} - \mathbf{c}^k)_i \leq u_i \Rightarrow \lambda_i^{k+1} = (\lambda^{k+1} - \mathbf{M}\lambda^{k+1} - \mathbf{c}^k)_i. \tag{13c}$$

Case (13c) reduces to,

$$(\mathbf{M}\lambda^{k+1})_i = -\mathbf{c}_i^k, \tag{14}$$

which for a suitable choice of $\mathbf{M}$ and back substitution of $\mathbf{c}^k$ gives,

$$\lambda_i^{k+1} = (\mathbf{M}^{-1}(\mathbf{N}\lambda^k - \mathbf{b}))_i. \tag{15}$$

Thus, our iterative splitting method becomes,

$$\min(u_i^k, \max(l_i^k, (\mathbf{M}^{-1}(\mathbf{N}\lambda^k - \mathbf{b}))_i)) = \lambda_i^{k+1}. \tag{16}$$

This is termed a projection method. To realize this, let $\lambda' = \mathbf{M}^{-1}(\mathbf{N}\lambda^k - \mathbf{b})$ then,

$$\lambda^{k+1} = \min(\mathbf{u}^k, \max(\mathbf{l}^k, \lambda')), \tag{17}$$

is the $(k+1)^{\text{th}}$ iterate obtained by projecting the vector λ' onto the box given by $\mathbf{l}^k$ and $\mathbf{u}^k$. Valid splittings of $\mathbf{A}$ are

$$\mathbf{M} = \mathbf{D} \quad \wedge \quad \mathbf{N} = -\mathbf{D} - \mathbf{U}, \tag{18a}$$
$$\mathbf{M} = \mathbf{D} + \mathbf{L} \quad \wedge \quad \mathbf{N} = -\mathbf{U}, \tag{18b}$$
$$\mathbf{M} = \mathbf{D} + \omega\mathbf{L} \quad \wedge \quad \mathbf{N} = (1-\omega)\mathbf{D} - \omega\mathbf{U}, \tag{18c}$$

for $\leq \omega \leq 2$. $\mathbf{L}$, $\mathbf{D}$ and $\mathbf{U}$ are strict lower triangular, diagonal, and strict upper triangular parts of $\mathbf{A}$. These choices results in the projected versions of the Jacobi, Gauss–Seidel and Successive Over Relaxation (SOR) methods respectively. When using the Gauss–Seidel splitting (18b), the resulting PGS method (16) can be efficiently implemented by a forward loop over the components and a component wise projection. Pseudocode for this is,

```
1 : for k = 1 to k_max do
2 :   for i = 1 to n do
3 :     λ'_i ← (−Σ_{j=1}^{i−1} A_{i,j} λ_j − Σ_{j=i+1}^{n} A_{i,j} λ_j − b_i) / A_{i,i}
4 :     λ_i ← min(u_i, max(l_i, λ'_i))
5 :     for all j dependent on i do
6 :       (l_j, u_j) ← update(λ_i)
7 :     next j
8 :   next i
9 : next k
```

To our knowledge no known convergence theorems exist for (16) in the case of variable bounds $l(\lambda)$ and $u(\lambda)$. However, for fixed constant bounds the formulation can be algebraically reduced to that of a LCP formulation. In general, LCP formulations can be shown to have linear convergence rate and unique solutions, when A is symmetric positive definite [4]. However, the **A** matrix equivalent of our frictional contact model is positive symmetric semi definite and uniqueness is no longer guaranteed, but existence of solutions are [4].

5 HEURISTICS FOR CONVERGENCE IMPROVEMENTS

If permutations to the ordering of contacts are made prior to (7) then this is equivalent to use a different sorting order in the splitting (16). Adding a permutation Π to the PGS method alters line 2 of the pseudocode,

$$\begin{aligned} 2a: &\quad \textbf{for } p = 1 \textbf{ to } n \textbf{ do} \\ 2b: &\quad i \leftarrow \Pi(p) \end{aligned}$$

There exists a sorting which yields substantial improvements in convergence behavior [3]. Knowing the optimal sorting heuristic for a given problem will improve overall performance.

We have researched numerous heuristics, measuring improvements in convergence, speedup and accuracy. In an effort not to obfuscate the interesting and important results, we will only present the most interesting or promising heuristics. We have divided the heuristics into three groups, physics-based, geometry-based and splitting-based. To represent physics-based heuristics, we have chosen an impulse propagation permutation (IPP) heuristic. This is interesting as it is based on the impulse-based formulation of rigid body dynamics. From the group of geometry-based heuristics, a coordinate permutation (CP) heuristic is examined. Inspired by [9], the splitting-based heuristics are represented by both an extreme staggered permutation (ESP) heuristic as well as a greedy staggered permutation (GSP) heuristic.

Impulse Propagation Permutations: The PGS method solves the contact problem sequentially, which is numerically similar to sequential impulse propagation methods [11]. However, the basis of the PGS method is a simultaneous contact model, while a sequential model is used for impulse propagation methods. Impulse propagation methods attempt to model the propagation of collision impulses through rigid bodies, so perhaps a similar physical principle is beneficial with the PGS method. The idea is to apply a permutation of the problem, mimicking impulse propagation. We base the permutation on a sorting of the relative contact velocities **y**. The sorting is performed as a preprocessing step, the same sorting is used throughout the PGS method. The permutation could be incrementally updated, possibly improving the convergence. This increases the time complexity of each Gauss–Seidel iteration by $\mathcal{O}(nlgn)$. Experience showed that this was to costly, so we only performed an ascending and decreasing pre-processing sorting. Results are presented for an ascending ordering permutation heuristic.

Coordinate Permutations: Consider computing contact impulses for a stack of boxes, this problem has a distinctly dominating up/down direction. If one solves for contacts from bottommost boxes before uppermost boxes, then the solution may propagate in a bottom-up fashion similar to [5]. This indicates there might be certain directions, where shocks can be propagated and vanish in a single sweep. We sort the contact points by their position along each coordinate axis, considering both ascending and descending ordering. Different choices for coordinate axes has also been examined. We also considered a symmetric blocked approach. Blocking was introduced to make sure normal impulses where solved prior to the dependent friction impulses. This set of heuristics are represented by a descending ordering by y coordinates.

Staggered Permutations: A simple adaption of the staggered approach consists in a permutation such that the normal and frictional impulses are separated into two disjoint sequences. There are two approaches, either one alternates between normal impulse and friction impulse at each iteration or one could perform a number of iterations on one set of impulses before switching to iterating a number of times on the other set of impulses. The number of iterations between the switches is determined by a given rule. The latter principle is adopted for the GSP heuristic. The GSP heuristic computes an error measure for the normal impulses and frictional impulses, and the set with the largest error measure is the next to be iterated over. In some cases this heuristic would iterate over one set of impulses until convergence, before moving on to the other set.

As in [9], the full problem can also be split into two coupled subproblems, where one alternate between solving normal and friction impulses in each iteration. To realize the benefit of this extreme staggered approach, let us first split (3) into two coupled subproblems by partitioning the matrix-vector equation into two coupled equations according to the normal and friction sequences such that,

$$\begin{bmatrix} \mathbf{J}_n^T\mathbf{u} \\ \mathbf{J}_t^T\mathbf{u} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{nn}\lambda_n + \mathbf{A}_{nt}\lambda_t + \mathbf{b}_n \\ \mathbf{A}_{tt}\lambda_t + \mathbf{A}_{tn}\lambda_n + \mathbf{b}_t \end{bmatrix}, \quad (19)$$

where $\mathbf{A}_{ab} = \mathbf{J}_a\mathbf{M}^{-1}\mathbf{J}_b^T$, where $a,b \in n,t$. This problem can be decoupled into two subproblems,

$$\mathbf{J}_n^T\mathbf{u} = [\mathbf{A}_{nn}\lambda_n + \mathbf{A}_{nt}\lambda_t + \mathbf{b}_n] \quad (20)$$

and

$$\mathbf{J}_t^T \mathbf{u} = [\mathbf{A}_{tt}\lambda_t + \mathbf{A}_{tn}\lambda_n + \mathbf{b}_n]. \quad (21)$$

Assuming that λ_t is constant in (20) and λ_n is constant in (21), we can define the constants $\mathbf{b}'_n = \mathbf{A}_{nt}\lambda_t + \mathbf{b}_n$ and $\mathbf{b}'_t = \mathbf{A}_{tn}\lambda_n + \mathbf{b}_t$, who are updated when we change subproblem. Instead of solving one problem $\mathbf{A} \in \mathbb{R}^{n\times n}$, we solve two subproblems $\mathbf{A}_{nn} \in \mathbb{R}^{\frac{n}{3}\times\frac{n}{3}}$ and $\mathbf{A}_{tt} \in \mathbb{R}^{\frac{2n}{3}\times\frac{2n}{3}}$.

6 EXPERIMENTS AND RESULTS

We define a residual function from (10) as,

$$\mathbf{H}(\lambda) = \min(\mathbf{u}(\lambda) - \lambda, -\min(\lambda - \mathbf{l}(\lambda), \mathbf{y})) \quad (22)$$

The natural merit function of (22) is,

$$\Theta(\lambda) = \frac{1}{2}\mathbf{H}(\lambda)^T\mathbf{H}(\lambda), \quad (23)$$

which we use as an absolute error measure for the PGS method. We use Q-convergence measures for our analysis [13]. A comparison of convergence rates is done by visual inspections of logarithmic plots of $(iteration, \log(\Theta))$, observe Figure 2.

The test data has been generated using a number of setups from [14]. The setups uses the negative *z*-axis for the direction of gravity and the *x*-axis and *y*-axis span the horizontal plane. The only exception is the *diku*-setup, where the negative *y*-axis is the direction of gravity. For each setup, **A** has been examined by plotting nonzero elements and eigenvalues. Many setups have a spectral radius $\rho(\mathbf{M}^{-1}\mathbf{N}) \geq 1$, which is cause for concern when considering convergence proofs for the Gauss–Seidel method for linear equation systems. The **A**-matrices are positive semi definite, having a large ratio of zero valued eigenvalues, ranging from (50%-90%) – only the *diku*-setup has no zero valued eigenvalues.

To clarify the experimental results, the test data has been partitioned into equivalency classes [5]. For this paper, only one representative data set from each class is presented.

Non-structured: Few contacts are present, **J** is small.

Loose structured: Large number of contacts without too much structure. This means that normal and tangent directions will be varied. This results in a large **J** with some redundancy.

Dense structured: Large number of contacts with similar normal and tangent directions. This results in a large **J** with a high degree of redundancy. This increases the number of zero valued eigenvalues in **A**.

All tests were performed on a system with Intel Core 2 Duo P8600 2.4GHz CPU and 3GB RAM, running Windows XP SP 3 after a clean boot. In total, 35 heuristics were examined using 10 data sets. The heuristics were implemented and evaluated using MatLab. For each iteration, the error (23) and computation time were measured. Most of the plots exhibit piecewise linear convergence, and seem to converge towards some local minimum of the error function. The time measurements have been used to compute the time speedup

Setup	Type	# Contacts	# Bodies
diku	Non-structured	105	1001
card	Loose structured	154	43
box stack	Dense structured	68	10

Table 1: Sample setups and their complexity.

$$Speedup = \frac{t_{PGS}}{t_{heuristic}}. \quad (24)$$

Table 3 and 2 show the speedups that were significantly different from 1.

Impulse Propagation Permutation: Inspection of Figure 2 shows that the IPP heuristics performs similar to the pure PGS method. There seems to be no benefit from using this heuristic.

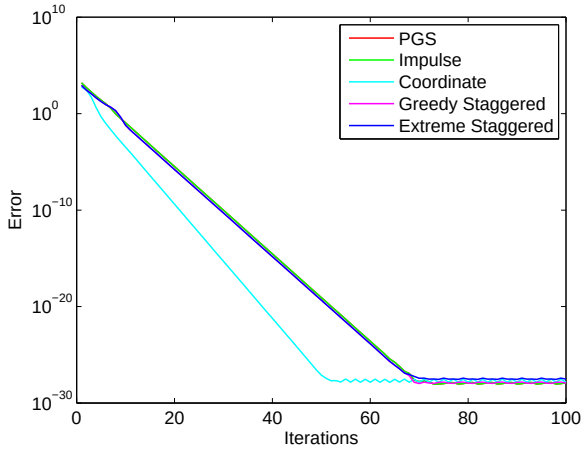
Coordinate Permutation: A number of CP heuristic variations have been investigated sorting by *x*, *y*, and *z* coordinates. The most interesting results arose when sorting descending by *y* coordinate, which is shown on Figure 2. This particular permutation performs well on dense and non-structured setups, but badly on loose structured setups. While not shown, sorting by descending *y* coordinates performs better than sorting by the direction of gravity.

Extreme Staggered Permutation: The ESP heuristic performs similar to the pure PGS method. However, it uses much less time per iteration than the pure PGS method, see Table 2. The speedup is more pronounced for the larger data sets than for the smaller data sets.

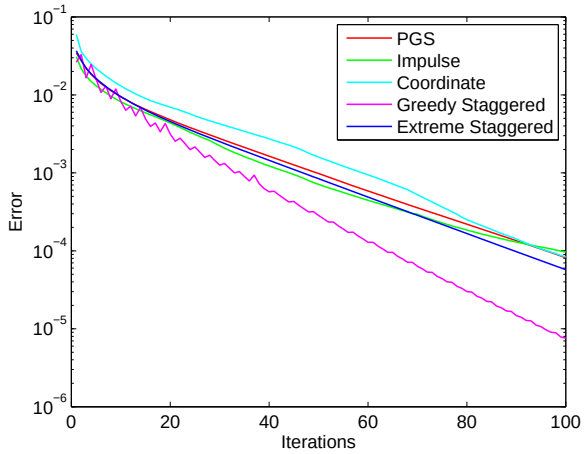
Setup	Type	Time Speedup
diku	Non-structured	1.4
card	Loose structured	2.2
box stack	Dense structured	1.9

Table 2: Speedup measures of the extreme staggered permutation heuristic compared to the pure projected Gauss–Seidel method. Numbers larger than 1 indicate smaller time usage per iteration.

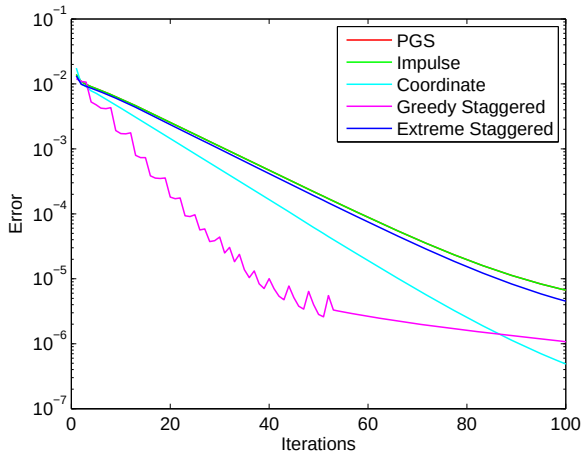
Greedy Staggered Permutation: As Figure 2 shows, the GSP heuristic performs at least as well as the pure PGS method, and often better. When changing from normal iterations to friction iterations (or vice versa), a short burst in the rate of convergence is experienced. This causes the jagged shape of the convergence plots. Another noticeable feature of the GSP heuristic is the speedup. As Table 3 shows, the GSP heuristic is roughly twice as fast as the pure PGS method.



(a) Non-structured setup.



(b) Loose structured setup.



(c) Dense structured setup.

Figure 2: Comparative convergence plots for four different heuristics. Convergence for the PGS method without heuristics is included as a baseline for the comparison. In some cases, PGS is obfuscated by the impulse propagation permutation heuristic.

Setup	Type	Time Speedup
diku	Non-structured	1.7
card	Loose structured	2.0
box stack	Dense structured	2.7

Table 3: Speedup measures of the greedy staggered heuristic compared to the pure projected Gauss–Seidel method. Numbers larger than 1 indicate smaller time usage per iteration.

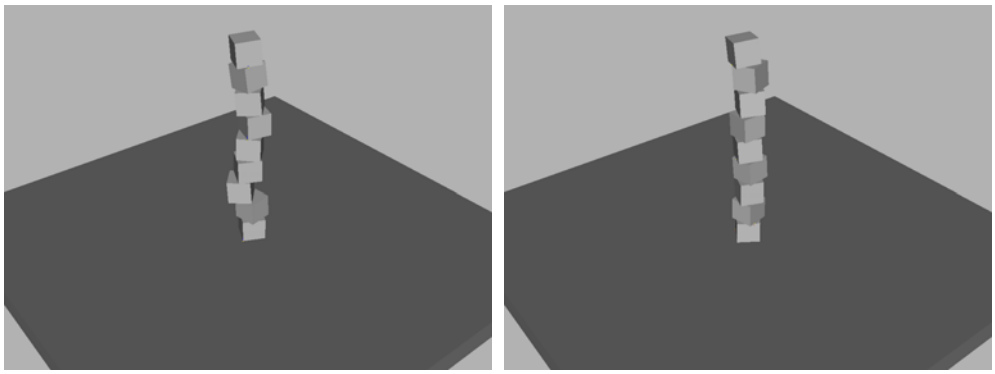
The GSP heuristic has been implemented in OpenTissue [14]. A comparison of the visual improvement over the PGS method, when used on the *box stack*-setup is shown in Figure 3. Note that the box stack is more stable when using the GSP heuristic, compared to using the pure PGS method.

7 CONCLUSION AND DISCUSSION

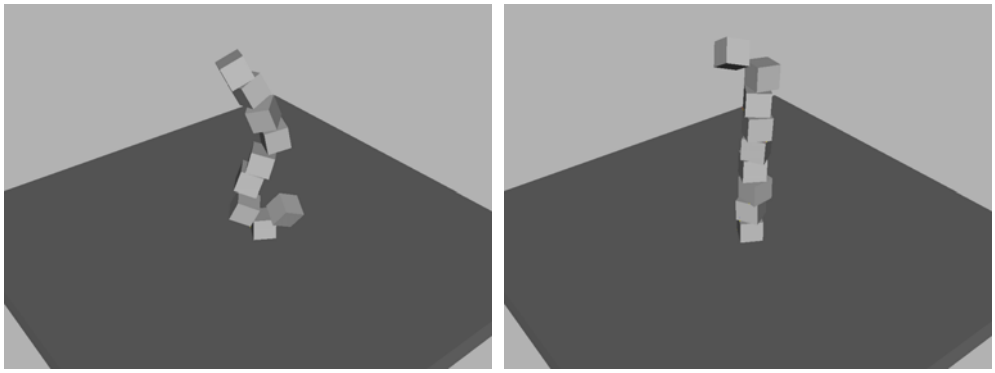
Based on our test data, using a greedy choice for deciding between performing a normal or friction iteration results in an improved convergence rate in most cases, but consistently lower time usage per iteration. The extreme staggered permutation (ESP) heuristic splits the problem into two sub problems, thereby reducing the time usage per iteration.

Sorting by the *y*-axis yields improvements in the rate of convergence. We believe this is due to contact points of each sub sequence being spread over the entire problem. It is not surprising that exploiting geometry information can improve convergence rate. However, the problem lies in obtaining this prior knowledge at run time, unless such information is given at design time. Therefore, as a general heuristic coordinate permutation seem to have only little practical usage. Another interesting result is that any of the symmetric coordinate permutations seems to perform worse than the pure projected Gauss–Seidel (PGS) method.

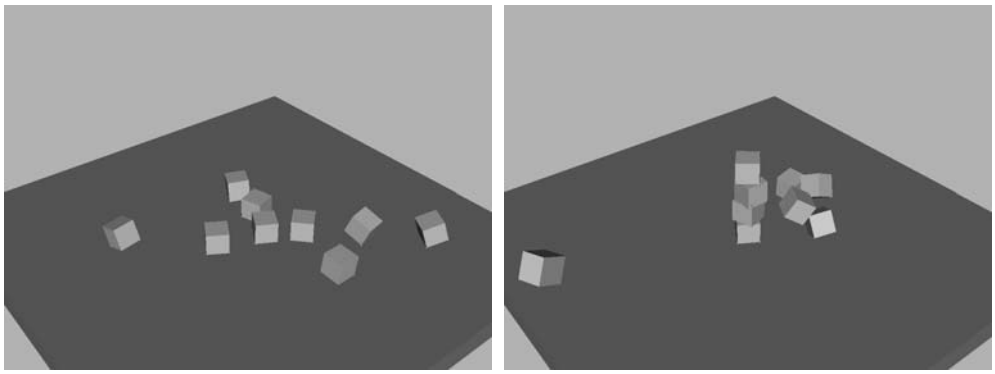
In our opinion, staggered permutations are the most promising heuristics. It is interesting to hypothesize on why this is so. Are their success the result of improved numerical behavior of the method when decoupling normal impulses from friction impulses? Normal impulses control the bounds of the friction impulses, whereas the friction impulses have an indirect, yet significant, effect on normal impulses through the dynamics of multiple contacts. Or is it because the friction model used in the nonlinear complementarity problem (NCP) formulation is too poor a model? Should future work focus on creating improved contact models for interactive simulation that has a better friction model component? Or should future numerical schemes be based on the splitting idea? The results by [17] suggest that convergence rate for the friction component of the NCP formulation is much slower than for the normal component, This indicates to us that the NCP formulation is a poor friction model. In [9] an accurate



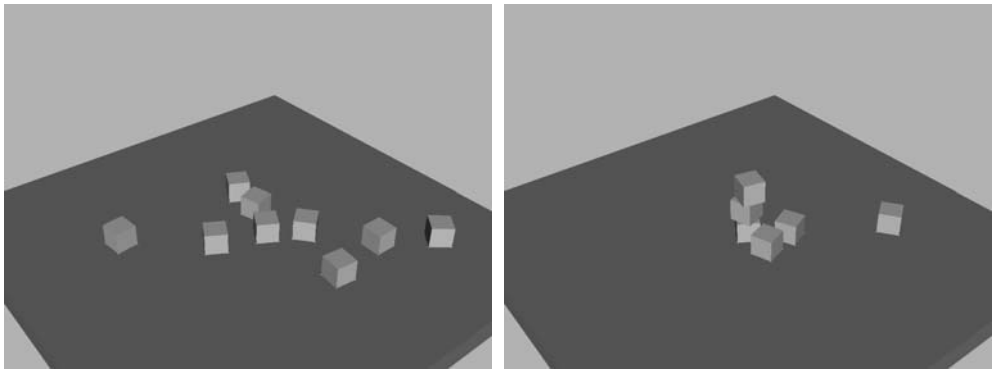
(a) 1.0 secs (b) 1.0 secs



(c) 2.0 secs (d) 2.0 secs



(e) 4.0 secs (f) 4.0 secs



(g) 10.0 secs (h) 10.0 secs

Figure 3: The box stack handled by pure projected Gauss–Seidel on the left and using the greedy staggered heuristic on the right. Note that using the greedy staggered heuristic makes the box stack more stable. After 10 seconds, the bottom three boxes are still stacked when using the GSP heuristic.

linear complementarity problem (LCP) formulation is used to model friction, and they obtain nice results for friction. This again indicates that the current friction model widely used in interactive rigid body simulators is too poor. We will leave these questions open for future work on interactive rigid body simulation.

REFERENCES

[1] Mihai Anitescu and Florian A. Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics. An International Journal of Nonlinear Dynamics and Chaos in Engineering Systems*, 14(3):231–247, 1997.

[2] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 23–34, New York, NY, USA, 1994. ACM.

[3] Richard L. Burden and J. Douglas Faires. *Numerical Analysis*. Books/Cole Publishing Company, 1997.

[4] Richard Cottle, Jong-Shi Pang, and Richard E. Stone. *The Linear Complementarity Problem*. Computer Science and Scientific Computing. Academic Press, February 1992.

[5] Kenny Erleben. Velocity-based shock propagation for multibody dynamics animation. *ACM Trans. Graph.*, 26(2):12, 2007.

[6] Roy Featherstone. *Robot Dynamics Algorithms*. Kluwer Academic Publishers, second printing edition, 1998.

[7] Eran Guendelman, Robert Bridson, and Ronald Fedkiw. Nonconvex rigid bodies with stacking. *ACM Trans. Graph.*, 22(3):871–878, 2003.

[8] James K. Hahn. Realistic animation of rigid bodies. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 299–308, New York, NY, USA, 1988. ACM.

[9] Danny M. Kaufman, Shinjiro Sueda, Doug L. James, and Dinesh K. Pai. Staggered projections for frictional contact in multibody systems. *ACM Trans. Graph.*, 27(5):1–11, 2008.

[10] Victor J. Milenkovic and Harald Schmidl. A fast impulsive contact suite for rigid body simulation. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):189–197, 2004.

[11] Brian Vincent Mirtich. *Impulse-based dynamic simulation of rigid body systems*. PhD thesis, University of California, Berkeley, 1996.

[12] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animation. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 289–298, New York, NY, USA, 1988. ACM.

[13] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer Series in Operations Research. Springer-Verlag, New York, 1999.

[14] OpenTissue. Opensource project, physics-based animation and surgery simulation, www.opentissue.org, 2009.

[15] Carol O'Sullivan, John Dingliana, Thanh Giang, and Mary K. Kaiser. Evaluating the visual fidelity of physically based animations. *ACM Trans. Graph.*, 22(3):527–536, 2003.

[16] Stephane Redon, Abderrahmane Kheddar, and Sabine Coquillart. Gauss least constraints principle and rigid body simulations. In *In proceedings of IEEE International Conference on Robotics and Automation*, 2003.

[17] Morten Silcowitz, Sarah M. Niebe, and Kenny Erleben. Nonsmooth Newton Method for Fischer Function Reformulation of Contact Force Problems for Interactive Rigid Body Simulation. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)*, 2009.

[18] David E. Stewart. Rigid-body dynamics with friction and impact. *SIAM Review*, 42(1):3–39, 2000.

[19] David E. Stewart and Jeff C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal of Numerical Methods in Engineering*, 39(15):2673–2691, 1996.

[20] Jeff C. Trinkle, James Tzitzoutis, and Jong-Shi Pang. Dynamic multi-rigid-body systems with concurrent distributed contacts: Theory and examples. *Philosophical Trans. on Mathematical, Physical, and Engineering Sciences*, 359(1789):2575–2593, December 2001.

A comparative analysis of B-spline deformation models in 3D shape matching

T.R.Langerak
University Medical Center Utrecht
Heidelberglaan 100
3584 CX Utrecht, The Netherlands
robin@isi.uu.nl

Y. Song
Delft University of Technology
Landbergstraat 15
2628 CE Delft, The Netherlands
Y.Song@tudelft.nl

ABSTRACT

Lattice-based B-spline driven deformation has become a useful method in shape matching applications. Here, the challenge is to find a configuration of the B-spline deformation model that effectuates a deformation that spatially aligns one shape (the source) to another shape (the target). Literature study indicates that few B-Spline deformation based algorithms were implemented that target polygonal meshes. In contrast, in the field of medical image registration, B-spline deformation has been extensively applied in matching shapes that use a voxel-based shape representation. For exploring the opportunities of applying these voxel-based methods to the shape matching of polygonal meshes, in this paper we propose to match polygon meshes by transforming them to voxel models and apply established techniques from the field of medical imaging. Two voxel-based methods are selected and implemented: Global Optimization methods, which globally optimize the B-spline model, and Markov Random Field methods, which locally optimize the B-spline model. These methods are compared to parameterized B-spline-based shape matching methods previously proposed by the authors. These methods directly match polygon meshes. Results indicate that the proposed methodology outperforms the parameterized approach in terms of accuracy and computation time and therefore is a promising alternative to existing methods.

Keywords

Shape matching, B-spline deformation, freeform deformation, Markov Random Fields, active shape models.

1. INTRODUCTION

In 1986, Sederberg and Parry introduced the concept of lattice-based B-spline driven deformation that allowed the freeform deformation of (in their case) solid geometric models [Sed86]. B-spline driven freeform deformation makes use of the correspondence between a target shape and an overlying grid of B-Spline control points. A change in the configuration of the B-Spline control points results, through a trivariate Bernstein polynomial function, in a deformation of the underlying shape (see figure 1). B-spline deformation has turned out to be an intuitive method of shape modeling, as the nature of the B-Spline paradigm allows both global and local deformation. Long after its introduction, B-spline driven deformation has been and is being applied in shape modeling applications (e.g. [Igar99], [Karp06], [Song06], [Pern05]). However, there is also another important application of B-spline driven freeform deformation: that of shape matching. In shape matching the challenge is to find a transformation that spatially aligns two shapes. This transformation can be given as a transformation matrix, but it can also be enacted by a matrix of B-Spline control points.

B-Spline driven shape matching techniques have been applied to reverse engineering [Lang07][Ver01] and shape retrieval. In addition, B-spline driven shape matching has found an application in the area of medical image registration [Glo08], [Rue99], [Mattes03]. Here, the main advantage of the B-spline deformation model is that it is successful in mimicking natural tissue deformation and expressing anatomical variance.

When looking closely at the existing literature on B-spline driven shape matching, one cannot fail to notice that shape matching has been given much more attention in the field of medical image registration than it has in the registration of polygonal meshes or solid models.

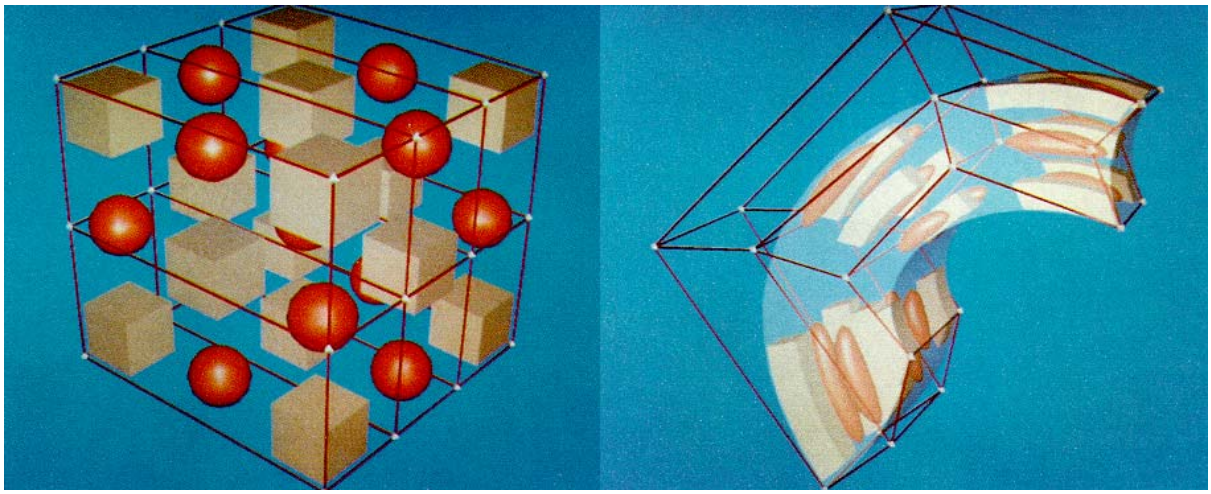


Figure 1: *Example of B-Spline driven freeform deformation (taken from [Sed86])*

Partly, this can be explained by the fact that for voxel models, B-spline driven shape matching is less computationally costly. Because the validity of voxels is easier to maintain (through interpolation) than the validity of polygonal models, computation costs are considerably lower, especially for large and/or complex shapes. In this paper we therefore propose a new technique for matching shapes represented by polygonal meshes (see figure 2).

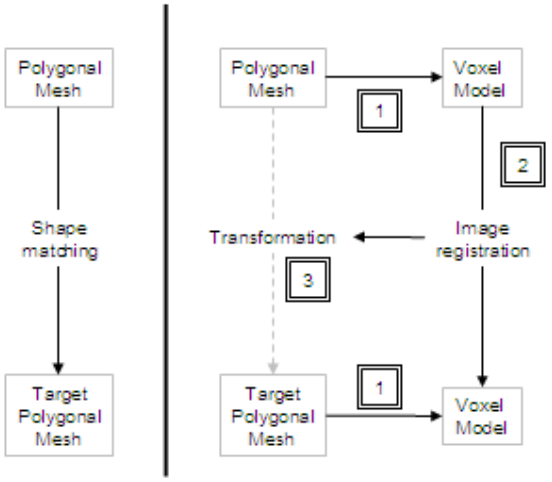


Figure 2: *The direct (left) and the proposed shape matching method (right)*

In a first step, both meshes are converted to a voxel model. Then, using proven techniques from the field of medical image registration, these voxel models are registered. This results in a transformation that can also be applied to the original polygonal mesh to align it with the target mesh, after which only some finetuning is needed. The purpose of this approach is to reduce the computation cost of the shape matching of polygonal meshes, and hopefully increase the accuracy.

In this paper we will investigate two medical image registration methods: global optimization and Markov Random Field minimization. Both will be applied to polygonal models after they have been converted to a voxel model, and the result will be compared to a method that makes use of parametric models for matching polygonal models without converting them to voxel models (i.e. as depicted on the left side of figure 2). This method is an adaptation of a method previously proposed by the authors ([Song05],[Lang07], [Lang08]).

In section 2 we will first define the B-Spline deformation model. In section 3, we will review the three employed shape matching techniques. In section 4 we will compare the performance of all three techniques when applied to four polygonal test models and investigate if the techniques that are taken from the field of medical image registration outperform parametric models. Finally, in section 5 we will draw conclusions and speculate on future research.

2. B-SPLINE DEFORMATION MODELS

The basic principle of the B-spline deformation model is the duality between the parametric coordinates of the model and the spatial coordinates of the shape that is being deformed. The basic elements of a B-spline deformation model are its control points. Although theoretically there is no reason for a specific starting point of the B-spline control points, usually they are assumed to be initially positioned in a uniform lattice, as in figure 1, left. Alternatively, they can be positioned with regard to a previously identified shape [Song05], or in lattices of arbitrary topology [Mac96].

2.1 Definition of the Deformation Model

In three dimensions, the B-spline deformation model $\mathcal{B}$ can be defined as $\mathcal{B}=\{P,d\}$ with P being a set of $l\times m\times n$ control points $P_{ijk}=(x_{ijk},y_{ijk},z_{ijk})$. For *i, j* and *k* it holds that $0\le i<l$ $0\le j<m$ and $0\le k<n$; d is the degree of the B-spline model. Without loss of generality we assume here that B-Splines are uniform and rational, for the reason that most of the methods that are dealt with in this paper have only been described for these types of B-Splines. A NURBS deformation model would also incorporate knot vectors and control point weights and although this makes the B-spline deformation more complex, in an application to shape matching the only relevant difference lies in the dimensionality of the shape similarity function.

A deformation T can be denoted as a function $T(I,B,\Delta)$, where Δ is a set of $l\times m\times n$ vectors that contains, for each control point, a three-dimensional spatial transformation vector $\Delta_{ijk}=(\Delta x_{ijk},\Delta y_{ijk},\Delta z_{ijk})$. The coordinates of the B-spline control points can be expressed both in a (x,y,z) coordinate system and in a (u,v,w) coordinate system. The (x,y,z) coordinates indicate the spatial location of the control point. These coordinates can be directly manipulated manually or automatically, and are therefore the main focus of both shape modeling and shape matching techniques. The (u,v,w) coordinates indicate the position of the control point in the parametric space of the B-spline model and it is an important notion of B-spline deformation that this position is invariant under freeform deformation.

Given the (u,v,w) coordinates of an arbitrary point in an image I, the corresponding (x,y,z) coordinates can easily be found using the trivariate Bernstein polynomial:

$$I(x,y,z)=\sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n}\mu P_{ijk}\text{, where}$$

$$\mu=\binom{l}{i}\binom{m}{j}\binom{n}{k}(1-u)^{l-i}(1-v)^{m-j}(1-w)^{n-k}u^{i}v^{j}w^{k}$$

Then, the deformation of an arbitrary point in the image I can be described as:

$$T(I(x,y,z),\mathcal{B},\Delta)=\sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n}\mu(P_{ijk}+\Delta_{ijk})-I(x,y,z)$$

Unfortunately, (u,v,w) coordinates are not always readily available and must first be found. Given the (x,y,z) coordinates of a point, this can be done by iterative subdivision of the B-spline control point lattice, for example by using one of the many variants of Newton's iteration.

2.2 The Shape Matching Problem

In this paper we will focus on an application of the lattice-based B-spline deformation model to shape matching. Using the definitions given before, this problem can be defined as follows:

Given a source image I_s, a target image I_t, a shape similarity function $\delta(\ )$ and a B-Spline deformation model B, find the set of translation vectors Δ such that $\delta(T(I_s,\mathcal{B},\Delta),I_t)$ is minimal.

In other words, to align two images, coordinates for each control point in the B-spline deformation model must be found such that the resulting deformation minimizes the distance between source and target image. Since there are many strategies to approach this problem, without loss of generality the following assumptions are made in this paper:

1. *Only the source image is deformed*

   In some applications, it may be logical to deform both the fixed and the moving image and end up somewhere in the middle [Yang08], to average the deformation fields of both, or to enforce that both registrations are consistent [Chri01]. Such strategies may be beneficial in cases where one or both of the images contain noise. In general, a deformation of the target image can be approximated as a deformation of the source image through a process of interpolation, but this process may suffer from many off-topic difficulties and for simplicity's sake we therefore assume that deformation only occurs in one direction.

2. *B-spline models are non-hierarchical*

   A common strategy for B-spline driven shape matching uses a multi-resolution approach, in which the resolution of the B-spline control point mesh increases in a step-wise fashion, so that in each step the level of detail of the procedure is increased [Szel97]. By performing global deformations first, and gradually increasing the resolution of the deformation, local deformation becomes less dependent on the global image similarity. However, in each resolution, the formulation of the problem is the same and because our findings can easily be generalized to multiple resolutions, we assume that the shape matching procedure only takes place within a single resolution.

3. *Source and target image have been affinely registered prior to B-spline deformation.*

   In the normal practice of image registration, a rigid or affine transformation is performed before applying a B-spline deformation, to make

sure that scale and position do not have an effect on the B-spline deformation and to reduce the risk of getting stuck in a local minimum of the search space. Similar to multi-resolution registration, this improves the local effects of a B-spline deformation. Because a rigid or affine transformation can in principle be expressed as a strictly constrained B-spline deformation, such a registration does not compromise the validity of our deformation model.

2.3 Regularization

Although B-spline driven deformation is a powerful deformation model that is particularly well-suited to performing local deformations, there is also a risk: because the variables of the shape matching problem are the B-spline control points rather than the underlying image, it may happen that a configuration of the B-spline control point network results in a deformation under which the underlying image violates some validity constraints. The most common violation of image validity is folding, which occurs when two or more control points swap their relative position. Rather than checking the validity of the underlying image during the shape matching procedure, often a regularization mechanism or penalty is included in the optimization routine, such that the problem of shape matching can be reformulated as:

Given a source image I_s*, a target image* I_t*, a shape similarity function* $\delta(\ )$*, a B-Spline deformation model* $\mathcal{B}$ *and a regularization function* ρ*, find the set of translation vectors* Δ *such that* $\delta(T(I_s), I_t) + \rho(\Delta)$ *is minimal*

The difficulty here lies in constructing the regularization function and throughout the literature several options have been proposed.

The most common approach is to relate the regularization function to the elastic energy of the B-spline deformation [Rue99][Kybic03] [Sorz05], such that the translation of a single B-spline control point is penalized depending on the magnitude of the translation vector. In [Hart00] it has been shown that folding can be detected using the Jacobian determinant of the deformation field and in [Rohl03] it has been shown that deviations of this Jacobian can be used to obtain an incompressibility constraint that can be used to guarantee a volume-preserving deformation. Staring et al. [Star06] propose a local rigidity constraint that can be used to penalize large deformation in some areas, but less so in other areas.

3. DEFORMATION METHODS

As the deformation that is enacted by the B-spline model depends on the position of its control points, a logical approach to the optimization problem described in section 2 is to treat the coordinates of these control points as variables of the minimization problem. Although this seems straightforward, there are several approaches to this problem, which will be discussed in this section. This section will review two approaches that are used in the area of medical image registration, and will also discuss a parametric approach that the authors of this paper have successfully applied in the area of reverse engineering. In section 4, we will test all three methods in an application to polygonal meshes to see if they lead to comparable results.

3.1 Global Optimization

The most straightforward way of finding a deformation that minimizes the distance between source and target image is a global optimization approach. A global optimization approach treats all *n* B-spline control point coordinates as if they are coordinates of a global vector Δ, which can be defined as $\Delta = \{\delta x_0, \ldots, \delta y_1, \ldots, \delta z_n\}$, where $n = i \cdot j \cdot k$

This vector can be minimized using any known multi-dimensional function optimization technique. Most approaches use a gradient descent (GD) approach [Rue99][Matt03], in which an iterative approach to the minimization of Δ is taken. In each iteration step, a number of points are sampled in the target image and projected backwards onto the source image. The distance between the sample points and their backwards projection is used to calculate the distance between the source and target image and in the next step the direction of further optimization is chosen to be the direction of steepest descent of the distance function. This method is straightforward but computationally costly because it requires a computation of the gradient of the distance function.

Although gradient descent is the most popular global optimization approach, other (gradient-based) methods are also possible. Klein et al. [Klein07] compare different methods, including gradient descent, quasi-Newton, nonlinear conjugate gradient, simultaneous perturbation, evolution strategy, Robbins-Monro and Kiefer-Wolfowitz. Although small differences between the methods can be observed, it is unclear to what extent these differences are application-independent.

3.2 Markov Random Fields

Another approach to the shape matching problem, in which there is no need to compute the gradient, is to formulate it as a discrete Markov Random Field (MRF) objective function [Glo08]. Basically this boils down to treating the deformation vectors for each point as independent, such that the movement of a single control point is evaluated regardless of the movement of its neighbors. Note that under certain constraints, this approach is similar to Powell's direction set approach to global optimization.

Clearly, such a search covers a much larger part of the search space, involving all the risks and benefits that go along with it. In addition, a MRF approach requires a much more thoroughly defined (and, ideally, an adaptive) regularization term: the independent movement of control points increases the risk of folding and violation of other validity constraints. A primary-dual linear programming approach can be used to quickly compute the optimal MRF [Komo07].

3.3 Parametric Models

Both of the previous methods assume that no information is available when translating the B-spline control points. In other words: these methods look for *any* deformation that is optimal given a certain shape similarity measure: it is not guaranteed, however, that this deformation is valid or that it is the least complex way of deforming the source image. Imagine, for example, the alignment of two 2D pictures of faces: in this case from the point of view of the shape similarity function it is valid to deform the source image in such a way that the eyes are swapped while the rest of the model is kept constant. However, in practice this deformation is invalid and unnecessary and therefore an undesirable result of the shape matching procedure.

In this particular example, it makes much more sense to incorporate knowledge of the shapes in question into the shape matching procedure. Models on the way a shape can be deformed are called Active Shape Models (ASM) or Active Appearance Models (AAM) [Cootes95]. In the application of reverse engineering, a similar concept is the freeform feature model [Song05],[Lang07]. In terms of the definitions given earlier in this paper, these models operate by enforcing a set of displacement vectors $\Delta^{\pi} = \left\{\delta^{\pi} x_0, \ldots, \delta^{\pi} y_1, \ldots, \delta^{\pi} z_n\right\}$, such that the shape matching procedure is reduced to finding the set of scalar values $\Pi = \left\{\pi_0, \ldots, \pi_n\right\}$, such that $\Delta = \left\{\pi_0 \delta^{\pi} x_0, \ldots, \pi_1 \delta^{\pi} y_1, \ldots, \pi_n \delta^{\pi} z_n\right\}$. If Δ^{π} is unknown, then it can be estimated and iteratively maximized using an expectation-maximization approach (as suggested in [Lang08]), but such an estimation is only meaningful if there is indeed a (perceived) parametric difference between source and target image. If this is not the case, the shape matching routine is likely to become over-parameterized. Π can be optimized using global optimization methods or MRF models, but the latter would be difficult to implement because a regularization penalty cannot be directly defined on Π. Song et al. [Song05] use a quasi-Newton approach to match a parameterized shape, or feature, to a target shape. In [Lang07] an evolutionary approach to global optimization is proposed, which is computationally costly but turns out to be successful in traversing the search space in meaningful directions.

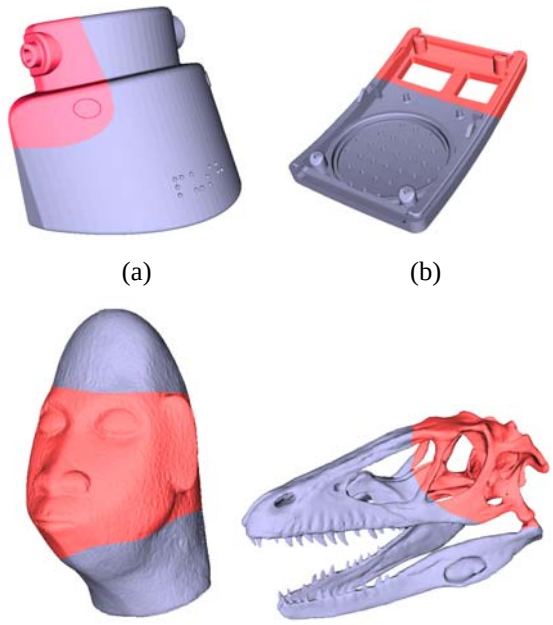
3.4 A Comparative Analysis

The main difference between global optimization and Markov Random Fields lies in two aspects: computational complexity and allowed deformation. Global optimization allows less deformation during the shape matching procedure, but is more stable as a result and converges to a solution in a much smoother way. The fact that the global deformation is taken into account during the optimization means that it is slower, especially so when the gradient is used. In an application to the registration of voxel models, the need for a gradient descent approach is justified by the fact that shape similarity measures are often dependent on other aspects than spatial correspondence.

Markov Random Fields methods have the advantage that they explore much more of the search space, and although this also increases the risk of getting stuck in a local minimum, there are plenty of strategies available in the literature to reduce this risk. That deformation is not looked at from a global perspective allows for fast computation, for example using linear programming. MRF optimization has been reported to be up to four times as fast as global optimization.

4. RESULTS

To test the merit of the method proposed in section 2 we have applied both shape matching using parametric models and shape matching using the two reviewed approaches to the registration of voxel models to four polygonal meshes. These meshes have been selected to pose a variety of problems that are commonly encountered in shape matching problems, specifically varying polygon density and complex topology. In addition, the test cases had several characteristics that can be expected to be a challenge in converting them to voxel models, most importantly the aforementioned varying polygon density and connectivity. The four test models are shown in figure 3. Test models a) and b) are industrial parts that have been obtained from the practice of computer-aided design and that pose various challenges: model a) exhibits a large variety in the level of detail while model b) has a complex topology. Models c) and d) were taken from the internet. Model c) has a straightforward topology but a large number of polygons, due to the texture of the skin of the figure. Model d) poses a challenge because of its topology. In addition, some parts of the model are connected only through thin strips, which may form a challenge when converting the model to a voxel model. In table 1, data for all four models is given.



(a) (b)

(c) (d)

Figure 3: *The four test shapes*

Of each model, a region was selected to serve as the target polygonal mesh (see figure 3). This region was chosen to be smaller than the source image, because this is a requirement of the reviewed image registration methods due to the backwards projection of the sample points used to compute the shape similarity. Table 1 also displays the data for the selected regions, as well as the dimensions of the voxel models that were generated.

Table 1: *Basic data on experiment models*

	Model (a)	*Model (b)*	*Model (c)*	*Model (d)*
Model:				
#polygons	46930	57914	71924	53688
#voxels	53x57x38	112x61x23	62x97x63	72x59x123
Selected Region				
#polygons	6441	2967	48703	6083
#voxels	31x11x33	42x61x23	62x68x63	64x52x48

The original polygonal meshes were chosen as the source images and are denoted as I_s^0 to I_s^3. The selected regions were used as the target images and denoted as I_t^0 to I_t^3.

The proposed method proceeded in the following steps:

1. Noise was added to the target images as follows: a B-spline control point lattice with a resolution of 20x20x20 was constructed around each target image, such that the entire part was contained in the lattice. All B-spline deformation vectors were initialized with a zero displacement vector, to which white Gaussian noise in the domain $[-1,1]$ was added. The resulting deformation was applied to the polygonal meshes and the displacement vectors were recorded as the set of vectors Δ_{noise}.
2. The source images were registered to the target images using a parametric model that was initialized by the normalized vector set Δ_{noise}, such that it indicated the direction but not the magnitude of the deformation that was performed in step 1. Although in practice Δ_{noise} is not known, in this case we felt that using it was justified because no obvious parameterization is available. In section 5 we will argue why this does not affect our findings.
3. Both source and target images were voxelized using the VoxelModeller algorithm implemented in the Visualisation Toolkit (VTK) Library. The voxelization took less than a minute for models a, b, and d, and a little over two minutes for model c.
4. All voxelized source images were registered to the corresponding voxelized target image using both an MRF approach and a global optimization approach, using a 20x20x20 B-spline control point grid.
5. The transformation that resulted from step 4 was applied to the original polygonal meshes.

For the registrations, the following methods were used:

- For the gradient descent registration of the voxel models, we made use of the open source software library Image Registration Toolkit (ITK).
- For the registration of voxel models using Markov Random Fields, we used the DROP package, as described in [Glo08]
- To register the polygonal meshes we used a reimplementation of the method given in [Lang07].

For the experiments with the voxel models, mutual information was used as a shape similarity measure. For the polygonal meshes, the Mean Directed Hausdorff Distance was used. Values for the regularization penalty were experimentally determined. Unfortunately this had to be done for

each test model individually and therefore the values for the regularization penalty could not be compared.

The experiments were done using a computer with four 1.6 GHz processors and 8 GB of RAM memory. Table 2 shows the results of the experiments, where accuracy is measured as the distance in mm. of the source and target image after shape matching.

Table 2: Results of the direct registration

(acc./time)	*GO*	*MRF*	*Par.*
Model (a)	4.8/6min	3.9/7min	3.4/24min
Model (b)	4.5/5min	4.1/6min	3.4/16min
Model (c)	4.8/6min	4.1/6min	3.3/48min
Model (d)	6.2/8min	5.3/8min	2.8/31min

Of the three methods, parameterized registration was the most accurate. This is not surprising, because this method alone made use of additional information regarding the optimal direction of a search for the most optimal deformation. To come to a more fair comparison between the methods, in a post-processing step we applied the parameterized shape matching method to the results of the voxel-based methods until A) the accuracy as given in the rightmost column of table 2 was achieved and B) until convergence (but forced to stop when the time given in table 2 was reached). A) gives an indication of the improvement in computation time that can be achieved and B) gives an impression of the improvement in accuracy that can be achieved. In table 3, the computation times and accuracy including this post-processing step are given. As an example, for model (c), Global Optimization and MRF minimization achieved an accuracy of 3.3 mm. in respectively 25 and 27 minutes, where this took 48 minutes using a parametric approach. In respectively 37 minutes and 44 minutes, an accuracy of 2.9, respectively 3.0 was achieved, compared to the 3.3 using a parametric approach.

Table 3: Results of the post-processing stage

	A		*B*	
	GO	*MRF*	*GO*	*MRF*
Model (a)	3.4/4min	3.4/4min	2.8/12min	2.6/17min
Model (b)	3.4/2min	3.4/3min	3.0/16min	3.1/11min
Model (c)	3.3/19min	3.3/21min	2.9/31min	3.0/38min
Model (d)	2.8/4min	2.8/5min	2.5/19min	2.3/24min

5. CONCLUSION AND DISCUSSION

The purpose of this paper was to demonstrate that voxelizing a polygonal mesh may be beneficial for the registration of the mesh. Voxel images can be registered much quicker than polygonal meshes, because the validity maintenance of a voxel model is much easier than that of a polygonal mesh.

Our results show that for the four presented test models, voxel-based shape matching is indeed much quicker than matching the polygonal meshes directly. In addition, in a post-processing, the results of the voxel-based methods can be used to improve the accuracy of the shape matching. As we said in section 4, making use of the noise vector does not affect our results: without this additional information, parametric shape matching would have performed even worse.

Although the results of these tests are interesting, one must keep in mind that four test models is not enough to reach a strong conclusion on the results of the reviewed methods. In addition, the parametric approach to shape matching has not been optimized to the extent that image registration techniques have been optimized. Optimization of the parametric approach may lead to better and faster results, but this also applies to the post-processing step of which the results are shown in table 3.

Nevertheless, the results indicate that, to obtain a better registration accuracy, it may pay off to first voxelise a polygonal mesh and then apply known registration methods. We intend to more thoroughly investigate this line of research in the future.

6. ACKNOWLEDGEMENTS

Test models a and b were kindly provided by NewProducts. Test models c and d were taken from the internet. Our work made extensive use of the ITK open source library for image registration methods and also of the DROP package, developed by Glocker et al. in München.

7. REFERENCES

[Chri01] Christensen, G.E., Johnson, H.J., Consistent Image registration, IEEE Transactions on Medical Imaging, Vol. 20, No. 7, pp. 568-582, 2001.

[Cootes95] Cootes, T.F., Taylor, C.J., Cooper, D.H., Graham, J., Active shape models - their training and application. Computer Vision and Image Understanding, Vol. 61, pp. 38-59, 1995.

[Glo08] Glocker, B., Komodakis, N., Tziritas, G., Navab, N., Paragios, N., Dense Image registration through MRFs and efficient linear programming, Medical Image Analysis, Vol. 12, No. 6, pp. 731-741, 2008.

[Han08] Sass Hansen, M, Larsen, R., Glocker, B., Navab, N., Adaptive Parametrization of

Multivariate B-splines for Image Registration, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1-8, 2008.

[Hart00] Hartkens, T., Hill, D.L.G., Maurer, C.R., Martin, A.J.,Hall, W.A., Hawkes, D.J., Rueckert, D., Truwit, C.L., Quantifying the intraoperative brain deformation using interventional MR imaging, Proc. Int. Soc. Magn. Reson. Med., Vol. 8, 2000.

[Igar99] Igarashi, T., Matsuoka, S., Tanaka, H., *Teddy: A sketching interface for 3D freeform design*, Proceedings of SIGGRAPH, pp. 409-416, 1999.

[Karp06] Karpenko, O.A., Hughes, J.F., *SmoothSketch: 3D free-form shapes from complex sketches*, Transactions on Graphics, Vol. 25, No. 3, pp. 589-598, 2006.

[Klein07] Klein, S., Staring, M., Pluim, J.P.W., Evaluation of optimization methods for nonrigid medical image registration using mutual information and b-splines, IEEE Transactions on Image Processing, Vol. 16, No. 12, pp. 2879-2890, 2007.

[Komo07] Komodakis, N., Tziritas, G., Approximate labeling via graph-cuts based on linear programming. IEEE Transactions on Pattern Analysis and Machine, Vol. 29, No. 8, pp. 1436 - 1453, 2007.

[Kybic03] Kybic, J., Unser, M., Fast parametric elastic image registration, IEEE Transactions on Image Processing, Vol 12, No.11, pp. 1427-1442, 2003.

[Lang07] Langerak, T.R., Vergeest, J.S.M., An evolutionary strategy for free form feature identification in 3D CAD models, Full Paper Proceedings of the WSCG conference, 2007, Plzen.

[Lang08] Langerak, T.R., Freeform feature recognition and manipulation to support shape design, Ph.D. Thesis, Delft University of Technology, 2008.

[Mac96] MacCracken, R., Joy, K.I., Freeform deformation with lattices of arbitrary topology, Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pp. 181-188, 1996.

[Mattes03] Mattes, D., Haynor, D.R., Veselle, H, Lewellen, T.K., Eubank, W., PET-CT Image registration in the chest using free-form deformations, IEEE Transactions on Medical Imaging, Vol. 22, No. 1, pp. 120-128, 2003.

[Pern05] Pernot, J.-P., Falcidieno, B., Giannini, F.; Léon, J.-C., Fully free-form deformation features for aesthetic shape design, Journal of Engineering Design, Vol. 16, No. 2, pp. 115-133, 2005.

[Rue99] Rueckert, D., Sonoda, L.I., Hayes, C., Hill, D. L. G., Leach, M. O., Hawkes, D. J., Nonrigid Registration Using Free-Form Deformations: Application to Breast MR Images, IEEE Transactions on Medical Imaging, Vol. 18, No. 8, pp. 712-721, 1999.

[Szel97] R. Szeliski and J. Coughlan, Spline-based image registration, Int. J.Comput. Vis., vol. 22, pp. 199–218, 1997.

[Sed86] Sederberg, T.W, Parry, S.R., Free-form deformation of solid geometric models, Computer Graphics, Vol. 20, No. 4, pp. 151-160, 1986.

[Song05] Song, Y., Vergeest, J.S.M., Bronsvoort, W.F., Fitting and manipulating freeform shapes using templates, Journal of computing and information science in engineering, Vol. 5, No. 2, pp. 86-94, 2005.

[Sorz05] Sorzano, C.O.S., Thevenaz, P., Unser, M., Elastic registration of biological images using vector-spline regularization, IEEE Trans Biomed Eng, Vol. 52, No. 4, pp. 652-663, 2005.

[Star06] Staring, M., Klein, S., Pluim, J.P.W., A rigidity penalty term for nonrigid registration Medical Physics, Vol. 34, No. 11, pp. 4098-4108, 2007.

[Tanner00] Tanner, C., Schnabel, J.A., Chung, D., Clarkson, M.J., Rueckert, D., Hill, D.L.G., Hawkes, D.J., Volume and shape preservation of enhancing lesions when applying nonrigid registration to a time series of contrast enhancing MR breast images, Lecture Notes in Computer Science, Vol 1935, pp. 327–337, 2000.

[Rohl03] Rohlfing, T., Maurer, C.R., Bluemke, D.A., M., Jacobs, M.A., Volume-Preserving Nonrigid Registration of MR Breast Images Using Free-Form Deformation With an Incompressibility Constraint, IEEE Transactions on Medical Imaging, Vol. 22, No. 6, pp. 730-741, 2003.

[Ver01] Vergeest, J.S.M., Spanjaard, S., Horváth, I., Jelier, J.J.O., Fitting freeform shape patterns to scanned 3D objects, Journal of Computing and Information Science in Engineering, Vol. 1., No. 3, pp. 218-224, 2001.

[Yang08] Yang, D., Li, H., Low, D.A., Deasy, J.O., El Naqa, I., A fast inverse consistent deformable image registration method based on symmetric optical flow computation, Physics in Medicine and Biology, Vol. 53, pp. 6143-6165, 2008.

Reliable Soft Shadows over Implicit Surfaces

Jorge Flórez-Díaz
Institut d'Informàtica i Aplicacions
Campus Montilivi
Universitat de Girona
Spain (17071), Girona, Catalonia
jeflorez@ima.udg.edu

Mateu Sbert
Institut d'Informàtica i Aplicacions
Campus Montilivi
Universitat de Girona
Spain (17071), Girona, Catalonia
mateu@ima.udg.edu

ABSTRACT

This paper introduces an efficient and reliable algorithm to create soft shadows for ray traced implicit surfaces. In a classical soft shadow creation process, many rays have to be traced to calculate how much of the area light is visible from a point. In our approach, we trace from a point a beam that covers the area light. If the beam is not occluded, the point can be safely discarded because the use of Interval Arithmetic guarantees that the point is fully lit by the area light. In this case, extra rays are not traced, thus extra intersection tests are not required. Our soft shadow algorithm is then 38% to 59% faster than an algorithm based in a regular number of beams traced for each point. Besides, the using of beams based on Interval Arithmetic guarantees that thin details of the surfaces are not lost during the process.

Keywords

Soft shadows, Implicit Surfaces, Beam tracing, Interval Arithmetic.

1. INTRODUCTION

An implicit surface can be defined as the set of points from the equation defined by $f(x,y,x)=0$, where $f : \Omega \subseteq \boldsymbol{R^3} \rightarrow \boldsymbol{R}$. Finding methods to perform a reliable ray tracing process of these surfaces has been the subject of study by several authors in the last two decades [Kal89a,Mit90a,Har97a]. However, the most extended technique to perform such work is a root finding process using Interval Arithmetic [Mit90a,Cap00a,San03a].

Ray tracing of implicit surfaces is a slow process. Furthermore, the use of Interval Arithmetic decreases the performance of such algorithms. This is because every interval operation requires many floating point operations. For that reason, the creation of effects like soft shadows are often avoided due to the vastly increased number of rays required.

Another problem is that some special implicit surfaces are still not correctly rendered. This occurs because the rays miss the thin parts of the surfaces crossing a pixel, causing aliasing problems. This is more noticeable in the shadows produced by the implicit surfaces.

In this paper, we propose an efficient and reliable algorithm to create soft shadows for implicit surfaces using Interval Arithmetic. Our method is based on the use of shadow beams instead of rays, having the following advantages:

- A shadow beam can be used to replace the process of many shadow rays. If the beam is not occluded by any object in the scene, then the trace of the single rays is avoided, saving the computational time required for such intersection test.
- Our reliable beam can detect any feature (even the smallest ones), avoiding aliasing problems in the shadows.

Previous Works

There are a huge number of articles related with soft shadows in literature. The most widely used algorithms are based in rasterization techniques like shadow maps [Kir03a,Arv04a,Ann08a], in which a pre-filtered depth map of the shadows are created, and shadow volumes [Asr03a,Leh06a] which defines the shadow boundary using a geometric algorithm for a silhouette extraction. Guennebaud et al [Gue06a] developed an algorithm which computes a percentage of light seen from every pixel in the image. This is done by backprojecting the occluders in a shadow map onto the light plane and determining the magnitude of the occlusion. These techniques can run very fast in graphics hardware.

However, they can not be easily adapted to work with a pure ray tracing process for implicit surfaces.

An approach that can be easily applied to create soft shadows for ray tracing is Monte Carlo sampling [Coo84a]. Here, many rays have to be traced against the area light, which is inefficient for implicit surfaces where the root finding process is slow. To solve that problem, coherence approaches can be applied.

In pencil tracing [Shi87a], the rays nearby to a special ray (called the axial ray) are grouped. These rays (called paraxial rays) are represented by a 4D matrix that represents the deviations in position and direction from the axial ray. This matrix system can be combined with the matrices for every surface in the environment to represent the propagation of the rays. The disadvantage of this method is that the surfaces have to be smooth, because the system can not deal with discontinuities. In those cases, the method requires of individual rays instead of pencils.

In cone tracing [Ama8a], the rays are represented by a cone conformed by an apex, a center line and a spread angle. To calculate reflections and refractions, the new center line is calculated using a standard ray tracing technique. This technique can deal with the aliasing problems in the soft shadows by means of the calculation of the part of the cone blocked by the objects. However, due to the complexity in the calculation of intersections and parts of the cone covered by the objects, the method only deals with spheres, planes and polygons.

Beam tracing [Hec08a] replaces individual rays with beams. A beam consists in a set of rays with a common apex crossing planar polygons. When a beam intersects an object, a new beam with a new polygonal section is generated. The remainder of the beam can be complex forms that require a method that can operate with arbitrary polygons. However, this technique is slow, only works for planar polygons facets and require complex intersections test.

There are some techniques that improve the beam tracing process, which can be used to accelerate the soft shadow generation. [Over07a] introduced a technique based in algorithms for efficient beam-triangle intersection, and beam-kd-tree traversal to generate the soft shadows. In [Car09a] a frustum tracing technique is introduced. Here, shadow rays are generated in every frustum on demand. This technique is well suited for SIMD architectures, and was specially designed for the Larrabee architecture of Intel. These techniques represent the beam by means of four rays, which are traversed in the acceleration structures. Those techniques can be easily adapted to work with implicit surfaces. However, they could miss small features of the implicit surfaces that lie inside the beam represented by four unconnected rays.

In [Over99a] there is a proposal to create an approximation of the soft shadow for implicit surfaces. However, it only works for simple implicit surfaces and generates only approximations of the shadows.

2. PRELIMINARIES

Our proposal is based in the use of beams and Interval Arithmetic to accelerate the creation of the soft shadows, as well to guarantee that any part of the surfaces is not missed during the beam traversing. Besides, our algorithm was implemented for GPU using the Nvidia's CUDA architecture to improve the rendering time. Using a CPU, a ray tracer using Interval Arithmetic could take minutes for naïve implicit surfaces [Cap00a,San03a]. Other approaches based on GPU have demonstrated that the rendering time for the ray tracing of implicit surfaces could be decreased up to 3 orders of magnitude [Col08a,Woo04a].

Beam Definition

Using interval arithmetic, a beam is defined by:

$$X = c_x + T(X_s - c_x)$$

$$Y = c_y + T(Y_s - c_y)$$

$$Z = c_z + T(Z_s - c_z)$$

Where X, Y, Z; X_s, Y_s, Z_s; T are intervals (in this paper, intervals are defined as capital letters). (c_x, c_y, c_z) is the origin or view point; X_s, Y_s, Z_s are the intervals defining the direction of the beam, and T is the interval parameter of the beam, which determines how much the beam is advancing in the direction X_s, Y_s, Z_s.

The intersection between a beam and an implicit surface defined by $f(x, y, z) = 0$, is defined by:

$$0 \in F(X_s, Y_s, Z_s, T)$$

where $F(X_s, Y_s, T)$ is called the "*inclusion function*" which is equivalent to:

$$f(c_x + T(X_s - c_x), c_y + T(Y_s - c_y), c_z + T(Z_s - c_z))$$

So, the intersection test for the beam consists in finding a space of solutions (T) for all the rays that conforms the beam. The value T can be found using a classical Interval Bisection method or an Interval Newton method [Cap00a].

Interval Arithmetic guarantees that for any value $x \in X$, $f(x) \subseteq F(X)$, where $F(X)$ is the inclusion function. Hence, the advantage of the beam definition using Interval Arithmetic is that any part

of the surface crossed by the pixel is not missed (see figure 1).

If $0 \notin F(X_s, Y_s, Z_s, T)$ then it is sure that there is not an intersection between the beam and the surface for any value $t \in T$.

If $0 \in F(X_s, Y_s, Z_s, T)$ then it is possible that there are some intersections between any of the rays included in the beam and the implicit surface. In this case, if a subdivision process is performed over the parameter of the beam T, the current value of T must be subdivided and every new interval evaluated again. The subdivision algorithm needs to arrive to machine precision because in that case $0 \in F(X_s, Y_s, Z_s, T)$ is often achieved [Cap00a].

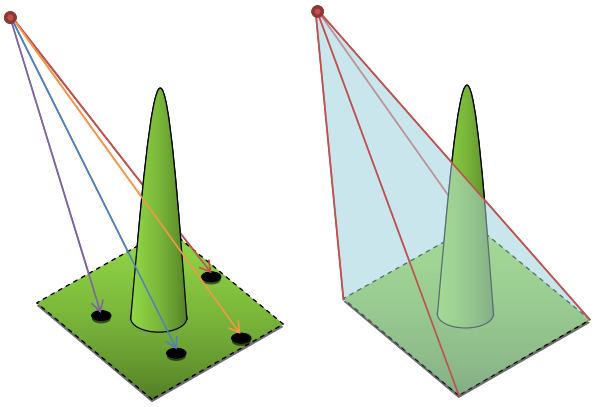


Figure 1. (Left) Some rays can miss thin parts of an implicit surface. (Right) Using beams, it is possible to discover such thin details.

Acceleration Structure

In our implementation we use a regular grid to accelerate the beam traversal. The nature of the rendered scenes, composed by arbitrary implicit surfaces uniformly distributed, makes a regular grid a good candidate to accelerate the beam tracing process [Fuj01a]. Moreover, using a regular grid the complexity of the beam traversing is reduced.

We found that a regular grid is an approach that fits perfectly in the SIMT architecture (Single-Instruction, Multiple-threads) of CUDA. Here, each independent thread can perform the evaluation of a cell of the grid, so the evaluation of many cells can be performed very fast. In our implementation this time is meaningless compared with the time spent in the rendering process.

To create the regular grid, the object space in the scene is subdivided in a predefined number of boxes or cells, and each one is scanned to look for the implicit surfaces crossing them.

Having an implicit surface $f(x,y,z)=0$, to know if the surface crosses a cell defined by the intervals (X,Y,Z), the united extension $F(X,Y,Z)$ is used. If $0 \in F(X,Y,Z)$, the region may be crossed by the surface. In the other case, the region can be definitively discarded.

Beam Traversal

To determine the boxes intersected by the beam, the direction of the beam in x, y or z (in space coordinates) is selected. This is done taking the normalized directions of the four vectors composing the corners of the beam and selecting the one with the bigger absolute value. The boxes are scanned advancing in the coordinate selected as direction, but only for the boxes covered by the other two coordinates of the beam.

Figure 2 represents a 2D scenario to facilitate the understanding of the process. In this case, the output is a line instead of an area. The propagation direction is supposed to be in the z axis. The adaptation of this process to a 3D case is straightforward.

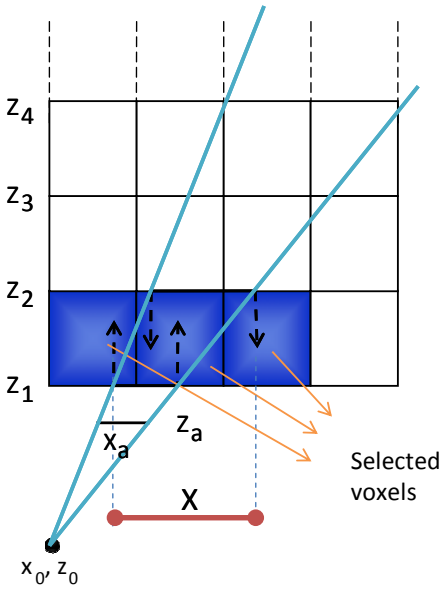


Figure 2. Traversing of a beam inside a regular grid structure.

Every row in direction z is checked, but only the cells covered by the interval ***X*** are considered for the evaluation of the intersection test (cells in blue). The interval covering the cells in x coordinate is calculated as follows:

$$X = x_0 + T(X_a - x_0)$$

having:

$$T = (Z - z_0) / (Z_a - z_0)$$

in which X_a and Z_a are the intervals representing the direction of the beam, and x_0, y_0 are real values representing the viewpoint. The value of Z is obtained taken the minimum and maximum values of z for the current row of cells in the grid.

The algorithm continues looking for cells covered by ***X*** in every row in the grid.

In every cell, the corresponding interval value of the parameter of the beam (T_C) must be calculated. This value represents the minimum and maximum value in which the roots are searched for the parameter *T*. This value is calculated as:

$$T_C = [\max[\underline{T_z}, \underline{T_x}], \min[\overline{T_z}, \overline{T_x}],$$

where $\underline{T_z}, \underline{T_x}$ are the lower bounds, and $\overline{T_z}, \overline{T_x}$ are the upper bounds of the intervals T_z, T_x. Moreover:

$$T_z = (V_z - z_0) / (Z_a - z_0), \quad T_x = (V_x - x_0) / (X_a - x_0),$$

where the cell is defined by the intervals V_x and V_z.

Root Searching

When a beam intersects a cell crossing a part of a surface, a root must be searched in the space T_c corresponding to the current cell. Usually, recursion is used for the root searching algorithms. However, in the GPU the recursion is not allowed. For that reason, we made an algorithm which work making bisections over the value T_c and evaluating the inclusion function in the two sections created in every step. We use two variables to control the bisection process: one variable, *ns,* keeps the number of subdivisions currently performed over the parameter T_c in power of two (number of bisections = 2^{ns}), and the other, *cs,* is pointing to the section currently scanned. For instance, if *ns* = 2 and *cs* = 3, then *Tc* is currently subdivided in four sections, and the section in which the inclusion function is evaluated is the third one. This means that $cs < 2^{ns}$ during the bisection process.

The subdivision process must start with ns = 1 (T_c is subdivided in two sections), and *cs* = 1 (pointer to the first section). If we find that $0 \in F(X_s, Y_s, T)$ (the root could be in the current section), the current section is bisected, the level of subdivision is incremented (*ns* = *ns* + 1) and the *cs* is set to the first section of the current subdivision level (*cs* = *cs**2 - 1).

If for the current section $0 \notin F(X_s, Y_s, T)$ (there are no roots in this section), then we must check if the pointer is in the first or the second section. If the remainder of *cs*/2 is different than 0, then *cs* is pointing to the first section. In this case, *cs* is set to the next section (*cs* = *cs* + 1). If the remainder of *cs*/2 is equal than 0, then *cs* is in the second section of the current subdivision. In that case, the two sections are rejected, the level of subdivision is reduced (*ns* = *ns* - 1), and two new sections are selected, just following the two evaluated sections in the last step (*cs* = *cs*/2 + 1).

The process continues evaluating pairs of sections until a section reached a predefined small size. In our implementation, we finish the process when the width of the currently evaluated section is less than 10^{-5}. The process is also finished if $cs > 2^{ns}$. This occurs when the bisection process did not find any intersection between the beam and the surface.

3. ALGORITHM SPECIFICATION

In our implementation, a PBO (Pixel Buffer Object) is created and sent to the GPU. We use the PBO to keep the final color values corresponding to every pixel of the window. After that, the surface to be rendered can be defined by the user. The surfaces were previously loaded, so the user only has to define an index indicating which surface has to be rendered. Once the surface is defined, the CUDA programs are executed, and the CPU waits until the results are collected from the GPU. Finally, OpenGL is used to show the contents of the PBO in a window.

Creation of the Grid

In our algorithm, the first step performed in the GPU is the definition of a grid to accelerate the beam tracing process. This task can be performed directly in CUDA or can be performed in CPU. For the last case, the data of the grid must be saved in an array to be sent to the GPU.

Beam Casting Process

Having the grid in the GPU, a beam casting process is performed, tracing a beam for every pixel in the screen. The objective is to trace a beam covering all the area of the pixel, and traverse the beam to find the intersections with the surfaces. For this task, we divided the screen in many tiles, which are sent to the GPU to be processed in one block each one. The result of this task is saved in an array with size equal to the number of pixels in the screen. The array has two floats for each position. Those values are used to keep the lower and upper bounds of the space of solutions (*T*) for the corresponding pixel. This value can be replaced in the definition of the beam to obtain the corresponding intersection values for *X, Y* and *Z*.

Because the intersection points are intervals, we use the midpoints to obtain the initial point of the shadow beam.

Soft Shadow Beam Tracing

Having the initial point, a shadow beam is traced from the initial point at a distance $\propto$ to avoid self intersections. The beam must cover all the area of the light.

In our implementation, we define the area lights as squares that are parallel to the x,y or z axis. We use three intervals to define the light (L_x,L_y,L_z), but one

of them is a point-wise interval. The intervals defining the light are used as the direction of the beam.

We use the shadow beam to analyze what happens with the rays included in the beam. There are two possible situations:

- Case 1: An occluder is not found between the initial point and the light. It means that all the rays inside the beam are not occluded by any surface in the scene.
- Case 2: the beam intersects a surface in the scene. It means that some or all the rays that conform the beam are occluded by an object.

In the first case, the point is not covered by a shadow. This means that further rays are not needed; so we finish for this point and continue with the next. This represents a lot of time saved in the calculation of the intersection test for new rays, which is the more expensive step in a ray tracing process (see figure 4).

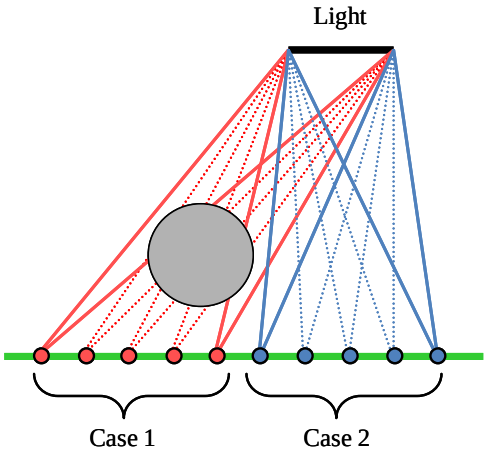


Figure 4. Two cases detected during the beam tracing process. Case 2 can represent a lot of points in a scene, saving the tracing of new rays.

For the second case more rays are needed. It is possible to apply a classical Monte Carlo sampling or any other method based in rays. In our implementation, we opt to trace more sub-beams instead or rays to guarantee that small features are not missed. Our idea is to avoid aliasing problems in the visualization of the shadow (see figure 5).

If the beam intersects any of the surfaces in the scene, the area light is subdivided in many tiles. Many sub-beams starting in the same point are traced against these tiles. The count of sub-beams hitting surfaces over the total number of traced beam is used to create an index of the percent of the area of the point under the shadow. The value of this index varies from 0 for points not covered by shadow, to 1 for surfaces in the umbra

In our implementation, the light is subdivided in a regular number of tiles, tracing new beams against them. Initially, we thought that subdivision process like a quadtree could reduce the number of beams needed, and hence reducing the processing time. However, we found that our implementation using a quatree for the subdivision of the light, was up to 1.5 times slower than an implementation using a regular subdivision. This occurs because GPU is faster with a completely parallel approach, in which every beam can be traced independently of the others. A quadtree requires more conditionals, and its irregular structure makes this approach slower than the tracing of a regular and predefined number of sub-beams against the light.

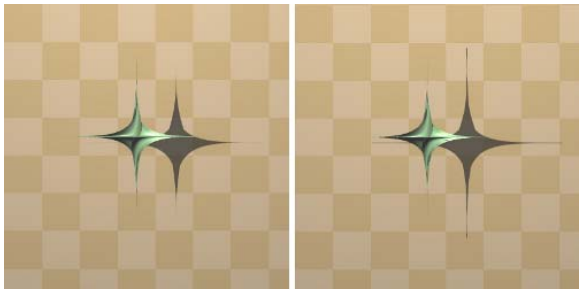


Figure 5. A twister surface with thin features. Left: using rays, the shadow is not well defined. Right: using beams, the thin details are revealed in the shadow.

4. EXPERIMENTATION

To test our algorithm, we used four surfaces: Orthocircle, Chubs, Kusner Schmitt and McMullen (figure 6).

The soft shadows were generated using 49 sub-beams. We found that this number of beams is good enough to obtain the results in figure 6. We use an area light of *0,1* of the size of the area of the screen. The images were generated at a resolution of *512x512* pixels in a GPU NVIDIA 9600M GT.

Table 1 shows the results of our algorithm compared to an algorithm in which 49 sub-beams are traced for every intersection found during the beam casting process. Our method can drastically reduce the number of beams needed for the soft shadow creation. Time reduction in all the cases is over 38% (arriving up to 59% for the Orthocircle) for the shadow creation process.

In Table 1, the column “initial beams occluded” indicates the number of initial beams traced, for which an occlusion was detected. Note that the number of “pixels in shadow”, which are detected when the sub-beams are traced, is smaller than the pixels detected by the “initial beam occluded”. This occurs because the overestimation of the intervals produces a beam that increases its real size. For that

reason, some pixels are detected to be in a shadow but they are not (see figure 7).

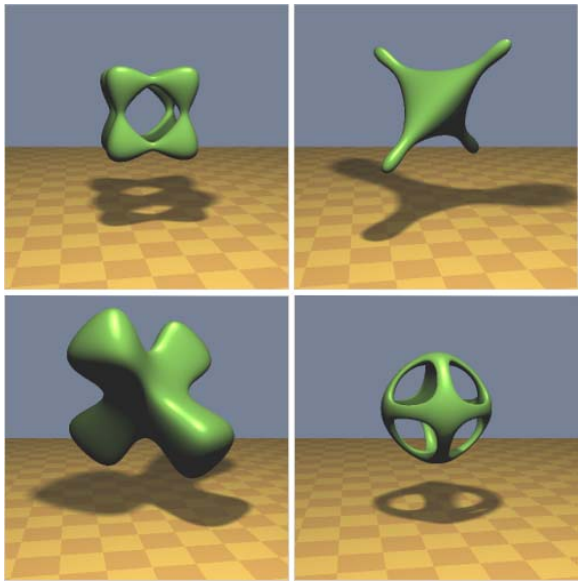


Figure 6. Surfaces used in our experimentation, from left to right and top to bottom: Chubs, Kusner-Schmitt, McMullen and Orthocircle.

The overestimation is proportional to the width of the used intervals. In this algorithm, this effect depends of the size of the light: for a bigger light, more pixels will be detected to be in a shadow when they really are not. However, if the intervals are subdivided, and each one is computed independently, the effect of overestimation is reduced [rev05a]. In our algorithm, when the small beams are traced (the size of the intervals is smaller) a correct representation of the soft shadow is found. The overestimation represents an extra cost in the rendering time, because 49 rays are traced from some points fully lit by the light. However, this matter does not affect the quality of the image. In the case of a really big light, instead of tracing an initial beam, it is possible to trace four, and perform the same analysis presented in this paper.

The efficiency of the algorithm is related to the number of pixels covered by a shadow. For example, the Orthocircle have a 5% of the pixels under a shadow, and the improvement is 59% over the approach based in a regular number of beams. For the McMullen surface, the shadow covers the 21% of the pixels, with an improvement of 38%.

Another issue to be taken into account is that a beam detects pixels under a shadow where a ray can miss thin features of the surface. The same occurs for the borders of the surface: the beam detects more pixels under a shadow than an approach based in rays. Indeed some of these pixels could be fully lit. The overestimation contributes to obtain this effect. However, this issue does not affect the quality of the images obtained (see figure 6).

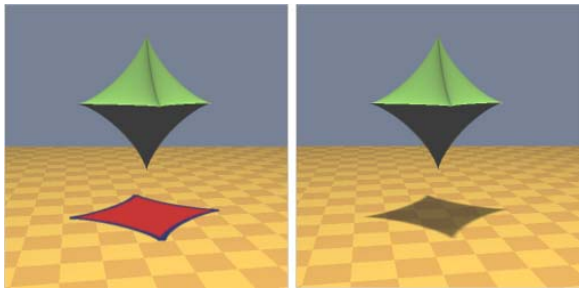


Figure 7. Left: The blue and red regions indicate the pixels detected to be in a shadow by the initial beams. The red region indicates the pixels that are indeed under the shadow. Right: the surface and its soft shadow.

5. CONCLUSIONS

In this paper we introduced a reliable soft shadow algorithm for implicit surfaces. Here, we propose the use of beams based on interval beam arithmetic to accelerate the rendering process and to detect thin features of the implicit surfaces.

We prove that using beams, the creation of soft shadows can be accelerated. Using only one beam it is possible to detect if a pixel is under a shadow. If occlusions are not detected, then the tracing of more rays for the same point are avoided, thus saving computational time. This is more efficient than tracing many rays to detect the soft shadows in every intersection found during a ray casting process.

Besides, Interval Arithmetic allows the creation of convex hulls in which none of the solutions is lost. A beam based on Interval Arithmetic can guarantee that either small or thin parts of the surfaces are not missed in the final image.

As a future work, we are going to study the application of Affine Arithmetic in order to improve the computational time. Affine arithmetic promises to reduce overestimation problems, and to keep all the advantages of the interval arithmetic.

6. ACKNOWLEDGEMENTS

This work has been partially funded by the Spanish Government (Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica, Ministerio de Ciencia y Tecnología) through the co-ordinated research project MEC TIN2007-68066-C04-01 and by the government of Catalonia through 2009 SGR643.

7. REFERENCES

[Ama8a] Amanatides J. Ray Tracing with Cones. Computer Graphics 1984, 18, 3, 129-185.

			Our method (using initial beams)			Using 49 beams		
Surface	Beam casting (secs.)	Pixels in shadow	Initial beams occluded	Time (secs.)	No. Beams	Time (secs.)	No. Beams	%time saved
Orthocircle	1,23	13.328	14.961	1,66	719.859	4,08	7.101.962	59%
Chubs	1,53	21.932	27.357	3,82	1.340.493	6,57	7.343.091	42%
Kusner Schmitt	2,24	22.374	44.979	5,65	2.203.971	9,44	7.395.374	40%
McMullen	2,28	55.231	73.386	6,84	4.085.914	11,06	8.749.244	38%

Table 1. Results of the experimentation.

[Ann08a] Annen, T., Dong, Z., Mertens, T., Bekaert, P., Seidel, H., and Kautz, J. 2008. Real-Time All-Frequency Shadows in Dynamic Scenes. ACM Transactions on Graphics. 27, 3, 1–8. 2008

[Arv04a] Arvo J. and Westerholm J. Shadows Using a Single Light Sample Hardware Accelerated Soft Shadows using Penumbra Quads. Journal of WSCG, Vol.12, No.1-3, 2004

[Asr03a] Assarsson U., Akenine-Möller T.: A geometry based soft shadow volume algorithm using graphics hardware. ACM Transactions on Graphics. 22, 3. 511–520. 2003

[Cap00a] Capriani O., Hvidegaard L., Mortensen M. and Schneider T. Robust and efficient ray intersection of implicit surfaces. Reliable Computing. 1, 6,, 9-21, 2000.

[Car09a] Carsten B. and Wald I. Efficient Ray Traced Soft Shadows using Multi-Frusta Tracing. Proceedings of the Conference on High Performance Graphics. 135-144. 2009.

[Col08a] Collange S., Flórez J. and Defour D. Interval Library based in Boost Interval. Proceedings of the International Conference on Real Numbers and Computers, 61-72. 2008

[Coo84a] Cook, R., Porter, T., and Carpenter, L. Distributed Ray Tracing. Computer Graphics (Proc. of SIGGRAPH ’84) 18, 3, 137–144.

[Fuj01a] Fujimoto, A. Tanaka, T. Iwata, K. ARTS: Accelerated Ray-Tracing System. IEEE Computer Graphics and Applications. 6,4,16-26, 1986.

[Gue06a] Guennebaud G., Barthe L., and Paulin M. Realtime soft shadow mapping by backprojection. In Eurographics Symposium on Rendering, 227–234, 2006.

[Har97a] Hart J. Sphere tracing: A geometric method for the antialiased ray tracing of Implicit Surfaces. The Visual Computer, 12, 10:527–545, 1997.

[Hec8a] Heckbert P. and Hanrahan P. Beam Tracing polygonal Objects. Computer Graphics 1984, 18 (3), 119 – 127.

[Kal89a] Kalra D. and Barr A. Guaranteed ray intersection with implicit surfaces. Computer Graphics (Siggraph proceedings), 23:297–206, 1989.

[Kir03a] Kirsch F. and Döllner J. Real-Time Soft Real-time soft shadows using a single light sample. Journal of WSCG (Winter School on Computer Graphics 2003), 11, 1, 2003.

[Leh06a] Lehtinen, J., Laine, S., and Aila, T. 2006. An Improved Physically-Based Soft Shadow Volume Algorithm. Computer Graphics Forum (Proceedings of Eurographics ’06) 25, 3.

[Mit90a] Mitchell, D. Robust ray intersection with interval arithmetic. Proceedings on Graphics Interface ’90, 68–74, 1990.

[Over07a] Overbeck R., Ramamoorthi R.,and Mark W. A Real-time Beam Tracer with Application to Exact Soft Shadows. Eurographics Symposium on Rendering 2007.

[Over99a] Overveld K., Mark T. and and Wyvill B. Soft Shadows for Soft Objects. Proceedings of Fourth Eurographics Workshop on Implicit Surfaces, Bordeaux, France. 1999.

[Rev05a] Revol, N. and Rouillier, F. Motivations for an Arbitrary Precision Interval Arithmetic and the MPFI Library. Reliable Computing, 11, 4, 275-290, 2005

[San03a] Sanjuan-Estrada J., Casado L. and García I. Reliable algorithms for ray intersection in computer graphics based on Interval Arithmetic. XVI Brazilian Symposium on Computer Graphics. 35 – 44, 2003.

[Shi87a] Shinya M., Takahashi T. and Naito S. Principles and applications of pencil tracing. Computer Graphics 1987, 21, 4, 45-54.

[Wil78a] L. Williams. Casting curved shadows on curved surfaces." SIGGRAPH '78, pp. 270-274. New York, USA, 1978.

[Woo04a] Wood A, Brendan M. and Scott K. Ray tracing arbitrary objects on the GPU. Proceedings of Image and Vision Computing, 21-23, 2004.A GPU

A Hierarchical Automatic Stopping Condition for Monte Carlo Global Illumination

Holger Dammertz
Ulm University
James-Franck-Ring
Germany, 89081 Ulm

Johannes Hanika
Ulm University
James-Franck-Ring
Germany, 89081 Ulm

Alexander Keller
mental images GmbH
Fasanenstrasse 81
Germany, 10623 Berlin

Hendrik P.A. Lensch
Ulm University
James-Franck-Ring
Germany, 89081 Ulm

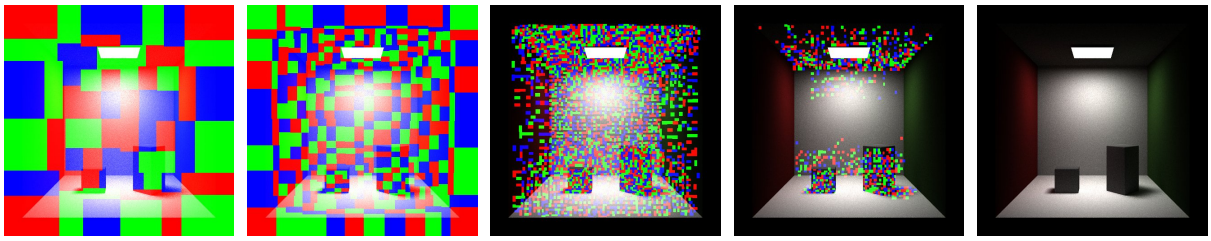


Figure 1: An example sequence of blocks. Initially large blocks (shown in saturated colors) are recursively refined and terminated individually as the image converges. Note that complex areas in the image, where indirect illumination dominates the appearance (e.g. the ceiling), are detected and refined by the algorithm.

ABSTRACT

We introduce a hierarchical image-space method to robustly terminate computations in Monte Carlo image synthesis, independent of image resolution. The technique consists of a robust convergence measure on blocks which are either recursively subdivided or terminated independently, using a criterion which separates signal and noise based on integral estimates from two separate sample sets. The technique can be easily implemented, as the evaluation of the error measure only requires a second framebuffer and a list of non-terminated blocks. Based on the stopping criterion, one can furthermore sample both the image plane as well as the light sources adaptively. Adaptive sampling reduces the number of samples required to gain the same root mean square error to one quarter in some of our test cases.

Keywords

Monte Carlo, Global Illumination

1 INTRODUCTION AND PREVIOUS WORK

Monte Carlo methods are among the most general techniques to solve the global illumination problem. Unbiased Monte Carlo methods are guaranteed to converge to the correct solution but the number of samples required for generating an image where the remaining noise is hardly visible is not known in advance.

We therefore propose a stopping algorithm which robustly separates the remaining noise in the rendered image from the variation due to the scene content, which is easy to implement, requires very little additional memory and we also show how to use it to adaptively sample from the light sources. Following the idea of Dippé and Wold [DW85], we estimate the rate of change as the difference between images generated with two different sample sets.

A lot of work has been done to quantify the perceived differences by the human visual system [Mys98, BM98, RPG99, YPG01, SCCD04, FP04, SGA*07, Dal93, MDMS05], error measures based on confidence intervals, contrast, variance, saliency and entropy have been investigated. For simplicity and fast evaluation, we base our stopping criterion on the remaining color noise in relation to the logarithmic luminance of the sample, but any other error measure could be used as well.

Rigau et al. [RFS03] presented a similar method based on entropy, but they only consider the image as a whole and sub-pixel refinements, not blocks of pixels.

Hachisuka et al. [HJW*08] also subdivide a hierarchy of samples very similar to the MISER algorithm [PTVF92], but our error measure is based

on two integral estimates, which is able to separate signal from noise. Also, additional reconstruction is not necessary because sampling is dense enough (at least one sample per pixel), and the samples do not have to be stored explicitly in our case.

In [ODR09] the authors describe an adaptive wavelet rendering that is similar to our approach but their scaling functions are more restrictive than our blocks and additionally we also adaptively sample the light source and are not restricted to 2d.

For a reliable error estimate, it is not sufficient to look at the variation inside a single pixel, due to the limited number of samples [Mit87]. To overcome this, we evaluate the error measure on a hierarchy of blocks, which are recursively refined only as the remaining error drops below a certain resolution-independent threshold (see Figure 1 for an illustration of the refinement process). This makes sure the algorithm adapts to the true signal, not a noisy estimate. Assuming that the samples available so far have not completely missed any major light contribution (i.e. one "firefly"-path is enough), our scheme operates conservatively.

Painter and Sloan [PS89] also used a hierarchy, an image-space *k*d-tree, but explicitly stored all samples and all levels of the hierarchy. We, on the other hand, simply store the integral estimates of the two sampling sets per pixel in two framebuffers. Furthermore, we only keep track of the leaf blocks, which significantly simplifies bookkeeping.

Based on our convergence estimate, adaptive sampling can be performed by simply supersampling only those blocks which are not yet terminated.

Note that this is different to most existing adaptive sampling approaches, since it is not based on placing samples in regions considered interesting at an early stage but splits and terminates rather conservatively. This is why the common problem of missed features such as small objects is not a problem.

Furthermore, as the technique is framebuffer-based, it is completely independent of the underlying rendering algorithm.

2 A HIERARCHICAL CRITERION

To make sure the algorithm does not stop computations before all important features have been detected, a lot of samples should be drawn before making a decision (see Figure 2 for an illustration, and Figure 3 for a comparison to a pure per-pixel approach). Since it is not desirable to shoot a lot of possibly unnecessary rays per pixel, we initially base the evaluation of the error on the image as a whole.

As the variance in the image will be distributed unevenly, depending on the problem and the type of path space sampler used, the stopping decision should be done locally. This is reflected by splitting the blocks when the algorithm has enough confidence not to have missed any important features, i.e. the per-block error measure drops below a certain threshold ε_s.

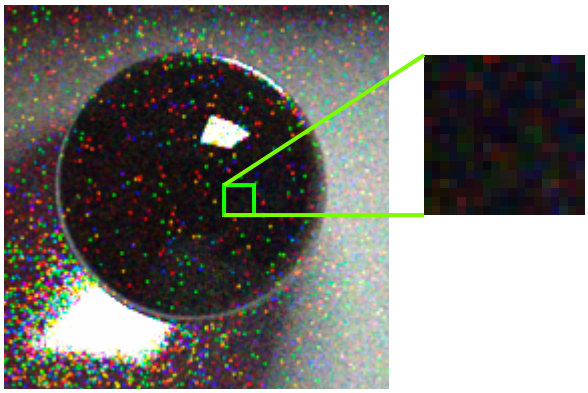


Figure 2: A caustic rendered with way too low sampling density. It is not sufficient to make decisions based on only a few samples, i.e. on small blocks or even pixelwise. The magnified block for example appears to be converged already, but it still misses important features (evident through adjacent bright pixels) and thus needs more samples.

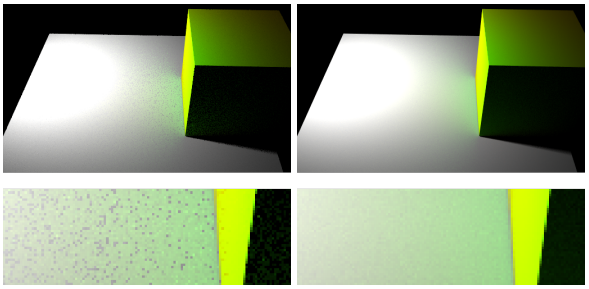


Figure 3: This shows the problem of using a per pixel termination criterion. Due to the nature of Monte Carlo sampling some pixels terminate too early which is clearly visible in the left image. The right image shows our proposed termination criterion with equal sample count.

A block only consists of an axis-aligned bounding box covering an area of the image. The blocks are managed in a linear list. That is, if a block is split, it is simply removed and the two resulting child blocks, disjointly covering the same area of the image, are added to the list again. This way, no explicit hierarchy has to be maintained.

The algorithm continues by drawing new samples (one for each pixel in each remaining block in our implementation). If stratification in image space can be guaranteed, e.g. by using a simple backward path tracer, only non-terminated blocks have to be sampled further. If certain sampling methods cannot be restricted in this way, such as for example light tracing methods, their contribution can still be used, as the stopping condition is not directly coupled to adaptive sampling. In fact, it is also possible to sample adaptively from the lights (see Section 3.2).

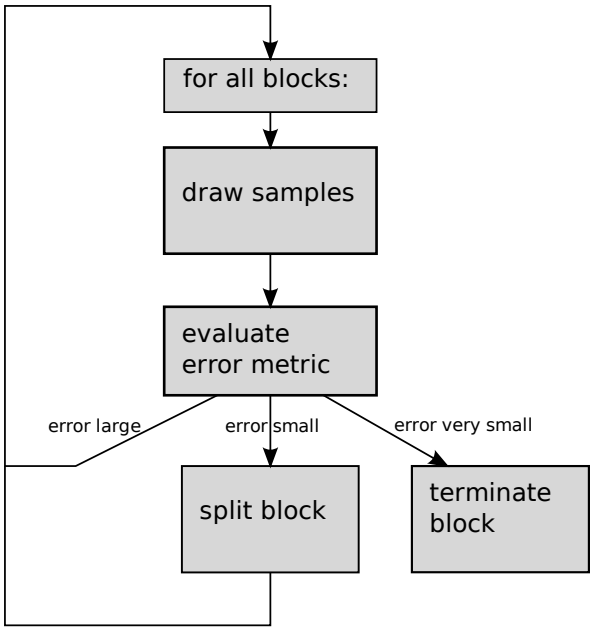


Figure 4: Illustration of a rendering algorithm using the stopping algorithm.

The image is converged when the error of all blocks is smaller than a threshold $\varepsilon_t < \varepsilon_s$. For an overview of the algorithm see Figure 4. An example sequence of blocks is depicted in Figure 1. Note how the algorithm automatically adapts the block size to the image and is thus independent on the resolution.

2.1 Error Metric

We use an error metric based on the pixels in the final image, but to get a robust criterion we always evaluate a block of pixels. We compare two independently computed images with the same number of samples similar to [DW85, DS04].

In the implementation, the evaluation of the error metric can be simplified by introducing just a single additional accumulation buffer and keeping the normal accumulation buffer used to compute the final image. In the second buffer we accumulate only samples every second rendering pass. This is based on the simple observation that computing a single image I with an even sample count can be split into two images A and B with $I = A/2 + B/2 \Rightarrow B/2 - A/2 = I - A$. Thus, using an RGB buffer we estimate the per pixel error as

$$e_p = \frac{|I_p^r - A_p^r| + |I_p^g - A_p^g| + |I_p^b - A_p^b|}{\sqrt{I_p^r + I_p^g + I_p^b}}.$$

The square root in the denominator is motivated by the logarithmic response of the human visual system to luminance. The term here behaves similarly, is easier to evaluate and was found to yield slightly better results.

The error per block is computed by summing over all pixels:

$$e_b = \frac{r}{N} \sum_p e_p$$

where N is the number of pixels contained in this block and r is a scaling factor computed as $\sqrt{A_b/A_i}$. A_i is the area of the image, A_b the area of the block under consideration.

2.2 Block Splitting and Block Termination

The user specifies a single error value v that is used in all further decisions. From this user-specified value we compute two error thresholds ε_s (splitting) and ε_t (termination). For simplicity we use $\varepsilon_t = v$ but it is of course possible to rescale v to a more intuitive parameter range. In all our tests we used $v = 0.0002$. Given ε_t, we compute $\varepsilon_s = 256 \cdot \varepsilon_t$. This is an empirical choice that worked well in all our tested scenes.

The splitting is performed axis-aligned by choosing the axis where the block has the largest extent. The split position is chosen such that the error measure is as equal as possible on both sides.

3 RESULTS

In this section we analyse our proposed stopping criterion for different scenes and additionally examine how well the proposed error metric can be used as an adaptive sampling criterion. In the first subsection we use a path tracer with next event estimation. In Section 3.2 we propose a simple method to extend this adaptive sampling to a light tracing algorithm.

3.1 Image Space

Figure 5 on the left shows a simple test scene of a diffuse sphere illuminated by an area light source casting a large smooth shadow. The right image shows the distribution of the number of samples in the final image as a heat map. White is the maximum number of samples and black the minimum. As each block is sampled uniformly the block structure is still apparent in this image. The graph in Figure 6 shows the number of samples required by the normal rendering algorithm and by our technique to achieve an RMS error below 0.01. In this simple scene the sample count was reduced from $700M$ to $400M$ (57%). This indicates that our error metric is meaningful also with respect to the RMS error. The evaluation of the error metric costs about 10% of the rendering time in our implementation.

As another example we show a living room scene in Figure 7. Again on the left the final image can be seen and on the right the sample count distribution. This is a more realistic scene as it is closed and strong indirect illumination is present. The complexity is distributed almost equally over the whole image. This explains

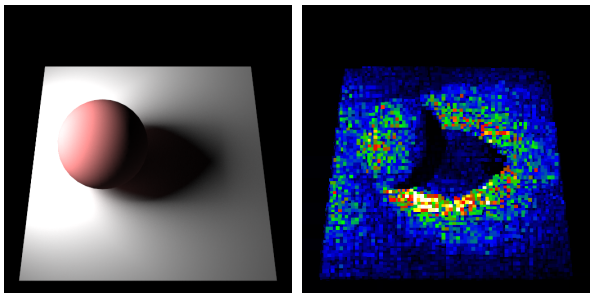


Figure 5: A simple sphere scene rendered with the proposed stopping condition (left) and the respective heat map (right) representing the number of samples required for a converged block. Most effort has to be spent in the penumbra regions.

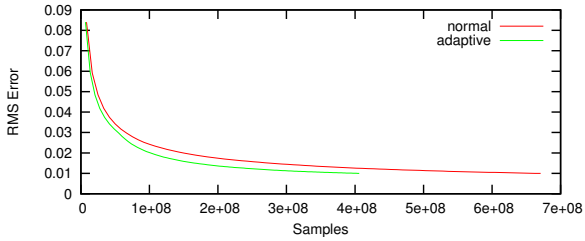


Figure 6: This graph shows the RMS error of the image shown in Figure 5 with increasing number of samples, for normal and adaptive sampling.

why in this scene sampling adaptively does not present a huge benefit. This is also visible in the RMS error graph shown in Figure 8. Still our termination criterion works robustly and also isolates the two small areas in the image where the illumination is more complex due to glossy objects and a small caustic cast by the monkey head on the table (see Figure 9).

3.2 Light Space

In this section we show how to extend our criterion to also be able to adaptively sample in a light tracing setting [Vea97]. A light tracer starts paths only from the light sources and connects hit points to the camera. This is very well suited to compute caustics but it is not possible to directly refine samples on the image plane as it is unknown where exactly a light sample will connect. For our termination criterion this is not a problem as it works equally well independent of the used rendering algorithm. Adaptive sampling of the image plane is not straightforward for light tracing, so all samples need to be distributed equally.

To facilitate adaptive sampling also from the light sources, we back-project the image error metric to an importance map around each light source. Such an importance map consists in a quantized hemisphere around each emitting triangle, thus representing only outgoing directions, ignoring the starting point on the triangle and higher-dimensional bounces. This works well under the assumption that triangles are small with respect to the illuminated area. This can always be achieved by subdividing the emitting triangles which does not change the final image. We chose this approach over explicitly storing the starting point as well to reduce the dimensionality and thus the size of the map. This also simplifies the integration into an existing rendering system. The algorithm proceeds by calculating an initial image calculating a few samples. After that each new light sample accumulates the remaining image error back into its importance map bin at the light source side. This way, the image error metric is back-projected with a delay of one iteration. The importance map can then be used in the next iteration to importance sample light directions (similar to [CAM08]).

As test scene we chose a distorted glass object casting a large caustic (see Figure 10). The top row of Figure 11 shows three importance maps at three different iterations. The bottom row shows the associated image space error blocks. While the top row is heat-map colored by the importance, the colors in the bottom row are solely to distinguish the different blocks. It can be clearly seen how large areas of the hemisphere do not contribute at all to the final image and also how the termination of regions in image space is reflected in the hemispherical importance map. Figure 12 shows the RMS error graph using no adaptivity at all and using the described back-projected importance map. In this case the number of samples required to get an RMS error below 0.01 is reduced from $570M$ light paths to $160M$ light paths (28%).

4 CONCLUSION

We introduced a stopping condition for Monte Carlo image synthesis which automatically adjusts itself to image resolution, robustly adapts to local variance, and is very simple to implement on top of any rendering system. Due to this simplicity it can be easily integrated in GPU based ray tracing systems. The criterion can also be used to facilitate adaptive sampling. Additionally, we have shown how back-projection of the image error metric onto hemispheres around the light sources can be used to profit from adaptive sampling also when starting paths at the light sources.

In the future, it should be possible to extend the system to complete bi-directional light transport. That is, combine adaptive image space sampling and starting the light paths in important directions based on the back-projected image space error at the time and also consider all deterministic connections. It would also be interesting to extend the method to higher dimensionality, similar to Hachisuka et al. [HJW*08], i.e. to remove the dependency on the framebuffer. Furthermore, a mathematical analysis of the variance in path space should be done to gain some knowledge about the worst case integration error at a given sampling density. This way, the initial sampling rate required to assure no fea-

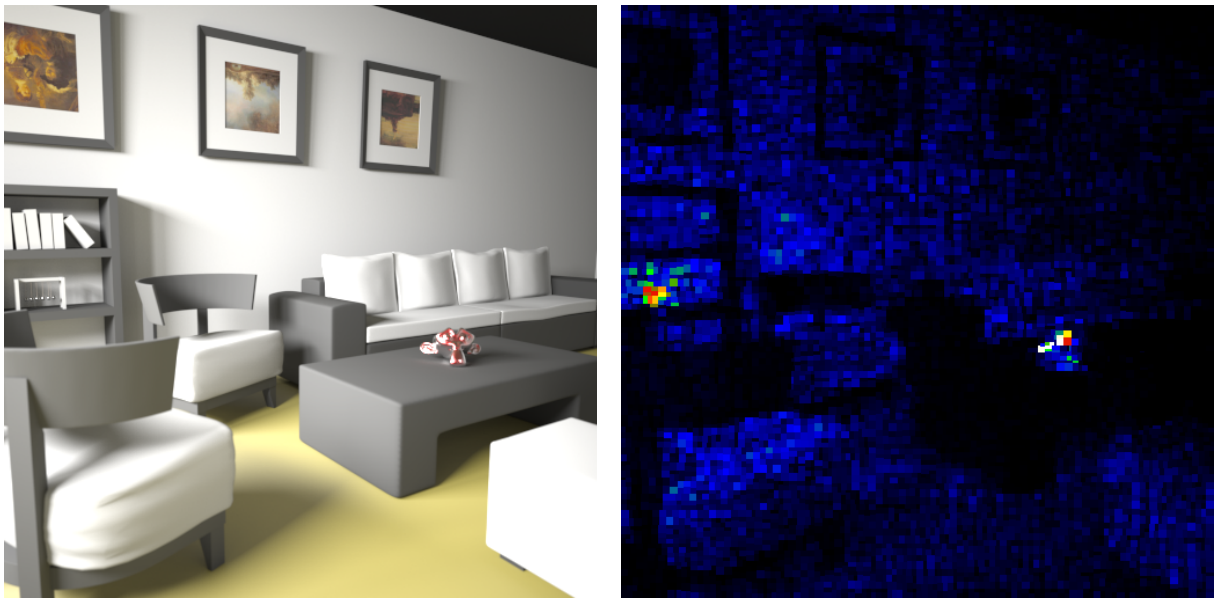


Figure 7: Final image (left) and sample density heat map (right) for the living room scene. The algorithm robustly finds the two spots with significantly higher variance than the rest of the image.

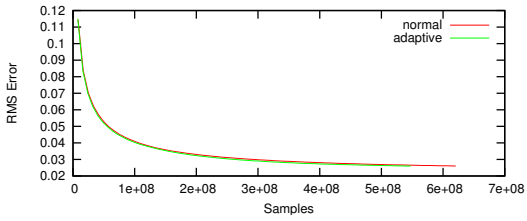


Figure 8: This graph shows the RMS errors of normal and adaptive sampling of the image shown in Figure 7 with increasing number of samples.

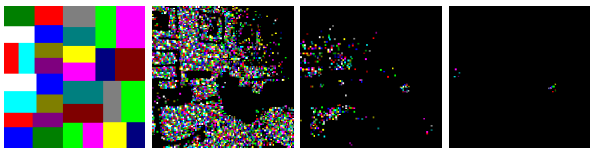


Figure 9: Four images taken from the sequence of active blocks when rendering the living room scene seen in Figure 7. The difficult spots are isolated quickly.

tures are missed when first evaluating the error measure could be determined.

The adaptivity of the method could also be improved upon, if one does not only wish to use it as a termination condition, but also increase efficiency. In analogy to building fast spatial acceleration hierarchies for ray tracing, the image space block splits could be optimized to cut off empty space, i.e. always separate blocks which are likely to terminate in the next iteration.

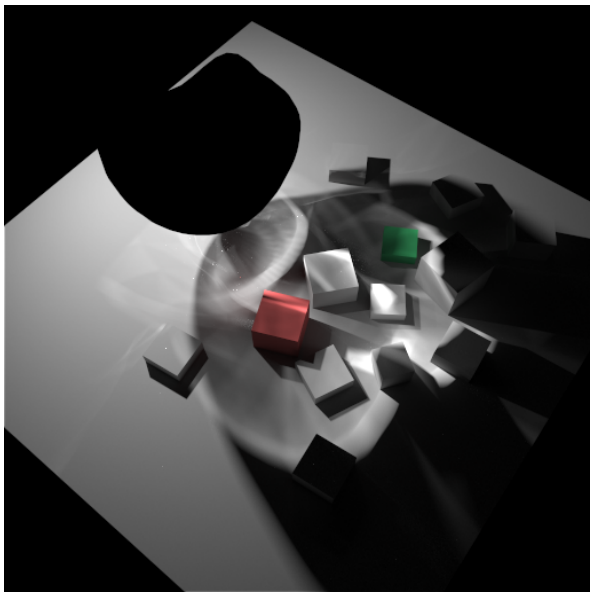


Figure 10: The light tracing test scene. The glass object is black because a pure light tracer cannot deterministically connect singular materials to the camera.

5 ACKNOWLEDGMENTS

This work has been partially funded by mental images GmbH and the DFG Emmy Noether fellowship (Le 1341/1-1).

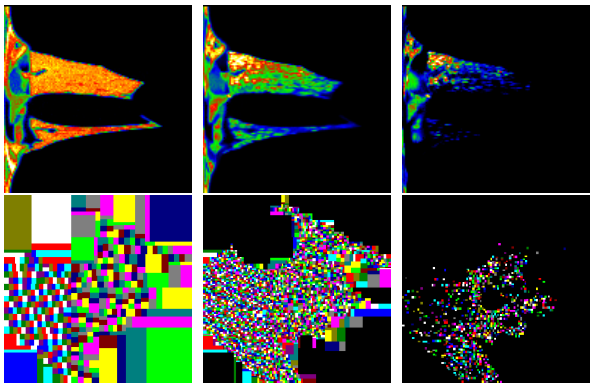


Figure 11: Hemispherical importance map at the light source (top row) and respective image error blocks for the caustic scene (Figure 10).

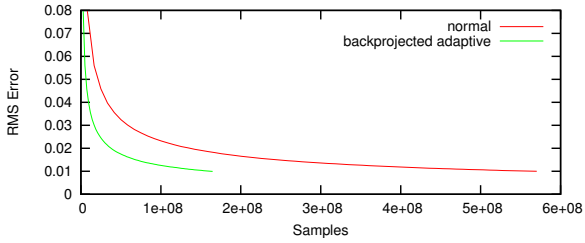


Figure 12: RMS error graph for the light tracing experiment comparing no adaptivity and our proposed back-projection. The curves terminate when the RMS error drops below 0.01.

REFERENCES

[BM98] BOLIN M., MEYER G.: A perceptually based adaptive sampling algorithm. *ACM Transactions on Graphics (Proc. SIGGRAPH 1998)* (1998), 299–309.

[CAM08] CLARBERG P., AKENINE-MÖLLER T.: Practical product importance sampling for direct illumination. In *Computer Graphics Forum (Proc. of Eurographics 2008)* (2008), pp. 681–690.

[Dal93] DALY S.: The visible differences predictor: an algorithm for the assessment of image fidelity. 179–206.

[DS04] DMITRIEV K., SEIDEL H.-P.: Progressive path tracing with lightweight local error estimation. In *Vision, modeling, and visualization 2004 (VMV-04)* (Stanford, USA, 2004), Girod B., Magnor M., Seidel H.-P., (Eds.), Akademische Verlagsgesellschaft Aka, pp. 249–254.

[DW85] DIPPÉ M., WOLD E.: Antialiasing through stochastic sampling. *Computer Graphics (Proc. SIGGRAPH '85)* (1985), 69–78.

[FP04] FARRUGIA J., PÉROCHE B.: A progressive rendering algorithm using an adaptive perceptually based image metric. *Computer Graphics Forum 23* (2004), 605–614.

[HJW*08] HACHISUKA T., JAROSZ W., WEISTROFFER P., DALE K., HUMPHREYS G., ZWICKER M., JENSEN H. W.: Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH '08)* (2008).

[MDMS05] MANTIUK R., DALY S., MYSZKOWSKI K., SEIDEL H.-P.: Predicting visible differences in high dynamic range images - model and its calibration. In *Human Vision and Electronic Imaging X, IS&T/SPIEś 17th Annual Symposium on Electronic Imaging* (2005), Rogowitz B. E., Pappas T. N., Daly S. J., (Eds.), vol. 5666, pp. 204–214.

[Mit87] MITCHELL D.: Generating antialiased images at low sampling densities. *Computer Graphics (Proc. SIGGRAPH '87)* (1987), 65–72.

[Mys98] MYSZKOWSKI K.: The visible differences predictor: applications to global illumination problems. In *Rendering Techniques '98 (Proc. of the Sixth Eurographics Workshop on Rendering)* (1998), pp. 233–236.

[ODR09] OVERBECK R. S., DONNER C., RAMAMOORTHI R.: Adaptive Wavelet Rendering. *ACM Transactions on Graphics (SIGGRAPH Asia 09) 28*, 5 (2009).

[PS89] PAINTER J., SLOAN K.: Antialiased ray tracing by adaptive progressive refinement. *Computer Graphics (Proc. SIGGRAPH '89)* (1989), 281–288.

[PTVF92] PRESS W. H., TEUKOLSKY S. A., VETTERLING W. T., FLANNERY B. P.: *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, New York, NY, USA, 1992.

[RFS03] RIGAU J., FEIXAS M., SBERT M.: Refinement criteria for global illumination using convex functions. In *Compositional Data Analysis Workshop* (2003).

[RPG99] RAMASUBRAMANIAN M., PATTANAIK S., GREENBERG D.: A perceptually based physical error metric for realistic image synthesis. *ACM Transactions on Graphics (Proc. SIGGRAPH '99)* (1999), 73–82.

[SCCD04] SUNDSTEDT V., CHALMERS A., CATER K., DEBATTISTA K.: Top-down visual attention for efficient rendering of task related scenes. In *In Vision, Modeling and Visualization* (2004), pp. 209–216.

[SGA*07] SUNDSTEDT V., GUTIERREZ D., ANSON O., BANTERLE F., CHALMERS A.: Perceptual rendering of participating media. *ACM Trans. Appl. Percept. 4*, 3 (2007), 15.

[Vea97] VEACH E.: *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, 1997.

[YPG01] YEE H., PATTANAIK S., GREENBERG D.: Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. In *In ACM Transactions on Graphics* (2001), pp. 39–65.

Classification of regions extracted from scene images by morphological filters in text or non-text using decision tree

Wonder Alexandre Luz Alves
Department of Computer Science
Institute of Mathematics and Statistics
University of São Paulo, Brazil
wonder@ime.usp.br

Ronaldo Fumio Hashimoto
Department of Computer Science
Institute of Mathematics and Statistics
University of São Paulo, Brazil
ronaldo@ime.usp.br

Abstract

We present in this work a new method to classify regions extracted from scene images by morphological filters in text or non-text region using a decision tree. Our technique can be divided into three parts. Firstly, we extract a set of regions by a robust scheme based on morphological filters. Then, after a refinement, a set of text attributes is obtained for each region. In the last step, a decision tree is built in order to classify them as text or non-text regions. Experiments performed using images from the ICDAR public dataset show that this method is a good alternative for practical problems involving text location in scene images.

Keywords: text region classification, text localization, text information extraction, morphological filters.

1 INTRODUCTION

In the last years, several algorithms for *text information extraction* (TIE) from scene images have been proposed [4, 9, 7, 12, 19, 18]. In spite of such studies, it is not easy to design a general-purpose TIE system [11] for scene images, since texts that are present in these type of images are considered as an integral part of the scene and their presence are almost always accidental and non-intentional. Owing to this factor, text occurrences in these scene images can be significantly different from one another with respect to slopes, sizes, font styles, illumination and also they can be partially occluded. In Fig. 1, we show some examples of scene images.

Figure 1: Scene Images extracted from the ICDAR dataset [13].

According to Jung et al. [11], the TIE problem can be divided into five subproblems: **(1)** detection; **(2)** localization; **(3)** tracking; **(4)** extraction and enhancement; and **(5)** recognition. Our work focuses on the second subproblem whose main objective is to locate text regions within an input image. Since text regions usually are composed by characters aligned along a line such that for any two adjacent characters there is a high contrast with a uniform background, we can consider, as the first step for solving the problem of text region localization, the problem of localizing region candidates that contain uniform background and similar shapes with high contrast with respect to their background that are aligned along a line. As the second step, we can label these candidates into text or non-text regions using a built classifier. Thus, let us define a *text region candidate* in the following way.

Definition 1 (Text Region Candidate) *Let ε be a small positive real number. A text region candidate R is defined as the smallest involving rectangle with a uniform background that contains objects with similar shapes and high contrast (with respect to their background) positioned along a line such that for any two adjacent objects C_i and C_{i+1} belonging to R, we have that $dist(C_i, C_{i+1}) < \varepsilon$, where dist is a distance function.*

Text location methods can be classified into two types: texture-based [4, 12] and region-based [6, 9, 7, 18, 19]. The texture-based methods assume that the character attributes are different from their background in the text region making them possible to be discriminated. Techniques such as Gabor filter, wavelets, fast Fourier transform can be used to extract text attributes from potential text

regions in order to classify them. The region-based methods [6, 9, 7, 18, 19] use several attributes already present in the image. In general, their procedure can be divided into three parts: (i) obtain a set of region candidates by emphasizing the high contrast between text and their background; (ii) then, merge the obtained candidates by a geometrical analysis of their spatial arrangements; (iii) and finally, determine if the obtained regions are text or non-text using heuristics or classifiers.

In this work, we present a new region-based method to classify regions extracted from scene images by morphological filters in text or non-text region using a decision tree. Our technique can be divided into three parts. Firstly, a set of region candidates (connected components) is selected by using a robust scheme based on morphological filters. A refinement of the obtained region candidates is performed in order to emphasize regions with text presence. Then, a set of text attributes is obtained for each region. Finally, a decision tree is built in order to classify them as text or non-text regions. Experiments performed using images from the ICDAR public dataset show that this method is a good alternative for practical problems involving text location in scene images.

After this short introduction, Section 2 presents a brief review of some related works with the commonly main hypotheses used for text region localization. Then, our method is presented in Section 3. In Section 4, we show the experimental results. Finally, Section 5 concludes this work and points out some future research.

2 RELATED WORK

Region-based methods usually have the strategy of selecting a set of region candidates and then eliminate the non-text regions. In this context, morphological filters have been widely used to extract set of text region candidates [6, 7, 9, 18, 19]. Once these candidates are extracted, some methods use heuristics [6, 9, 19], others classifiers [10, 18] to select the text regions.

Wu et al. [19] proposed a scheme based on morphological filters to locate candidates for text regions based on the high contrast between texts and their background, and afterwards they applied some heuristics to find the actual text regions. Hasan and Karam's method [9] uses morphological filters to extract region candidates and then non-text regions are eliminated by heuristics. Lixu et al. [7, 6] proposed a method based on the difference of top-hat transforms to segment region candidates and then subsequently text regions are reconstructed using conditional dilations. Retornaz and Marcotegui [18] developed a method based on ultimate opening to extract the characters from text regions, and thereafter they use a classifier based on linear Fisher discriminant to classify text regions. Renjie et al. [10] proposed a two stage text region classification: (i) eliminate impossible regions using heuristics; (ii) and then separate text from non-text regions using an SVM classifier. The same ideas to classify text regions are used in [3, 8, 18].

This work, as well as others [9, 10, 18, 19], assumes as true some hypotheses that take account the contrast between text regions and their background and the geometric regularity of the font within the text region. Such hypotheses (called here as *text region hypotheses*) are classified as following:

1. *Contrast*
   (a) There is a contrast between text region and its background.
   (b) The text gray levels within the same region are similar.
2. *Font Geometry*
   (a) Characters within the same region:
   - they have similar heights and widths.
   - they are aligned along a line.
   - distances between any two adjacent characters are similar.
3. *Prior Knowledge*
   (a) A text region consists of at least three characters.

3 PROPOSED METHOD

The proposed method consists of three stages (see Fig. 2): (i) extraction of regions; (ii) selection of regions; and (iii) classification of the selected regions in text and non-text.

Extraction of regions: In this stage, a set of region candidates is extracted from the input image using morphological filters.

Selection of regions: The next stage consists of refining the extracted regions in order to emphasize the most probable text regions. This is done by using heuristics based on the text region hypotheses.

Classification of regions: The last stage of the method consists of obtaining a set of features from the selected regions that will be later used as an input to a decision tree in order to classify them as text or non-text regions.

3.1 Extraction of Region Candidates

The extraction of region candidates from the input image must be orientation invariant and also highly noise tolerant. It is well know that morphological operators can be successfully used to accomplish these tasks. In Fig. 3, we show a simple flow chart of our extraction procedure.

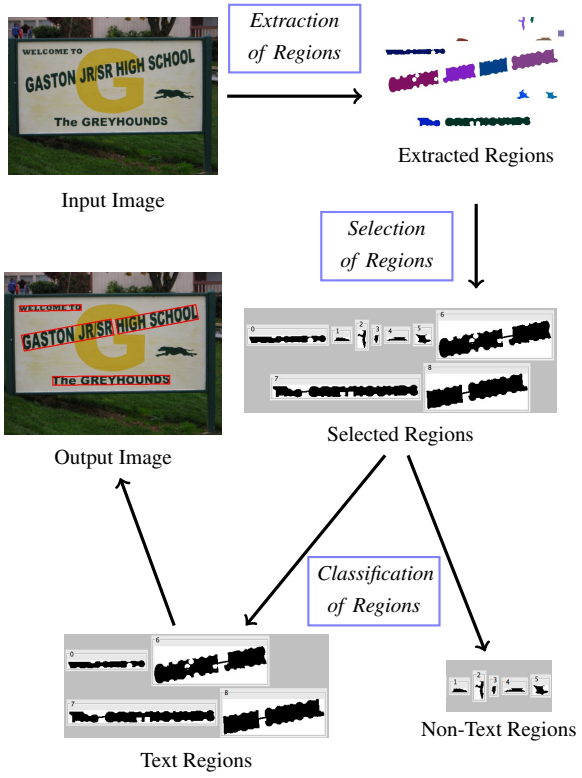


Figure 2: Overview of the proposed method.

The extraction procedure of the region candidates (based on the contrast hypotheses) can be briefly described by the following steps. (1) Firstly, if the input f is a color image, then it is converted into a gray scale image using the following equation [5]:

$$f(x) = \lfloor 0.299 \cdot f_r(x) + 0.587 \cdot f_g(x) + 0.114 \cdot f_b(x) \rfloor,$$

where f_r, f_b, f_g are the three RGB components of the input image. (2) Then, an opening and a closing top-hats are applied to the image f using as the structuring element (SE) the disk of radius λ, producing two output images f_w and f_c, respectively. (3) Afterwards, build the image f_m by taking the maximum between f_w and f_c pixel-by-pixel. The image f_m contains all the characters with thickness less than or equal to the radius λ of the SE. (4) The next step is to apply the closing to the f_m by a 3×3 square SE in order to close small borders within the extracted regions. (5) In the sequence, the image f_m is binarized by the local Otsu's thresholding method [15] obtaining the binary image f_a. (6) Then, an area-opening filter is applied to f_a (producing the binary image f_b) to eliminate connected components smaller than the area of the disk of radius λ. (7) Thereafter, the image f_b is decomposed into n connected components $\Lambda(f_b) = \{R_1, R_2, \ldots, R_n\}$ to be latter used in the next parts. In Fig. 4, we present an example of the application of extraction procedure.

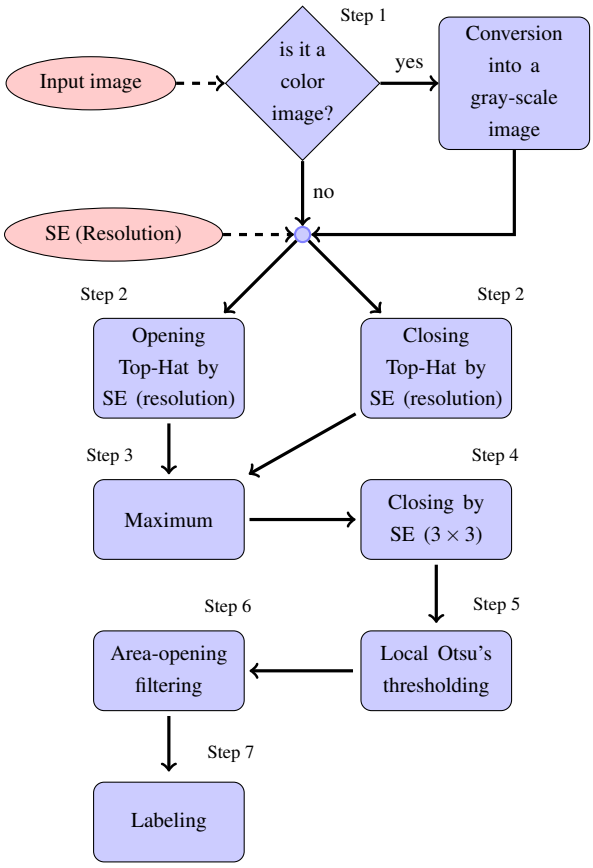


Figure 3: A simple flow chart showing our procedure for extraction of candidate regions.

3.2 Selection and Merging of Regions

After obtaining the set $\Lambda(f_b)$ of region candidates from the input image f, a subset $\zeta \subseteq \Lambda(f_b)$ with the potential text regions are *selected* based on the contrast and geometric hypotheses. Afterwards, a geometric analysis is performed on the selected regions to *merge* the ones that may belong to the same text region.

Selection of Regions Based on the rectangularity criteria [9, 19], we build a subset $\zeta \subseteq \Lambda(f_b)$ containing the potential text regions. Let $R \in \Lambda(f_b)$ be a connected component candidate for a text region and let R^θ denote the rotated version of R along its longest axis [5]. The connected component R is said to be a text region candidate if the following statements hold:

1. the density must be between 0.2 and 0.95, that is,

$$0.2 < \frac{|R|}{W_{R^\theta} \cdot H_{R^\theta}} < 0.95;$$

2. the width of the rotated connected component R^θ must be larger than its height, that is,

$$\frac{W_{R^\theta}}{H_{R^\theta}} > 1.5,$$

(a)

(b)

Figure 4: Extraction of region candidates using morphological filters. (a) Input image; (b) connected components obtained by the application of the extraction region candidate procedure.

where $|R|$ denotes the area of the component R; W_{R^θ} and H_{R^θ} are the width and height of the rotated connected component R^θ along the direction θ, respectively. The first criterion eliminates small and large components that are impossible to be text regions; while the second criterion is a characteristic that must be satisfied by the most characters. These two criteria (based on the works [9, 19]) can eliminate a large number of regions that could not be text regions.

Merging of Regions Two distinct components $R_i, R_j \in \Lambda(f_b)$ belong to the same text region if their orientations, heights and alignments are similar. So, R_i and R_j are in the same text region if the following statements hold:

1. The difference between their orientations must be less than 15 degrees, that is,

$$\begin{aligned} d_\theta(R_i,R_j) &= \min\{|\theta_{R_i}-\theta_{R_j}|, \\ &\quad |\theta_{R_i}-\theta_{R_j}+360|, \\ &\quad |\theta_{R_i}-\theta_{R_j}|-360\} < 15, \end{aligned} \tag{1}$$

   where θ_{R_i} and θ_{R_j} are the orientations of R_i and R_j, respectively. This criterion comes from the work [19].

2. The heights of R_i and R_j must be similar, that is,

$$d_H(R_i,R_j) = |H_{R_i^\theta} - H_{R_j^\theta}| < \min\left\{H_{R_i^\theta}, H_{R_j^\theta}\right\}, \tag{2}$$

   where H_{R^θ} is the height of the rotated component R^θ. This criterion was inspired from the work [18].

3. The third criterion requires that the centroids of R_i and R_j must be similar, that is,

$$\begin{aligned} d_C(R_i,R_j) &= \|CenR_i - CenR_j\|^2 \\ &< \max\left\{W_{R_i^\theta}, W_{R_j^\theta}\right\}, \end{aligned} \tag{3}$$

   where $CentR$ is the position of the centroid of the component R and W_{R^θ} is the width of the rotated component R^θ. This criterion was inspired from the work [19].

4. Let L_R be the longest axis of R obtained by the equation $y = m_R \cdot x + b_R$. Then, the distance between the axes L_{R_i} and L_{R_j} of R_i and R_j, respectively, can be defined as:

$$\begin{aligned} d_L(R_i,R_j) &= \frac{|y_{Cen_{R_j}} - m_{R_i}\cdot x_{Cen_{R_j}} - b_{R_i}|}{2\sqrt{1+m_{R_i}^2}} \\ &+ \frac{|y_{Cen_{R_i}} - m_{R_j}\cdot x_{Cen_{R_i}} - b_{R_j}|}{2\sqrt{1+m_{R_j}^2}}, \end{aligned}$$

   where x_{Cen_R} and y_{Cen_R} are the coordinates of the centroid Cen_R. This criterion requires that

$$d_L(R_i,R_j) < \min\left\{H_{R_i^\theta}, H_{R_j^\theta}\right\}. \tag{4}$$

   This criterion was inspired from the work [19].

Although all these four criteria come from other works [18, 19] (and consequently, they are not so original), the thresholds given by the right side of the last three inequalities (Eqs. 2, 3 and 4) are computed using the values taken from regions instead of single characters or constant numbers (differently from the corresponding original works [18, 19]). This approach makes our method more robust when applied to a set of distinct images.

Based on these four criteria, it is possible to decide if R_i and R_j are merged or not, that is, $R' \leftarrow \{R_i, R_j\}$ will be put into ζ and, at the same time, R_i and R_j are taken out from ζ.

The main advantage of applying the merging of regions is that the attributes (features) taken from regions with larger number of characters are likely to give more information to the classifier that will be used later to discriminate text and non-text regions. Besides, text regions that have been fragmented in the extraction of regions stage can be recovered in this stage.

In Fig. 5, we present an example of the application of these criteria applied to the connected components of the image shown in Fig. 4(a).

(a)

(b)

Figure 5: Results by the application of the selection and merging region candidates. (a) Image containing the labeled connected components of $\Lambda(f_b)$ after candidates extraction; (b) image showing (marked by green rectangles) the selected regions obtained from ζ.

3.3 Classification of Text Region by Decision Tree

After selecting a set ζ that contains all the potential text regions, the next step consists of building a classifier to discriminate whether a given element $R \in \zeta$ is a text or a non-text region. In the following, we will describe how to obtain features from the connected components $R \in \zeta$ that will be later used for classification. Most ideas presented in this section for feature extraction have been taken from the work [9, 19]. Differently from [9, 19], we use these features to construct a vector that will be later used as the input for the decision tree classifier.

Features Extraction The features considered for text and non-text region classification are extracted based on the hypotheses described in Section 2. Given a region $R \in \zeta$, the x-projection can be used to extract geometric features from R [1, 19]. If R is an actual text region, then the characters presented in R have similar widths and heights and in addition their centroids are aligned along the same line. The x-projection technique projects onto a line all pixels within the region R, column by column (see Fig. 6). The deepest valleys provide important information to segment the characters in R. Let XP_R be the vector that stores the x-projection of R. Then, the deepest valleys of XP_R can be detected by thresholding it. In this way, several characters C_i can be extracted from R. Let $\overline{W_R}$ and $\overline{H_R}$ be, respectively, the average width and height of all characters $C_i \in R$. Then, the width and height variances can be calculated, respectively, by the following equations:

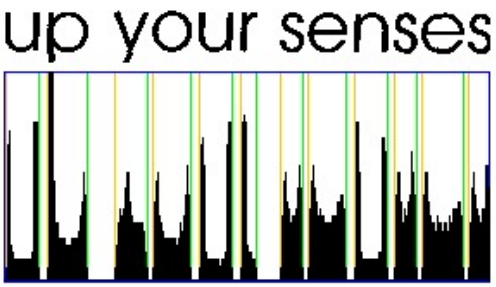


Figure 6: Example of the result by the application of x-projection. Note that yellow and green lines indicate, respectively, the beginning and the end of a character segmentation.

$$\sigma^2_{W,R} = \frac{1}{N_C} \cdot \sum_{C_i \in R} (W_{C_i} - \overline{W_R})^2, \quad (5)$$

$$\sigma^2_{H,R} = \frac{1}{N_C} \cdot \sum_{C_i \in R} (H_{C_i} - \overline{H_R})^2, \quad (6)$$

where N_C is the number of characters within R. In addition, all character centers form a line and satisfy the equation $y = m^C_R \cdot x + b^C_R$, where m^C_R and b^C_R can be easily obtained by linear regression [16]. Then, the linearity of R can be measured by

$$Lin(R) = \frac{1}{N_C} \cdot \sum_{C_i \in R} \frac{|y_{C_i} - m^C_R \cdot x_{C_i} + b^C_R|}{\sqrt{(m^C_R)^2 + 1}}. \quad (7)$$

Besides considering geometric aspects of R, we also take account that all characters within a text region must have similar gray levels. Let μ be the average gray level of all characters $C_i \in R$ within the image input f. Then, the homogeneity of R can be measured by

$$Hom(R) = \frac{1}{N_P} \cdot \sum_{C_i \in R} \sum_{(x,y) \in C_i} (f_{(x,y)} - \mu)^2 \quad (8)$$

where N_P is the number of pixels of all characters $C_i \in R$. This measure was inspired from the work [9].

Therefore, the considered features to classify text and non-text regions are:

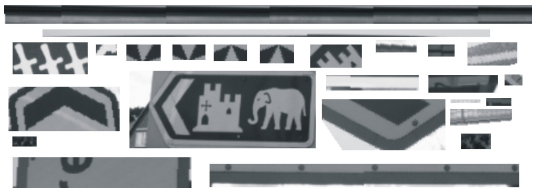
1. *Contrast based feature*
   (a) Gray level homogeneity (Eq. 8).
2. *Font geometric based features*
   (a) Number of characters obtained from the x-projection;
   (b) Height variance of the characters (Eq. 5);

(c) Width variance of the characters (Eq. 6);

(d) Linearity of the characters (Eq. 7).

Classifier Design In this work, a decision tree is used to classify $R \in \zeta$ in text or non-text region. For that, a training set $P = \{(p_i, c_i) \in \mathbb{R}^5 \times \{1,2\} : i = 1,2,\ldots,n\}$ of labeled feature vectors (obtained by the application of the measures described in the previous subsection to text and non-text regions) was built. In Figs. $7(a)$ and $7(b)$, we present some examples of text and non-text regions used to build the set P.

(a)



(b)

Figure 7: Examples of (a) text regions (class 1) and (b) non-text regions (class 2).

We have used the C4.5 algorithm [17] (implemented in WEKA[1] software) for training the decision tree. The training set P has 683 patterns extracted from 50 images of ICDAR dataset [13] in which 203 patterns belong to class 1 (text region) and 480 patterns to class 2 (non-text region). The traditional cross-validation technique has been used to validate the classifier performance [2]. The training error obtained for the built decision tree is 4%.

4 EXPERIMENTAL RESULTS

In order to evaluate the proposed method, we have selected other 84 images from the ICDAR dataset with different colors, illuminations, scales and orientations as well as images with partial occluded text regions. Like other methods [9, 18, 19], the performance of the proposed method is measured in terms of recall and precision rates.

The 84 test images contain 367 text regions (total text regions) with at least 3 characters. Performed tests showed that 323 text regions were correctly classified (true positives) and only 7 were wrongly assigned as text regions (false positives). In this case, the recall and precision rates are 88% and 97%, respectively. Table 1 shows the obtained confusion matrix and Fig. 8 presents some images with text regions marked with red rectangles after the application of the decision tree classifier.

Table 1: Confusion matrix.

Classes	True positive	False positive
Text region	323(88%)	7(1.9%)
Non-text region	(98%)	44(12%)

Since our method builds the set ζ of the potential text regions (see Section 3.2) and this set may not contain all the actual text regions within the input image, we can evaluate the proposed method by calculating the recall and precision rates considering only the text regions in ζ. In our experiment, the cardinality of ζ is 347. This means that 20 text regions of the input images have not been detected by the selection and merging procedure. Thus, in this case, the recall and precision rates are 93% and 97.8%, respectively. Table 2 presents the confusion matrix considering only the selected regions.

Table 2: Confusion matrix (considering only the selected regions).

Classes	True positive	False positive
Text region	323(93%)	7(2.1%)
Non-text region	(97.9%)	24(7%)

In order to have a comparison between our method and at least another one, we have applied the Wu's method [19] to the same image dataset. Perfomed tests showed that only 280 text regions were corrected classified, corresponding to the rate of 77%. However, the Wu's method had a larger number of false positives (75 regions wrongly assigned as text regions) corresponding to the rate of 20%. This happens mainly because Wu's method uses simple thresholds (constant values) in several heuristics for text region classification and consequently it is not sufficiently robust to deal with a large number of images containing text regions with different scales and orientations. Table 3 presents the confusion matrix obtained by Wu's method [19].

Table 3: Confusion matrix of method [19].

Classes	True positive	False positive
Text region	280(77%)	75(20%)
Non-text region	(80%)	87(23%)

We should remark that the performance of Wu's method [19] can be drastically affected in the feature extraction stage where it would have a big difficult to

[1] http://www.cs.waikato.ac.nz/ml/weka/

(a) (b) (c) (d) (e) (f)

Figure 8: Scene images. $(a-f)$ The classified text regions are marked by red rectangles.

extract text regions containing characters that (1) are not horizontally aligned along a line or (2) have different sizes. This is due to the fact that in this stage it uses a fixed (1×7) horizontal line as a SE to perform two morphological operators: an opening and a closing.

We should remark that the performance of Wu's method [19] can be drastically affected in the feature extraction stage, since, at this stage, it uses a fixed (1×7) horizontal line as a SE to perform two morphological operators: an opening and a closing. Consequently, it would have a big difficult to extract text regions containing characters that (1) are not horizontally aligned along a line or (2) have different sizes. Just for a quick comparison, Fig. $9(a)$ shows an application of Wu's method [19] to a scene image; while in Fig. $9(b)$, we show the result for the same image using our method.

5 CONCLUSION AND FUTURE WORK

We presented a new method to classify regions extracted from scene images by morphological filters in

(*a*)

(*b*)

Figure 9: Scene images containing text regions with different scales. Result from the application of (*a*) Wu's method [19] and (*b*) our method.

text or non-text region using a decision tree. Our technique can be divided into three parts. Firstly, we extract a set of regions by a robust scheme based on morphological filters. Then, after a refinement, a set of text attributes is obtained for each region. In the last step, a decision tree is built in order to classify them as text or non-text regions. The obtained results show a good performance of text region classification with the overall recall and precision rates equal to 88% and 97%, respectively. These results show that our method can be a better alternative for text localization in scene images. For future work, we envisage the following points: (*i*) perform a comparative analysis of our method with others using the metric proposed in [14]; (*ii*) propose a scheme for extracting text region candidates with different character sizes presented in the input image using the ultimate closing and opening operators [18]; (*iii*) propose an algorithm to adjust the exact location of the classified text region in the input image.

ACKNOWLEDGEMENTS

We would like to thank the financial support from CNPq (www.cnpq.br) and FAPESP (www.fapesp.br).

REFERENCES

[1] Rodolfo P. dos Santos, Gabriela S. Clemente, Tsang Ing Ren, and George D.C. Cavalcanti. Text line segmentation based on morphology and histogram projection. *Document Analysis and Recognition, International Conference on*, 0:651–655, 2009.

[2] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, second edition, 2000.

[3] Sergio Escalera, Xavier Baró, Jordi Vitrià, and Petia Radeva. Text detection in urban scenes. *Catalan Association of Artificial Intelligence, 12th International Congress of the*, 2009.

[4] Julinda Gllavata, Ermir Qeli, and Bernd Freisleben. Detecting text in videos using fuzzy clustering ensembles. *Multimedia, International Symposium on*, 0:283–290, 2006.

[5] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[6] Lixu Gu. Character region identification from cover images using dtt. In *ICADL*, pages 388–397, 2004.

[7] Lixu Gu, Toyohisa Kaneko, Naoki Tanaka, and R M Haralick. Robust extraction of characters from color scene image using mathematical morphology. *Pattern Recognition, International Conference on*, 2:1002, 1998.

[8] Shehzad Muhammad Hanif and Lionel Prevost. Text detection and localization in complex scene images using constrained adaboost algorithm. *Document Analysis and Recognition, International Conference on*, 0:1–5, 2009.

[9] Y. M. Y. Hasan and L. J. Karam. Morphological text extraction from images. *Image Processing, IEEE Transactions on*, 9(11):1978–1983, 2000.

[10] Renjie Jiang, Feihu Qi, Li Xu, and Guorong Wu. Detecting and segmenting text from natural scenes with 2-stage classification. In *ISDA '06: Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications*, pages 819–824, Washington, DC, USA, 2006.

[11] Keechul Jung, Kwang In Kim, and Anil K. Jain. Text information extraction in images and video: a survey. *Pattern Recognition*, 37(5):977–997, 2004.

[12] Kwang In Kim, Keechul Jung, and Jin Hyung Kim. Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(12):1631–1639, 2003.

[13] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. Icdar 2003 robust reading competitions. *Document Analysis and Recognition, Int. Conf. on*, 2:682, 2003.

[14] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, R. Young, K. Ashida, H. Nagai, M. Okamoto, H. Yamamoto, H. Miyao, J. Zhu, W. Ou, C. Wolf, J. M. Jolion, L. Todoran, M. Worring, and X. Lin. ICDAR 2003 Robust Reading Competitions: Entries, Results and Future Directions. *International Journal on Document Analysis and Recognition - Special Issue on Camera-based Text and Document Recognition*, 7(2-3):105–122, 2005.

[15] J. R. Parker and P. Federl. An approach to licence plate recognition. In *Proceedings of Visual Interface*, 1997.

[16] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical recipes in C: the art of scientific computing*. Cambridge University Press, New York, NY, USA, 1988.

[17] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[18] Thomas Retornaz and Beatriz Marcotegui. Scene text localization based on the ultimate opening. In *Proceedings*, volume 1, pages 177–188. Instituto Nacional de Pesquisas Espaciais (INPE), 2007.

[19] Jui-Chen Wu, Jun-Wei Hsieh, and Yung-Sheng Chen. Morphology-based text line extraction. *Machine Vision Application*, 19(3):195–207, 2008.

Modeling trajectories of free moving objects with smooth flow fields

Mykhaylo Nykolaychuk, Christian Rössl, Holger Theisel, Klaus Richter

Fraunhofer IFF
Magdeburg Germany
{mykhaylo.nykolaychuk,klaus.richter}@iff.fraunhofer.de

Otto-von-Guericke-Universität
Magdeburg Germany
{roessl,theisel}@isg.cs.uni-magdeburg.de

ABSTRACT

We present a new method for smooth global analysis and representation of image-based motion data with topological methods for flow fields, useful for further activity analysis. A video sensor captures motion of subjects and yields discrete samples of velocity vectors in the image domain. We show how to construct a smooth, continuous, globally defined bidirectional flow field which approximates the directional part of given samples. We then apply a topological analysis of this continuous flow field which yields a segmentation into regions of similar flow behavior, i.e., regions of similar motion. This segmentation (topology) can be seen as global model of typical motions and used for further analysis like detecting atypical motions. We tested our method empirically and provide results for two different scenarios with human motion and traffic motion, respectively.

Keywords: **motion analysis, flow field topology**

1 INTRODUCTION

Motion analysis is an important research topic in computer vision with a great variety of applications. In the present scenario, our goal is to "learn" typical motion within a spatial domain that is captured by digital video. Such analysis provides the fundamentals to classify motion and to decide algorithmically whether subjects are moving in an admissible way.

There are many applications for such a tool. Examples are automatic monitoring of traffic, e.g., in a factory environment with self-directed vehicles and human workers interacting. The aim is to detect potentially dangerous situations such that action can be taken to avoid any damage or injuries.

Another example is automatic control of security areas like airports, where atypical behavior of subjects within the moving crowd is detected. It is quite easy to imagine that vector fields may provide a good tool to model such crowd movement.

In this paper, we present a method for motion analysis based on *vector field topology* which was introduced in [4]. The input is a video sequence from which trajectories of moving subjects are extracted. Figure 1 shows example scenes, arrows indicate possible motion directions. The set of all trajectories are converted into a flow field, more specifically into a tensor field as vector orientation, i.e., their sign, should not have any influence. We then regard the topology of the tensor field to segment it into regions of similar flow behavior. The resulting segmentation may serve as basis, e.g., for methods that detect atypical flow. A typical result of our method is shown in Figure 10.

The remainder of the paper is organized as follows. In Section 2 we start with previous work on the field of image-based motion detection and trajectories classification. In Section 3 we describe the data acquisition process. Section 4 presents our approach to generate a bidirectional vector field from scattered vector samples. Topology extraction is discussed in Section 5, and we show experimental results in Section 6. Section 7 concludes the paper with a summary and discussion on future work.

2 RELATED WORK

Video-based analysis systems generally have a very complex structure. They span different levels of abstraction: from the low-level detection and tracking of moving objects in video streams (trajectories) to the high-level behavior analysis (scene understanding) [16]. A common aim is to describe the observed data and to detect atypical or threatening events in real-time, i.e., to find a high-level interpretation. This becomes complicated when complex situations (i.e. scenes containing many objects and interactions) in a completely unsupervised ways are observed and evaluated.

Our work can be seen as bridge between these two levels. The input to our method are extracted trajectories, and the output is a global representation as the so-called topological skeleton of these motion data, which provides a segmentation.

Analysis of human motion and analysis of traffic are two big areas of research. We refer to surveys in [7] and [11] for detailed overview of human and traffic motion analysis respectively.

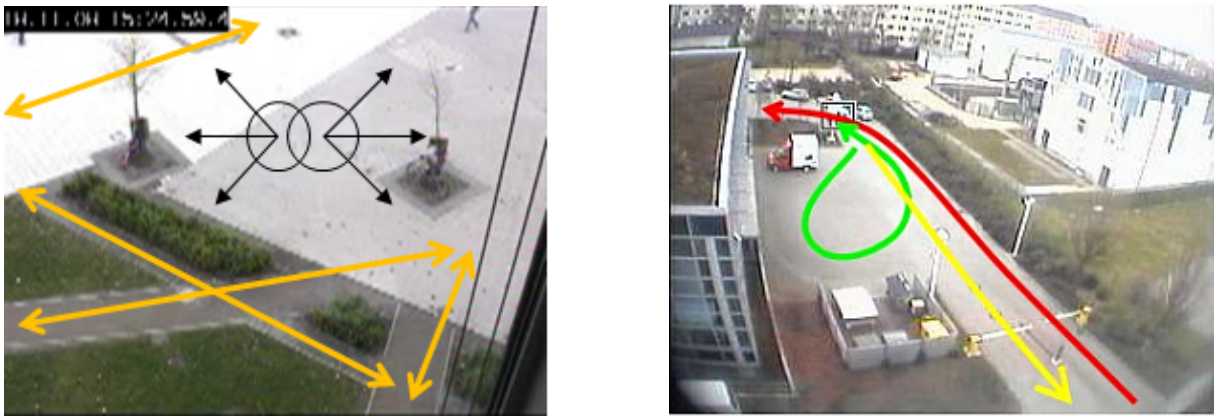


Figure 1: Motion scenes of persons (left) and vehicles (right)

Most related to our work is the approach of Hu et al. [6] which considers environments where tracking of individual objects is hard or even impossible (i.e. crowds): first, flow vectors are computed for each video frame, then after some filtering a global motion flow field is generated, and representative modes (sinks) for each motion pattern are extracted and tracked. A collective global representation of the discovered motion patterns (super tracks) is used for event-based video matching. This approach is suitable when no individual analysis of moving objects is required. In contrast our method provides potential for both, global and individual analysis.

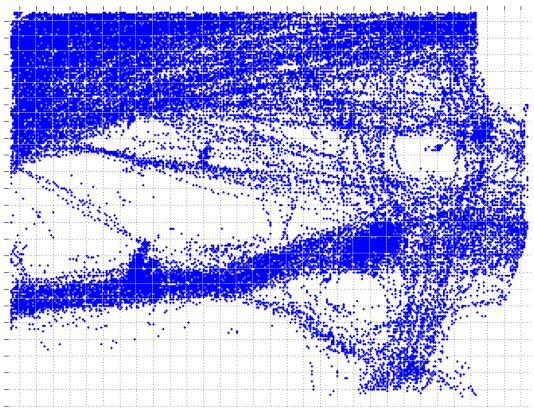


Figure 2: Raw motion data

Motion data analysis using trajectory data gained increasing interest recently [8, 10]. A lot of trajectory classification methods have been developed in the fields of pattern recognition [1] and video surveillance [14]. One important characteristic of those methods are that they use the shapes of whole trajectories to do classification. For example such as the hidden Markov model (HMM) [1] by modeling a whole trajectory with a single mathematical function.

Many approaches employ neural networks such as self-organizing maps (SOM) and use whole trajectories for classification. For instance in [14], each trajectory is encoded to a feature vector using its summary information (e.g., the maximum speed). But some valuable information (e.g., time, motion direction, etc.) could be lost due to this encoding.

Moving-object anomaly detection is a hot topic of research and very closely related to trajectory classification. Li et al. [10] propose an anomaly detection method based on motifs (i.e., trajectory features). This method is limited to extracting motifs without taking other attributes into account.

Morris and Trivedi [12] learn topology scene descriptors (POI) and modeled the activities among POIs with HMMs. The approach is suitable to detect abnormal activities and shows good performance when used in structured scenes.

Pusiol et al. [17] propose a method to learn scene logical regions (scene topology) in an unsupervised way. Topology of the scene is learned using the regions where the person usually stands and stops, so-called slow regions. Transitions among these slow regions are learned as primitive events from which histograms are generated to recognize activities of other subject.

Generation of semantic-based trajectories can be used to simulate real-world behavior. Pfoser and Theodoridis [15] present several examples of how real-world movement characteristics can be described with appropriate semantics. While we do not aim at a semantic analysis in this work, we are confident that our resulting global model of motion may serve as basis for such analysis.

Another approach to classify trajectories is clustering (see, e.g., [8]). However, significant features are likely to appear only locally at parts of trajectories, they do not characterize trajectories globally as a whole. Also discriminative features appear not only as common movement patterns but also as regions.

In contrast to this, we propose a different approach based on a *global* model: significant features are extracted as *topology* of motion data which are represented by a smooth flow field.

3 DATA ACQUISITION

We use a video camera in combination with optical flow to extract motion data from a video sequence. We chose an off-the-shelf video sensor Vitracom SiteView EP for this purpose.

In our setup the sensor is fixed and captures a certain region (see Figure 1). For any detected motion of any subject we are provided with a sequence of a subject's id, its positions in image-space and a time-stamp. These data can be used to construct trajectories of individual subjects. All our computations are performed in image-space which is sufficient for our purpose as in this work we do not combine data from multiple sensors.

Due to the complexity of the scene the acquired data suffers from disturbing effects like distortion, overlapping, occlusion, or fusion of moving objects. In order to extract reliable raw data, we apply a smoothing step on the individual trajectories. The rationale for this preprocess consists firstly in the fact that filtering in this way respects the temporal and spatial coherence as motions are processed individually. Secondly, we can rely on a simple and efficient finite differences scheme for curve smoothing rather than smoothing scattered vector data. In order to achieve this, we minimize a certain bending energy for trajectory smoothing.

Finally, we overlay velocity samples from all trajectories within the image domain. The result is a set of vector samples scattered in the domain. Figure 2 shows such raw data.

4 VECTOR FIELD GENERATION

While the scattered vector samples describe our vector field, we cannot apply them directly for topology extraction and segmentation. We need to generate a suitable continuous parametric model from the given discrete set of data points. A standard approach is least-squares approximation of a bivariate function, e.g., tensor-product B-spline.

Unfortunately, this approach does not work in our setting: our goal is fitting a *bidirectional* vector field (or a tensor field), i.e., we want to fit accurately the direction of velocity vectors but not their orientation. This *orientation-invariance* cannot be modeled by a linear least-squares fit.

In the following, we provide a different formulation of the problem which leads to solving an eigenvalue problem.

Given are samples of velocity vectors $\mathbf{v}_i = (u_i, v_i)^\top \in \mathbb{R}^2, i = 1,\ldots,m$ at points $(x_i, y_i)^\top$ in the image plane. We want to find a parametric vector field

$$\mathbf{w}(x,y) = \sum_{j=1}^{n} b_j(x,y)\,(U_j, V_j)^\top \;, \qquad (1)$$

such that $\mathbf{w}(x_i, y_i)$ is as parallel as possible to $\mathbf{v}_i$. Here, $b_j(\cdot)$ represent any suitable bivariate basis functions, e.g., tensor-product B-splines. And we have to determine the unknown coefficients $U_j, V_j,\ j = 1\ldots,n$.

Note that we are only interested in matching the direction of vectors and disregard both, orientation and magnitude. The first property is essential to our approach, while the latter one can easily be restored in a second step by a linear least-squares fit of a scalar magnitude field.

4.1 Eigenvalue problem

In order to model the problem as a minimization, we reformulate it: instead of $\mathbf{w}$ and $\mathbf{v}$ being parallel, we require that $\mathbf{w}(x_i, y_i)$ are as *orthogonal* to $\mathbf{v}_i^\perp =: (-v_i, u_i)^\top$ as possible, where $(\cdot)^\perp$ denotes a rotation by $\frac{\pi}{2}$. For the best fitting bidirectional vector field we have

$$E = \sum_{i=1}^{m} \left(\mathbf{w}(x_i, y_i)^\top \mathbf{v}_i^\perp \right)^2 \to \min \;, \qquad (2)$$

i.e., we penalize non-orthogonality. We rewrite (2) in matrix notation such that

$$E = \mathbf{r}^\top \mathbf{r} \;.$$

Then we have each element of $\mathbf{r} \in \mathbb{R}^m$ as

$$\begin{aligned} \mathbf{r}_i &= \mathbf{w}(x_i, y_i)^\top \mathbf{v}_i^\perp \\ &= -v_i\, \mathbf{b}(x_i, y_i)^\top \mathbf{U} + u_i\, \mathbf{b}(x_i, y_j)^\top \mathbf{V} \\ &= \left(u_i \mathbf{b}(x_i, y_i)^\top \right) \mathbf{V} - \left(v_i \mathbf{b}(x_i, y_i)^\top \right) \mathbf{U} \;. \end{aligned}$$

Here, we write the two components of $\mathbf{w}(x,y)$ in (1) as scalar products $\mathbf{b}(x_i, y_i)^\top \mathbf{U}$ and $\mathbf{b}(x_i, y_j)^\top \mathbf{V}$, respectively, with a vector $\mathbf{b} = (b_1(x,y), \ldots, b_n(x,y))^\top$ of basis functions and coefficient vectors $\mathbf{U} = (U_1, \ldots, U_n)^\top$ and $\mathbf{V} = (V_1, \ldots, V_n)^\top$.

Then $\mathbf{r}$ is expressed as

$$\mathbf{r} = \mathbf{C}\mathbf{V} + \mathbf{D}\mathbf{U} = (\mathbf{C}, \mathbf{D})\,(\mathbf{U}, \mathbf{V})^\top \;,$$

and the definition of the matrices $\mathbf{C}, \mathbf{D} \in \mathbb{R}^{m \times n}$ is given by the element-wise notation above.

For the quadratic energy term $E = \mathbf{r}^\top \mathbf{r}$ we obtain

$$\begin{aligned} E &= (\mathbf{V}^\top, \mathbf{U}^\top) \begin{pmatrix} \mathbf{C}^\top \mathbf{C} & \mathbf{C}^\top \mathbf{D} \\ \mathbf{D}^\top \mathbf{C} & \mathbf{D}^\top \mathbf{D} \end{pmatrix} \begin{pmatrix} \mathbf{V} \\ \mathbf{U} \end{pmatrix} \\ &=: \mathbf{x}^\top \mathbf{A}\, \mathbf{x}^\top \;. \end{aligned}$$

It is obvious that the global minimum $E = 0$ is attained at $\mathbf{x}^\top = (\mathbf{V}^\top, \mathbf{U}^\top) = \mathbf{0}$. This trivial solution, however, is not the one we are looking for. By adding the additional constraint $||\mathbf{x}||^2 = 1$ we avoid the trivial solution.

(Note that we are interested only in directions of vectors and not in their magnitude, so there is an infinite number of feasible solutions which differ only in scale. Hence, we may restrict the above norm of $\mathbf{x}$ to any positive number.)

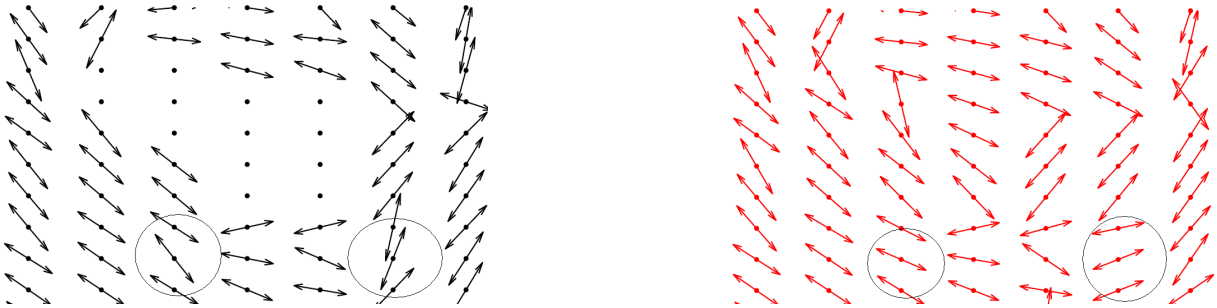


Figure 3: Vector field without (left) and with (right) regularization.

Introducing the Lagrange multiplier λ the minimization problem now reads as

$$E' = \mathbf{x}^\top \mathbf{A}\,\mathbf{x}^\top - \lambda(\mathbf{x}^\top \mathbf{x} - 1) \to \min$$

The matrix $\mathbf{A}$ is symmetric and positive definite, and we find the minimum by setting the gradient $\nabla E' = 0$, which yields

$$\mathbf{A}\mathbf{x} - \lambda \mathbf{x} = 0\,,$$

and which in turn means that the minimum is attained for an *eigenvector* of $\mathbf{A}$. In fact, we are looking for the eigenvector corresponding to the smallest eigenvalue.

There are efficient numerical methods for solving the eigenvalue problem, i.e., for computing the smallest eigenvalue and hence a single eigenvector.

While the dimensions of the matrix $\mathbf{A} \in \mathbb{R}^{2m\times 2m}$ can be rather large — on a $N \times N$ grid we have $m = N^2$ — $\mathbf{A}$ is sparse if the basis functions $b_j(x,y)$ have local support which is the case for B-splines. State-of-the art algorithms exploit the sparsity pattern of the system matrix [9].

In summary, we compute a parametric vector field $\mathbf{w}(x,y)$ from discrete samples $\mathbf{v}_i$ by solving an eigenvalue problem. The solution satisfies the condition that $\mathbf{w}$ is as parallel as possible to vectors $\mathbf{v}_i$ in least-squares sense.

4.2 Regularization

There may be relatively large parts of the domain, which do not contain any vector samples. We are aware of the fact that the approximation method described above may not be able to "interpolate" meaningful vectors across such regions. (Instead, the solution to the eigenvalue problem yields arbitrary vectors of very small magnitude, see Figure 3 (left).)

This is a well-known effect that depends on the particular choice of basis functions: roughly speaking, one cannot expect a contribution from basis functions without having enough samples within their support. For B-splines, this means the Schoenberg-Whitney conditions should be satisfied, which is generally not the case in our setup (see, e.g., [5]) The standard approach to solve this problem is adding a regularization term, i.e., taking into account additional constraints usually on smoothness. We penalize the norm of first order partials. The eigenvalue problem then reads as

$$(\mathbf{A} + \alpha \mathbf{R}^\top \mathbf{R})\mathbf{x} - \lambda \mathbf{x} = 0\,,$$

where $\mathbf{R}^\top \mathbf{R}$ captures the regularization term, and $\alpha > 0$ is a small weight. Figure 3 shows the effect of regularization.

4.3 Velocity bias

The formulation as is effectively yields a weighted least-squares approximation: (2) does not strictly penalize angles but includes the magnitude of the samples. For our purpose, such weighting is neither meaningful nor does it provide better results. There is no reason, why faster movement should influence the fitting of the directional component. For this reason, we normalize all samples such that $||\mathbf{v}_i|| = 1, i = 1,\ldots,m$. Note that this does not interfere with recovering speed as a scalar value: this would be second step resulting in least-squares fitting a scalar field. As we are interested in vector field topology, speed is currently not of interest to us.

4.4 Discussion of our setting

The formulation of the problem allows for an arbitrary choice of basis functions b_j. A natural and well-established choice are tensor product B-splines. Higher polynomial degree q yields higher order smoothness, i.e., C^{q-1} continuity, of the bidirectional flow field $\mathbf{w}$ as well as higher order approximation. In our experiments with $q = 1,2,3$ we found that in fact the simplest choice $q = 1$, i.e., a bi-linear function $\mathbf{w}$ shows sufficiently good results. At the same time, this is the most efficient model to evaluate. In some situations we even observed that higher order smoothness does not capture "turbulent" regions of the sample set as well, because due to larger support, more samples with potentially diverging directions are taken into account.

Regions without any data, i.e., regions with no observed motion, should be masked out for our application, they provide trivial segments. Regularization generally yields plausible data for such regions and improves vector field near these segment boundaries.

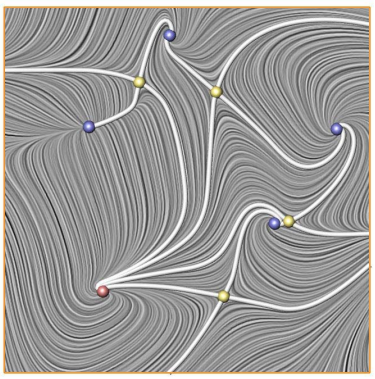


Figure 4: Example of a topological skeleton of a vector field

However, we found that for our purpose the difference in using or not using a regularization term is not significant.

5 TOPOLOGY EXTRACTION

In this section we show how to extract the topological skeleton of the smooth bidirectional flow field $\mathbf{w}$. This skeleton provides the segmentation of the image domain in regions of similar motion.

5.1 Topological skeleton

One of the most important features of a vector field is its *topological skeleton* which has been introduced as a visualization tool in [4]. The topological skeleton of a 2d vector field $\mathbf{v}(x,y)$ essentially consists of a collection of *critical points*, i.e., $\mathbf{v}(x,y) = \mathbf{0}$ and special stream lines called *separatrices* which separate the flow into areas of different flow behavior.

The attractiveness of the topological skeleton as a visualization tool lies in the fact that even a complex flow behavior can be expressed (and visualized) by using only a limited number of graphical primitives. At the same time, the topological skeleton yields a segmentation of the domain.

Helman and Hesselink [4], consider first order critical points, i.e. critical points with a non-vanishing Jacobian. Based on an eigenvector analysis of the Jacobian matrix, these critical points are classified into *sources*, *sinks*, *centers* and *saddles*. For the description of the topology, saddle points are of particular interest. In addition so called *boundary switch points* [2] may separate regions of different inflow/outflow behavior across the boundary of the flow.

Figure 4 shows an example of a vector field together with the topological skeleton. For an excellent introduction to vector field topology and related computational methods we refer to [19].

5.2 Extraction of the topological skeleton

For extracting the topological skeleton, we proceed in two steps. First, we find critical points. Then we integrate streamlines staring from critical points to construct separatrices and hence segment boundaries.

Finding critical points.

We consider a bidirectional flow which is essentially a 2nd order symmetric *tensor field* $\mathbf{T}(x,y)$ where

$$\mathbf{T} = \begin{pmatrix} w_1(x,y)^2 & w_1(x,y)\,w_2(x,y) \\ w_1(x,y)\,w_2(x,y) & w_2(x,y,)^2 \end{pmatrix},$$

with $\mathbf{w}(x,y) = (w_1(x,y), w_2(x,y))^\top$.

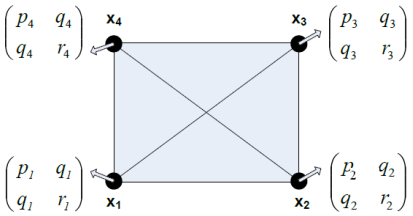


Figure 5: Cell subdivision into triangles

Then critical points (or *degenerate points*) are points (x_c, y_c) in the domain where the eigenvalues λ_1 and λ_2 of $\mathbf{T}(x_c, y_c)$ are equal and hence we have

$$\mathbf{T}(x_c,y_c) = \mathbf{Q}\begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix}\mathbf{Q}^\top = \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix},$$

where $\lambda = \lambda_1 = \lambda_2$ [3, 18]. This means, in degenerate points *any* vector is an eigenvector to $\mathbf{T}$. This is valid in any coordinate system $\mathbf{Q}^\top$.

Such points are characterized by the two conditions

$$\begin{cases} \mathbf{T}_{11}(x_c,y_c) - \mathbf{T}_{22}(x_c,y_c) = 0 \\ \mathbf{T}_{12}(x_c,y_c) = 0 \end{cases}$$

Finding critical points corresponds to finding roots of the two components of the above equations.

Finally, critical points are classified in *wedges* or *trisectors* based on certain partials:

$$a = \frac{\partial}{\partial x}(\mathbf{T}_{11} - \mathbf{T}_{22}),\quad b = \frac{\partial}{\partial y}(\mathbf{T}_{11} - \mathbf{T}_{22}),$$
$$c = \frac{\partial}{\partial x}\mathbf{T}_{12},\quad d = \frac{\partial}{\partial y}\mathbf{T}_{12},$$

(see [3]), with $\delta = ab - cd$, for a wedge $\delta > 0$ and for a trisector: $\delta < 0$.

The detection of critical points consists essentially of a numerical root finding algorithm. As we are not interested in finding higher order critical points, we simplify the setting such that elements of the tensor field $\mathbf{T}$ are interpolated independently.

Even more, we prefer linear interpolation to bi-linear interpolation within grid cells (or generally higher order interpolation). For linear interpolation we subdivide each quad cell uniformly into either two (inserting

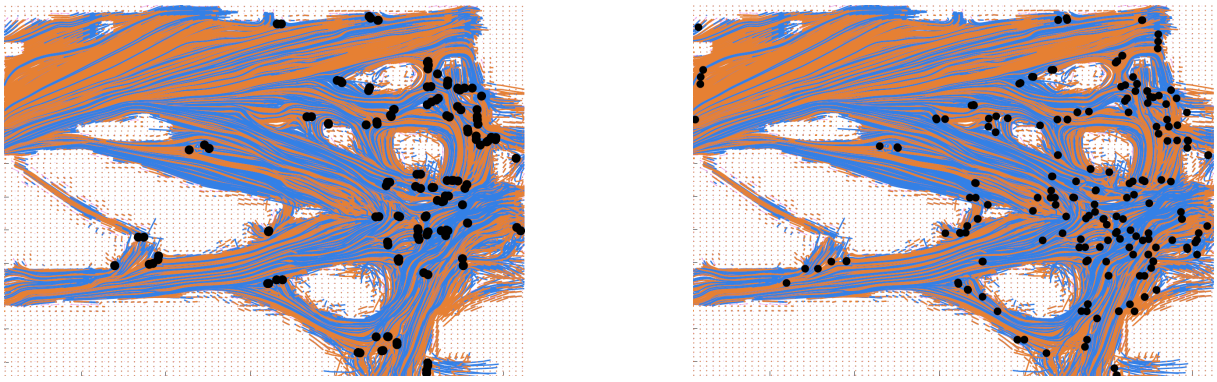


Figure 6: Detected degenerate points for linear (left) and bilinear (right) interpolation.

the diagonal) or four (inserting the quad's barycenter, Figure 5) triangles and formulate the above condition in terms of barycentric coordinates. This is due to two reasons: firstly, we have to consider a lower polynomial degree for root finding, which makes this task more efficient. And secondly, the number of detected critical points decreases, and we find exactly one or no (first order) critical point within each linear piece.

Figure 6 compares amount and distribution of critical points for linear and bi-linear interpolation. In this example, we detect 62 and 145 degenerate points for linear interpolation splitting each cell into two or four (as shown in the Figure 5) triangles, respectively. Bilinear interpolation yields 228 degenerate points. For our results we use linear interpolation and $1:4$ subdivision of cells as this constitutes a good compromise between efficiency and finding all relevant points.

Finding separatrices.

Separatrices are streamlines which emanate from degenerate points in two (wedge) or three (trisector) directions. We use a 4th order Runge-Kutta method for numerical integration on $\mathbf{w}$. The standard algorithm for vector field integration is modified such that the orientation is chosen at the starting point and kept consistent during integration.

6 RESULTS

We tested our approach with two real-world data sets where motion of vehicles and motion of human subjects were captured, respectively (see also [13]). The video sensor is fixed for both setups and provides a perspective view of the scenes (Figure 8 and 10, right).

The vehicle data was acquired over a 24 hour period. Vehicle motion is more homogeneous in this case. Humans were captured in the area where many people walk in different directions over a 30 minutes time interval.

The motion data is rather complex: there are several obstacles, and persons can move freely around them, in general. However, we expect most motion along several paths though the region. Computations were performed on 56×56 (vehicles) and 53×53 (human subjects) grids, respectively. Our current, non-optimized implementation requires approximately 5 seconds for computation of each, the flow field and the critical points. Integration of separatrices takes about 20 seconds.

Figure 7 (left) shows the raw data of vehicle movements as acquired from the sensor. The least-squares approximated bidirectional vector field is shown in Figure 7 (right). There are homogeneous regions in the scattered raw data, such as the region near the gateway, which results in homogeneous vector field regions. Regions of less homogeneous motion, e.g, where particular trajectories intersected, are less homogeneous in the bidirectional vector field. Regions without motion data yield zero vectors in the vector field. This shows effectiveness of the bidirectional vector field model and its generation.

Figure 8 visualizes the extracted tensor field topology for the moving vehicles on the grid (left) and as an overlay on the perspective image (right). Regions of similar motion are well-separated, and this example shows nicely that the topological skeleton (black) conveys the properties of the vector field very well with only a limited number of primitives. There are rather smooth regions where vehicles enter the scene at the gateway. Fundamental motions such as vehicles turning are present. There are also several regions with high amount of degenerate points: in fact, these regions represent critical regions within the domain. The white circle marks an area with a barrier that lets the cars in and out. The region near the trash cans marked by the black circle is highly frequented by both, vehicles and human subjects.

The motion of human subjects shown in Figure 9 is more complex, hence the resulting vector field is less homogeneous. Figure 10 visualizes topology of the motion and makes it straightforward to detect regions with no motion, homogeneous and less homogeneous motion. The white circle marks a region where different directions of motion intersect, and hence this region shows a rather complex behavior. Note that such arbi-

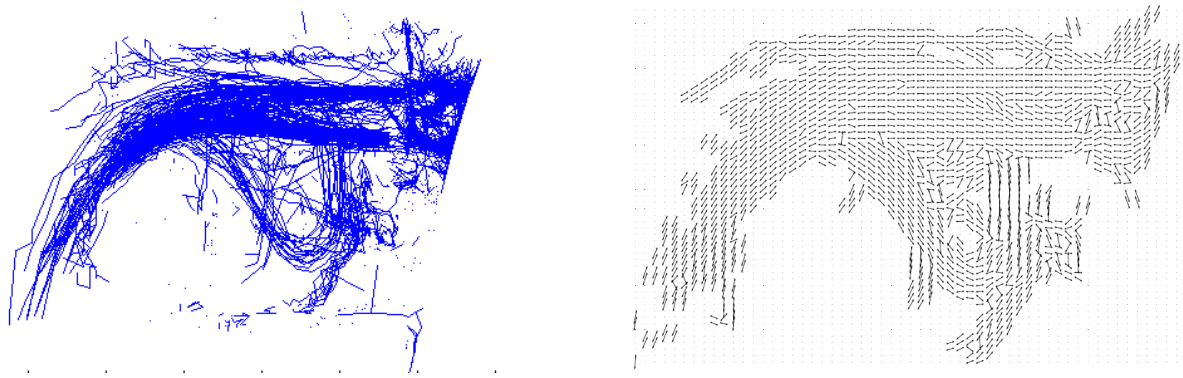


Figure 7: Raw data (left) and generated vector field (right)

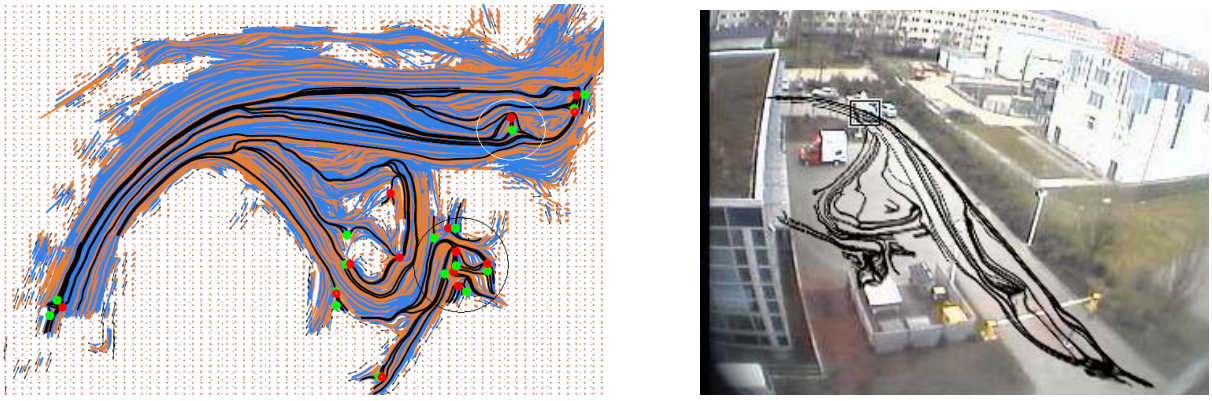


Figure 8: Topology of vehicles motion trajectories

trary intersections cannot be represented accurately by the underlying tensor field. In fact the whole region should be represented by a single (higher order) critical point. Such interpretation of the results and classification is left for future work.

In summary, we observe that the topological skeleton provides a meaningful segmentation of motion within the scene and give a global representation of the monitored area.

7 CONCLUSIONS

We demonstrated how methods for analyzing flow fields can be applied to analysis of motion data. In order to achieve this, we first approximate scattered velocity data with a smooth bidirectional vector field, i.e., essentially with a tensor field. Then we extract its topological structure for segmentation of the domain into regions of similar motion. Topological skeleton is an appropriate global representation of typical motions in a monitored area. Our results prove the efficiency of this approach.

The output of our method may serve as a basis for further analysis. Assume that the extracted topological skeleton reflects a typical or normal condition for the observed region. The change of the skeleton, i.e. the deviation from normal movements can then be quickly identified. Another application of our method is classification of individual movements or extraction of basic movements. This requires additional object tracking.

For future work we plan to extract fundamental motions of our data which enables on-the-fly classification of trajectories in categories of similar motion. At the same time newly acquired motion should be integrated in the current segmentation, i.e. real-time update of topological skeleton. Conversely, such classification could be used to fit flow fields for every motion category which in turn can be used to improve the segmentation. It might be useful to simplify and/or smooth the topological skeletons, especially when data from multiple sensors or data from a moving sensor are combined.

An ultimate goal is to infer specific actions from motion data in order to identify non-admissible actions in any kind of security area.

ACKNOWLEDGMENTS

The authors are partially funded by the German Federal Ministry of Eductaion and Research (BMBF) within the ViERforES project (no. 01IM08003C).

REFERENCES

[1] F.I. Bashir, A.A. Khokhar, and D. Schonfeld. Object trajectory-based activity classification and recognition using hidden markov models. *IP*, 16(7):1912–1919, July 2007.

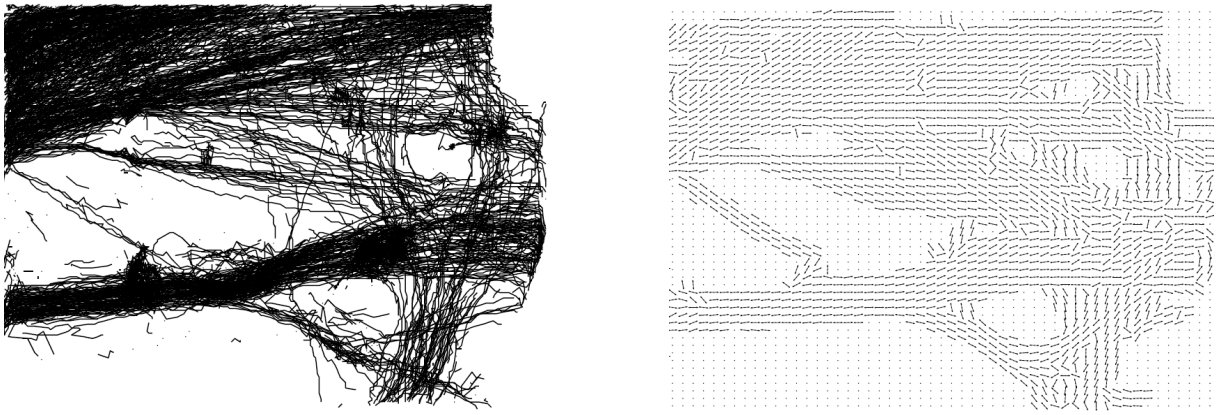


Figure 9: Raw motion trajectories and estimated vector field

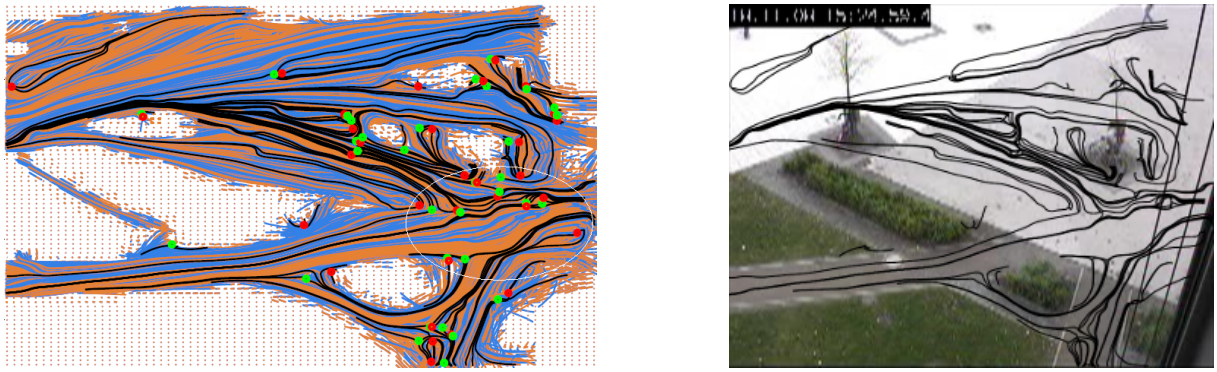


Figure 10: Topology of persons motion trajectories

[2] W. de Leeuw and R. van Liere. Collapsing flow topology using area metrics. In D. Ebert, M. Gross, and B. Hamann, editors, *Proc. IEEE Visualization '99*, pages 149–354, Los Alamitos, 1999.

[3] T. Delmarcelle and L. Hesselink. The topology of symmetric, second-order tensor fields. In *IEEE Visualization*, pages 140–147, 1994.

[4] J. Helman and L. Hesselink. Representation and display of vector field topology in fluid flow data sets. *Computer*, 22(8):27–36, 1989.

[5] Josef Hoschek and Dieter Lasser. *Grundlagen der geometrischen Datenverarbeitung*. B.G.Teubner, Stuttgart, 1992.

[6] Min Hu, Saad Ali, and Mubarak Shah. Detecting global motion patterns in complex videos. In *ICPR*, pages 1–5, 2008.

[7] V. Kastrinaki, M. Zervakis, and K. Kalaitzakis. A survey of video processing techniques for traffic applications. *Image and Vision Computing*, 21:359–381, 2003.

[8] Jae-Gil Lee, Jiawei Han, Xiaolei Li, and Hector Gonzalez. Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering. *PVLDB*, pages 1081–1094, 2008.

[9] R. B. Lehoucq, D. C. Sorensen, and C. Yang. Arpack users' guide: Solution of large scale eigenvalue problems with implicitly restarted arnoldi methods., 1997.

[10] Xiaolei Li, Jiawei Han, Sangkyum Kim, and Hector Gonzalez. Roam: Rule- and motif-based anomaly detection in massive moving object data sets. In *Proceedings of 7th SIAM International Conference on Data Mining*, 2007.

[11] Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2):90–126, 2006.

[12] Brendan T. Morris and Mohan M. Trivedi. Learning and classification of trajectories in dynamic scenes: A general framework for live video analysis. In *AVSS '08: Proceedings of the 2008 IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance*, pages 154–161, 2008.

[13] Mykhaylo Nykolaychuk. Analyse von Bewegungstrajektorien mit Techniken der Strömungsvisualisierung. Master's thesis, Otto-von-Guericke Universität Magdeburg, Fakultät für Informatik, June 2009.

[14] Jonathan Owens and Andrew Hunter. Application of the self-organizing map to trajectory classification. In *VS '00: Proceedings of the Third IEEE International Workshop on Visual Surveillance (VS'2000)*, page 77, 2000.

[15] Dieter Pfoser and Yannis Theodoridis. Generating semantics-based trajectories of moving objects. In *International Workshop on Emerging Technologies for Geo-Based Applications*, pages 59–76, 2000.

[16] C. Piciarelli and G. L. Foresti. On-line trajectory clustering for anomalous events detection. *Pattern Recognition Letters*, 27(15):1835–1842, 2006.

[17] Guido Pusiol, Francois Bremond, and Monique Thonnat. Trajectory based primitive events for learning and recognizing activity. In *Second IEEE International Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS2009)*, 2009.

[18] X. Tricoche, G. Scheuermann, and H. Hagen. Scaling the topology of symmetric, second-order planar tensor fields. In G. Farin, B. Hamann, and H. Hagen, editors, *Hierarchical and Geometrical Methods in Scientific Visualization*, pages 171–184. Springer, Berlin, 2002.

[19] Tino Weinkauf. *Extraction of Topological Structures in 2D and 3D Vector Fields*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik (H. Theisel), March 2008.

Facial Feature Detection and Tracking with a 3D Constrained Local Model

Meng Yu
University of St. Andrews, U.K.
yumeng@cs.st-andrwes.ac.uk

Bernard P. Tiddeman
University of St. Andrews, U.K.
bpt@cs.st-andrews.ac.uk

ABSTRACT

In this paper, we describe a system for facial feature detection and tracking using a 3D extension of the Constrained Local Model (CLM) [Cris 06, Cris 08] algorithm. The use of a 3D shape model allows improved tracking through large head rotations. CLM uses a joint shape and texture appearance model to generate a set of region template detectors. A search is then performed in the global pose / shape space using these detectors. The proposed extension uses multiple appearance models from different viewpoints and a single 3D shape model. During fitting or tracking the current estimate of pose is used to select the appropriate appearance model. We demonstrate our results by fitting the model to image sequences with large head rotations. The results show that the proposed 3D constrained local model algorithm improves the performance of the original CLM algorithm for videos with large out-of-plane head rotations.

Keywords: Active appearance models, Multi-view face models, Constrained local model, Facial feature tracking, Facial feature detection

1 INTRODUCTION

This paper describes a method for tracking human face features using a 3D shape model and view-dependent feature templates. We match the 3D face model to previously unseen 2D video sequences of human faces by applying a shape constrained search method, using an extension of the constrained local model algorithm.

The original CLM algorithm [Cris 06] works with limited rotations from the front face view. The extension to the algorithm proposed here works not only on the front face view but also on the face with large head rotations in videos. The proposed multi-view CLM consists of a 3D shape model and several 2D texture models from multiple views.

In our implementation, the shape model is first given some suitable initialisation (approximate rigid body alignment, scaling). In each subsequent iteration square region are sampled around each feature point and projected into the allowed appearance model space. The shape and pose parameters are then found that maximise the correlation between the synthesised appearance template patches and patches extracted around the current estimates of the feature point locations in image space. The proposed algorithm is a view based, in that we switch appearance models depending on the current estimate of the face orientation.

After a brief review of face and face feature detection, we will describe the model building and fitting methods in more detail, followed by experimental results demonstrating the performance of the proposed multi-view CLM method.

WSCG 2010 conference proceedings,
WSCG'2010, February 1 – February 4, 2010
Plzen, Czech Republic.

2 RELATED WORK

The problems of facial feature detection and tracking have received a great deal of attention in the literature, here we only cover the more immediately relevant work. Active Shape Models (ASM) [Coot 95] use Principal Component Analysis (PCA) to learn the main axes of variation from a training set of labelled examples. Fitting the shape model to a new image involves local searches for matching features alternated with projection of the shape estimate back into the allowed model space.

Active Appearance Models(AAMs) [Coot 98, Coot 01a] use the same PCA based shape model as ASMs together with a PCA based model of appearance (i.e. shape normalised texture). It has been used for face modelling and recognising objects [Lani 97, Jone 98], fitting unseen images [Gros 05, Peyr 07], tracking objects[Ahlb 01, Steg 01] and medical image processing [Coot 01b, Mitc 01]. The original implementation [Coot 01a] learnt a linear model relating the error image (between the model and the image) and the required parameter updated at each time step. Following the forwards additive algorithm [Luca 81], the inverse additive algorithm [Hage 98], and the forwards compositional algorithm [Shum 01], Mathews and Baker [Bake 01, Bake 02, Matt 04] derived more mathematically elegant methods in

which the updates are always calculated in the average shape and then concatenated with the current guess. This inverse compositional method allows the pre-computation of the gradient images and inverse Hessian matrix for greater efficiency. Later work demonstrated that the inverse compositional algorithm is only really suitable for person-specific fitting and tracking, and that simultaneous estimation of the shape and appearance parameters was required for robust face fitting [Gros 05].

Cristancce et al. [Cris 06, Cris 08] proposed a patch based algorithm to model a deformable object. Their Constrained Local Model (CLM) algorithm is another method with the similar appearance model to that used in the AAMs [Coot 98]. It learns the variation in appearance of a set of template regions surrounding individual features instead of triangulated patches. The fitting algorithm first finds the best match of the combined shape-appearance model to the current guess, then searches locally using a non-linear optimiser to find the best match to the model. Further study on patch based appearance models have been carried out – exhaustive local search (ELS) algorithm [Wang 07], generic convex quadratic fitting (CQF) approach [Wang 08] and Bayesian constrained local models (BCLM) [Paqu 09]. The approach has been proven to outperform the active appearance models (AAMs) [Coot 01a] as it is more robust to occlusion and changes in appearance and no texture warps are required. ELS, CQF and BCLM are all gained some improvements over CLM fitting to certain databases. In this work, we will use the original normalised cross correlation error metric and concentrate on the extensions to large head rotations.

Active appearance models (AAMs) [Coot 98, Coot 01a] were originally formulated as 2D and most of the algorithms for AAM fitting have been a single-view [Coot 02]. Automatically locating detailed facial landmarks across different subjects and viewpoints, i.e. 3D alignment of a face, is a challenging problem. Previous approaches can be divided into three categories: view (2D) based, 3D based and combined 2D+3D based. View based methods [Coot 00, Zhou 05, Fagg 05, Peyr 08], train a set of 2D models, each of which is designed to cope with shape or texture variation within a small range of viewpoints. We have found for some applications that switching between 2D views can cause notable artefacts (e.g. in face reanimation). 3D based methods [Blan 99, Romd 02, Bran 01, Jone 98, Vett 97, Zhan 04], in contrast, deal with all views by a single 3D model. 3D Morphable model fitting is an expensive search problem in a high dimensional space with many local minima, which often fails to converge on real data. 2D+3D based methods [Xiao 04, Hu 04, Kote 05, Ramn 08] used AAMs and estimated 3D shape models to track faces in videos, but these algorithms are generally most suitable in the person specific context. Our proposed multi-view CLM algorithm is a 3D extension of the CLM algorithm [Cris 06, Cris 08] which could be more useful for fitting to unseen face images or tracking.

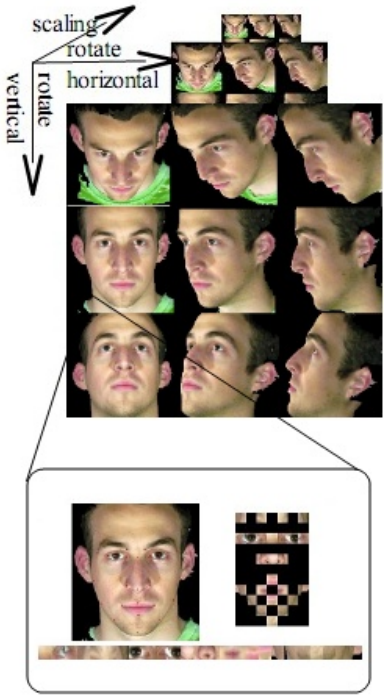


Figure 1: The Mult-view CLM consists of a shape model and several appearance models from different views. There are 15 views and 3 scales used to cover all the likely circumstances in the application. (There are only 9 views in the figure since the views from the right side are approximately mirrorring copies of the ones from the left side.)

3 ALGORITHM

3.1 An Overview

The model (*Figure* 1) consists of a model of 3D shape variation and 15 models of the appearance variations in a shape-normalised frame. A training set of labelled images, where key landmark points are marked on each example object, is required. We use landmark points placed on a set of 3D face models to generate the 3D shape model. The appearance model for each view is found by rendering the face model from the appropriate viewpoint and sampling square patches from the rendered image about the projected location of the feature point.

We use 14 subjects (8 males, 6 females) performing 7 posed expressions (neutral, happy, sad, disgust, surprise, fear, anger) and 7 posed vise mes (/ah/, /ch/, /ee/, /k/, /oo/, /p/, /th/) captured using a stereophotogrametric system (www.3dMD.com). From the set of landmark

Figure 2: Example of training images

points, a statistical model of shape variation can be generated using Principal Component Analysis (PCA). We extract a 20x20 block of pixels around each feature point at each of 3 spatial scales. (*Figure* 2) These patches are vectorised and used to build the appearance model. All the features are formed into a 500x20 block of pixel strip before the PCA analysis is applied.

In the original CLM work a combined shape and appearance models was created by performing PCA on the combined shape and appearance parameter vectors, and the search was carried out in this space. The use of multiple appearance models in out algorithm would require the use of multiple combined models. In order to simplify switching of the appearance model with a single shape model in this work we use separate models of shape and appearance instead of using a combined model. In future work we will experiment with switching with a combined model.

3.2 Shape Model

The shape model is built from normalised shape coordinates $s(x,y,z)$. To calculate the principal components, the templates are aligned to the average template (in a standard position) by performing a rigid body and scale transform. The covariance matrix of the vectorised points (templates), X, is created using the formula:

$$Cov = \frac{1}{N-1} \sum_{i=0,j=0}^{m,n} (X_i - \overline{X})(X_j - \overline{X}) \qquad (1)$$

where $\bar{X}$ is the mean of the vectorised points, N is the number of the templates. The model is then obtained by applying Jacobi's method to the covariance matrix to find the eigenvectors and eigenvalue, which represent the principal components and their distributions.

$$s = \bar{s} + P_s b_s \qquad (2)$$

where $\bar{s}$ is the mean shape, P_s is a set of orthogonal modes of variation derived from the shape templates training set and b_s is a set of shape parameters. The equation can then be used to reconstruct new shapes by varying the given shape parameters.

Figure 3: The first strip is the CLM texture model for a particular view. The second strip holds the stencil channel which is used to exclude hidden or background pixels.

The two-dimensional coordinates of the shape model can be calculated with the following equation:

$$s_{2d} = M \cdot V \cdot (\bar{s} + P_s \cdot b_s) \qquad (3)$$

where V is a vector of the pose (translation, rotation, scaling) transforming parameters $T_x, T_y, T_z, S, \theta, \phi, \gamma$ and M is the opengl frustum projection matrix.

3.3 Appearance Models

To build a model of the appearance, we render each 3D face model in our training set from a particular viewpoint and sample a square patch around each feature point. By transforming the face with different scale, rotation, shift and lighting parameters, we build a set of texture patches. After vectorising all the patches, PCA analysis is applied to the textures from a particular viewpoint and scale to build an appearance model:

$$g = \bar{g} + P_g b_g \qquad (4)$$

where $\bar{g}$ is the mean normalized gray-level vector, P_g is a set of orthogonal modes of variation derived from appearance training sets, the texand b_g is a set of gray-level parameters. We build 45 appearance models, one for each of 15 viewpoints across 3 different scales.

3.4 Other Features

To increase the stability with varied backgrounds, we use visibility information from the rendered patches to estimate occluded pixels. We grab the stencil channel (*Figure* 3) from the rendering canvas when we extract the texture patches to mark out the edges between the face and the background. We also compare the depth of the projected landmark with the depth buffer to identify self occlusion, and set the grabbed stencil buffer values to zero for occluded points. In the current work, we use a fixed visibility model for each viewpoint based on the average for the model.

Multi-scale techniques are standard in computer vision and image processing. They allow short range models to extend over longer ranges and optimisation to be achieved in fewer steps. In our model, a Gaussian filter is used as the convolution function to build a multi-scale texture pyramid. The processing time is much shorter with lower resolution images. So we fit the unseen image with the lowest resolution image first to improve the performance. When fitting we also use a Gaussian pyramid built for each frame and then the CLM search is applied at each layer from the coarsest to the finest. The process can be illustrated in *Figure* 4.

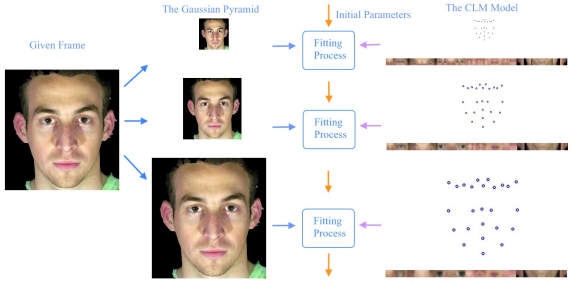


Figure 4: A skeleton of the three scales image searching technique.

Figure 5: Multiple appearance models.

3.5 Texture Model Selection

In the proposed algorithm, there are a global three-dimensional shape model and fifteen texture models (three in vertical and five in horizontal). One additional step to the original algorithm is the selection of the texture model while searching with the multi-view CLM algorithm. For tracking face movements, the algorithm needs to select the correct texture model for the current pose automatically. As each texture model is built from a certain view, we can use the rotation parameters θ, ϕ to estimate the view by testing the criteria shown in *Figure* 5. θ and ϕ can be obtained from the current estimate of head rotation using one of the shape template updating methods.

The texture model selection process is given by the following steps repeatedly until the end of the tracking.

1. The multi-view CLM algorithm is applied to the given frame accompanied with the initial parameters.
2. A set of new parameters are obtained including θ and ϕ which is the estimated rotation angles for the current face pose.
3. To estimate the next frame, θ and ϕ is then passed into the texture model selection module to choose the proper appearance model.

3.6 Search Algorithm

With the texture model selection algorithm, we can extend the searching method [Cris 06] for use with a three-dimensional model.

For a given set of initial points, $X = (x_0, y_0, z_0, x_1, y_1, z_1 \cdots, x_{n-1}, y_{n-1}, z_{n-1})$, the initial pose parameters V are estimated for the shape model built previously. Then the multi-view appearance CLM algorithm shown in *Figure* 6 is applied.

1. Initialise with the global face detector.
2. Estimate the initial pose parameters V.
3. Repeat
   (a) Repeat
      i. Compute the feature coordinates, s, and extract the feature patches, g.
      ii. Estimate the texture model from the pose parameters V.
      iii. Synthesise the feature patches from the updated coordinates and the selected texture model.
      iv. Apply the alpha channel feature, the hidden points feature to the extracted and synthetic feature patches.
      v. Optimise the error metrics with the shape template updating methods to get a new set of pose and shape parameters, V, b_s.
   (b) Until converged.
4. Until converged for all selected scales.

3.7 Shape updating methods

The original CLM algorithm [Cris 06] used the Nelder-Meade simplex algorithm [Neld 65] to optimize the Cross Correlation. This algorithm works by using N+1 samples in the N dimensional parameter space. Each iteration the worst sample is discarded and a new sample is added based on a set of simple heuristics. In this work, we instead use Powell's method [Pres 07] as this is supposed typically to require fewer function evaluations than the Nelder-Meade algorithm.

Optimisation techniques based on off-the-shelf non-linear optimisers like those described above are typically slow to converge. We compare optimisation using Powell's method with a direct method for optimising the global NCC using an estimate of the Jacobean and Hessian matrices and solving a linear system and a quadratic equation [Tidd 07].

We also compare the techniques described above with minimisation of the sum of squared errors (SSE) as an error metric. This is similar to the above, requiring the Jacobean and inverse Hessian matrices and solution of a linear system. This method is essentially equivalent to the additive inverse compositional AAM alignment, [Hage 98, Matt 04] wrapped in a slightly different fitting algorithm.

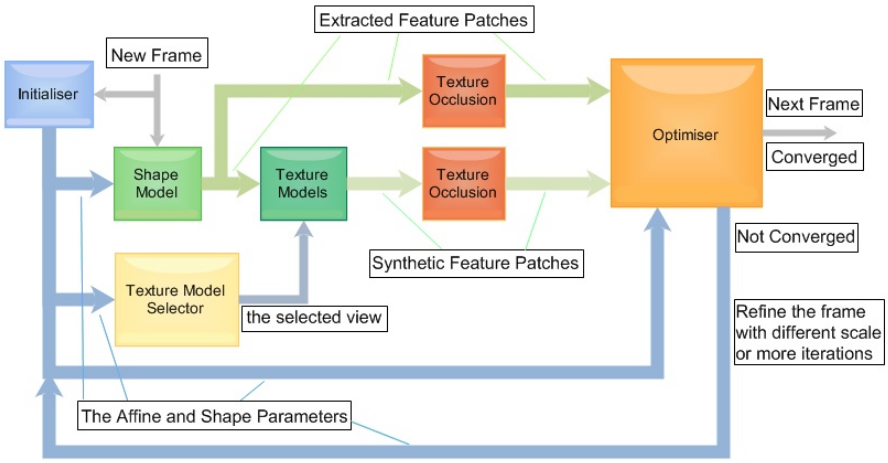


Figure 6: Multi-view CLM tracking process

4 RESULTS

We have evaluated the proposed algorithms using a mixture of synthetic and real data. Synthetic data is generated by rendering multiple 3D face scans from different viewpoints, and is useful because the 3D models provide accurate ground-truth data. We also test the algorithms on real video data with hand-labelled feature points.

We have designed two experiments to evaluate the proposed multi-view appearance 3D CLM. The first to compare single-view CLM to the proposed multi-view CLM. The second set of experiments compare the various optimisation algorithms within the multi-view CLM framework.

4.1 Synthetic Data Experiments

This experiment aims to compare the performance of the proposed multi-view 3D CLM algorithm to the single-view CLM algorithm. A set of face sequences with fixed expression and head rotation of over 40 ° were synthesised using rendered models captured from the 3dMD system. We use 10 sequences comprising over 700 images in the experiment. We applied both the 2D single view and the 3D multi-view methods to the same set of face sequences using the FastNCC algorithm [Tidd 07] as the optimisation method. An illustration can be seen in *Figure* 7.

The statistical results of fitting are shown in *Figure* 8 and *Table* 1. We can see that the fitting with the multi-view CLM algorithm converges better. A Student t-test (p=0.05) has been carried out for the the errors d_e from both methods. The hypothesis is $\mu_0 < \mu_1$, where μ_0 is the error d_e with single-view CLM algorithm and μ_1 is with multi-view CLM algorithm. The sig. is $8.16e-5$. The fitting speed is also improved. Therefore, the fitting process is more stable with multi-view CLM algorithm.

The alignment accuracy is measured with the following equation:

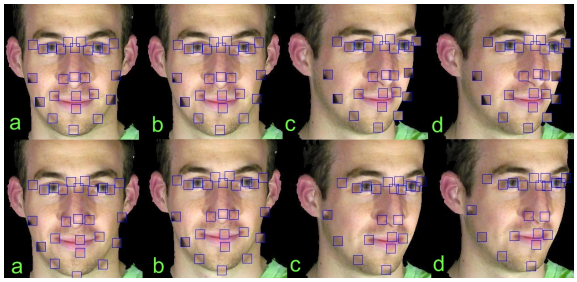


Figure 7: Each row consists of a set of selected frames from a tracking sequence with the synthetic texture patches drawn on which indicates the location of the features. The results in the first row is from the single-view approach and the second row is from the multi-view approach. When the rotating angle reaches certain degrees (b,c), the algorithm continues tracking the face well by auto-switching the appearance model to a side view model while the patches start getting off the position with single-view model.

Time (ms)	Multi-view CLM	Single-view CLM
mean	445.4	457.5
variance	46.3	65.1

Table 1: The speed comparison between the multi-view CLM algorithm and the Single-view CLM algorithm. The experiments are applied to synthetic images.

$$d_e = \frac{\sum \sqrt{(x_{std} - x_c)^2 + (y_{std} - y_c)^2}}{Nd} \quad (5)$$

where x_{std}, y_{std} represent the manually placed "ground truth" feature points locations, x_c, y_c represent the tracked feature points locations, d represents the distance between the center of the eyes and N is the number of features.

To make the further investigation, we compare the average errors's frame by frame as we can see in *Figure* 9. In the experiments, the first frame is the frontal face image and the face rotates one degree per frame. For

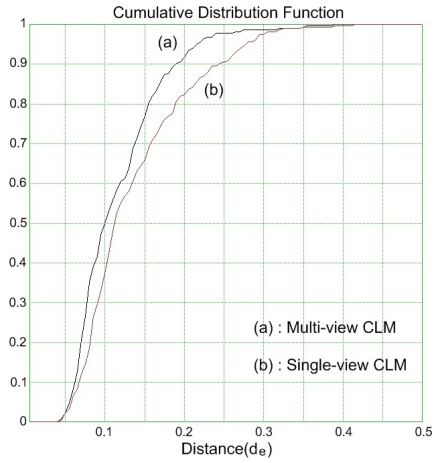


Figure 8: The error d_e comparison between the multi-view CLM algorithm and the Single-view CLM algorithm. The experiments are applied to synthetic images.

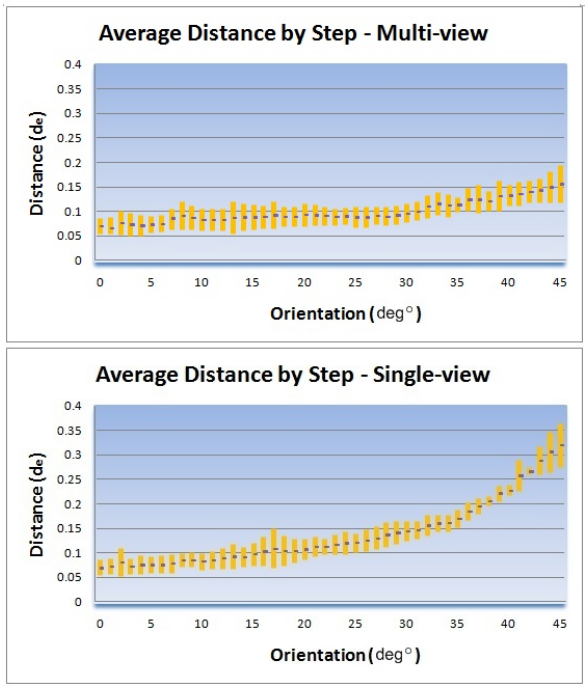


Figure 9: The figure contains the mean error d_e for each step (1 degree). The upper figure is for the Multi-view 3D algorithm and the lower is for the Single-view 2D algorithm.

the multi-view 3D CLM method the fitting errors are approximately constant across different views. For the Single-view CLM algorithm, we can see that the tracking is robust within the first 15 frames but the fitting becomes increasingly inaccurate as the rotation angle increases, as one would expect. The fitting is fairly stable up to about 20°–30°, which gives an indication of the optimal range of each appearance model.

A texture model from one view could possibly cover the rotation range around 20°–25°. However, it has to be cautious about the criterion area because the fitting

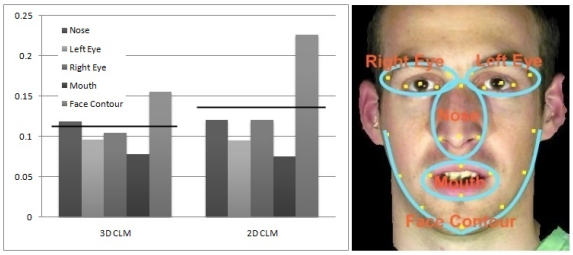


Figure 10: The average errors between the tracked feature points and the corrected feature points of the faces divided into different regions. The horizontal line across each group is the average error of all the face feature points.

could fail at that area referring to *Figure* 7. In order to do that, we choose a different texture model every 30° which makes each texture model cover 15° for each direction. This requires a fifteen texture model system to cover about 100° in the vertical direction and 160° in the horizontal direction.

We also investigate the performance of the algorithms for different facial regions. We split the face into jaw, mouth, nose and two eye regions and calculate the errors separately (*Figure* 10). The proposed algorithm improves mostly the performance on the face contour and the right eye area, which is somehow hidden during rotation. The performance in these areas benefits from the fixed visibility model applied to the multiple appearance models.

4.2 Real Data Experiment

Another experiment is carried out to investigate the accuracy and stability of the proposed algorithm on real video data. The video data consists of four different subjects showing expression, speech and some head rotation (1280 frames in total) (*Figure* 11) These images and subjects are independent of the CLM training sets.

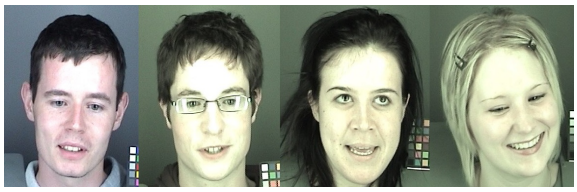


Figure 11: Example images from the text video clips fitting to.

The image sequences are roughly initialised with the face detector described in [Chen 08] before a CLM search is applied to the frame and the following frames while tracking. Three optimising methods are used for the experiments: Powell's method [Pres 07], FastNCC algorithm [Tidd 07] and the Gauss-Newton method [Hage 98, Matt 04], a maximum of 4 iterations are used per frame while tracking.

The results are shown in *Figure* 12 and *Figure* 13. For Powell's method and Gauss-Newton algorithm, nearly

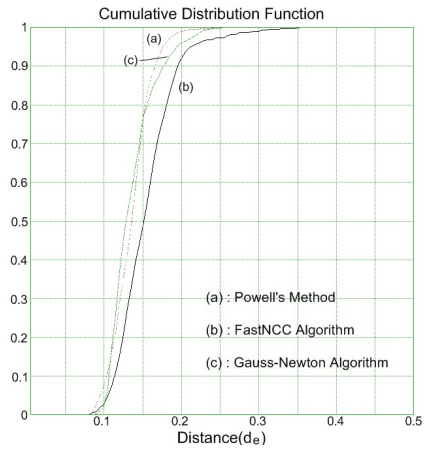


Figure 12: Results of fitting performance with three optimising algorithms to sets of real images.

Sig.	Powell's Method	FastNCC
Gauss-Newton	0.4658	2.8e-10

Table 2: Results from two samples Student T-Test (p=0.05) for Gauss-Newton algorithm against Powell's method and Gauss-Newton Algorithm against FastNCC algorithm. The hypothesis is that $\mu_0 < \mu_1$, where μ_0 is the error d_e for Gauss-Newton algoithm, μ_1 is the error d_e for the other two methods.

80 % of the points are within 0.15 d_e. For FastNCC algorithm, nearly 80 % of the points are within 0.17 d_e. The proposed multi-view CLM algorithm gives robust performance for all three optimising methods. Collaborating with the anova test shown in (*Table* 2), the Gauss-Newton algorithm and the Powell's method are more stable and accuracy.

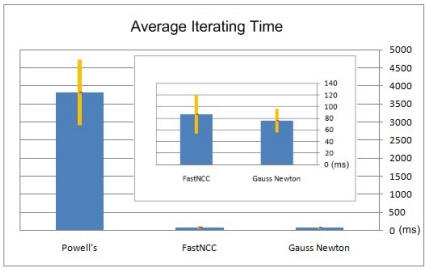


Figure 13: Average iteration time estimated on the fitting using three optimising algorithms on the set of real images.

The iteration time is measured roughly and could be further optimised. The results are shown in *Figure* 13. The Gauss-Newton algorithm is the fastest method. The FastNCC algorithm performs at the same level as those two methods. Powell's method takes much more time than the other two methods.

5 CONCLUSION AND FUTURE WORK

The presented multi-view 3D CLM algorithm is based on the original Constrained Local Model search [Cris 06, Cris 08]. There are 15 texture models built from different views of faces and a 3D shape model in the algorithm. For each view, the selected texture model and the shape model forms a combined model and applies a constrained local search on the given image. This algorithm can be used in locate and track human face features in sequences with large head rotations.

Based on the experiments carried out, we have shown that the proposed multi-view 3D algorithm gives better results when fitting to unseen images with large head rotations (the images are captured from 3dMD). The fitting is more robust (*Figure* 8, 9, 10) and efficient (*Table* 1) than the single-view CLM algorithm [Cris 06]. The effects of choice of the optimisation algorithm was marginal for the data tested here, but further experimentation with a wider range of video data would be needed to confirm this.

Future research will involve lighting and colour and more self occlusion factors, which could improve the matching rates under variant lighting, colour conditions and even with unexpected occlusions. To achieve that, more constrained searching should be applied while tracking and a more stable tracking failure recovery system should be introduced.

The fitting results (*Figure* 10) showed that the face contour part is not matched as well as internal facial features. We will investigate improved background modelling and contour alignment methods to improve facial contour alignment and tracking.

REFERENCES

[Ahlb 01] J. Ahlberg. "Using the active appearance algorithm for face and facial feature tracking". In: *International Conference on Computer Vision Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-time Systems*, pp. 68–72, 2001.

[Bake 01] S. Baker and I. Matthews. "Equivalence and efficiency of image alignment algorithms". In: *IEEE IEEE Transactions on Computer Vision and Pattern Recognition*, pp. 1090–1097, 2001.

[Bake 02] S. Baker and I. Matthews. "Lucas-Kanade 20 Years On: A Unifying Framework: Part 1". Tech. Rep. CMU-RI-TR-02-16, Robotics Institute, University of Carnegie Mellon, Pittsburgh, PA, July 2002.

[Blan 99] V. Blanz and T. Vetter. "A morphable model for the synthesis of 3D faces". In: *Computer graphics, annual conference series (SIG-GRAPH)*, pp. 187–194, 1999.

[Bran 01] M. Brand. "Morphable 3D models from video". In: *IEEE computer society conference on computer vision and pattern recognition*, pp. 456–463, 2001.

[Chen 08] J. Chen and B. P. Tiddeman. "Multi-cue Facial Feature Detection and Tracking". In: *International Conference on Image and Signal Processing*, pp. 356–367, 2008.

[Coot 00] T. F. Cootes, K. Walker, and C. Taylor. "View-Based Active Appearance Models". In: *IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 227–232, IEEE Computer Society, Washington, DC, USA, 2000.

[Coot 01a] T. F. Cootes, G. J. Edwards, and C. J. Taylor. "Active appearance models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 681–685, 2001.

[Coot 01b] T. F. Cootes and C. J. Taylor. "Statistical models of appearance for medical image analysis and computer vision". In: *SPIE Medical Imaging*, pp. 236–248, 2001.

[Coot 02] T. F. Cootes and P. Kittipanyangam. "Comparing Variations on the Active Appearance Model Algorithm". In: *British Machine Vision Conference*, pp. 837–846, 2002.

[Coot 95] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. "Active shape models - Their training and application". *Computer Vision and Image Understanding*, Vol. 61, pp. 38–59, 1995.

[Coot 98] T. F. Cootes, G. J. Edwards, and C. J. Taylor. "Active appearance models". *European Conference on Computer Vision*, Vol. 2, pp. 484–498, 1998.

[Cris 06] D. Cristinacce and T. Cootes. "Feature detection and tracking with Constrained Local Models". In: *British Machine Vision Conference*, pp. 929–938, 2006.

[Cris 08] D. Cristinacce and T. Cootes. "Automatic feature localisation with constrained local models". *Pattern Recognition*, Vol. 41, No. 10, pp. 3054–3067, 2008.

[Fagg 05] N. Faggian, S. Romdhani, J. Sherrah, and A. Paplinski. "Color Active Appearance Model Analysis Using a 3D Morphable Model". In: *Digital Image Computing on Techniques and Applications*, p. 59, IEEE Computer Society, Washington, DC, USA, 2005.

[Gros 05] R. Gross, I. Matthews, and S. Baker. "Generic vs. Person Specific Active Appearance Models". *Image and Vision Computing*, Vol. 23(11), pp. 1080–1093, November 2005.

[Hage 98] G. D. Hager and P. N. Belhumeur. "Efficient Region Tracking With Parametric Models of Geometry and Illumination". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, pp. 1025–1039, 1998.

[Hu 04] C. Hu, J. Xiao, I. Matthews, S. Baker, J. Cohn, and T. Kanade. "Fitting a Single Active Appearance Model Simultaneously to Multiple Images". In: *British Machine Vision Conference*, September 2004.

[Jone 98] M. J. Jones and T. Poggio. "Multidimensional Morphable Models: A Framework for Representing and Matching Object Classes". In: *International Journal of Computer Vision*, pp. 107–131, Springer Netherlands, 1998.

[Kote 05] S. Koterba, S. Baker, I. Matthews, C. Hu, J. Xiao, J. Cohn, and T. Kanade. "Multi-View AAM Fitting and Camera Calibration". In: *IEEE International Conference on Computer Vision*, pp. 511–518, IEEE Computer Society, Washington, DC, USA, 2005.

[Lani 97] A. Lanitis, C. J. Taylor, and T. F. Cootes. "Automatic interpretation and codeing of face images using flexible models". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19(7), pp. 742–756, 1997.

[Luca 81] B. D. Lucas and T. Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

[Matt 04] I. Matthews and S. Baker. "Active appearance models revisited". *International Journal of Computer Vision*, Vol. 60, pp. 135–164, 2004.

[Mitc 01] S. C. Mitchell, B. P. F. Lelieveldt, R. J. Geest, J. G. Bosch, J. H. C. Reiber, and M. Sonka. "Multistage hybrid active appearance model matching: Segmentation of left and right ventricles in cardiac MR images". In: *IEEE Transanctions on Medical Image*, pp. 415–423, 2001.

[Neld 65] J. A. Nelder and R. Mead. "A simplex method for function minimization". *Computer Journal*, Vol. 7, pp. 308–313, 1965.

[Paqu 09] U. Paquet. "Convexity and Bayesian Constrained Local Models". *IEEE Transactions on Computer Vision and Pattern Recognition*, pp. 1193–1199, 2009.

[Peyr 07] J. Peyras, A. Bartoli, H. Mercier, and P. Dalle. "Segmented AAMs Improve Person-Independent Face Fitting". In: *British Machine Vision Conference*, 2007.

[Peyr 08] J. Peyras, A. J. Bartoli, and S. K. Khoualed. "Pools of AAMs: Towards Automatically Fitting any Face Image". In: *British Machine Vision Conference*, p. , 2008.

[Pres 07] W. H. Press., S. A. Teukolsky., W. T. Vetterling, and B. P. Flannery. *Numerical recipes - The art of scientific computing*. Cambridge University Press, 2007.

[Ramn 08] K. Ramnath, S. Koterba, J. Xiao, C. B. Hu, I. Matthews, S. Baker, J. F. Cohn, and T. Kanade. "Multi-View AAM Fitting and Construction". In: , Ed., *International Journal of Computer Vision*, pp. 183–204, 2008.

[Romd 02] S. Romdhani, V. Blanz, and T. Vetter. "Face identification by fitting a 3D morphable model using linear shape and texture error functions". In: *European Conference on Computer Vision*, pp. 3–19, 2002.

[Shum 01] H. Y. Shum and R. Szeliski. *Panoramic vision: sensors, theory, and applications*, Chap. Construction of panoramic image mosaics with global and local alignment, pp. 227–268. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[Steg 01] M. B. Stegmann. "Object tracking using active appearance models". In: *Danish Conference Pattern Recognition and Image Analysis*, pp. 54–60, 2001.

[Tidd 07] B. P. Tiddeman and J. Chen. "Correlated Active Appearance Models". In: *IEEE Transactions on Signal-Image Technology & Internet-Based Systems*, pp. 832–838, 2007.

[Vett 97] T. Vetter and T. Poggio. "Linear object classes and image synthesis from a single example image". In: *Pattern Analysis and Machine Intelligence*, pp. 733–742, 1997.

[Wang 07] Y. Wang, S. Lucey, J. Cohn, and J. M. Saragih. "Non-rigid Face Tracking with Local Appearance Consistency Constraint". In: *IEEE International Conference on Automatic Face and Gesture Recognition*, September 2007.

[Wang 08] Y. Wang, S. Lucey, and J. F. Cohn. "Enforcing convexity for improved alignment with constrained local models". In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, June 2008.

[Xiao 04] J. Xiao, S. Baker, I. Matthews, and T. Kanade. "Real-time combined 2D+3D active appearance models". In: *the IEEE computer society conference on computer vision and pattern recognition*, pp. 535–542, 2004.

[Zhan 04] Z. Zhang, Z. Liu, D. Adler, M. F. Cohen, E. Hanson, and Y. Shan. "Robust and Rapid Generation of Animated Faces from Video Images: A Model-Based Modeling Approach". *International Journal of Computer Vision*, Vol. 58, No. 2, pp. 93–119, 2004.

[Zhou 05] Y. Zhou, W. Zhang, X. Tang, and H. Shum. "A Bayesian Mixture Model for Multi-View Face Alignment". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 741–746, IEEE Computer Society, Washington, DC, USA, 2005.

A Two-Step Minimization Algorithm For Model-Based Hand Tracking

O.Ben Henia
Liris-CNRS UMR 5205
Université de lyon1
France
obenheni@liris.cnrs.fr

M. Hariti
Liris-CNRS UMR 5205
Université de lyon1
France
mhariti@liris.cnrs.fr

S. Bouakaz
Liris-CNRS UMR 5205
Université de lyon1
France
sbouakaz@liris.cnrs.fr

ABSTRACT

Model-based methods to the tracking of an articulated hand in a video sequence could be divided in two categories. The first one, called stochastic methods, uses stochastic filters such as kalman or particle ones. The second category, named deterministic methods, defines a dissimilarity function to measure how well the hand model is aligned with the hand images of a video sequence. This dissimilarity function is then minimized to achieve the hand tracking. Two well-known problems are related to the minimization algorithms. The first one is that of local minima. The second problem is that of computing time required to reach the solution. These problems are compounded with the large number of degrees of freedom (DOF) of the hand (around 26). The choice of the function to be minimized and that of the minimization process can be an answer to these problems. In this paper two major contributions are presented. The first one defines a new dissimilarity function, which gives better results for hand tracking than other well-known functions like the directed chamfer or hausdorff distances. The second contribution proposes a minimization process that operates in two steps. The first one provides the global parameters of the hand, i.e. position and orientation of the palm, whereas the second step gives the local parameters of the hand, i.e. finger joint angles. Operating in two stages, the proposed two-step algorithm reduces the complexity of the minimization problem. Indeed, it seems more robust to local minima than a one-step algorithm and improves the computing time needed to get the desired solution.

Keywords: Hand tracking, minimization algorithm, dissimilarity function.

1 INTRODUCTION

Hand gestures take a fundamental role in inter-human communication of daily life. Their use has become an important part of human-computer interaction in the two last decades [WH99][Tos06][Ste04]. Data gloves are commonly used as input devices to capture and track human hand motion by attaching some sensors to the hand to measure the joint angles and the spatial positions of the hand directly. Unfortunately, glove-based devices are expensive, frail and not user friendly.

Vision-based approaches offer promising alternatives to capture and track human hand motion with affordable cameras. However, building a fast and effective vision-based hand motion tracker is challenging. This is due to the high dimensionality of the pose space, the ambiguities due to occlusion, the lack of visible surface texture and the significant appearance variations due to shading.

In this paper, a parametric hand model is used. Articulated hand motion is decoupled to its global hand motion and local finger motion, in which global motion is parameterized as the rotation and translation of the palm, and local motion is parameterized as the joint angles of the hand. The hand motion tracking is first formulated as an optimization task, where a dissimilarity function between the projection of the hand model under articulated motion and the observed image features, is to be minimized. A two-step iterative algorithm is then proposed to minimize this dissimilarity function. This is achieved by means of a simplex approach proposed by Nelder and Mead [NM65].

The next section presents a brief literature survey of hand motion tracking. Section 3 describes the used 3D hand model as well as the dissimilarity function between the projection of the hand model and the corresponding image features. Section 4 details the hand motion tracking algorithm. Before concluding, experimental results from synthetic and real data are presented in section 5.

2 LITERATURE REVIEW

Approaches to the tracking of hand motion could be divided into two classes. In the first class, view-based approaches are usually used to recover hand pose through classification or regression techniques [RASS01][SKS01]. A set of hand features is labeled

with a particular hand pose, and a classifier is trained from this data. Due to the high dimensionality of the space spanned by possible hand poses, it is difficult, or even impossible to perform dense sampling. Thus, these approaches are well suited for rough initialization or recognition of a limited set of predefined poses. This family of approaches is often used in applications where computing time takes precedence over tracking accuracy. This is the case of a human computer interface (HCI) application where only few hand poses are considered, e.g. a real-time hand gesture recognition system using classifiers was proposed in [IKS07].

The second class incorporates model-based approaches. These approaches use an articulated three-dimensional (3D) hand model and provide more precise hand tracking. The underlying kinematic structure is usually based on the biomechanics of the hand. Therefore, the 3D hand model is often represented as a hierarchical one with approximately 26 degrees of freedom (DOF) [KX06][WLH05].

Models that have been used for tracking are based on planar patches [WLH01], deformable polygon meshes [HH96] or generalized cylinders[DF98]. Stenger et al. [SMC01a] used quadric surfaces like cones and ellipsoids. Other authors proposed a fine mesh obtained by scanning the hand to track[BKMM*04]. However, even if the fine mesh could increase the tracking accuracy, it remains restrictive to scan the hand to track.

A model-based tracking system uses a geometric hand model whose projection is registered to the observed image. An error function between image features and model projection is computed, and the parameters of the hand model are then adapted such that this cost is minimized. Tracking in such a system is performed by means of methods that could be classified into two categories, stochastic and deterministic ones. Stochastic methods exploit the state space to seek an estimation of the hand poses. Stenger et al. [SMC01a] used an Unscented Kalman Filter (UKF) to update the model projection pose to minimize the geometric error between the model projection and a video sequence on the background. They argued that their approach is well adapted for the hand tracking problem due to the nonlinear nature of the latter arising from the DOF of rotation. The particle filter, also known as the Condensation algorithm [IB98] in the computer vision community, has been applied to the hand tracking problem in [KX06]. It goes beyond the unimodal Gaussian assumption of the Kalman filter by approximating arbitrary distributions with a set of random samples. The advantage is that the particle filter can deal with clutter and ambiguous situations more effectively, by not placing its bet on just one hypothesis. However, a major concern is that the number of particles required increases exponentially with the dimension of the state space.

On the other hand, deterministic methods typically track by performing an iterative minimization of a cost function which measures how well the model is aligned with the images. Rehg and Kanade[RK94][RK95] proposed two possible cost functions, the first based on edge features[RK94], the second on template registration [RK95]. In the first method, they used the Gauss-Newton algorithm to minimize their edge-based cost function. The second method of template registration was used to deal with self-occlusion. Using two cameras and dedicated hardware, their DigitEyes tracking system achieved a rate of 10 frames per second when using local edge features.

Ouhaddi and Horain[OH99] suggested two cost functions. The first is defined as a non-overlapping surface between the model silhouette and the hand one extracted from an image, whereas the second as a distance between the model and image contours. For the two methods, they used three standard optimization approaches to minimize their cost functions, named Levenberg-Marquardt, downhill simplex [NM65] and Powell approaches. They obtained the best results for minimizing the non-overlapping surface by means of the downhill simplex approach. Using a single camera, they tracked simple hand motion like translation of the hand and abduction/adduction of fingers.

Delamarre and Faugeras [DF98] pursued a stereo approach to hand tracking. A stereo correlation algorithm was used to estimate a dense depth map. Note that deterministic methods do not escape from the problem of local minima. Bray et al. [BKMSG04] proposed an optimization method for hand tracking based on Stochastic Meta-Descent (SMD), which is a gradient descent with local step size adaptation that combines rapid convergence with excellent scalability. However, this method requires the knowledge of the derivative of the function to be minimized.

Our proposed work belongs to the class of model-based approaches for hand tracking. Indeed, we use a parametric hand model which is defined as a hierarchical one with 26 DOF. To reduce our model to 22 DOF, we impose constraints on the finger motion, such as linear dependence of finger joint angles. Furthermore, the state space of the hand is decoupled into global pose parameters and finger motion parameters. These sets of parameters are estimated in a two-step iterative algorithm. Specifically, a new dissimilarity function between the hand images and the projection of the hand model is defined and then minimized using a simplex approach proposed by Nelder and Mead [NM65].

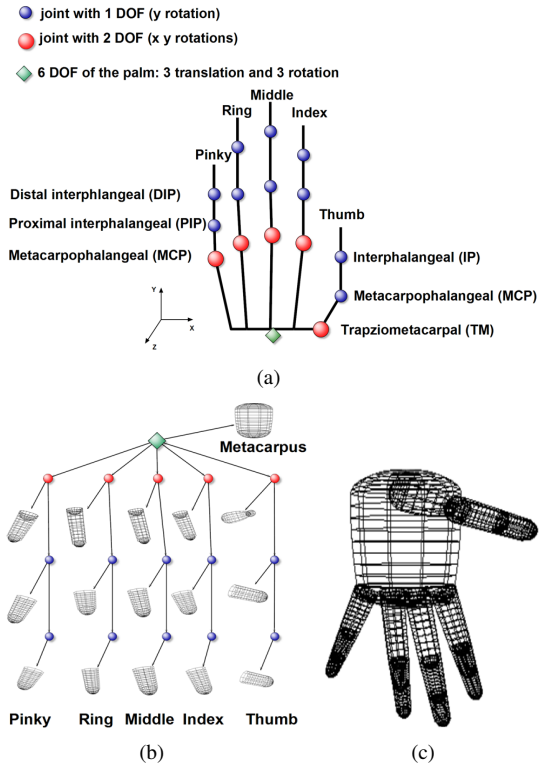


Figure 1: (a) Squeletal representation of the 3D model (b) Hierarchical representation of the 3D model (c) 3D hand model appearance

3 MODEL AND DISSIMILARITY FUNCTION

3.1 The Hand Model

The human hand is a complex and highly articulated structure. Several models have been proposed in the literature to represent the latter. In [HH96] a 3D deformable point distribution model was implemented. This model can not accurately reproduce all realistic hand motion. Indeed, since this model is not based on a rigid skeleton, fingers can be warped and reduced to ensure tracking of the hand gestures.

On the other hand, skeleton animation based model was used in [BKMSG04][OH99]. This kind of models is usually defined as hierarchical transformations representing the DOF of the hand: position and orientation of the palm, joint angles of the hand(Fig:1(a)). The variation in the values of DOF animates the 3D hand model. Using this kind of models, we can estimate not only the position and orientation of the hand, but also the joint angles of fingers.

In our proposed work, we use a parametric hand model which is conforming to the H-Anim standard[1]. The H-Anim model is often used in the 3D animation field. We can highlight its particularity to separate the kinematic part (motion) from the appearance one. This model consists of a hierarchy of 3D transformations (rotation, translation) allowing easy control of its animation by modifying only the involved transformations. Regarding the appearance, objects called segments are placed in this hierarchical representation (Fig.1(b)) to provide the shape of this model (Fig.1(c)). To change the appearance of this model, we modify the objects representing the latter. In our proposed work, these object segments are modeled by quadric surfaces as shown in Fig.1(c).

In our hand model, the base is a palm and five fingers are attached to the palm. The palm is modeled as a kinematic structure with six DOF. Three of six DOF correspond to the position of the palm. The other three DOF correspond to the orientation of the palm. Each finger is modeled as a four DOF kinematic chain. Two of four DOF correspond to the metacarpophalangeal joint (MCP) and its abduction (see Fig.1(a)). Note that for the thumb these two DOF correspond to the trapeziometacarpal joint and its abduction(see Fig.1(a)). The other two DOF correspond to the proximal interphalangeal joint (PIP) and the distal interphalangeal joint (DIP) (see Fig.1(a)).

Therefore, our hand model has 26 DOF. To reduce this number of DOF, we apply a constraint defined by Lin et al. [LWH00], which consists in exploiting dependencies between the angles of the proximal interphalangeal joint (PIP) and the distal interphalangeal joint (DIP) : $\theta_{DIP} = 2/3\theta_{PIP}$. Applied to our hand model, this constraint yields 22 DOF to be estimated.

For the model appearance, we use a set of quadrics approximately representing anatomy of a real human hand(Fig.1(c)). The palm is modeled using a truncated ellipsoid, its top and bottom closed half-ellipsoids. Each finger is composed by three truncated cones, i.e. one for each phalanx. Hemisphere was used to close each truncated cone. The major advantage of this model shape representation is its simplicity to be adapted to any hand to track compared with models based on 3D scans.

3.2 Dissimilarity Function

Many model-based approaches have been proposed in the literature to achieve low-cost monocular hand tracking. These approaches make use of a dissimilarity function, which measures how well the 3D hand model is aligned with the hand images. The most often used dissimilarity functions exploit either silhouette or edge features extracted from the images of a video sequence. The directed chamfer distance[Bor88] is a famous function, which computes a dissimilarity between two edge images I_a and I_b as a distance d_C between two edge pixel sets A and B of I_a and I_b, respectively. The dis-

[1] Humanoid-Animation(H-ANIM) is an approved ISO standard for humanoid modeling and animation. website :www.h-anim.org

tance d_C from set A to B with respect to metric d is defined as:

$$d_C(A,B) = \frac{1}{|A|} \sum_{a_i \in A} \min_{b_j \in B} d(a_i, b_j) \quad (1)$$

where $|A|$ is the cardinal number of the pixel set A and $d(a_i, b_j)$ the Euclidian distance between a_i and b_i. Another well-known dissimilarity function is the directed hausdorff distance d_H,which is proposed in[HKKR93] and defined as the maximum of all distances from each edge pixel in set A to its nearest neighbor in set B:

$$d_H(A,B) = \max_{a_i \in A} \{\min_{b_j \in B} d(a_i, b_j)\} \quad (2)$$

where the Euclidian distance $d(a_i, b_j)$ is identical to the one described in Eq.1. A disadvantage of the edge-based dissimilarity functions is their sensitivity to outliers. Indeed, few pixels are only considered to carry out the tracking of articulated objects. On the other hand, Ouhaddi and Horain [OH99] proposed a non-overlapping function which makes use of the hand silhouette. Their dissimilarity function could be defined as the uncommon area of two surfaces: the hand silhouette and the model projection.

The above dissimilarity functions were used to track simple hand motion in a video sequence composed from images acquired by a single camera. Indeed, Ouhaddi and Horain [OH99] used their non-overlapping function to track some global hand motion such as the position of the palm, or simple local finger motion like abduction/adduction, but not local and global motion together. Stenger et al[SMC01b] evaluated their algorithm for tracking three global DOF of the hand(two translations and one rotation).

To achieve more robust monocular hand tracking, we propose a new dissimilarity function which combines the non-overlapping surface and the directed chamfer distance. Our dissimilarity function assesses the difference between the model projection and the hand silhouette. Let H_s and M_s be the respective hand silhouette(Fig.2(a)) and model projection M_p (Fig.2(b)). Let NOS be the $N_r \times N_c$ image containing the non-overlapping surface(Fig.2(d)), where N_r and N_c are the numbers of rows and columns, respectively. A pixel (i,j) of this image is defined as:

$$NOS_{ij} = \begin{cases} 1 & \text{if the pixel (i,j) belongs to} \\ & ((H_s \cup M_s) - (H_s \cap M_s)) \\ 0 & \text{otherwise} \end{cases}$$

The size of the non-overlapping surface evaluates the dissimilarity between the hand image and the model projection. To achieve more tracking robustness, we propose to weight the NOS's pixels by their distance to the hand contours. For this purpose, we extract first the

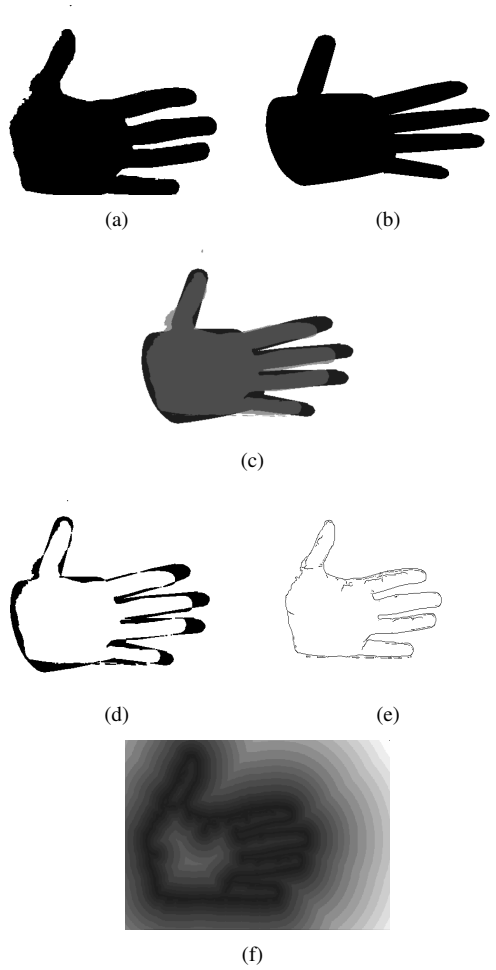


Figure 2: The different computing steps for our dissimilarity function. (a) Hand silhouette H (b) Model projection M_p(c) Superposition of the model projection and hand silhouette (d) Non-overlapping surface NOS (e) Hand contours (f) Distance map D

hand contours(Fig.2(e)) from a hand image H, and an $N_r \times N_c$ distance map D(Fig.2(f)) is then computed. An element D_{ij} of D represents the distance of the pixel (i, j) to the hand contours. We define our dissimilarity function which compares a hand image H with the model projection associated with the parameters R, T and θ, where R and T represent the global motion (three rotations and three translations) and θ represents the local motion(fingers articulation angles). Our dissimilarity function d_F is defined as follows:

$$d_F(NOS, D) = \sum_{i=1,j=1}^{N_r,N_c} NOS_{ij} * D_{ij} \quad (3)$$

4 THE TRACKING ALGORITHM

The tracking of the hand gestures in a video sequence is performed by seeking the parameters of the hand model which reproduce the hand motion. We achieve this step

by minimizing our dissimilarity function for each frame of the video sequence. This minimization gives the parameters of the hand model which align the pose model with hand pose. We assume that the parameters of the hand model are close to the solution associated with the first frame of the video sequence. For the remainder of the video sequence, the minimization algorithm exploits the parameters of the hand model obtained at the previous frame. We use the Nelder-Mead method [NM65] which is an iterative minimization algorithm. This choice is justified by two main reasons. Firstly, the use of this method does not require the knowledge of the derivative of the function to be minimized. The second reason relates to the processing of the algorithm itself. Indeed, the simplex descent explores different directions for each iteration. These various explorations can be achieved in parallel to increase computation time.

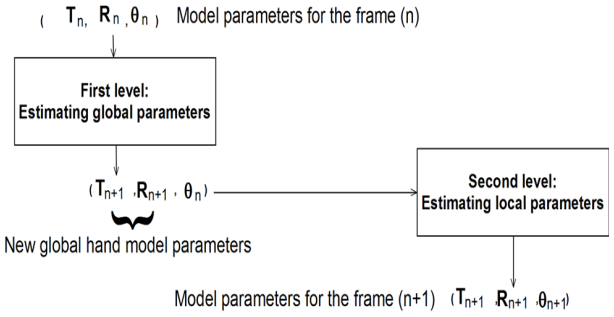


Figure 3: Two-step estimation process

Due to the high dimensionality of the search space, we decouple the minimization algorithm in two steps(Fig.3). The first one estimates the position and orientation of the hand. The parameters representing the joints angles of fingers are fixed, whereas those representing the position and orientation of the palm are processed by the minimization algorithm. The processing is reversed at the second step, i.e. the orientation and position parameters are first fixed to those obtained in the first step, and the joint angles of the hand are then estimated. Besides simplifying the minimization problem, this approach can be justified by the slow variation of the hand pose in two successive frames.

	Tracking algorithm	Frame 1	Frame 50	Frame 100
Ground Truth				
d_C	One-Step			
	Two-Step			
d_H	One-Step			
	Two-Step			
d_{NOS}	One-Step			
	Two-Step			
d_F	One-Step			
	Two-Step			

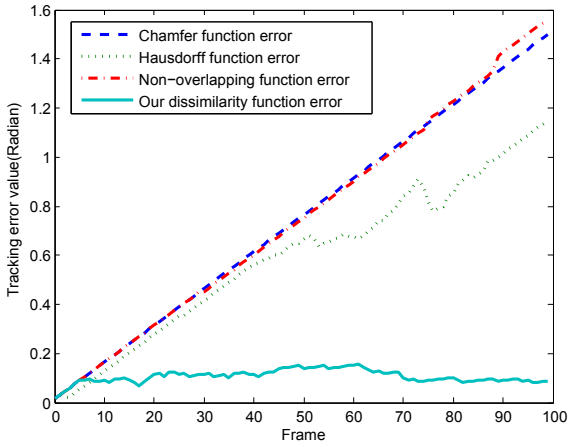
Table 1: One-Step and Two-Step tracking algorithm results using different dissimilarity functions : directed chamfer distance(d_C) directed hausdorff distance(d_H), non-overlapping function(d_{NOS})and our dissimilarity function (d_F).

5 EXPERIMENTAL RESULTS

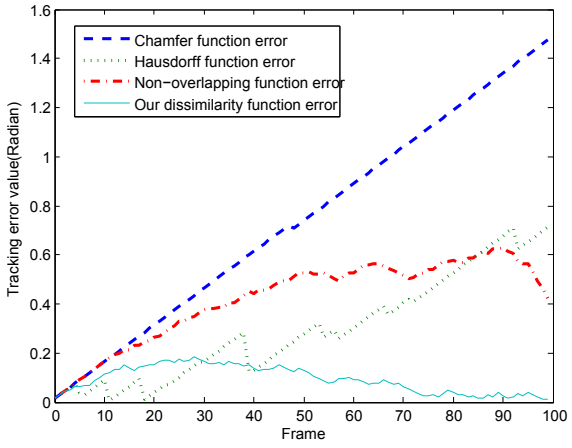
The performance of the two-step iterative algorithm is first evaluated for tracking hand motion appearing in synthetic and real images. We evaluate also our dissimilarity function by comparing it with the non-overlapping function. For the first test benchmark, a video sequence of one hundred 320x240 synthetic images of the hand model is acquired (Table 1). To obtain this sequence of images we vary three global parameters and four local ones. The global parameters are the X-translation, Z-translation and Z-rotation of the palm.

The local parameters are the metacarpophalangeal MCP(see Fig1(a)) X-rotations of the hand fingers except the thumb. The results of the hand model tracking in a synthetic video are shown in table1. The same table

illustrates the outputs of the one-step and the two-step iterative algorithms. We can see in table 1 that our dissimilarity function(d_F) provides the best results when it is compared with other well-known dissimilarity functions like the directed chamfer distance(d_C)[Bor88], the directed hausdorff distance (d_H)[HKKR93] and the non-overlapping function(d_{NOS})[OH99].



(a)



(b)

Figure 4: Z-Rotation tracking error.(a) Error obtained by the one-step iterative algorithm (b) Error obtained by the two-step iterative algorithm

To estimate the error of the tracking algorithm, we compute a difference between the tracking results and the ground truth. The tracking error is then plotted as a curve in Fig.4, in which we only consider the Z-Rotation. The curves represented in Fig.4 show that our dissimilarity function is more efficient than other functions compared with. Indeed, using our dissimilarity function the average tracking error is about 0.09 radian, while this error can exceed 1 radian with the others functions such as the directed chamfer distance. The same figure shows also that the two-step algorithm yields more robust results than the one-step algorithm. Specifically, in the case of the non-overlapping surface, the two-step algorithm clearly improves the tracking performance. Indeed the average tracking error drop 0.76 radian with the one-step algorithm to 0.42 radian with the two-step algorithm. Our observations concerning the Z-Rotation tracking error could be still valid for other global and local parameters.

The processing is performed using a PC Intel-Centrino with duo-core processor and Nvidea graphic card (GeoForce 8600MGT), which is used to compute the model projection. For this purpose, we exploit the graphic library OpenGL[2] to obtain the model images. Besides robustness, we argue that the two-step iterative algorithm is faster than the one-step iterative one. Indeed, for synthetic video sequence,the processing rate is approximately about 11 frames per second for the two-step iterative algorithm, and 8 frames per second for the one-step iterative algorithm running on the same machine.

Another video sequence of four hundred frames of 320x240 pixels is processed using our algorithm. In this video sequence we track all the DOF of the hand except ones of the thumb. Indeed, the thumb is fixed and thus 18 parameters are estimated. For the six global parameters, we set experimentally the iteration number parameter of the downhill simplex to 10. This parameter is also set experimentally to 5 in order to estimate the local motion of each finger. Fig.5 shows the robustness of the two-step tracking algorithm compared with the one-step tracking algorithm. Indeed, the fingers motion tracking is more accurate for the two-step algorithm especially in the last frames of the video sequence. In fact, we can highlight that our two-step algorithm is more efficient to track some complex finger motion than the one-step algorithm, as we can see in the last frames of the video sequence shown in Fig.5. We can note also that the two-step algorithm is more powerful for tracking much DOF of the hand. Indeed, the processing rate for the video sequence (Fig.5) is about 5 frames per second for our two-step algorithm, and 2 frames per second for the one-step algorithm.
We evaluate also our algorithm for tracking more complex hand motions. As we can see in Fig.6 the proposed algorithm cannot deal with the self-occlusion problem since the hand tracking is lost after a few tens of images. Indeed the hand motion becomes very complicate to track when the x or y rotation of the palm becomes important. This problem is incontrovertible for monocular hand tracking. Wang and Provic[WP09]have tried to deal with the self-occlusion problem using a color glove. Their color glove allows to differentiate the palm down pose from the palm up one. They used a view-based approach but they could not estimate exactly the depth position of the palm. The self-occlusion problem still unresolved for monocular hand tracking.

[2] www.opengl.org

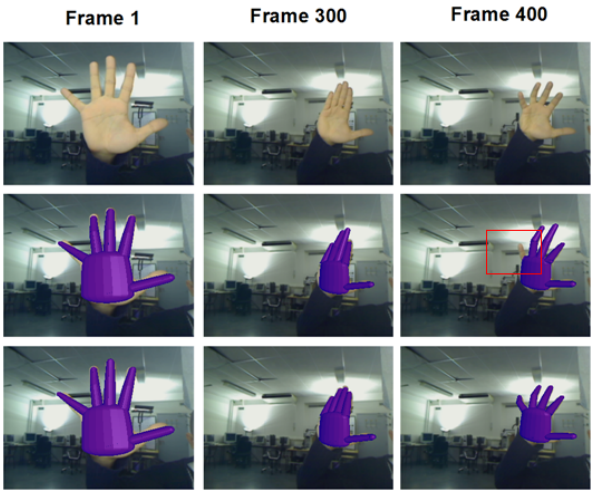


Figure 5: Tracking results for real images. The top row shows the selected frames of the video sequence, the middle row shows the tracking results of the one-step algorithm and the bottom row shows the tracking results of the two-step algorithm. The tracking loss is framed in red.

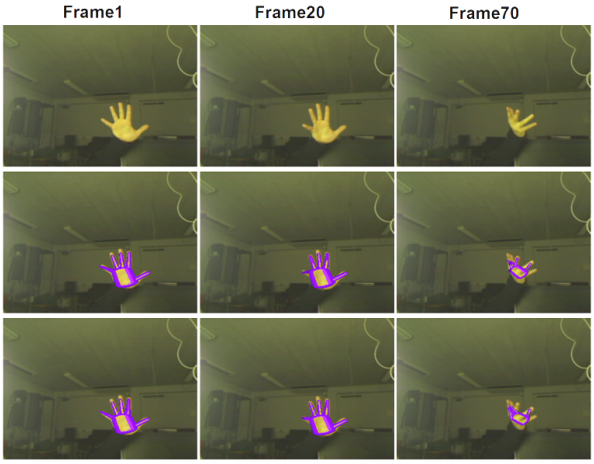


Figure 6: Self-occlusion problem illustrated with a video sequence of 70 frames

6 CONCLUSION AND FUTURE WORK

The main obstacle to articulated hand motion tracking is the large number of degrees of freedom (around 26) to be recovered. Search algorithms, either deterministic or stochastic, fall foul of exponential computational complexity. Deterministic approaches formulate the hand tracking problem as a minimization one, in which a dissimilarity function comparing a hand pose with a 3D hand model is to be minimized. The high number of the DOF compound the main problem of minimization algorithms: local minima. To overcome this problem, we present first a new dissimilarity function which gives better results when it is compared with other well-known functions, like the directed chamfer or hausdorff distances as shown in the experimental results section. To minimize this dissimilarity function we propose a two-step algorithm operating on two stages. The first one gives the global DOF of the hand, i.e. position and orientation of the palm, whereas the second step provides the local DOF of the hand, i.e. finger joint angles. Operating in two stages, the proposed two-step algorithm reduces the complexity of the minimization problem. Indeed, it seems more robust to local minima than a one-step algorithm and improves the computing time needed to get the desired solution as shown in the experimental results section.

When using a single camera it remains difficult to deal with complex motion such as X-rotation or Y-rotation. Indeed, we are confronted with the problem of self-occlusion. To solve this problem, we plan to spend several cameras. Using multiple cameras, we will be able to deal with more complex hand motion.

A study in deepening course reveals that a GPU implementation of our method will significantly increase its computational power.

7 ACKNOWLEDGMENTS

Our research works have been achieved as part of the PlayALL project, which aims to develop a cross-platform toolchain standard for entertainment and video games. We thank very much the Cap Digital Cluster and the General Directorate for Enterprises (DGE) for their financial support. We also thank Brice Michoud for its very helpful comments.

REFERENCES

[BKMM*04] BRAY M., KOLLER-MEIER E., MUELLER P., GOOL L. V., SCHRAUDOLPH N. N.: 3d hand tracking by rapid stochastic gradient descent using a skinning model. In *1st European Conference on Visual Media Production (CVMP)* (March 2004), Chambers A., Hilton A., (Eds.), pp. 59–68.

[BKMSG04] BRAY M., KOLLER-MEIER E., SCHRAUDOLPH N. N., GOOL L. V.: Stochastic meta-descent for tracking articulated structures. In *IEEE Workshop on Articulated and Nonrigid Motion, Conference on Computer Vision and Pattern Recognition (CVPR)* (Washington, DC, USA, 2004), IEEE Computer Society, p. 7.

[Bor88] BORGEFORS G.: Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence 10*, 6 (1988), 849–865.

[DF98] DELAMARRE Q., FAUGERAS O.: Finding pose of hand in video images: a stereo-based approach. In *IEEE Proc. of the third International Conference on Automatic Face and Gesture Recognition* (1998), IEEE Computer Society, pp. 585–590.

[HH96] HEAP T., HOGG D.: Towards 3d hand tracking using a deformable model. In *Face and Gesture Recognition* (1996), pp. 140–145.

[HKKR93] HUTTENLOCHER D. P., KLANDERMAN G. A., KL G. A., RUCKLIDGE W. J.: Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15* (1993), 850–863.

[IB98] ISARD M., BLAKE A.: Condensation conditional density propagation for visual tracking. *International Journal of Computer Vision 29* (1998), 5–28.

[IKS07] IKE T., KISHIKAWA N., STENGER B.: A real-time hand gesture interface implemented on a multi-core processor. In *MVA* (2007), pp. 9–12.

[KX06] KATO M., XU G.: Occlusion-free hand motion tracking by multiple cameras and particle filtering with prediction. *IJCSNS International Journal of Computer Science and Network Security 6*, 10 (2006), 58–65.

[LWH00] LIN J., WU Y., HUANG T.: Modeling the constraints of human hand motion. In *HUMO '00: Proceedings of the Workshop on Human Motion (HUMO'00)* (Washington, DC, USA, 2000), IEEE Computer Society, p. 121.

[NM65] NELDER J. A., MEAD R.: A simplex method for function minimization. *The Computer Journal 7*, 4 (January 1965), 308–313.

[OH99] OUHADDI H., HORAIN P.: 3d hand gesture tracking by model registration. In *Proc.IWSNHC3DI99* (1999), pp. 70–73.

[RASS01] ROSALES R., ATHITSOS V., SIGAL L., SCLAROFF S.: 3d hand pose reconstruction using specialized mappings. In *ICCV* (2001), pp. 378–385.

[RK94] REHG J. M., KANADE T.: Visual tracking of high dof articulated structures: An application to human hand tracking. In *European Conference on Computer Vision* (1994), Springer-Verlag, pp. 35–46.

[RK95] REHG J. M., KANADE T.: Model-based tracking of self-occluding articulated objects. In *ICCV* (1995), pp. 612–617.

[SKS01] SHIMADA N., KIMURA K., SHIRAI Y.: Real-time 3-d hand posture estimation based on 2-d appearance retrieval using monocular camera. In *RATFG-RTS '01: Proceedings of the IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems (RATFG-RTS'01)* (Washington, DC, USA, 2001), IEEE Computer Society, p. 23.

[SMC01a] STENGER B., MENDONCA P. R. S., CIPOLLA R.: Model-based 3d tracking of an articulated hand. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on 2* (2001), 310.

[SMC01b] STENGER B., MENDONCA P. R. S., CIPOLLA R.: Model-based hand tracking using an unscented kalman filter. In *Proc. British Machine Vision Conference, volume I* (2001), pp. 63–72.

[Ste04] STENGER B. D. R.: *Model-Based Hand Tracking Using A Hierarchical Bayesian Filter*. PhD thesis, University of Cambridge, 2004.

[Tos06] TOSAS M.: *Visual Articulated hand tracking for Interactive Surfaces*. PhD thesis, University of Nothingham, 2006.

[WH99] WU Y., HUANG T. S.: Vision-based gesture recognition: A review. In *GW '99: Proceedings of the International Gesture Workshop on Gesture-Based Communication in Human-Computer Interaction* (London, UK, 1999), Springer-Verlag, pp. 103–115.

[WLH01] WU Y., LIN J. Y., HUANG T. S.: Capturing natural hand articulation. In *ICCV* (2001), pp. 426–432.

[WLH05] WU Y., LIN J., HUANG T. S.: Analyzing and capturing articulated hand motion in image sequences. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 27*, 12 (2005), 1910–1922.

[WP09] WANG R. Y., POPOVIĆ J.: Real-time hand-tracking with a color glove. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers* (New York, NY, USA, 2009), ACM, pp. 1–8.

Enhanced Visual Depth Cues for Collocated Visuo-Haptic Augmented Reality

B. Knörlein
knoerlein@vision.ee.ethz.ch
Computer Vision Laboratory
ETH Zurich

G. Székely
szekely@vision.ee.ethz.ch
Computer Vision Laboratory
ETH Zurich

M. Harders
harders@vision.ee.ethz.ch
Computer Vision Laboratory
ETH Zurich

ABSTRACT

Our current research focuses on the application of visuo-haptic augmented reality in medical training. The setup developed in this context enables collocated haptic interaction with scene objects. In order to allow natural manipulations, provision of appropriate depth cues becomes a crucial factor. Therefore, we have included fast occlusion handling and shadow synthesis in our augmented environment. The occlusion map is initialized using a plane sweep approach, followed by an edge-based optimization via a Mumford-Shah functional. For obtaining the depth map three head mounted cameras are used and a left-right consistency check is performed to provide robustness against half occlusions. Shadowing is implemented via shadow mapping, considering both real and virtual objects. All steps have been implemented on GPU shaders and are performed in real-time.

Keywords: Augmented reality, occlusion handling, shadow casting.

1 INTRODUCTION

Augmented Reality (AR) extends the real environment with virtual components. In a video-see-through system, virtual objects are superimposed onto the image stream of the real world. By integrating a collocated haptic device into the setup, also haptic augmentations can be provided. Our current research focuses on the application of such visuo-haptic AR setups in medical training of open surgical procedures. In this setting, a user actively manipulates the augmented, collocated environment, leading to occlusions between real and virtual objects.

Depth perception has been identified as a major concern in AR systems [28]. This becomes even more critical if interactions with scene objects are performed. Occlusions and shadows provide key cues for visual perception to infer depth, thus also increasing immersion into the simulated environment [4, 29, 22]. Absence or incorrect presentation of these effects can lead to sensory conflicts. Therefore, we developed approaches to incorporate appropriate visual depth cues into our AR rendering pipeline. We focused on accurate occlusion handling and shadow casting. Moreover, since low latency in a visuo-haptic system is crucial for maintaining veridical haptic perception [15], we also targeted providing these visual cues in real-time.

For occlusion handling we first obtain a depth map using a plane sweep algorithm. For this step three cameras integrated into a head mounted display are being used. Thereafter, the depth is used to provide an initial guess of occlusions, which is further optimized through a segmentation process using a Mumford-Shah functional. The occlusions are optimized by incorporating the total variation of the color images to align the occlusions with image edges. The main contribution of our approach is that no a priori information about the environment is required. This is in contrast to most other approaches, which first perform a segmentation in order to extract the possible occluders. Finally, the reconstructed depth is employed to cast shadows in the augmented scene. The details of these steps will be explained below.

2 RELATED WORK

Resolving occlusions is still a major issue in current AR research. Approaches can be divided into four categories: model-, segmentation-, contour-, and depth-based techniques. The former two require a priori information about the real environment. Model based approaches use the shape and pose of the occluder [6], while segmentation-based techniques incorporate knowledge about the appearance of the occluder or the background [7, 25]. The latter two approaches have proven to be more flexible. Nonetheless, silhouette-based approaches often need manual input or are not feasible in real-time [2, 17], and occlusions obtained from depth, in general, lack fidelity [32, 27]. Novel approaches make use of a combination of different information for the detection of occluders. A depth recon-

Figure 1: RGB images of left, center, and right camera.

struction step is often combined with a previous segmentation, thus requiring knowledge about appearance [18, 14, 31, 19]. To bypass these problems, a more sophisticated approach, has been presented in [33]. They suggest to employ a probabilistic framework combining color, depth, and neighborhood information to handle occlusions. We also combine depth reconstruction with image segmentation. However, we first reconstruct the depth of the scene and then use this information in a segmentation process in combination with the camera images. No a priori information of the scene is therefore required. Different approaches can be used for the depth reconstruction, e.g. shape from shading [8], but application in AR has to be performed in real time.

Several techniques have been used for shadow casting in AR, e.g. shadow volumes using stencil buffer [10] or shadow textures [13]. More sophisticated approaches use image based information [30] or target the detection, fusion and adjustments of virtual and real shadows [24, 12, 23]. While knowledge about scene depth has been used in other contexts in AR [9, 16], the application to shadow casting for dynamic objects has not been considered before. We use a soft shadow algorithm combining shadow maps with the scene depth reconstruction. A similar approach for casting static shadows has recently been reported in [20].

3 VISUO-HAPTIC ENVIRONMENT

We use a stereoscopic video see-through setup with three dragonfly2 firewire cameras, weighing in total less than 300g. Two of the cameras are placed in front of the head mounted display (HMD) to record the live streams shown to the user. They are aligned with the HMD displays with a baseline of 65mm. A third, centered camera is mounted on the HMD to support the depth reconstruction process. The HMD provides a 40 degree field of view (FOV). To increase camera overlap for the depth reconstruction, the image size of the cameras is increased towards the opposite side by 60 pixels (i.e. 5 degrees). However, only a 400x300 pixel region corresponding to 40 degrees FOV is presented to the user. The center camera acquires 500x300 pixel images at a FOV of about 50 degrees. The colorspaces of the three cameras have been precalibrated to each other by using a color checker. An example view of the environment captured by the three cameras is shown in Figure 1. The head pose of the user is provided by an external infrared tracking system. Haptic feedback is rendered on a PHANToM 1.5 interface, simulating the kinesthetic force response from virtual scene objects. The overall setup and the HMD used are depicted in Figure 2. More details on the system and the calibration procedures can be found in [11]. Known real objects in the scene, e.g. the haptic device stylus or the table, are registered in the environment and represented by virtual counterparts. The latter can for instance be used for shadow casting as well as occlusion handling.

4 DEPTH RECONSTRUCTION

In order to recover the scene depth in real-time we extend the approach suggested in [3]. Depth is reconstructed via a plane sweep technique. The basic idea of this class of algorithm is to sweep the space between a near and far limit with parallel planes at an arbitrary number of discrete depths. Pixels in a reference image are projected onto corresponding pixels in additional test images at each plane depth. Thereafter, the errors between the RGB values of these pixels are determined. The depth of a pixel in the reference image is then set to that of the plane resulting in a minimal error measure. Cornelis et al. further increase the robustness of this technique by performing a hierarchical plane sweep with connectivity constraints.

When reconstructing depth in a video-see through setup, camera baseline and FOV are given by the hardware specifications. Moreover, the working distance is in general below 1m. This leads to strong

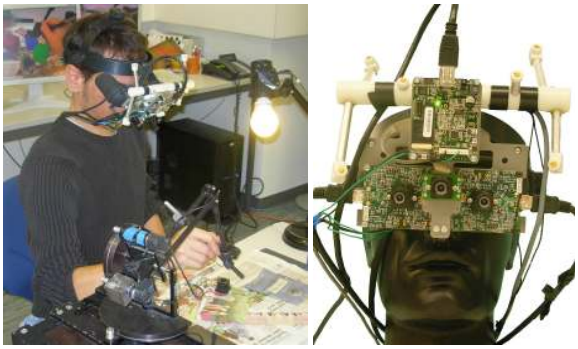


Figure 2: Collocated visuo-haptic AR environment (left) and three camera HMD setup (right).

half occlusions and differences in object appearance in the images (see Figure 1). The original approach by Cornelis et al. is not sufficiently robust in this case. Smooth transitions or wrong depth values for the half occluded regions result. Therefore, we have applied several modifications to their approach.

Coarse level depth initialization: Our depth reconstruction process is performed on a four-level hierarchy. The coarsest level is a 8x8-downsampling of the original reference image. In the first step, the depth for the center view at the coarsest level is determined. The space from 1000 cm to 10 cm is swept by 64 planes. These are equally distributed in reciprocal depth space, thus providing a more equally distributed sampling in image space. For a plane located at depth d, the pixel at $\mathbf{p}_c = (x_c, y_c)$ with color value $\mathcal{C}_c$ in the reference image is projected onto the left and right view, resulting in image coordinates $\mathbf{p}_l^d$ and $\mathbf{p}_r^d$. Using homogeneous coordinates the projection from view a to b is given by

$$\begin{pmatrix} x_b^d \\ y_b^d \\ w_b^d \end{pmatrix} = [\mathbf{Pr}_b^a|\mathbf{0}] \begin{pmatrix} x_a \\ y_a \\ 1 \\ 1 \end{pmatrix} + \frac{1}{d}\mathbf{Pt}_b^a \qquad (1)$$

$\mathbf{Pr}_b^a$ and $\mathbf{Pt}_b^a$ are the 3×3 and 3×1 submatrices of the 3×4 projection matrix given by

$$[\mathbf{Pr}_b^a|\mathbf{Pt}_b^a] = [\mathbf{K}_b\mathbf{R}_b^a\mathbf{K}_a^{-1}|\mathbf{K}_b\mathbf{t}_b^a] \qquad (2)$$

with the internal camera parameters $\mathbf{K}$ and the transformation $[\mathbf{R}_b^a|\mathbf{t}_b^a]$ from view a to b. The projected pixels are then normalized resulting in the image coordinates $\mathbf{p}_{l,r}^d = (x_{l,r}^d/w_{l,r}^d, y_{l,r}^d/w_{l,r}^d)$ and the color values $\mathcal{C}_l(\mathbf{p}_l^d)$ and $\mathcal{C}_r(\mathbf{p}_r^d)$ are interpolated in the full resolution images. Based on these, the matching score for the depth d is determined through sums of squared differences (SSD).

$$\epsilon^d(\mathbf{p}_c) = \frac{SSD(\mathbf{p}_c, \mathbf{p}_l^d) + SSD(\mathbf{p}_c, \mathbf{p}_r^d) + SSD(\mathbf{p}_l^d, \mathbf{p}_r^d)}{3} \qquad (3)$$

determined for coordinates $\mathbf{p}_a$ and $\mathbf{p}_b$ and color values $\mathcal{C}_a$ and $\mathcal{C}_b$ via

$$SSD(\mathbf{p}_a, \mathbf{p}_b) = \frac{1}{|\mathcal{N}_W|} \sum_{\substack{\tilde{\mathbf{p}}_a \in \mathcal{N}_W(\mathbf{p}_a) \\ \tilde{\mathbf{p}}_b \in \mathcal{N}_W(\mathbf{p}_b)}} \|\mathcal{C}_a(\tilde{\mathbf{p}}_a) - \mathcal{C}_b(\tilde{\mathbf{p}}_b)\|_2^2 \qquad (4)$$

in the 3x3 neighborhood

$$\mathcal{N}_W = \{(x, y) : |x - x_0| \leq 1, |y - y_0| \leq 1\} \qquad (5)$$

The initial depth estimate is then obtained through

$$d_{rec}(\mathbf{p}_c) = \begin{cases} d & : \min_d(\epsilon^d(\mathbf{p^c})) \leq \epsilon_{\mathbf{max}} \\ \infty & : \epsilon^d(\mathbf{p}^c) > \epsilon_{max}\, \forall d \end{cases} \qquad (6)$$

Thus, if for a pixel no matching score below a threshold ϵ_{max} is found, we assume that a half occlusion occurs. However, it is still possible to determine depths for these points in the subsequent steps. Parameter ϵ_{max} is set to 0.01 for the normalized pixel color range $[0, 1]$.

Hierarchical refinement: In the next steps, the lower-level depth estimates are repeatedly upsampled to higher levels until the original resolution is reached. After upsampling one level, two iterations of a median filter are performed first to reduce the influence of outliers. Thereafter, the depth map is refined using a connectivity constraint. In this step, the depth values of neighboring pixel are evaluated for the current pixel and the best match is set. The search neighborhood is alternated in each iteration between an orthogonally-adjacent

$$\mathcal{N}_{ort} = \{(x, y) : |x - x_0| + |y - y_0| = 1\} \qquad (7)$$

and a diagonal one

$$\mathcal{N}_{diag} = \{(x, y) : |x - x_0| + |y - y_0| = 2\} \qquad (8)$$

Only valid, non-occluded depth values are considered in this step, thus resulting in a first set of neighboring depth values $\mathcal{D}_\mathcal{N}$. Note that these have either been determined during the initial sweep or during one of the following refinements. In order to obtain a piecewise smooth depth approximation, this set is then extended with the mean value of the neighboring depth values $\bar{d}_\mathcal{N}$. This set of possible depth values is then evaluated for the current pixel. To determine the best match, a modification of Equation 6 is used. For computing the SSD this time no neighborhood window is applied.

$$SSD_{fine}(\mathbf{p}_a, \mathbf{p}_b) = \|\mathcal{C}_a(\mathbf{p}_a) - \mathcal{C}_b(\mathbf{p}_b)\|_2^2 \qquad (9)$$

Note that to further prioritize a smooth reconstruction, the matching score of the mean depth is multiplied by 0.75 when finding the depth according to Equation 6. The best matching depth is then assigned to the currently examined pixel.

The described refinement process with connectivity constraints is carried out five times on each level. Additional iterations were not useful, since only negligible changes occurred. The resulting fine-level depth reconstruction for the central view is shown in Figure 3 (right).

Consistency check: The reconstructed depth of the center camera is finally projected onto the left and right view. In order to enhance the result, the previously described refinement step is applied two times, using the left and right view as starting reference images, respectively. However, especially in half occluded regions still mismatches are present. These can for instance be seen in the example image in Figure 3 (middle) in the occluded region right to the sponge.

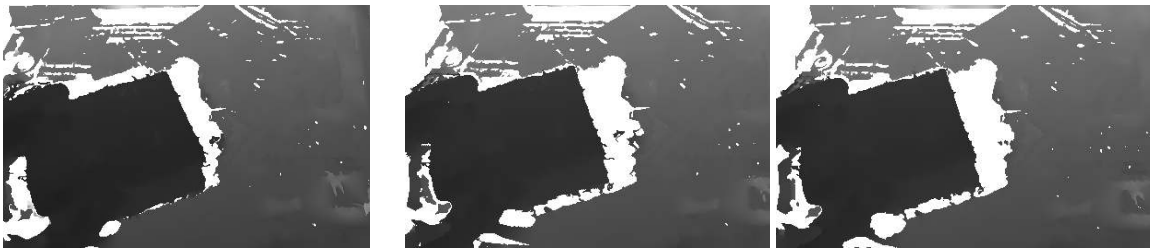


Figure 3: Depth reconstructed for center camera in Figure 1 (left). Fully refined depth map for right camera before and after outliers removal (right).

To remove these outliers, a left-right consistency check is carried out. For both views, pixels are projected from one into the other. The pixel is then backprojected into the first one by using the depth of the second view. If the distance in image space between the backprojected and the original pixel is below a threshold $dist_{max}$, the depth is assumed to be correct. Otherwise the left and right images do not describe the same world point and we set $d_{rec} = \infty$ for that pixel. To allow for small outliers and perspective distortions, $dist_{max}$ was set to 3 pixels.

After this outliers removal the neighbor refinement is again performed two times to fill gaps in the depth maps. The final result after consistency check is also depicted in Figure 3 (right).

In order to determine occlusions, the reconstructed depth d_{rec} could be compared to that of the virtual scene. However, the obtained depth map is still of insufficient quality, exhibiting several inaccuracies as can be seen in Figure 4 (left). Therefore, the determined depths are used to provide an initial guess in a further optimization step.

5 OCCLUSION OPTIMIZATION

The central idea of the occlusion optimization is to align the initial occlusion with image edges. This assumes that the occluding objects exhibit visible contours. To this end, we apply the Mumford-Shah segmentation model [21], resulting in a piecewise smooth approximation of image intensity by minimizing the energy functional

$$E_{MS} = \int_{\Omega} (u-g)^2 \, d\Omega + \alpha \int_{\Omega \setminus \Gamma} |\bigtriangledown u|^2 d\Omega + \beta \, len(\Gamma) \tag{10}$$

where g is the image, u the smooth approximation, and $len(\Gamma)$ the length of the set of evolved edges Γ in g. The positive tuning parameters α and β are related to scale and contrast in the image. Pock et al. [26] showed that the model can be applied, as well, to evolve a piecewise smooth approximation of a color image and a smooth depth map exhibiting the same set of edges Γ by using an iterative update process.

However, this scheme does not allow to account for missing depth values due to half occlusions. Therefore, we first determine a dense occlusion map and assume that half occlusions only effect image background. This is done by comparing the reconstructed depth map of the real environment d_{rec} with that of the virtual scene d_{vr}. In addition, the depth of any known precalibrated objects d_{pre}, e.g. the haptic device stylus, is also integrated. The initial dense occlusion map o is thus determined according to

$$o(x,y) = \begin{cases} 1 & : \; d_{rec}(x,y) < d_{vr}(x,y) \\ 0 & : \; d_{vr}(x,y) \leq d_{rec}(x,y) \\ 0.25 & : \; d_{vr}(x,y) \leq d_{pre}(x,y) \wedge \\ & \quad d_{vr}(x,y) = \infty \\ 0.75 & : \; d_{pre}(x,y) < d_{vr}(x,y) \wedge \\ & \quad d_{vr}(x,y) = \infty \end{cases} \tag{11}$$

An example of an initial occlusion map and the corresponding augmentation is illustrated in Figure 4 (left).

Starting from the initial RGB image $\mathbf{u}_0$ and occlusion map o_0, an iterative update process is then carried out, evolving both towards the edges Γ through linear diffusion. The RGB images are updated by

$$\mathbf{u}^{k+1}(x,y) = \frac{\mathbf{u}^0(x,y) + \sum_{(\tilde{x},\tilde{y}) \in \mathcal{N}(x,y)} \mu^k(x,y,\tilde{x},\tilde{y}) \mathbf{u}^k(\tilde{x},\tilde{y})}{1 + \sum_{(\tilde{x},\tilde{y}) \in \mathcal{N}(x,y)} \mu^k(x,y,\tilde{x},\tilde{y})} \tag{12}$$

where the μ^k are diffusion weights defined below. The neighborhood $\mathcal{N}$ is alternated in each iteration between the previously used $\mathcal{N}_{diag}$ and $\mathcal{N}_{ort}$ to remove sampling artifacts. Next, we update the occlusion map according to

$$o^{k+1}(x,y) = \frac{\delta o^0(x,y) + \sum_{(\tilde{x},\tilde{y}) \in \mathcal{N}(x,y)} \mu^k(x,y,\tilde{x},\tilde{y}) o^k(\tilde{x},\tilde{y})}{\delta + \sum_{(\tilde{x},\tilde{y}) \in \mathcal{N}(x,y,\tilde{x},\tilde{y})} \mu^k(x,y)} \tag{13}$$

where δ is a smoothing weight. However, the latter hinders evolution and does not allow for strong outliers. Pock et al. [26] apply a term including multiple depth hypotheses to encounter the problem. In our case, we use the matching cost ϵ of the assigned depth value $d_{rec}(x,y)$ to regulate the smoothness of the occlusion map with a maximum weight δ_{max}.

$$\delta = \begin{cases} 0 : & d_{vr}(x,y) = \infty \\ \delta_{max} \frac{\epsilon_{max} - \epsilon}{\epsilon_{max}} : & else \end{cases} \tag{14}$$

Thus, the error norm is removed if no correspondence can be detected, while it is preserved if an accurate

Figure 4: Occlusion map and augmentation before (left) and after refinement (right).

match is found. This results in stronger diffusion in regions where no or less accurate depth values could be found and weaker in those with accurate correspondences. The parameter δ_{max} was set to 100. The diffusion weights μ^k are given by

$$\mu^k(x,y,\tilde{x},\tilde{y}) = \frac{\mathcal{A}_\mathcal{N} \cdot \mathcal{B}_\mathcal{N}}{1 + \mathcal{B}_\mathcal{N} \cdot G(x,y,\tilde{x},\tilde{y})} \tag{15}$$

with $\mathcal{A}_\mathcal{N}$ and $\mathcal{B}_\mathcal{N}$ being positive constants and G a joint color-occlusion gradient. The former parameters depend on the constants α and β as well as on the specific neighborhood. In our case $\mathcal{A}_\mathcal{N}$ and $\mathcal{B}_\mathcal{N}$ were given as 15.54 and 705.09 for $\mathcal{N}_{ort}$, and 11.64 and 470.4 for $\mathcal{N}_{diag}$, respectively. Finally, the joint gradient term is specified by

$$\begin{aligned} G(x,y,\tilde{x},\tilde{y}) = \quad & \gamma \|\mathbf{u}(x,y) - \mathbf{u}(\tilde{x},\tilde{y})\|_2^2 \\ & + (1-\gamma)\,|o(x,y) - o(\tilde{x},\tilde{y})|^2 \end{aligned} \tag{16}$$

where the γ weights the influence of color and occlusion on the evolved edges. Since strong discontinuities are present in the initial occlusion map, we determine the gradients at the beginning of the iterations only from the color images by setting $\gamma = 1.0$. After 50 iterations the value is changed to 0.9. The overall update process is only carried out inside regions where virtual objects are rendered. If none is present, no information about occlusions is required.

Sufficient results using this optimization were achieved after 100 iterations. Figure 5 illustrates the evolution outcome for a subregion in the example scene of Figure 4. The left image shows the original RGB data located under the rightmost virtual cube, while the other two images depict the evolved edges and the diffused RGB data. The final evolved occlusion map as well as the augmentation is shown on the right of Figure 4. The final occlusion is determined by a binarization of the evolved result with a threshold of 0.5.

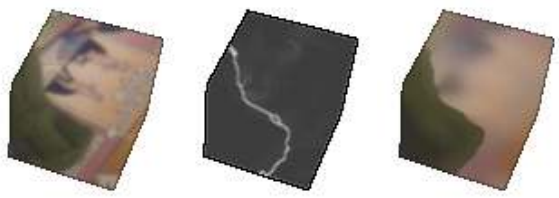


Figure 5: Original RGB image before, and evolved edges and diffused RGB image after optimization.

6 SHADOW CASTING

In AR shadow casting should occur between real and virtual objects. In general, a light emitter casts upon an occluding object, while light occluder and shadow receiver can both be real or virtual. By using shadow maps discussed in [20], convincing soft shadowing can be provided in real-time [1]. If a depth map of the scene is acquired, the approach shows another advantage. A key step for the algorithm is the determination of the depths of the first light-blocking objects. This allows to adjust the shadow map using the previously reconstructed depth map and thereby to include dynamic shadow casting of real objects.

When rendering virtual objects, shadows are cast from virtual and real entities. Therefore, the first occluder in either of the two environments has to be determined. To this end, shadow maps are determined for the virtual S_{virt} and the real S_{real} scene. The former is acquired by rendering the scene from the viewpoint of the tracked real light source. For the latter, the reconstructed depths of the camera views are rendered as point clouds from the view of the light source. Outliers are removed by median filtering. Moreover, the geometries of calibrated objects, e.g. the haptic device stylus or the table, are also integrated. For these, a shadow map S_{pre} is determined by rendering their virtual representation from the direction of the light source. A modified map is obtained by combining this shadow map with the real by determining

$$\tilde{S}_{real} = \begin{cases} S_{real} & : \quad S_{real} < (S_{pre} - t_{s1}) \\ S_{pre} & : \quad else \end{cases} \tag{17}$$

The threshold t_{s1} is introduced to increase robustness against outliers. In our examples it is set to 10mm. The final shadow map for rendering the augmented scene is then obtained as $S_{fin} = \min(\tilde{S}_{real}, S_{virt})$.

For rendering shadows on real surfaces, we currently, only determine virtual shadows on precalibrated objects, e.g. the table. This is done by casting shadows

Figure 6: Augmentation without and with shadows cast from a tracked real lightsource.

on the known real world geometry based on the determined virtual shadow map. Here, it has to be noted that occlusion handling has to be performed as well for the newly added shadows. In order to prevent occlusions from the real surface the shadow is cast upon an offset t_{s2} is added to the virtual depth d_{vr} when the occlusion map is initialized through Equation 11. This parameter is also set to 10mm.

Since we do not have information about the diffuse albedo of the real object, the shadowed pixel is modified by multiplying its color value with a shadow coefficient k_s and using an offset off to preserve an ambient light.

$$\mathcal{C}_{out} = \mathcal{C}_{in}\ (off + (1.0 - off)(1.0 - k_s)) \qquad (18)$$

The offset off is set to 0.5. For computing the penumbra of the shadows a percentage-closer filtering is applied [5]. Sixteen texture lookups are used for occluder search and shadow coefficient computation. These are performed based on a randomly rotated Poisson disk distribution. Results of the shadow casting approach are depicted in Figure 6.

7 RESULTS

Visual processing in our AR system is fully implemented on the GPU. Camera images are transferred to GPU texture memory, where all processing is carried out. Unconverted bayer images of all three cameras are uploaded, demosaiced by downsampling to half resolution, color adjusted by precalibrated 3x3 conversion matrices and undistorted. Based on these images the depth reconstruction is carried out. Thereafter,the virtual scene is rendered with shadows and the coarse occlusion map is determined and refined. Finally, the occlusion map is applied and the augmented scene is visualized to the user. Note that rendering the virtual scene and occlusion handing are only performed for the left and right camera only.

The overall process can be carried out at 20 fps on a GTX 285 graphics card. Timings for the different steps of the pipeline are provided in Table 1. The time for the occlusion refinement is given for a masked frame buffer corresponding to the virtual objects. The mask is rendered once to the depth buffer and regions outside the virtual drawing are culled by using an OpenGL depth test. The occlusion handling using the masking requires two rendering passes for either updating the color images or the occlusion map. An alternative implementation without masking can use rendertargets providing updates for both, the color images and the occlusion map, in one render pass. By using this technique occlusion handling can be performed in 38 ms for the full image. Example stereo frames of the full application are shown in Figure 7.

Nevertheless, the depth reconstruction has some limitations. Accurate depth maps can be acquired if the scene is well textured. However, homogeneous areas in the background or repetitive patterns decrease the reliability. In these cases, color keying or background subtraction can be employed to solve for occlusions. Furthermore, specular highlights can lead to additional errors in the depth reconstruction. Still most of these problems can be resolved in the occlusion refinement step.

The choice of the parameter ϵ_{max} directly influences the resulting depth map. A high threshold causes several outliers while lower values result in a sparse depth. The latter, however, is suitable for the occlusion optimization.

The total variation term of the occlusion optimization performs well without incorporation of a priori information. The fidelity of the occlusion is sensitive to the parameters α and β, which can be used to adjust to the scene. High α values result in a kernel (Equation 5) which provides stronger diffusion across image boundaries. In this case, the occlusion in half occluded regions becomes a smooth approximation. In contrast to this, higher values for β increase the influence of

Processing step	Performance in ms
Image upload	1.1
Demosaicing	0.4
Undistortion	0.3
Depth reconstruction	16
Virtual rendering with shadows	2.5
Occlusion refinement	24
Augmentation	0.3
Overall performance	47.6

Table 1: Performance of augmentation pipeline.

Figure 7: Examples of left and right view of final augmentation during collocated interaction with virtual objects.

edges and reduce the diffusion across image boundaries. However, some problems occur if only small regions are detected for either fore- or background during the depth reconstruction. In this case the refinement tends to prioritize the larger areas. Using an enlarged bounding box for virtual objects could solve this problem. In addition, another problem occurs if no edge between fore- and background objects is evident. This can lead to occlusions leaking from foreground objects into half occluded regions until the diffusion is stopped by a strong edge. Additional stereo or temporal constraints of the occluding contour could be used to encounter this problem. In addition, due to the resemblance of the procedure used to level set implementation, additional energy components, e.g. gradient vector flow or curvature forces could also be integrated. Finally, instead of determining the occluder with a fixed threshold, connectivity of occluding segments and adaptive thresholding could lead to better results.

The depth used for shadow casting was based on the camera view. Projection of the reconstructed depth map only produces correct results, if the perspectives are not too different. It would be better to recover the depth directly from the direction of the light source, for instance by using additional cameras. Noise in the depth maps in general reduces the quality of shadow casting.

8 CONCLUSION

In our current work we are developing a visuo-haptic AR system to be applied in medical training of open surgical procedures. In order to provide enhanced depth cues in the augmented scene, as well as to increase user immersion, we integrated real-time occlusion handling and shadowing in our system. The depth of the scene is recovered using a hierarchical plane sweep process combined with a left-right consistency check. Optimization applying a Mumford-Shah functional is carried out to obtain accurate occlusions. Moreover, the depth map is used for casting dynamic shadows.

Future work will investigate additional constraints for occlusion refinement, e.g temporal or stereo constraints as well as contour curvatures. In addition, a more de-

tailed quantitative evaluation of the approach will be carried out. Furthermore, we will also investigate possibilities for fusing real and virtual shadows in real-time.

Acknowledgments This work has been performed within the frame of the EU project Immersence IST-2006-27141.

REFERENCES

[1] L. Bavoil. Advanced soft shadow mapping techniques. In *GDC slides*, 2008.

[2] M. O. Berger. Resolving occlusion in augmented reality: a contour based approach without 3d reconstruction. In *CVPR*, pages 91–96, 1997.

[3] N. Cornelis and L. V. Gool. Real-time connectivity constrained depth map computation using programmable graphics hardware. *CVPR*, 1:1099–1104, 2005.

[4] D. Drascic and P. Milgram. Perceptual issues in augmented reality. In *SPIE Volume 2653: Stereoscopic Displays and Virtual Reality Systems III*, pages 123–134, 1996.

[5] R. Fernando. Percentage-closer soft shadows. In *SIGGRAPH*, page 35, 2005.

[6] M. Fiala. Dark matter method for correct augmented reality occlusion relationships. *HAVE*, pages 90–93, 2006.

[7] J. Fischer, H. Regenbrecht, and G. Baratoff. Detecting dynamic occlusion in front of static backgrounds for ar scenes. In *EGVE*, pages 153–161, 2003.

[8] D. Gelli and D. Vitulano. Surface recovery by self shading projection. *Signal Processing*, 84(3):467–473, 2004.

[9] G. Gordon, M. Billinghurst, M. Bell, J. Woodfill, B. Kowalik, A. Erendi, and J. Tilander. The use of dense stereo range data in augmented reality. In *ISMAR*, pages 14–23, 2002.

[10] M. Haller, S. Drab, and W. Hartmann. A real-time shadow approach for an augmented reality application using shadow volumes. In *VRST*, pages 56–65, 2003.

[11] M. Harders, G. Bianchi, B. Knoerlein, , and G. Székely. Calibration, registration, and synchronization for high precision augmented reality haptics. *TVCG*, 15(1):138–149, 2009.

[12] K. Jacobs, J.-D. Nahmias, C. Angus, A. Reche, C. Loscos, and A. Steed. Automatic generation of consistent shadows for augmented reality. In *GI*, pages 113–120, 2005.

[13] T. Kakuta, T. Oishi, and K. Ikeuchi. Virtual kawaradera: Fast shadow texture for augmented reality. In *CREST*, pages 79–85, 2005.

[14] H. Kim, D. Min, S. Choi, and K. Sohn. Real-time disparity estimation using foreground segmentation for stereo sequences. *Optical Engineering*, 45(3):037402(1–10), 2006.

[15] B. Knörlein, M. di Luca, and M. Harders. Influence of visual and haptic delays on stiffness perception in augmented reality. In *ISMAR*, pages 49 – 52, 2009.

[16] A. Ladikos and N. Navab. Real-time 3d reconstruction for occlusion-aware interactions in mixed reality. In *ISVC*, November 2009.

[17] V. Lepetit and M.-O. Berger. A semi-automatic method for resolving occlusion in augmented reality. *CVPR*, 2:225–230 vol.2, 2000.

[18] L. Li, T. Guan, and B. Ren. Resolving occlusion between virtual and real scenes for augmented reality applications. In *HCI (2)*, pages 634–642, 2007.

[19] Y. Lu and S. Smith. Gpu-based real-time occlusion in an immersive augmented reality environment. *Journal of Computing and Information Science in Engineering*, 9(2):024501(1–4), 2009.

[20] C. B. Madsen and R. E. Laursen. A scalable gpu-based approach to shading and shadowing for photorealistic real-time augmented reality. In *GRAPP (GM/R)*, pages 252–261, 2007.

[21] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, 42(5):577–685, 1989.

[22] T. Naemura, T. Nitta, A. Mimura, and H. Harashima. Virtual shadows - enhanced interaction in mixed reality environment. In *VR*, pages 293–294, 2002.

[23] G. Nakano, I. Kitahara, and Y. Ohta. Generating perceptually-correct shadows for mixed reality. *ISMAR*, 0:173–174, 2008.

[24] M. Nielsen and C. B. Madsen. Graph cut based segmentation of soft shadows for seamless removal and augmentation. In *SCIA*, pages 918–927, 2007.

[25] J. Pilet, V. Lepetit, and P. Fua. Retexturing in the presence of complex illumination and occlusions. In *Ismar*, pages 249–258, 2007.

[26] T. Pock, C. Zach, and H. Bischof. Mumford-shah meets stereo: Integration of weak depth hypotheses. In *CVPR*, pages 1–8, 2007.

[27] J. Schmidt, H. Niemann, and S. Vogt. Dense disparity maps in real-time with an application to augmented reality. In *WACV*, pages 225–230, 2002.

[28] T. Sielhorst, C. Bichlmeier, S. Heining, and N. Navab. Depth perception a major issue in medical ar: Evaluation study by twenty surgeons. In *MICCAI*, pages 364–372, 2006.

[29] N. Sugano, H. Kato, and K. Tachibana. The effects of shadow representation of virtual objects in augmented reality. In *ISMAR*, pages 76–83, 2003.

[30] P. Supan, I. Stuppacher, and M. Haller. Image based shadowing in real-time augmented reality. *IJVR*, 5(3):1–7, 2006.

[31] J. Ventura and T. Hollerer. Depth compositing for augmented reality. In *SIGGRAPH posters*, page 1, 2008.

[32] M. M. Wloka and B. G. Anderson. Resolving occlusion in augmented reality. In *SI3D*, pages 5–12, 1995.

[33] J. Zhu, Z. Pan, C. Sun, and W. Chen. Handling occlusions in video-based augmented reality using depth information. In *Computer Animation and Virtual Worlds, Special Issue*, page n/a, 2009.

A Simple Compression of Tri-Quad Meshes with Handles

Ruben G. Diaz.[1], Marcelo Dreux[1], Hélio Lopes[2], Thomas Lewiner[2]
[1]Deparment of Mechanical Engineering
[2]Department of Mathematics
Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 225, Gávea
Phone: +55 (21) 3527-1162/1639/1164
CEP 22453-900 - Rio de Janeiro - RJ - BRAZIL
rgomez@tecgraf.puc-rio.br, dreux@puc-rio.br, {lopes, lewiner}@mat.puc-rio.br

ABSTRACT

This paper extends a mesh compression method to deal with meshes composed of triangular and quadrilateral faces with genus. The proposed method is very useful given the fact that many CAD meshes have these characteristics. A quad extension has been introduced to the original EdgeBreaker encoding and decoding algorithm. The modification is simple and amounts to a case analysis of the possible compression labels assigned to two triangles originating from splitting a quadrilateral face. The new method entails to augmenting the C,L,E,R,S EdgeBreaker codes with the additional labels c,l,s.

Keywords: Mesh Compression, EdgeBreaker, Geometric Compression, Quad-Tri meshes.

1 INTRODUCTION

In many applications it is necessary to transmit 3D models over the Internet, and these models are usually large and complex. Due to these characteristics it is convenient to compress data in order to transmit them, in a simple and efficient way.

Boundary representation models consist of an indirect representation of a solid or mesh through its border description.

Many boundary representation models for 3D objects have been proposed, most of them represented by triangular meshes. Nevertheless, polygonal meshes with both triangle and quads are used in many CAD applications, asking for a compression scheme of the mesh model.

This work presents a data structure that is a simplification of the *HalfEdge* data structure. This new data structure is called *CHalfedge* and is used for the compression and decompression of static meshes composed by triangles and quads with or without handles.

CHalfedge does not support dynamic meshes, i.e., the mesh cannot be edited or adapted but can be used for progressive meshes transmission [15] [8].

2 RELATED WORK

2.1 Topological Data Structure

There are many well-known data structures used for boundary representation of solids in $\mathbb{R}^3$ [1] [14] [21]. Rantal et al. [16] presented an extension of the *HalfEdge* data structure [14] in order to deal with boundary for cut and paste operations with applications for feature based modeling. In their scheme, the boundary edge is represented as an edge with only one half-edge. *CHalfEdge*, the proposed data structure, uses the same strategy.

Guibas and Stolfi [6] proposed a generalization to include 2-manifolds, orientable or not, where Edge Functions and the corresponding *Edge-Algebra* have been implemented as the *Quad-Edge* data structure with a set of operators.

Dobkin e Lazlo [4] presented the *Facet-Edge* data structure and a set of operators to represent cell complexes that extended the work presented by Guibas and Stolfi. Brisson [2] generalized the last two works to include n-manifolds with the *Cell-Tuple* data structure and operators that are generically called Constructors.

Castelo et al [3] used the Handlebody theory to introduce the *Handle-Edge* data structure in order to represent combinatorial surfaces with boundary. They introduced a set of operators to create handles. Lopes and Tavares [13] presented the *Handle-Face* data structure and a set of operators to represent 3-manifolds.

2.2 Geometric Compression

Many mesh compression methods have been proposed, among them the Topological Surgery compression method proposed by Taubin [20]. This method

compresses the TST(Triangle Spanning Tree [20]) tree as well as its dual vertex tree. The MPEG-4 standard for 3D compression is based on the Topological Surgery technique [20], which was co-invented by Rossignac at IBM Institute.

Another compression technique, also proposed by Rossginac, is the EdgeBreaker compression [18] [17] for triangle meshes without handles. It uses the *CornerTable* data structure [12] to represent the connectivity of any manifold triangle mesh by the use of two integer arrays.

Rossignac also showed a decompression technique to triangular meshes that has been codified by the EdgeBreaker method. This decompression method is known as Wrap&Zip [19] and it has a quadratic compression in the worst case. The method presented by Rossignac is restricted to closed meshes, homeomorphic to a sphere, without boundary and without genus.

King et al [9] presented a codification and compression for quad meshes homeomorphic to a sphere. Although in this work the authors have mentioned that it is possible to extend the algorithm to surfaces with handles using irregular meshes (triangles and quads), they do not present an algorithm.

Lopes et al [12] extended the work of Rossignac [18] [17] of triangle meshes considering handles in a *CornerTable* data structure and Lewiner presented a codification for meshes with an arbitrary topology [11]. Lage et al [10] also uses a pseudo-representation of the *HalfEdge* data structure, like the representation used in this work, in order to create a scalable topological data structure only for triangle and tetrahedral meshes.

3 BASIC CONCEPTS

In this section some basic concepts of geometry and topology are presented [5]. In the m-dimensional Euclidean vector space, a *convex cell* $\sigma \in \mathbb{R}^m$ is a compact set (limited and closed), not empty, that is the solution of a finite set of linear equalities $f_i(v) = 0$ and linear inequalities $g_i(v)$ where $v = (x_1, x_2, ..., x_m) \mapsto \lambda_0 + \lambda_1 x_1 + \lambda_2 x_2 + ... + \lambda_m x_m = 0$.

A face, or a subcell of a cell σ, is obtained replacing some inequalities $g_i \geq 0$ by equalities. A vertex of a cell is a subcell with only one point. The cell σ is considered a subcell of itself. A subcell of σ that is not itself is called proper subcell. A *simplex* of dimension n is the convex hull of $n+1$ linearly independents points in $\mathbb{R}^m$.

The subcells of cell σ are defined by σ itself and not by the choice of a particular equations and inequations that represent them. This can be inferred by the following propositon:

All τ cells are a subcell of σ if and only if:

1. If P is the hyperplane generated by τ, then $P \cap \sigma = \tau$
2. No point of P belongs to a convex combination of two points of $\sigma - \tau$.

A n-dimensional complex cell K is a finite collection of i-dimensional cells $(i = 0, ..., n)$ in $\mathbb{R}^m$ under the following conditions:

1. If $\sigma \in K$ and $\tau < \sigma$ then $\tau \in K$.
2. If σ and $\rho \in K$ then σ e ρ are properly linked.

The 0, 1 and 2-cells of the complex K are called, respectively, the *vertices*, *edges* and *faces* of K.

The *underlying polyhedron* $|K| \in \mathbb{R}^m$ corresponds to the union of the cells in K. If a collection of cells $L \in K$ is a cell complex, then it is called a subcomplex of K.

A complex K is *connected* if it cannot be represented as a union of two non empty disjoint subcomplexes L and M without common cells. A *component* of a simplicial complex K is a connected subcomplex that is not contained in a larger connected subcomplex of K.

A combinatorial n-manifold is *orientable* when it is possible to choose a coherent orientation of its n-cells, i.e., two adjacent n-cells induce opposite orientations on their common $(n-1)$-cells. From now on, a *surface* and a *curve* will always mean respectively, a 2 and 1 dimensional connected oriented combinatorial manifold with or without boundary. The set of faces (2-cells), edges (1-cell) and vertices of a surface will be denoted $F(S)$, $E(S)$ and $V(S)$.

The mesh compression and decompression processes that are presented in this article are applied to combinatorial oriented surfaces without boundary. So it is necessary to respect some characteristics that ensures the preservation of the topological combinatorial aspects of the mesh. The next definition grants the existence of such characteristic for combinatorial surfaces:

Let NV, NE, e NF be respectively, the number of vertices, edges and faces of a combinatorial surface $\mathscr{S}$.

$$\chi(\mathscr{S}) = NV - NE + NF \tag{1}$$

Then the sum is a constant for all surfaces with the same topological type of $\mathscr{S}$. This constant is called Euler characteristic.

This definition can be expanded to include other components of the model.Consider an orientable surface S as a single connected component and let g be the number of genus and b the number of boundary curves of the surface. Then, the Euler-Poincaré formula is written as:

$$\chi(\mathscr{S}) = NV - NE + NF = 2 - 2g - b \tag{2}$$

The number of Euler-Poincaré $\chi(\mathscr{S})$ defines univocally the surface S. When the surface is oriented and has no boundary, the number of genus g of S,

knowing only the number of vertices, edges and faces of $S(NV,NE,NF)$ can be obtained by the Euler-Poincaré formula:

$$NF - NE + NV = 2 - 2g \tag{3}$$

$$\Downarrow$$

$$g = 1 - \frac{NF}{2} + \frac{NE}{2} - \frac{NV}{2} \tag{4}$$

The attachment of handles to a n-manifold component is relevant to a complete representation of boundaries, and is based on the HandleBody Theory [3] and [12].

4 CHALFEDGE DATA STRUCTURE

The CHalfEdge data structure proposed in this work is a simplification of the HalfEdge [14] data structure. It is suitable for polygonal meshes composed by triangles and quads and also uses the concept of half-edge that represents the association of a face with one of its boundary edges. The advantage of this simplification is that it works with two arrays of integers, V and M, which are later described.

Consider an oriented combinatorial surface S without border having NT triangles, NQ quads, NE edges and NV vertices. In this data structure the half-edges, the vertices and the faces are indexed by non-negative integers.

Each triangle is represented by 3 consecutive half-edges that define its orientation. Analogously, a quad is represented by four consecutive half-edges. For example, half-edges 0, 1 and 2 correspond to the first triangle; half-edges 3, 4 and 5 corresponds to the second triangle, half-edges 6, 7, 8 and 9 correspond to the first quad if there is a quad in the mesh and so on. A half-edge with index $h \in [0, 3NT-1]$ belongs to a triangle, and if $h \in [3NT, 3NT+4NQ-1]$ belongs to a quad.

The border of each triangle is implicitly represented by an oriented cycle of 3 half-edges. In the same way, an oriented cycle of 4 half-edges defines the border of a quad. In such a way, the surface S will have $3NT+4NQ$ half-edges. Assuming a counter-clockwise orientation, the next convention is adopted to index the half-edges of S:

- The indices of the half-edges that form the border of a triangle of index t are: $3t$, $3t+1$, and $3t+2$.
- The indices of the half-edges that form the border of a quad q are: $3NT+4q$, $3NT+4q+1$, $3NT+4q+2$, and $3NT+4q+3$.

The index of a triangle that has a half-edge h is $h.f = h \div 3$. If the half-edge h belongs to a quad, its index is $h.f = (h-3NT) \div 4$. The next(h.n) and previous (h.p) half-edges of h.f are obtained by the use of the following expressions:

Triangles

$$h.n = 3 * h.f + (h+1) \mod 3 \tag{5}$$

$$h.p = 3 * h.f + (h+2) \mod 3 \tag{6}$$

Quads

$$h.n = 3 * NT + 4 * h.f + (h+1) \mod 4 \tag{7}$$

$$h.p = 3 * NT + 4 * h.f + (h+3) \mod 4 \tag{8}$$

The CHalfEdge data structure represents the geometry of a surface S by the association of each half-edge h with its geometrical vertex index h.v. The edge adjacency between the neighboring triangles or quads of S is represented by the association of each half-edge h with its opposite half-edge h.m or M[h](mate), which has the same opposite geometrical edge(formally M[M[h]] = h, see Figure 1). The geometry and adjacency information are stored in two integer arrays, called the V and M tables, respectively.

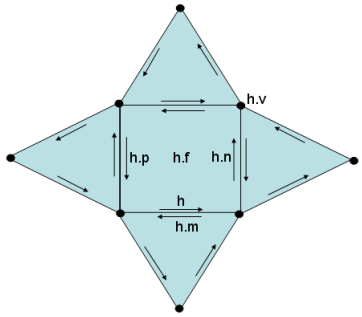
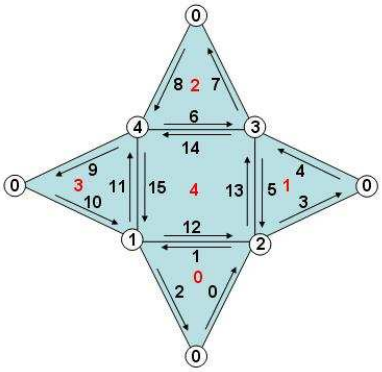


Figure 1: CHalfEdge notations.

According to the definition of combinatorial surface without bounding border, each edge is shared by only 2 faces. Two adjacent faces that share the same edge have half-edges with opposite orientation.

To illustrate the data structure tables consider the pyramid of Figure 2. When the half-edge h is associated to a boundary edge, its M[h] value is assigned to -1.



V[0] = 0	M[0] = 3
V[1] = 2	M[1] = 12
V[2] = 1	M[2] = 10
V[3] = 2	M[3] = 0
V[4] = 0	M[4] = 7
V[5] = 3	M[5] = 13
V[6] = 4	M[6] =14
V[7] = 3	M[7] = 4
V[8] = 0	M[8] = 9
V[9] = 4	M[9] = 8
V[10] = 0	M[10] = 2
V[11] = 1	M[11] = 15
V[12] = 1	M[12] = 1
V[13] = 2	M[13] = 5
V[14] = 3	M[14] = 6
V[15] = 4	M[15] = 11

Figure 2: A pyramid surface with its CHalfEdge tables.

Table V of Figure 2 indicates that half-edge 0 has vertex 0, half-edge 1 has vertex 2, and so on. Table M indicates that half-edge 0 has as its correspondent mate, or opposite, half-edge 3, half-edge 1 has its correspondent mate half-edge 12, and so on.

5 COMPRESSION OF TRI-QUAD MESHES

The algorithm for compression and decompression of tri-quad meshes proposed in this work is a generalization of the EdgeBreaker algorithm introduced by Rossignac [17].

The EdgeBreaker is a state machine that encodes a combinatorial surface without boundary S. At each state, a decision is made to move from a triangle Y to an adjacent triangle X. To perform this decision, all visited triangles and their incident vertices are marked.

Let Left and Right denote the other two triangles that are incident upon X. Let v be the vertex common to X, Left and Right. There are five possible situations denoted by the letters C,L,E,R and S (arranged for the mnemonic "Claire's"). The arrows in Figure 3 indicate the direction to the next triangle. Previously visited triangles are filled. The edge that crosses from Y to X is called the gate of one triangle to another.

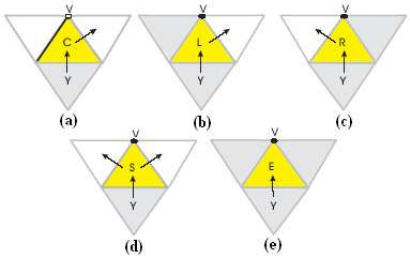


Figure 3: EdgeBreaker clers cases.

The EdgeBreaker algorithm walks to the right whenever is possible(C or L labels, see Figure 3a and 3b respectively). When the right triangle has already been visited it walks to the left(R label - Figure 3c). A triangle gets label S (split - Figure 3d) if the vertice v is already visited and both right and left triangles are not visited yet. In this case a branch in the dual spanning tree is created. Each of these branches has to be traversed and ends up with a label of type E (End - 3e). Whenever the traverse of a S triangle reaches the end of the right branch, the algorithm goes back to that S triangle and then traverses the left branch.

This kind of surface traversing is always possible when the surface is homeomorphic to a sphere. When the surface without boundary has g genus, there are some cases that when going back to the beginning of the branch generated by a label of type S, the left face of this branch has already been visited during the right traversing. Lopes et al [12], proposed in these cases a correct reconstruction by transmitting the edge to the left of the S triangle.

Lopes et al [12], proved that for a surface without border and with g genus, there are $2g$ S triangles of this type. Then the number $2g$ exactly corresponds to the number of handle operations of type 1 (see [12]) that must be applied to the surface during the decompression process to obtain its topology correctly.

5.1 EdgeBreaker Quad Extension

A simple way to compress a quad mesh is to divide each polygon into two triangles and apply the EdgeBreaker algorithm for triangle meshes. But, in the decompression process it is necessary to recognize which faces have been added in order to reconstruct the original mesh(see Figure 4).

King et al [9] suggest to divide a quad into two adjacent triangles, by implicitly adding a diagonal edge. This division is done in such way to force the second triangle to be always to the right of the first triangle. The quad subdivision rules also guarantee that the first triangle label is never R or E because the next triangle of the quad has not been visited. Furthermore, if the first triangle label is of type C, the second label cannot be of type L or E(see Figure 4).

For a triangle there are 5 possible combinations (see EdgeBreaker compression). For other types of polygons the number of combinations can be calculated by the recurrence relationship, based on the fact that a vertex that has not yet been visited can not be incident to an edge previously visited. The solution of this recurrence for polygon with n edges is the Fibonacci number $F(2n-1)$. Then for a quad there are $F(7) = 13$ possible combinations of edges and vertices not adjacent to a visited and not visited gate.

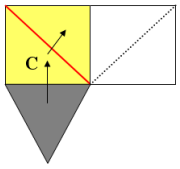


Figure 4: Quad division proposed by King [9].

The approach division leads to 13 possible pairs of label combinations for a quad: CC, CR, CS, LC, LL, LE, LR, LS, SC, SL, SE, SR and SS. In order to build the dual spanning tree, the original EdgeBreaker rules are used for each triangle generated by a quad subdivision.

5.2 Tri-Quad Mesh Codification

The algorithm developed in this work encodes an arbitrary surface by generating 3 separated information which are used to transmit a mesh over the Internet: a sequence of CLERS containing the labels c, l, s, C,

L, E, R, and S of each codified face of the surface; the geometry of the vertices and the handles. Thus, in order to compress a mesh composed by triangles and/or quads, must follow these rules:

1. If the face is a triangle, the labels C, L, R, E, or S are used, as in the original EdgeBreaker;
2. If the face is a quad, it is split into two triangles according to King's rule, and the first triangle is encoded with labels c, l or s. The second triangle receives labels C, L, R, E, or S.

The classification of the first quad triangle as c, l or s is the same as the original EdgeBreaker classification. The second quad triangle is always the exit of the quad. The use of a lower case label indicates that the next polygon is a quad.

Lopes et al [12] already mentioned approach is used when dealing with special cases of S triangles. In the next section it will be proved that even for surfaces with triangles and quads the number of transmitted edges is also 2g.

5.3 Analysis of the Compression Method

The compression algorithm creates the spanning tree of the dual graph, where graph nodes are the faces and graph lines represent adjacency between faces. It has complexity O(n), where n is the number of faces.

The algorithm traverses all the faces classifying them. Each face is visited only once, so the spanning tree in the dual graph has $NT+NQ-1$ lines that correspond to the edges that have gates from one face to another.

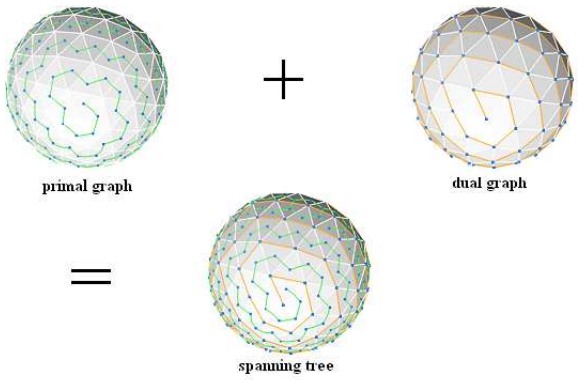


Figure 5: Compression in the primal and dual graph of the spanning tree.

Each visited face classified as c or C visits a new vertex. All vertices are visited so the total number of c or C labels is equal to NV minus the number of vertices of the initial face that can be 3 or 4 depending on being a triangle or a quad.

The algorithm creates a spanning tree in the primal graph of the surface, where the nodes of the graph are surface vertices and the lines of the graph are surface edges. The lines of the spanning tree of the dual graph correspond to the left edges of triangles labeled c or C and all the edges of the initial faces, except the gate. The number of spanning tree edges of the primal graph is, by definition $NV-1$ (see Figure 5).

The edges implicitly encoded by the algorithm are: the edge gates of the dual graph and the edges of the primal graph spanning tree. The total number of these edges is:

$$(NT+NQ-1)+(NV-1)=NV+NT+NQ-2 \quad (9)$$

The Euler-Poincare formula is:

$$NV-NE+(NT+NQ)=2-2g \quad (10)$$

So, the surface's total number of edges is defined as:

$$NE=NV+NT+NQ-2+2g \quad (11)$$

By comparing equations (9) and (10), the number of edges that has not been implicitly codified results exactly $2g$, which matches Lopes et al demonstration for triangular meshes [12].

5.4 Codification Cost

According to Euler-Poincaré formulation, a mesh with genus complies with the equation (10), where the number of edges, NE, is $(3NT+4NQ) \div 2$ and the number of faces, NF, is $(NT+NQ)$

After some simple algebraic manipulations you can reach the following equation:

$$NT+2NQ=2NV-4+4g \quad (12)$$

As already mentioned, eight symbols are used (c, l, s, C, L, E, R, and S) in order to codify and decodify tri-quad meshes. Hence, the number of necessary bits to represent a CLERS sequence is 3, since $2^3=8$ different numbers could be generated.

Triangles and quadrangles are coded with one and two CLERS labels, thus needing 3 and 6 bits, respectively.

To a tri-quad mesh the total number of bits is then (NT + 2NQ) * 3 bits. By substituting this expression in equation 12, the cost, in bits, to codify a tri-quad mesh without genus is:

$$cost=(2NV-4)*3bits \quad (13)$$

Therefore, the cost is less than or equal to 6 bits per vertex.

By the same token, the cost of a tri-quad mesh with genus is:

$$cost=(NT+2NQ)*3bits+ \quad (14)$$
$$2g*[log_2(3NT+4NQ)]*bits$$

As $NT + 2NQ = 2NV - 4 + 4g$, it follows that:

$$cost = (2NV - 4 + 4g) * 3bits + \\ 2g * [log_2(3NT + 4NQ)] * bits \quad (15)$$

When g/v « 1, which is usually the case, the cost per vertex tends to be equal 6.

6 DECOMPRESSION OF TRI-QUAD MESHES

Last section presented the idea of an algorithm to compress tri-quad meshes with handles. This section will introduce the mechanism to decompress meshes that have been codified with the method presented earlier. The decompression method builds the V and M array of CHalfEdge table by reading the stored CLERS, Geometry and Handles data.

6.1 EdgeBreaker Decompression

The methodology Wrap&Zip to decompress meshes presented in [19] and the strategy adopted by the algorithm Spirale Reversi presented in [7] will be used in this work to perform tri-quad mesh decompression. In order to rebuild a mesh codified by the algorithm presented in the last section, two steps are necessary.

The first step called Wrap decode and use the labels saved in CLERS codification to decide where a new triangle or quad will be added to the mesh that is being reconstructed. The result is a connected topological mesh that corresponds to the TST(triangle spanning tree) of the original mesh.

The second step of decompression is the Zip process, which binds the boundary curves that are not connected in order to end the reconstruction of connectivity and geometry of the surface. To correctly link the boundary edges of the mesh, the Zip process orients the free edges in counter clockwise orientation with labels of type L, R and E. Clockwise is used for labels of type C (See Figure 6).

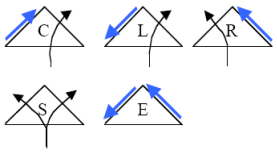


Figure 6: Edges orientation in the Wrap&Zip process.

The decompression process presented in this work also use two stacks as data structure in order to be able to perform a depth search and correctly rebuild the mesh without changing its topology. There is one stack for S labels and another for C labels.

The S stack indicates the left TST branch to be decompressed. The C stack is responsible for the Zip process, by considering the labels that have free edges after the Wrap process. The first element of the C stack will be the last to be decoded.

The Wrap&Zip process decodes and rebuilds the mesh in the same order that the triangles have been visited and coded by the compression process. This kind of decodification has the complexity time in the worst case $O(n^2)$, because this process needs to firstly search the encapsulated sub-sequences of labels S and E, to be able to correctly decode them.

The Spirale Reversi process decodes a mesh by reading the CLERS sequence in reverse order. In this case, it is not necessary to search encapsulated subsequences of labels S and E. This kind of decodification is done in linear time.

The decodification process is similar to the one presented in [7], but CLERS are read in the same order they are generated. Each triangle or quad of type C or c, is saved into a stack, and thus guaranteeing a backward reconstruction, i.e. a pseudo Spirale Reversi method in the Zip decompression process.

The decodification process is not a recursive algorithm because it depends on the data saved in S and C stacks.

7 RESULTS

It is shown some meshes (triangles/quad faces) used to perform the compression and decompression process with the corresponding CLERS.

For each encoded/decoded face it is applied a texture to show the type of CLERS label assigned to it. If a triangle is encoded the label textures show in Figure 7 are used:

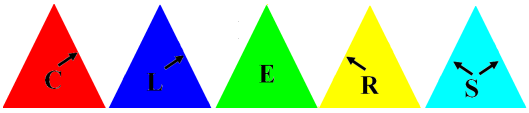


Figure 7: Triangles CLERS textures.

If the encoded/decoded element is a quad, it is divided into two adjacent triangles, as mentioned in section 6, so that there are three cases to consider for the first adjacent triangle when performing a split quad. These triangles are represented by the following label textures shown in Figure 8.

The second adjacent triangle of the quad is codified as if it were a normal triangle. The presented algorithm is able to encode/decode irregular meshes (triangles/quads) with or without an arbitrary number of genus.

Table 1 show some meshes used by compression algorithm proposed in this work, with their respective number of vertices, faces and handles.

Table shows CLERS frequency labels for triangles and quad meshes of table 1.

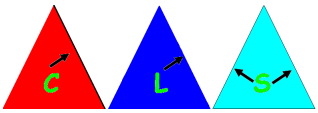


Figure 8: Quad CLERS textures.

Model	NV	NT	NQ	Handles
Bunny	4838	9672	0	0
Fandisk	6475	12946	0	0
cad1	12415	8688	8027	9
cad2	5215	6418	2014	5
knot	1997	0	1996	0

Table 1: Mesh models used to perform compression using CHalfEdge data structure

Model	Bunny	Fandisk	Cad1	Cad2	Knot
#C	4835	6472	4263	3210	0
#L	78	75	91	124	0
#E	377	170	151	254	0
#R	4005	6059	3924	2579	0
#S	376	169	154	250	0
#cC	0	0	422	260	400
#cR	0	0	7164	1372	1191
#cS	0	0	11	13	0
#lC	0	0	66	94	0
#lL	0	0	14	16	0
#lE	0	0	18	18	1
#lR	0	0	88	60	6
#lS	0	0	22	1	0
#sC	0	0	1	3	0
#sL	0	0	6	7	2
#sE	0	0	187	163	394
#sR	0	0	6	7	0
#sS	0	0	0	0	0

Table 2: CLERS frequency labels for tri-quad meshes

Figures 9, 10, 11, 12 and 13 show the meshes with triangles/quad encoded with the proposed method, presented in tables 1 and 2.

8 CONCLUSIONS

In conclusion, we have extended the simple implementation of EdgeBreaker to the connectivity graphs of triangle and quad meshes that represent topological manifolds with handles.

The compression and decompression algorithms use a simple data structure to represent the incidence and adjacency relationship of meshes with triangles and quads.

In addition, the use of an extra data structure (stack structure) in the compression and decompression algorithms enables to encode and decode the CLERS string in linear time, and there is also no need to perform search for encapsulated CLERS subsequences generated by the labels S and E. The decompression is done by a pseudo Spirale Reversi algorithm due to the use of the stack data structure.

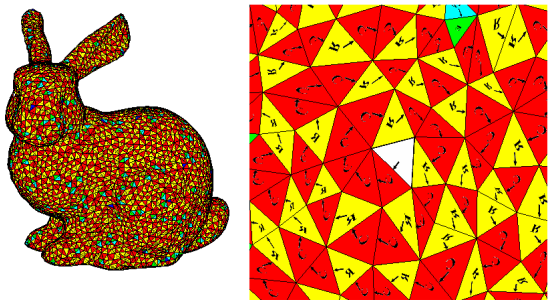


Figure 9: Mesh Bunny with 9672 triangles faces. On the right a detail of the generated compression traversal over the mesh.

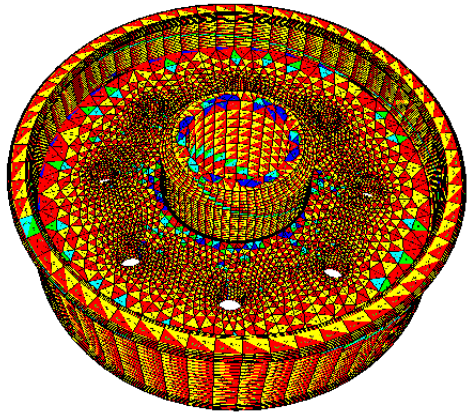


Figure 10: Cad1 mesh with 8688 triangles and 8027 quad faces with 9 handles.

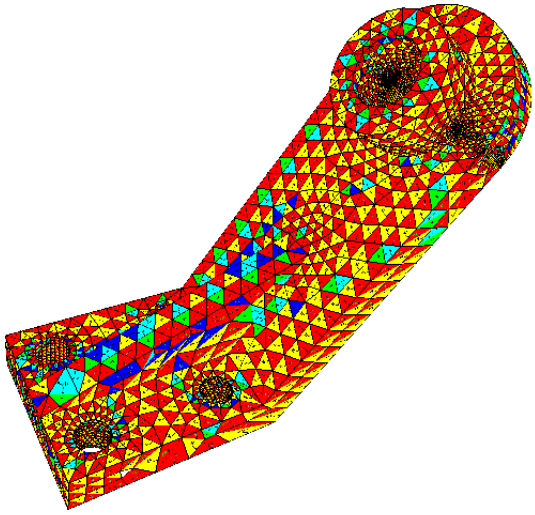


Figure 11: Cad2 mesh with 6418 triangles and 2014 quad faces, with 5 handles.

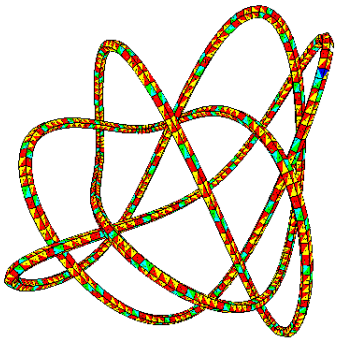


Figure 12: Knot mesh with 1996 quads.

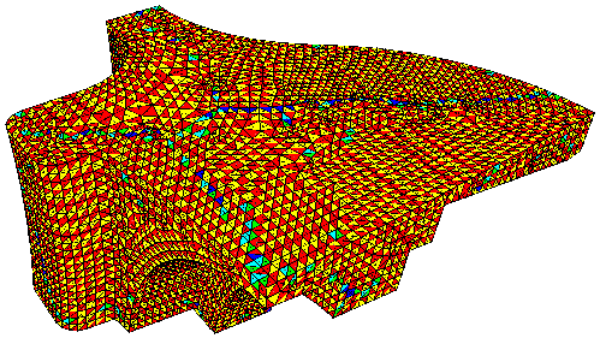


Figure 13: Fandisk mesh with 12946 triangles.

9 ACKNOWLEDGEMENT

The authors are grateful to ICAD/IGames/Vision Lab - the Intelligent CAD and Games Laboratory of PUC-Rio and MatMidia Laboratory of PUC-Rio. The first author is also grateful to CNPq, the Brazilian government research council which partially sponsored this research.

REFERENCES

[1] B. G. Baumgart. A polyhedron representation for computer vision. *AFIPS Proceedings*, (44):119–35, 1975.

[2] E. Brisson. Representing geometric strucutures in d dimesions:topology and order. *Discrete and Computational Geometry*, (9):387–426, 1993.

[3] A. Castelo, H. Lopes, and G. Tavares. Handlebody representation for surfaces and morse operations. *Proceedings on Curves and Surfaces in Computer Vision III*, pages 270–283, 1992.

[4] D.P. Dobkin and M. J. Lazlo. Primitives for the manipulation of three dimensional subdivisions. *Algorithmica*, (4):3–32, 1989.

[5] H. Edelsbrunner, M. J. Ablowitz, S. H. Davis, E. J. Hinch, A. Iserles, J. Ockendon, and P. J. Olver. *Geometry and Topology for Mesh Generation (Cambridge Monographs on Applied and Computational Mathematics)*. Cambridge University Press, New York, NY, USA, 2006.

[6] L.J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivision and the computation of voronoi diagrams. *ACM Transactions on Graphics*, (4):74–123, 1985.

[7] M. Isenburg and S. Snoeyink. Spirale reversi: reverse decoding of the edgebreaker encoding. *Computational Geometry: Theory and Applications*, pages 39–52, 2001.

[8] F. Kälberer, K. Polthier, and C. Tycowicz. Lossless compression of adaptive multiresolution meshes. In *22th Brazilian Symposium on Computer Graphics and Image Processing*, 2009.

[9] D. King, J. Rossignac, and A. Szmczak. Connectivity compression for irregular quadrilateral meshes. *GVU Tech Report GVU-GIT-99-36*, pages 1–21, 2000.

[10] M. Lage, H. Lopes, T. Lewiner, and L. Velho. Chf: A scalable topological data structure for tetrahedral meshes. *18th Brazilian Symposium on Computer Graphics and Image Processing*, pages 349–356, 2005.

[11] T. Lewiner, H. Lopes, J. Rossignac, and A. W. Vieira. Efficient edgebreaker for surfaces of arbitrary topology. *17th Brazilian Symposium on Computer Graphics and Image Processing*, pages 218–225, 2004.

[12] H. Lopes, J. Rossignac, A. Safanova, A. Szymczak, and G. Tavares. Edgebreaker: A simple implementation for surfaces with handles. *Computers&Graphics*, 27(4):553–567, 2003.

[13] H. Lopes and G. Tavares. Structural operators for modelling 3-manifolds. *Fourth ACM Siggraph Symposium on Solid Modelling*, pages 10–18, 1997.

[14] M. Mantyla. *An Introduction to Solid Modelling*. Computer Science Press, RockVille,MD, 1998.

[15] Massimiliano B. Porcu, Nicola Sanna, and Riccardo Scateni. Efficiently keeping an optimal stripification over a clod mesh. *WSCG (Journal Papers)*, pages 73–80, 2005.

[16] M. Rantal, M. Inui, F. Kimura, and M. Mantyla. Cut and paste based modelling with boundary features. *Proceedings of the second ACM/Siggraph Symphosium on Solid Modelling*, pages 303–312, 1993.

[17] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computers Graphics*, 5(1):47–61, 1999.

[18] J. Rossignac, A. Safanova, and A. Szymczak. A 3d compression made simple: Edgebreaker on a corner-table. *Proceedings of the Shape Modeling International Conference*, pages 278–283, 2001.

[19] J. Rossignac and A. Zzymczak. Wrap&zip decompression of the connectivity of triangle meshes compressed with edgebreaker. *Computational Geometry: Theory and Applications*, 14(1-3):119–135, 1999.

[20] G. Taubin and J. Rossignac. Geometric compression through topological surgery. *ACM Transactions on Graphics*, 17(2):84–115, 1998.

[21] K. J. Weiler. *Topological Structures for Geometric Modeling*. PhD thesis, Rensselaer Polytechnic Institute, 1986.

Vanishing Points Detection and Line Grouping for Complex Building Façade Identification

Wenting Duan
The University of Sheffield
Department of Electronic & Electrical Engineering
Mappin Street
Sheffield, S1 3JD
elq06wd@sheffield.ac.uk

Nigel M Allinson
The University of Sheffield
Department of Electronic & Electrical Engineering
Mappin Street
Sheffield, S1 3JD
n.allinson@sheffield.ac.uk

ABSTRACT

We describe a new method for automatically detecting vanishing points and group lines associated with building facades from a single uncalibrated image. Accurate vanishing points detection for buildings is of importance for building façade rectification and 3D scene reconstruction. The challenges arise from confusing scene clutter, occlusions, unusual building shapes and non-Manhattan street layout. Buildings usually possess many straight-line features, e.g., window and door openings, as well as their overall outline. We exploited these features and propose a robust line grouping technique. The method was evaluated on images from the Zubud-Zurich building database. The experiment shows the proposed method works well with varying building structures and image conditions, as well as filtering out "non-building" vanishing points (e.g. vanishing directions detected from road boundaries).

Keywords

Vanishing points, façade identification, line grouping, line segments detection.

1. INTRODUCTION

The rectification of building facades to their fronto-parallel view is an essential step in photogrammetry, 3D scene reconstruction and potential building recognition applications. Robust and accurate rectification requires the precise location of the extrapolated vanishing points obtained directly from, in our case, a single uncalibrated image. Buildings most commonly possess many straight-line features, e.g., window and door openings, as well as their overall outline. Exploiting these features and outlines is the basis of all published approaches. The challenges arise from confusing scene clutter, occlusions, unusual building shapes and non-Manhattan street layout. This paper presents an enhanced method for vanishing point detection that addresses these challenges.

General approaches for estimation of vanishing points are the Gaussian sphere, Hough transform, Expectation Maximisation (EM) and RANSAC-based methods. The Gaussian sphere method was first suggested by Barnard in 1983 [Bar83a]. A line segment from the image intersecting the Gaussian sphere forms a great circle, the vanishing points are found at the intersecting points of these great circles on the Gaussian sphere. The sphere surface is divided into accumulation cells, of which the ones with votes above a specified threshold are selected. This method transforms lines from unbounded to bounded space, so eliminating the singularity caused by a vanishing point located at infinity. Several processes part of the approach have since been improved; the tessellation of the Gaussian sphere [Qua89a], accumulator space [Bri91a] and projection of the intersection points for all line pairs instead of just line segments [Mag84a].

Another technique that maps line information to a bounded space is the Hough Transform [Lut94a]. Vanishing points can be obtained by applying the Hough transformation three times (known as the Cascaded Hough Transform), and was demonstrated by Tuytelaars et al. [Tuy98a]. Rother [Rot02a] pointed out that a drawback of mapping lines onto a bounded space can miss important geometric information such as the original distances between lines and points.

Kosecka and Zhang introduced a method using an EM algorithm to calculate the location of vanishing points for an uncalibrated camera [Kos02a]. This approach simultaneously groups lines parallel to real world axes and estimate dominant vanishing directions by weighting each individual line segment with a probability corresponding to every candidate vanishing direction. The problems of viewpoint change, occlusion, scene clutter and non-orthogonal building facades were addressed by our previous paper [Dua09a]. A refined approach was developed, which was based on an EM algorithm which gradually eliminated low probabilities and assigned lines to their associated group. Eildenauer and Vincze ran RANSAC several times on detected lines to obtain potential vanishing points. The EM algorithm was then used to refine the results [Wil07a]. Their method has been shown to work on images with structure not strictly meeting the Manhattan assumption [Cou03a].

Our work is most closely related to David's technique where RANSAC was used to detect vanishing points and to group lines, and line intersection was exploited to identify vanishing points corresponding to each building façade [Dav08a]. The images we are dealing with are uncalibrated, not conforming to the assumption of a Manhattan world. The aim is to detect all vanishing points and associated building façades appearing in a single image. Within an image of urban areas, detected lines can also arise from road boundaries and markings, vehicles, aerial cables, etc. Therefore, the EM algorithm, which requires an initialised line grouping, and RANSAC's randomly selected line segments, could erroneously pick up many non-building lines. Developing David's work, we suggest that selecting potential vanishing points and initial groups of lines based on a line orientation curve could reduce this type of error. A RANSAC based split and merge scheme was used to refine the vanishing point location. Finally, line intersection was used to find building façades and filter non-building vanishing points. In addition we show how geometric constraints can be utilised to reduce the exhaustive intersection process. In the Discussion, the line detection scheme used in our work was compared to the method used by David [Dav08a]. Parameters involved in our method were analysed to see how the overall grouping performance was affected by the value of the parameters. Example images demonstrate the robustness of our technique compared with previous work.

2. METHOD

2.1 Line Segments Detection

Due to the presence of geometric regularity in images containing man-made structures, lines can be easily derived from the image edges. The majority associated with building façades are also mutually parallel or orthogonal. As parallel lines of buildings intersect at vanishing points in the image under perspective transformation, line segments detected from building image are a reliable source to locate vanishing points. Firstly, we up-sampled the image if its size was less than 1M pixels in order to improve the quality of edges [Wil07a]. A Canny edge detector followed by non-maximum suppression, was used to find local edges. A binary image of the strongest edges is obtained after applying hysteresis thresholding. The line fitting stage is similar to Kosecka's method [Kos02a] originated in [Kah90a]. The normal edge orientation span [0 π] was equally divided into k bins. Take k = 18, (the set value 18 gives a bin size of 10°. This is chosen to give a reasonable bin size, enabling connected components to form within the set orientation range), hence the edge location are assigned with the same label if their orientation is within certain bin range (e.g. 10°<edge_orientation<20°). After getting 18 sets of edge pixels, every three sets with consecutive labels are grouped together for connected components analysis. To reduce repeated detection of lines and avoid boundary cut-off, the end label of the last grouping is used as the beginning label of the next. For example, (1,2,3), (3,4,5) and (5,6,7), etc. Edge lists of detected connected components with supporting points less than 15 pixels are removed. For each connected component, their supporting edge pixels are denoted as (x_i, y_i). The variance $\tilde{x} = x_i - \bar{x}$ and $\tilde{y} = y_i - \bar{y}$, where $\bar{x} = \frac{1}{n}\sum_i x_i$ and $\bar{y} = \frac{1}{n}\sum_i y_i$, was used to form the matrix D. The eigenvector $\boldsymbol{e_1}$ computed from matrix D, which corresponds to the largest eigenvalue, was used to derive the angle θ for the line we are trying to fit.

$$D = \begin{bmatrix} \sum_i \tilde{x}_i^2 & \sum_i \tilde{x}_i\tilde{y}_i \\ \sum_i \tilde{x}_i\tilde{y}_i & \sum_i \tilde{y}_i^2 \end{bmatrix} \quad (1)$$

$$\theta = \tan^{-1}(\boldsymbol{e_1}(2), \boldsymbol{e_1}(1)) \quad (2)$$

The line length is defined as the major axis length from its supporting edge pixels. The mid-point is also $(\bar{x}, \bar{y})$. Together with θ, the endpoints can be easily derived. In Figure 1, two example images and their detected lines are shown.

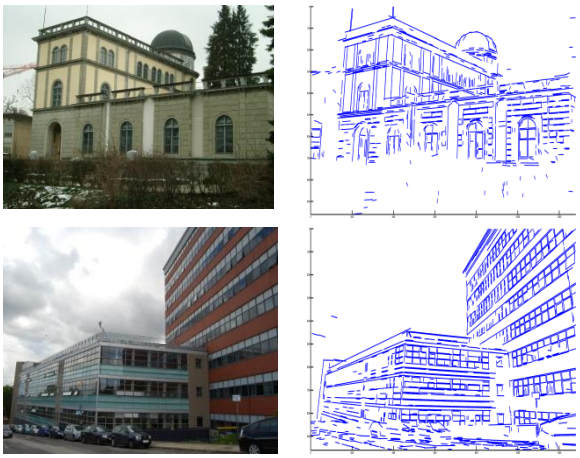


Figure 1. Two examples of line segments detected from building images.

2.2 Initial Vanishing Points Estimation and Line Grouping

According to the RANSAC based approach described by David [Dav08a] and Wildenauer and Vincze [Wil07a], two line segments were randomly selected to compute an estimation of vanishing point. Lines belonging to this direction were then grouped. The disadvantage of this method is that it will iterate exhaustively until: 1).a predefined *K* vanishing points candidate are found; or 2). Insufficient inliers are assigned. Instead, we suggest using a simple initial line grouping according to the line direction in order to create a guided selection of intersecting lines. Prior to the initial estimation scheme, each line segments are weighted by their length and colinearity. The weight range is [1, 2, 3], with every segment being equally weighted as 1 at the start. If a length is larger than 12.5% of the image size or it has more than two collinear lines, the value of 1 is added to the weighting respectively. The orientation histogram computed from the detected line segments with their associated weightings, i.e. segments with weightings of 2 are counted twice. This significantly boosts the peaks after fitting a curve to the Gaussian smoothed histogram. The curvature is computed as [Kos02a]:

$$\boldsymbol{C}(k) = h_\theta(k) - \frac{1}{s}\sum_{i=k-\frac{s}{2}}^{k+\frac{s}{2}+1} h_\theta(i) \qquad (3)$$

where k = 60 and s = 9. The starting side of the histogram is also padded with the last 5 bins at the ending side and so did the ending side before fitting the curve so that the peaks at the boundary can emerge (Figure 2(b)). The two minima located nearest to each unique peak are located and chosen as the orientation range for the initial line grouping. Hence, lines with directions falling into the defined range are grouped together. Those not assigned are grouped as the outliers as shown in black in Figure 2(c).

2.3 Split and Merge Scheme

As shown in Figure 2, this type of building structure can result in a fine initial grouping and its vanishing points can be easily detected using existing techniques. However, there are many buildings not so regular in appearance, such as the second example in Figure 1. This section shows how a split and merge scheme can appropriately group line segments.

For a homogeneous representation, $\boldsymbol{x_1} = (x_1, y_1, 1)$ and $\boldsymbol{x_2} = (x_2, y_2, 1)$. The lines are expressed in their normalised representation of the coincident infinite line $\boldsymbol{l_n}$ following the formula (4) for efficient computing of vanishing points:

$$\boldsymbol{l} = \boldsymbol{x_1} \times \boldsymbol{x_2} \qquad (3)$$

$$\boldsymbol{l_n} = \boldsymbol{l}/\sqrt{l_1^2 + l_2^2} \qquad (4)$$

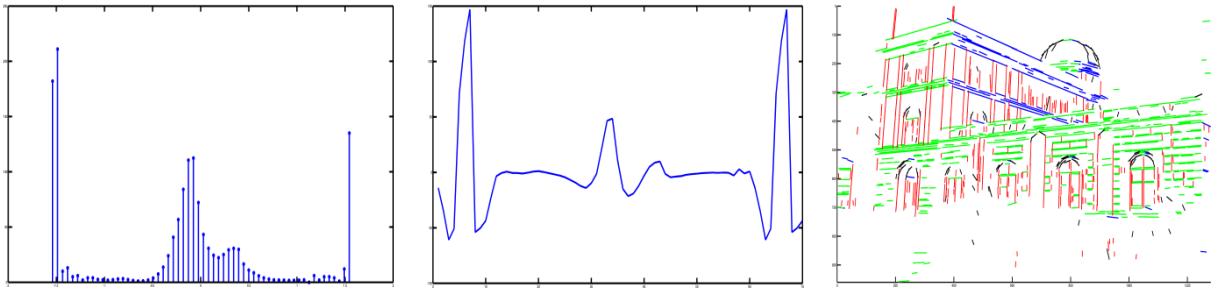
where $\boldsymbol{l} = (l_1, l_2, l_3)$. At the split stage, we assess each initially grouped line set. The line segments are sorted from highest to lowest weighting. As the lines with the higher weightings have a higher probability of giving a better estimation of vanishing point locations, two segments at the beginning of the line queue are intersected to produce the prototype vanishing point $\boldsymbol{v} = \boldsymbol{l_1} \times \boldsymbol{l_2}$. The angle θ_v and distance $\boldsymbol{d_v}$ to the prototype vanishing point from the rest of lines were calculated according to the formulas below:

$$\boldsymbol{m} = (x_m, y_m, 1) \qquad (5)$$

$$\boldsymbol{d_v} = \frac{|\boldsymbol{l_{nv}}.\boldsymbol{x_1}|}{\sqrt{\boldsymbol{l_{nv}}(1)^2 + \boldsymbol{l_{nv}}(2)^2}} \qquad (6)$$

$$\theta_v = \sin^{-1}(\boldsymbol{d}/_{0.5 * L}) \qquad (7)$$

The line $\boldsymbol{l_v}$ crosses the prototype vanishing point $\boldsymbol{v}$ and one of the assessing lines' midpoint $\boldsymbol{m}$. Its normalised version $\boldsymbol{l_{nv}}$ is calculated by (4) and was used to obtain $\boldsymbol{d_v}$. In (6), the distance between one of the prototype line's endpoints $\boldsymbol{x_1}$ and $\boldsymbol{l_{nv}}$ is obtained. Equation (7) measures the angle diversion of the prototype line from prototype vanishing point. The lines with θ_v smaller than σ_θ, and $\boldsymbol{d_v}$ smaller than σ_d are taken as inliers. If the number of inliers is larger than ψ_{min}, then the set of inliers is split from the testing line group forming a new line group. If one prototype line intersecting every other line within the group does not manage to form an inliers set, it is moved to the outlier group. The iteration will stop when there are less than ψ_{min} lines left in the prototype group, these leftover lines are also assigned to the outlier group.



(a) (b) (c)

Figure 2. Initial processing of Figure 1 (left): (a) line direction histogram; (b) curvature computation; (c) line grouping with outlier group marked black.

The initial outlier group is assessed last, because the assigned outliers from the previous groups are included. Another constraint added to the processing of this group is that the angle between each intersecting line pair needed to be smaller than 45° to continue. 45° is a practical limit angle for the parallel lines intersection to occur under perspective transformation. The vanishing points are recalculated for each line group formed. The parameters σ_θ and σ_d are 1° and 1, respectively in our experiment. This seems a rather strict threshold but the advantage is that accurate candidate parallel sets can emerge. ψ_{min} was set as 50 empirically for a reliable parallel line set to form. The point $\boldsymbol{v}$ lies on the line $\boldsymbol{l_n}$ if $\boldsymbol{v^T l_n} = 0$. This leads us to the linear least square estimation problem:

$$min_v \sum_{i=1}^{n} \left(\boldsymbol{l_i^T v}\right)^2 \qquad (8)$$

where n is the number of lines. We can rewrite this as

$$min_v \|\boldsymbol{Av}\|^2 \qquad (9)$$

The estimation of $\boldsymbol{v}$ is the eigenvector associated with the smallest eigenvalue of $\boldsymbol{A^T A}$. After the vanishing points are relocated, RANSAC was applied again to the outlier group for each vanishing direction. In The split stage has resulted in some over-grouped line sets, i.e. it is sensitive to line deviation from prototype vanishing points. The benefit is that some proper candidates of vanishing points start to emerge.

At the merge stage, a pair of line sets with similar vanishing directions is combined to calculate a new vanishing point based on (9). The inliers and outliers of the combined line group belonging to new vanishing point are found. If more than, say, 70% of lines from each merging set are inliers, the inliers are allowed to merge setting a new vanishing direction. The outliers are also moved to their associated set. The whole candidate line sets are also refreshed to eliminate the tested pair and to add the new set. Otherwise, nothing is changed and the next pair of sets is picked. The outlier group is again assessed at the end to find any inliers for the new estimated vanishing points. The resultant groups are shown in Figure 3.

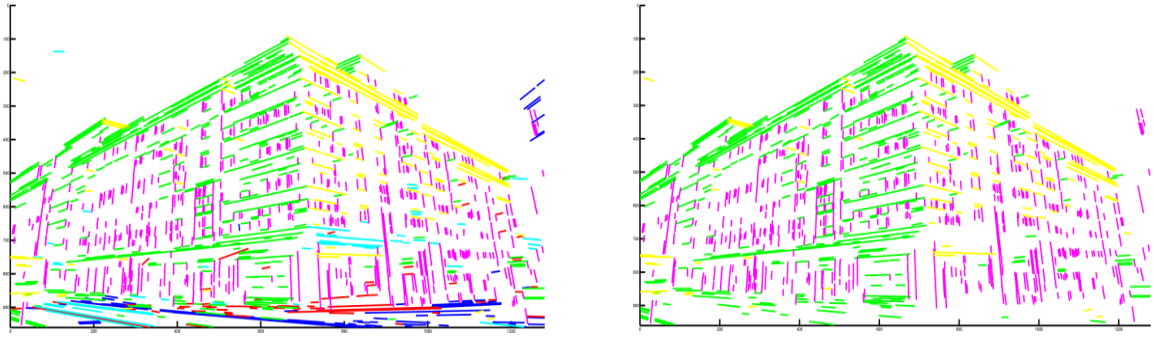
2.4 Filtering of Vanishing Points Associated with Building

Since the aim is to find vanishing points associated with building facades, and that there can be confounding line sets belonging to other regular objects such as roads or vehicles, we used a method similar to David [Dav08a] but less complicated to remove these unwanted groups. The example image in Figure 3 is used to demonstrate. In Figure 3(a) it is shown that at least two sets of lines are not associated with the building. The vanishing points corresponding to the vertical direction of the image are first selected. As the image is taken at ground level, vanishing points with negative value of y-axis are defined as vertical ones. Then, the intersection points between each of the vertical line set and every other line set are computed. The method is illustrated in Figure 4. Line segments in each set are indexed and extended by e_d pixels from their endpoints. A blank image of the same size as the original image is created for each participating line set. The lines in each group are indexed, and on their associated blank image, points consisting the extended line are labelled at their pixel location by line index *i*. The locations of labelled images with two or more recorded index are searched. This way, lines intersecting the vertical lines are found and the others filtered out. The line sets with lines intersecting infrequently are non-building sets.

3. EXPERIMENTAL RESULTS

The algorithm was tested on images from the Zubud-Zurich building database [Sha03a]. The images are in PNG format with size 640*480. The results were also compared with the previous paper [Dua09a] using EM algorithm for vanishing points detection and line grouping. First of all, the line

detection algorithm [Kah90a] was compared to Kovesi's method implemented in Matlab code [Kov07a] (used in David's [Dav08a] and Wildenauer's [Wil07a]. The parameters involved in the two methods were set to be the same, e.g. σ=1 in Canny detector, connected edge with length greater than 15 was accepted for line fitting. The result is shown in Fig 5 and Table 1. The advantages of the suggested method are: 1. Greater accuracy in line detection - this is clear on the major structural lines of the left frontal facade of the example building where the lines detected by Kovesi's method weren't parallel. 2. Quicker and computationally less complicated - the two methods were tested on the first eight images of the Zubud-Zurich building database [Sha03a]. From Table 1, we can see that the time required for Kovesi's method varies with the amount of line segments detected, whereas the current method is more stable and a lot faster. 3. A higher proportion of line segments associated with man-made structures were found; line segments associated with scene clutter such as trees were eliminated by the line detection procedure. The parameters ψ_{min}, σ_θ and σ_d were analysed to test how the overall grouping performance varies with different values. The overall grouping performance was measured by the average deviation distance from each final grouped line to their associated vanishing point. When incorrect groupings occurred, i.e. more or less groupings than actual line groupings appearing in the image were detected, a penalty weight was assigned to its measured distance. The results were obtained by running the proposed method on 30 images randomly selected from the Zubud-Zurich building database [Sha03a]. Figure 6 shows how the deviation distance varies with the setting of ψ_{min}. The dramatic changes around 30 and 70 are caused by the fact that in most cases: $\psi_{min} < 30$ leads to over-grouping (actual groups being split further) and $\psi_{min} > 70$ causes under-grouping (some groups not detected because minimum threshold set too high). Parameters σ_θ and σ_d were observed together, since individually they had a trivial effect compared to when they were changed in concert. As shown in Figure 6(b), as both σ_θ and σ_d increase, the average deviation distance from the final grouped lines to their associated VP also increases. In Figures 7 – 10, the results tested on different building image conditions are shown. In general, more line segments are detected with the new line detection method. The line sets are grouped more accurately, i.e. there are very few lines that are not associated with the detected vanishing direction. Figure 7 shows the results computed from an image taken with a sharp viewing angle to the building. The new method has shown to perform well while the EM based method fails to detect enough lines for its grouping. On buildings with multiple facades as shown in Figure 8, existing methods under the Manhattan assumption will fail to group correctly. However, the proposed method cannot handle buildings with a round structure. The lines belonging to the round part in Figure 8(a) are simply assigned to their closest vanishing direction. Figure 9 shows the results on building images with occlusion. The proposed method is actually more sensitive to occlusion where it tends to put the lines from the object causing occlusion as another group but still manages to find the groups associated with the building. A blurred image is shown in Figure 10. The method is weaker in these conditions than the EM based method. However, with fast improving camera and storage space, this may not be a problem in the future.



(a) (b)

Figure 3. Example image for demonstration of filtering out non-building segments: (a) computed group after the merge stage; (b) final line sets obtained after filtering stage.

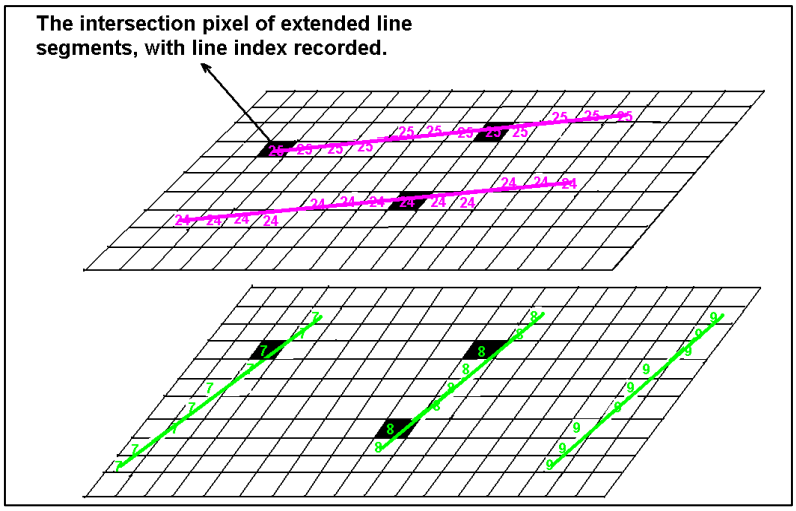
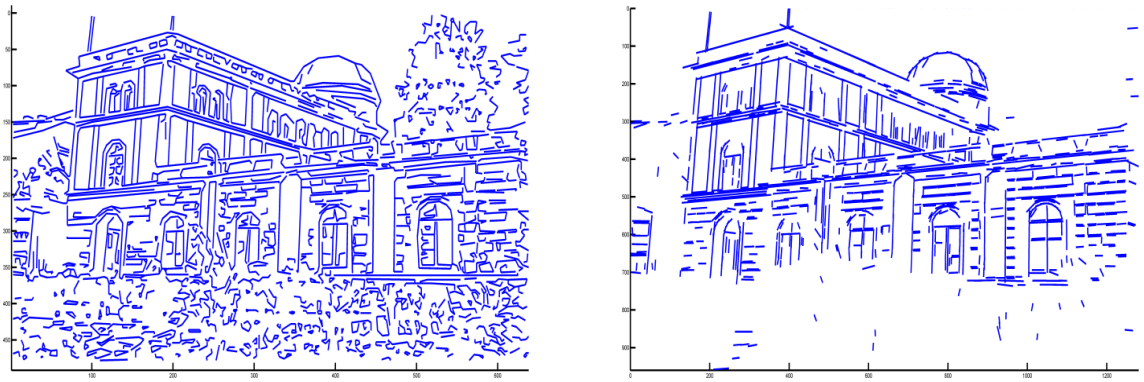


Figure 4. The illustration of how the intersections between lines were found.

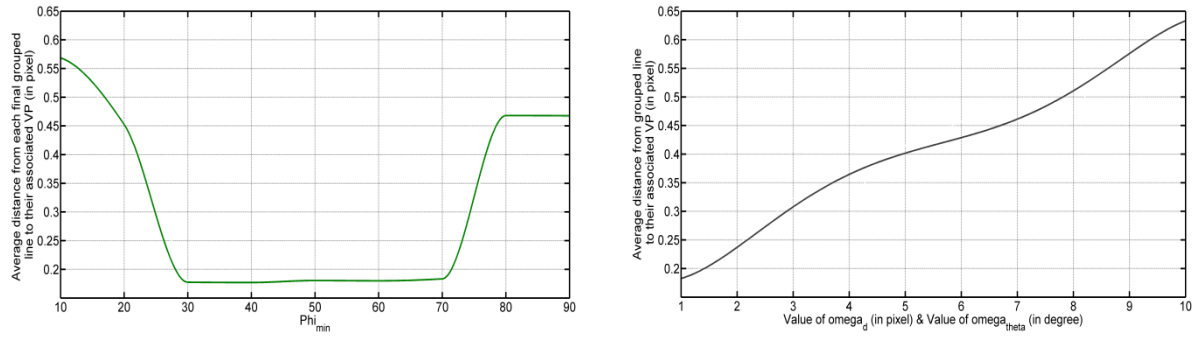


(a) (b)

Figure 5. (a) Line segments detected using Kovesi's method; (b) Line segments detected using Kahn's approach.

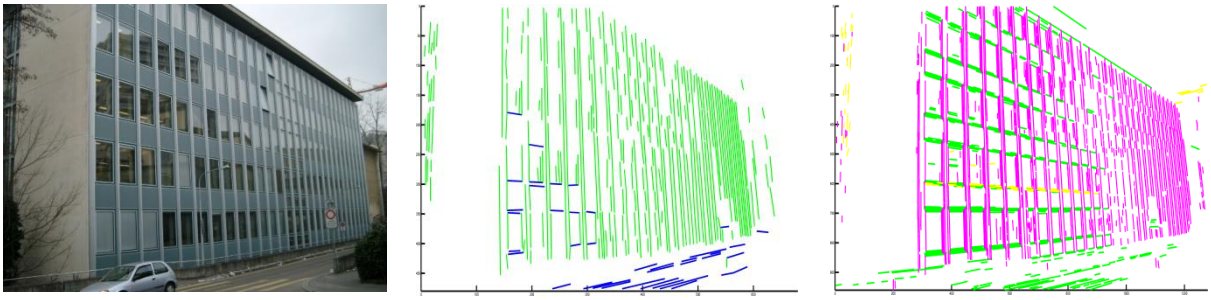
	Image1	Image2	Image3	Image4	Image5	Image6	Image7	Image8
No. lines detected by Kahn's method	1171	2513	1925	1210	1577	1034	2023	1570
No.lines detected by Kovesi's code	1058	801	1324	651	702	838	728	704
Time used for computing Kahn's method	17.96	17.48	19.82	16.48	15.23	15.92	14.97	15.39
Time used for Kovesi's original matlab code	426.31	121.24	688.08	59.09	67.06	190.63	37.04	43.26

Table 1. Comparison of two method on line segments detection in terms of line quantity and time used.



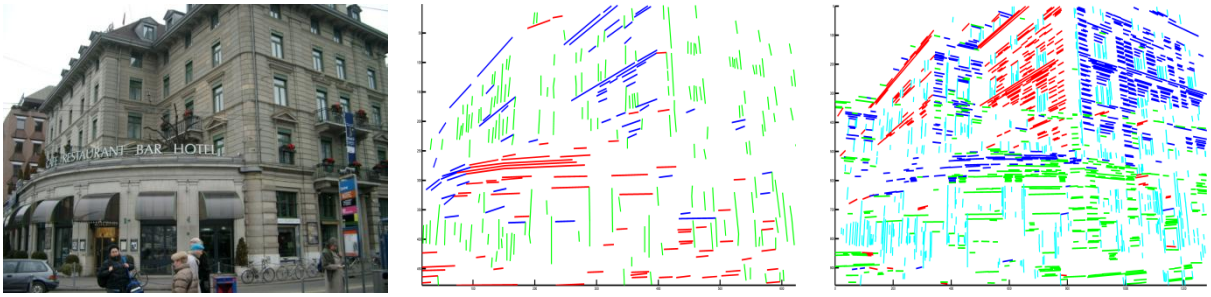
(a) (b)

Figure 6. Average deviation distance from final grouped lines to their estimated vanishing point VS (a) ψ_{min}; (b) σ_0 and σ_d.



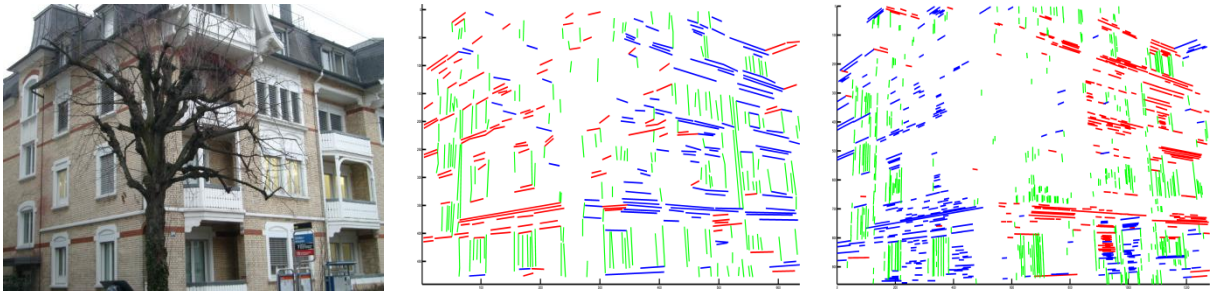
(a) (b) (c)

Figure 7. Line grouping results on a building image with sharp viewpoint. (a) original sample images; (b) line grouping results from previous EM based method; (c) results from our approach.



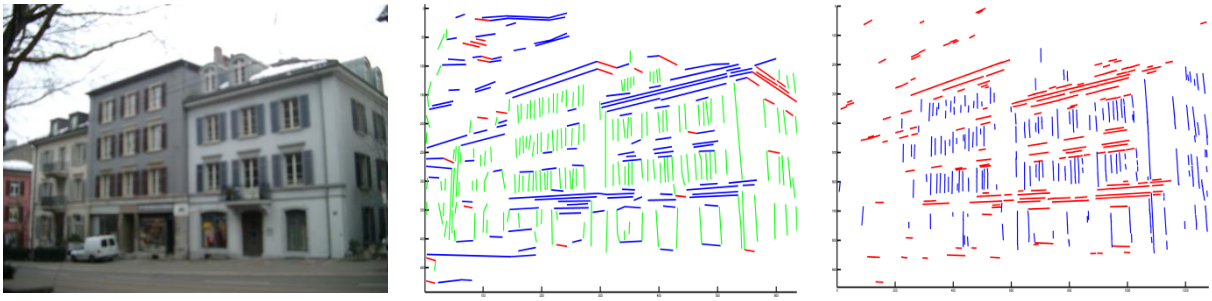
(a) (b) (c)

Figure 8. Line grouping results on buildings with multiple facades. (a) original images; (b) grouping results from previous EM based method; (c) grouping results from proposed approach.



(a) (b) (c)

Figure 9. Line grouping results on buildings with occlusion. (a) original images; (b) grouping results from previous EM based method; (c) grouping results from proposed approach.



(a) (b) (c)

Figure 10. Line grouping results on blurred building image. (a) blurred image sample; (b) line groups computed from previous EM method; (c) line groups computed from proposed approach.

4. CONCLUSIONS

A new approach for building planar surface associated vanishing points detection and line grouping has been presented. Working under uncalibrated camera conditions and with images not conforming to the Manhattan assumption, the experimental results have shown that the method is able to perform well on a wide range of building structures and conditions such as a cluttered environment. It also outperforms existing techniques in working in complex building images. Further improvement to this method would be the introduction of an error model as in [Cou03a]. In the future, we will investigate ways of integrating this method into 3D reconstruction of buildings and building recognition task.

5. REFERENCES

[Bar83a] Barnard, S. Interpreting perspective images. Artificial Intelligence, vol. 21, 1983.

[Bri91a] Brillault-O'Mahony, B. New method for vanishing point detection. Computer Vision, Graphics, and Image Processing, vol. 54(2), pp.289-300, 1991.

[Cou03a] Coughlan, J.M. and Yuille, A.L. Manhattan world: Orientation and outlier detection by Bayesian inference. Neural Computation, 15(5): 1063-1088, 2003.

[Dav08a] David, P. Detection of Building Facades in Urban Environments. Proc. of SPIE conf. on Visual Information Processing XV!!, vol. 6978, pp.9780, 2008.

[Dua09a] Duan, W. and Allinson, N.M. Automatic approach for rectifying building facades from a single uncalibrated image. Proc. of 6th International Conference on Informatics in Control, Automation and Robotics (ICINCO), vol.2, pp.37-43, 2009.

[Kah90a] Kahn, P., Kitchen, L. and Riseman, E.M. A fast line finder for vision-guided robot navigation. IEEE Transactions on PAMI, 12(11), pp.1098-1102, 1990

[Kos02a] Kosecka, J. and Zhang, W. Video compass. ECCV, vol. 2353, pp. 29-32, 2002.

[Kov07a] Kovesi, P.D. MATLAB and Octave functions for computer vision and image processing. School of Computer Science & Software Engineering, The University of Western Australia. Available from: http://www.csse.uwa.edu.au/~pk/research/matlabfns/

[Lut94a] Lutton, E., Maitre, H. and Lopez-Krahe, J. Contribution to the determination of vanishing points using Hough transform. IEEE Transaction on Pattern Analysis and Machine Intelligence, 16(4), pp.430-438, 1994.

[Mag84a] Magee, M.J. and Aggarwal, J.K. Determining vanishing points from perspective images. Computer Vision, Graphics and Image Processing, vol. 26, pp.256-267, 1984.

[Qua89a] Quan, L. and Mohr, R. Determining perspective structures using hierarchical Hough transform. Pattern Recognition Letters, vol. 9, pp.279-286, 1989.

[Rot02a] Rother, C. A new approach to vanishing point detection in architectural environments. Image and Vision Computing, vol.20, pp.647-655, 2002.

[Sha03a] Shao, T.S.H. and Gool, L.V. Zubud-zurich buildings database for image based recognition. *Technical report No. 260*, Swiss Federal Institute of Technology, 2003, http://www.vision.ee.ethz.ch/showroom/zubud/

[Tuy98a] Tuytelaars, T., Gool, L.V., Proesmans, M. and Moons, T. The Cascaded Hough Transform as an Aid in Aerial Image Interpretation. Proceedings of the Sixth International Conference on Computer Vision (ICCV'98), pp.67-72, 1998.

[Wil07a] Wildenauer, H. and Vincze, M. Vanishing point detection in complex man-made worlds. ICIAP, 2007.

A Multi-Cellular Orthographic Projection Approach to Image-Based Rendering

Carlos Saraiva	João Fradinho Oliveira	João Madeiras Pereira
IST/INESC-ID	IST/INESC-ID Rua Alves Redol, 9 1000-029 Lisboa	IST/INESC-ID
carlos.saraiva@ist.utl.pt	joao.oliveira@vimmi.inesc-id.pt	jap@inesc.pt

ABSTRACT

Image-based rendering (IBR) techniques take the approach of rendering new images based on existing ones, effectively separating the rendering complexity from the geometric complexity of the scene they represent. With the increasing complexity of scenes currently created, these techniques have gained renewed interest. This paper presents a multi-cellular IBR approach, which uses orthographic rather than perspective projected pre-computed images. Specifically we extend our single cell orthographically projected IBR approach to a multi-cell system, and show how the pixel error can be further reduced with the use of smaller and more numerous cells.

Keywords

Image-based rendering, orthographic projection, multi-cellular, real-time, impostors

1. INTRODUCTION

Despite the fast advance of hardware capabilities, the complexity of the scenes to render has also grown extremely fast. Even with today's modern hardware, complex scenes cannot be rendered in real-time by brute-force methods.

This introduces the need of simplifying the scene to be rendered, reducing its inherent complexity to levels that can be rendered faster, with as little loss in visual quality as possible.

While other efficient techniques have been described, to address the problem of massive data set viewing, they present some problems: they can be sensitive to the type of scene (some techniques don't deal well with CAD, others don't deal well with complex interiors, etc); some require manual intervention (selecting areas of interest) or have navigational constraints (i.e. the camera can only be placed in certain regions). As such, this paper presents an experiment to develop a method that does not suffer the aforementioned problems.

In our approach, we opted for a class of techniques labeled image-based rendering. This class of techniques strives to render the scene from a certain viewpoint given renders of the scene from other viewpoints. From these other renders, information of the scene can be extracted and reprojected into the new viewpoint. In previous work, Saraiva et al. presented a single cell IBR approach based on perspective projection of pre-computed orthographic projected images [Saraiva09] and have shown that the system created less pixel error than perspective projecting perspective projected images (see Figure 1 and 2). This gives the advantage of dealing well with any type of scene as long as it is renderable.

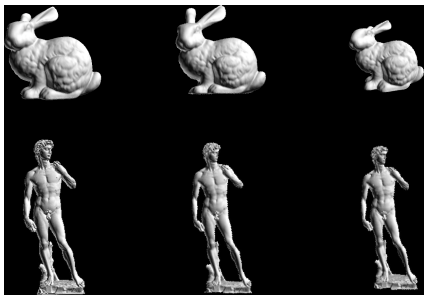


Figure 1. The Bunny and David models. The left shows the original model triangle rendering, the middle is with a texture generated with an orthographic projection and right shows with the texture generated with a perspective projection.

In this article we extend the system to a multi-cell IBR approach and show how the error is further redu-

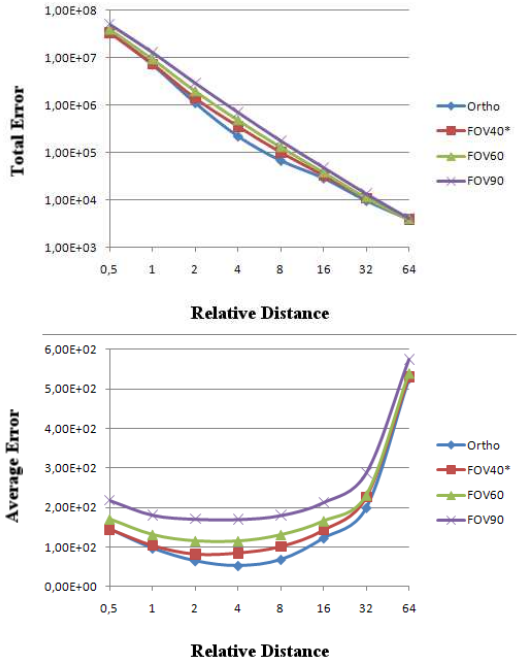


Figure 2. Single cell IBR - Pixel Error: For each increasing relative distance to the model, a series of images were created and subtracted from perspective triangle rendering with a FOV of 40°. Top: the sum of all pixel differences, where p_{dif} is the sum of absolute differences of each R,G,B components between both images. Bottom: the total error divided by the number of foreground bounding rectangle pixels.

ced with more cells using our system. In section 2 we review previous work. In section 3 we present the outline of the multicell system. In section 4 we address implementation issues specific with the reprojection of multi-cell orthographic images. In section 5 we present results. Finally, we draw conclusions in section 6 we conclude and outline future work in section 7.

2. PREVIOUS WORK

In the category of image-based rendering techniques, many methods can be considered, ranging from those that use no geometry at all (like resampling a set of images given the viewing parameters [McMillan95a] to more hybrid approaches that use both geometry and images to represent the underlying scene [Jeschke02].

It is also important to note that image-based rendering can be more useful when dealing with certain types of scenes which are not handled so well with geometric level of detail techniques. An example of such a scenario is the representation of a model constituted by "rough" curves (like a fractal): too much triangle reduction and what is supposed to be a curve can become noticeably linear; too small of a reduction can result in a great number of triangles to draw. Most image-based techniques do not suffer from this problem, as they are independent of the scene.

The techniques can also be classified by how the images are created: some use a predetermined set of possible viewing positions, creating images that represent more distant zones in the scene [Sajadi09]; others create images of objects or subvolumes of the scene and render the scene by representing each of the objects or subvolumes [Schaufler98].

This last category has the advantage that the scene can be represented from any viewpoint and not only a predetermined set. One possible approach useful for viewing models from an outside point of view is to render the model from a series of viewpoints and use those renders as textures placed into billboards, as seen in [Aliaga99].

In [Saraiva09], the camera positions for the generation of the impostors are chosen as the center of the triangles on an *n*-time subdivided icosahedron, enclosing the object (as seen in Figure 3, left), and at a distance (when using perspective) of the bounding sphere radius divided by the sine of half the field of view angle. For orthographic projection no displacement is made.

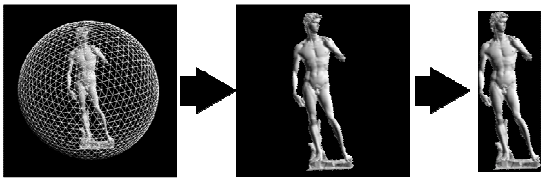


Figure 3. The camera positions; a generated texture; after cropping the texture.

Without correction, this technique can introduce distortion, since the projection of the scene into a plane done during the texture generation can differ from the projection used while viewing the scene with billboards.

This correction is performed by reprojecting an image generated from known viewing conditions into another viewing condition. This reprojection is a function that maps a texel to a number of pixels in the final image. However, several texels can be mapped onto a single pixel, so the texel that has the closest projection is used. Some techniques employ a z-buffer to this end; others are based on analysis of the image [McMillan95b].

Some methods of correction are CPU-based, where each texel in the original image is simply reprojected into the new viewpoint [McMillan97]. While highly parallel in nature, this kind of technique is not trivial

to reproduce in GPU using current graphical APIs, such as OpenGL or DirectX.

In order to exploit the capabilities of modern GPUs, the reverse is usually attempted: finding which source element is the one used in the destination element and sampling it. This problem is not trivial to solve since there is the possibility of the existence of multiple solutions, therefore, a search must be performed for accurate calculation of the source method to use.

These GPU-based methods change the sampled texture coordinate based on a structure that contains the displacement relative to the billboard's normal, typically a texture called a height map (like seen in [Kaneko01]).

Some techniques compensate for this displacement by applying an offset to the texture coordinate based on the view angle relative to the billboard and the height of the billboard at that point [Kaneko01]. While fast, this technique is not accurate and can present incorrect results.

Other techniques strive for absolute correctness, at the cost of performance. Such techniques typically perform a short-distance raycast on the height map, considering points by an increasing order of distance to the camera. If no solutions are skipped due to a large step size, the first solution found is the correct one [Tatarchuk06][Policarpo05]. This raycast process can also be sped up with a pre-computed acceleration structure, as seen in [Baboud06] and [Policarpo07].

3. MULTI-CELL IBR

3.1 Pre-processing

As a first step, the scene is divided into an hierarchy of regular cubic volumes. This can be done by calculating the bounding box of the scene, making it cubic and applying the octree algorithm [Jackins80] to that bounding box, which serves as the root node of the octree. The maximum depth of the octree should be chosen based on the available memory resources, as discussed later.

The second step pre-computes a series of impostor textures for each of these nodes. These impostors are generated using an orthographic projection, since they were found to have the least overall error. The method of generating these impostor textures is similar to the procedure of computing images in the single cell approach; the subdivided icosahedron used for calculating the camera positions is centered in the middle of the octree cell (instead of the centroid of the scene) which is circumscribed by the bounding sphere of the cell.

3.2 Runtime

The octree is traversed and nodes that are not framed by the camera can be culled. If a node that occupies an amount of screen space smaller than a certain threshold is reached, the impostor is used and the traversal stops. At close distances, for full detail, the original model triangle rendering could be used. However, if the choice to not use impostors was only based on distance, that would introduce problems, since then the approach would suffer from high polygonal densities (i.e. areas of the scene that have an extremely high polygon count). Therefore, the nodes that are indeed close enough to be represented geometrically are sorted by priority (distance to the camera) and rendered from the highest priority to the least, until a certain triangle budget is met. All remaining nodes are represented with the impostor with the smallest angle to the view direction.

3.3 Preliminary analysis

By using this approach of creating a representation of the cell as if viewed from the outside as in [Rusinkiewicz00], one gains freedom of navigation, since the camera no longer has to be constrained inside a cell. Also, the algorithm is almost automatic; the only manual intervention needed is setting parameters appropriate to the scene.

Multi-cell IBR solutions, however, are likely to occupy large amounts of disk space. This is accepted due to the relative low cost of disk space in the current days. Also, the algorithm can have an upper limit of disk space usage imposed to it (with a possible loss of performance if the limit is too small for the scene complexity): one can sort the nodes by the amount of primitives they contain. Then, the image generation can be applied to the nodes with the greatest number of primitives, in order, until a disk space budget is reached. It should also be noted that the method should not be used solely with impostors. It uses them to remove most of the scene complexity, but it should be complemented with other techniques (like geometric LODs, for example). However in [Saraiva09] we were somewhat surprised at the visual quality that was possible up close when examining 3D models with the reprojection of orthographic images, hence in this work we did not integrate LOD techniques, and aimed at improving the visual quality of less computationally intensive techniques such as IBR. Indeed in section 5 (results) we show that the combination of smaller IBR cells allows one to achieve this result.

3.4 Summary of steps

In pre-process: 1 – read geometry; 2 – create octree; 3 – calculate camera positions using subdivided icosahedron; 4 – texture creation and cropping.

At runtime: 1 – frustum culling; 2 – node sorting based on distance; 3 – for each node choose image with smallest angle to view direction.

4. MULTI-CELL IBR IMPLEMENTATION ISSUES

In this section we address implementations issues that arise when assembling multiple cell images of orthographic projected images. In section 4.1 we present the problem, in section 4.2 we consider a run-time solution using raycasting, and in section 4.3 our adopted solution.

4.1 Inter cell artifacts

A problem that arose on the multi-cellular approach (that was not present in the single-cell approach of [Saraiva09]) was the presence of "black line" artifacts on the model. An example of this artifact can be seen in Fig 4, which shows vertical and horizontal black lines spanning the Batalha monastery model.

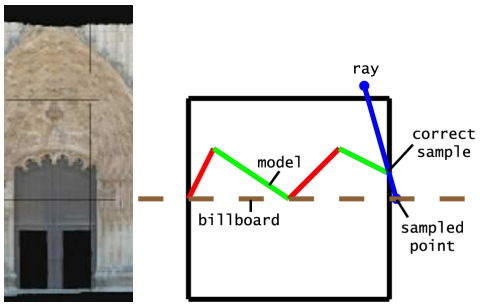


Figure 4. The incorrect sampling problem. The horizontal and vertical black lines artifact visible in the Batalha model (left).

These black lines are actually the edges of the octree node. This is due to the fact that since no correction was performed, the coordinates used to sample the texture were too far away from the center. An illustration of what occurs is depicted in Fig.4 (right).

In the image, the black box represents the region of space delimited by the octree node, and the red and green lines represent the mesh. The brown line represents the billboard plane and the blue line represents a ray from the camera into the volume. If it was done with the original model triangle rendering, the pixel would be colored green. However, since the geometry in the region is effectively flattened by the billboard process, the texture is sampled outside of the region, which is transparent (or non-existent, if it falls outside the billboard), generating the "black line" artifact, where one is actually seeing the background.

This error lessens as the distance to the region increases, since the rays become more and more parallel and, therefore, closer to the orthographic projection that was used to create the image. In practice, one can limit the maximum amount of error in the sampled coordinates by choosing a large enough distance to the billboard.

Attempts were made to correct this distortion. Like previously referred, this sort of correction can be done via CPU (highly undesirable) or the GPU. Attempts to minimize the error with small amounts of computation failed, as all solutions eventually degenerated to something similar to a raycast (presented in the next section).

4.2 The impracticality of raycast solution

Raycasting can be impractical, since it is a somewhat expensive operation, even in modern GPUs when using acceleration structures. On ideal conditions, the performance should depend only on the resolution of the display. However, this proved not to be true, due to the possibility of overdraw and limitations in the current graphics APIs.

Since we have our space divided into regions, without having any sort of occlusion information, we have to render all the regions (at least those that pass the frustum culling operation). In ideal conditions, only one ray would need to be cast per pixel on the screen. Since we desire the first intersection for that pixel, one could think a way to minimize the number of raycasts would be to order the raycasts for that pixel by sub-volume proximity to the camera. Once a ray hit in a region, that pixel would no longer need to have further raycasts performed.

However, the problem lies in how one can determine if that particular pixel has already seen an intersection. Usually to limit this sort of overdraw, two techniques are used: depth-testing and stencil-testing. One could program the GPU to reject fragments with depth greater than the corresponding fragment on the depth buffer. Something similar applies to the stencil tests. This does not solve the problem, though, since these tests are performed *after* the pixel/fragment shader (and raycast) is run, hence the gain would be nullified since the objective is to not do the raycast at all.

To lessen this effect (discarding fragments after an expensive pixel shader is run), graphics card manufacturers have applied a technique called *Early-Z*. This technique does the depth buffer test after the vertex shader is run, but before the pixel shader. Therefore, it lessens the number of fragments that have to go through a possibly expensive pixel shading.

This technique, however, is not part of the typical graphics pipeline, and so is not adjustable via an API. As such, the hardware can only apply this technique if it is sure that the final result is the same. This can only be done if one knows the depth of the fragment before the pixel shader is run. On most applications, this condition is verified; the depth of the fragment is the result of the interpolation of vertex depths.

By performing a raycast, we find out the correct depth of the fragment and can use that correct depth. But by doing so, one effectively is disabling *Early-Z* and increasing the overdraw rate significantly.

Since one needs to know the depth of the fragment in the framebuffer while executing the pixel shader and given that one cannot read and write into the same texture (or a value in a memory array) on a single pass, the use of shaders would not solve the problem. It is possible that more general-purpose APIs (like CUDA) provide a solution, but that goes beyond the purpose of the paper, which is to determine if the approach is viable or if it should be simply discarded.

The problem described, however, has a relatively simple workaround: although one might not know the exact depth of the fragment, one can usually bound that depth between two extreme values. If the Early-Z test fails on both extreme values, any value in between would also fail and the fragment could be discarded. This is not yet possible, however, due to limitations of current graphical APIs such as Direct3D and OpenGL, but is subject to change in the future. Even if we don't change the depth of the fragment (but correct the sampling coordinates), problems can still arise. Without correct depth, interpenetrating objects can display artifacts and techniques that depend on correct depth (like shadow mapping) can become no longer viable. As such, we chose not to correct neither the depth nor the coordinates, to see the viability of the technique with no runtime correction performed at all.

4.3 Cell growing solution

As one can observe, the lines seen due to the artifact are very thin, since the coordinates are only slightly wrong (with an error of one texel, for example). In practice this problem only arises when viewing an object at very close distances.

We found a solution that works well in practice. Instead of using only geometry inside the region to generate the images for the billboard, one can also use the surrounding geometry, which in essence "fills the gap". An illustration of this can be seen in Fig. 6.

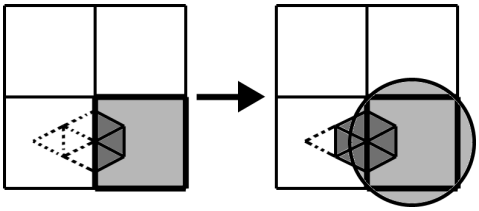


Figure 6. Filling the gap. The thick lines delimit the cell considered. In the left, the geometry is only considered in the area inside the cell (gray), while on the right, the surroundings are also considered.

Although visually incorrect, it tends to be an effective solution since the gap is generally small and color changes are usually smooth enough. To this end, the triangles considered to create the billboards were not only the ones in the cubic volume but the ones that intersect that cubic volume's bounding sphere. In Figure 7 we see a render of the same model after this correction is applied.

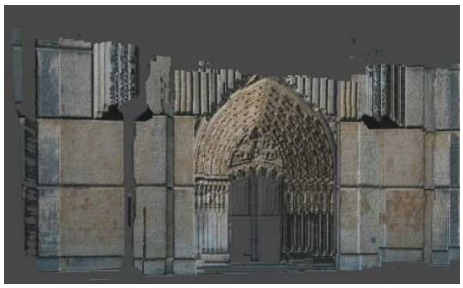


Figure 7. The "black line" artifact invisible, by filling the gap with surrounding geometry.

5. RESULTS

Figure 8 shows that the visual error associated with single cell orthographic projected IBR can be reduced by using more cells.

The test was done with the David model (8 million triangles), using an octree of depth 4, for a total of 218 cells (empty cells are ignored). The positions were chosen with a twice-divided icosahedron, which results in 320 images per cell. At a resolution of 128×128, after cropping, the average size ended up at 11MB per cell (2.41GB total). No compression was implemented, however a fast ZIP compression (using the deflate compression method [Deutsch96], with a 32KB dictionary and a word size of 32) reduced that size to less than a fifth (~18%). A slower and better compression further halved that size. This high compressibility, along with the low cost of disk space on the current days, makes the uncompressed large disk space occupation more acceptable.

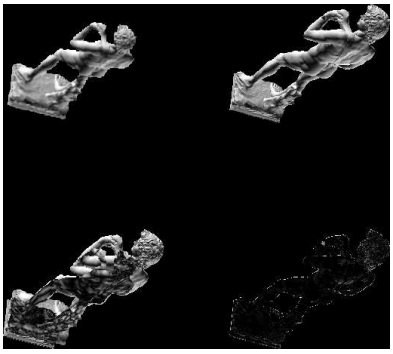


Figure 8. The David model in a single cell (left) and multi cell (right) representation. The top shows the render, the bottom shows the difference to the original model triangle rendering.

To test the effectiveness of this multi-cellular approach in solving the problem seen in the David and Batalha models, due to their large range of depths at certain angles, the camera was placed so that the vertical component was greater (to maximize the error) while still being able to see most of the model, at a relative distance of 1. The result can be seen in Figures 9-13 (cropped to fit the content).

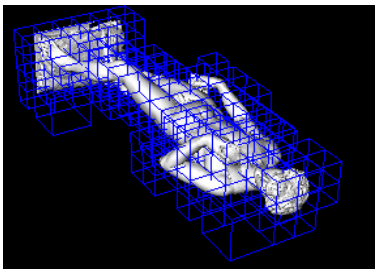


Figure 9. The wireframes of the 218 nodes.

Figure 10. The orthographic impostor approach, from another angle.

Figure 11. From the same angle as figure 10, with the original model triangle rendering.

Figure 12. The difference between figures 10&11.

Figure 13. The outline of the 218 billboards used to generate figure 10.

As one can see, this multi-cellular approach indeed lessened the problem of perspective error. This is due to two facts:

- since there are more cells, each cell occupies a smaller arc in the field of view, bringing it closer to the "original" orthographic projection, the difference with perspective multi-cell IBR can be seen in Figure 14.
- since the cells are smaller, the available gamut of depths is also reduced and so the flattening of the mesh does not change the depth significantly.

In terms of performance, the transfer of texture memory to the graphics card is fast enough to give the viewer 100+ FPS with a C++ implementation using OpenGL on an Intel E7200 with 2GB of main memory and a NVIDIA GeForce 8800GT. To stress the bandwidth limitations between the graphics card and main memory, the graphics card constantly deleted the textures after each render, requiring it to be reuploaded. A problem arises however in disk transfer bandwidth. When the camera rotates around the model too quickly, one texture for each of the 218 cells has to be loaded from disk, bringing performance down to about an average of 5FPS. This problem can be alleviated by caching [Yoon05].

It is hypothesized that this bandwidth problem could be greatly lessened if the textures were compressed and if they were prefetched (like TetraPuzzles [Cignoni04] does). We also note that when rendering triangles over IBR, fewer textures are needed hence further speeding up performance.

In terms of image quality, the lack of correction still brings some visual artifacts, as can be seen in the left wrist area in Figure 10 (with the difference clearly shown in Figure 12). This might not pose a problem, however: as the distance increases, these artifacts get smaller (in number and size), eventually making them unnoticeable. Since the technique should be complemented by some other method for nearby regions, these areas of larger distortion are replaced by correct pixels. An example of the triangle

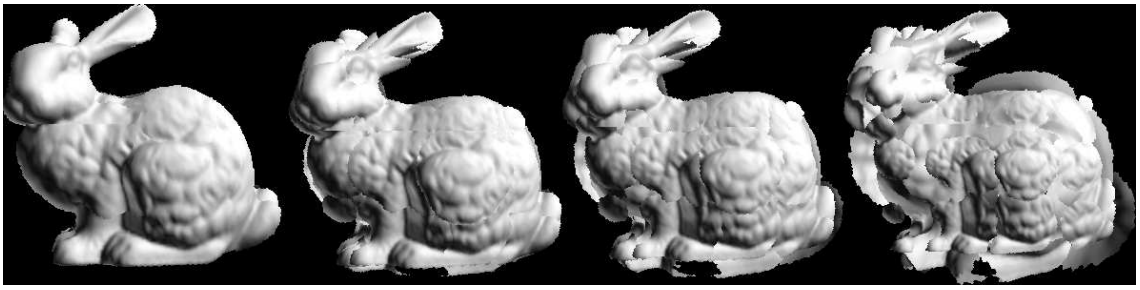


Figure 14. Multi-cell IBR (50cells with 320 images each). From left to right, perspective projection with 60° FOV of: Orthographic projected cells, Perspective projected cells of 40°, 60° and 90° respectively.

representation used at nodes where the center goes below a certain distance threshold to the camera (with impostors used in other cases) can be seen in Fig. 15.

Figure 15. The impostor solution with and without triangles drawn over (right and left, respectively). Notice the artifact on the left wrist disappears.

Figures 16-17 show the David model from another angle, but twice as close as the distance in Figures 9-13.

Figure 16. The David model, from yet another angle, but at a relative distance of 0.5 (impostor on the left and original model triangle rendering on the right).

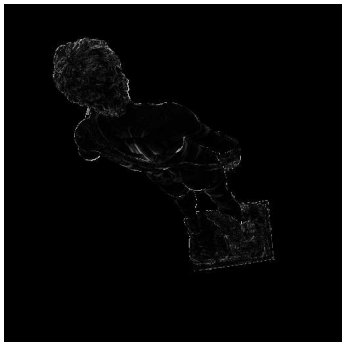


Figure 17. The difference between the two representations in figure 16.

6. CONCLUSIONS

In this paper, an experiment on the usage of simple multiple cell impostors (billboards) was described, in order to determine the feasibility of such a solution for real-time rendering of massive data sets. We have shown that with more cells the visual error is reduced when compared to single cell approaches.

The solution implemented, as described, possesses some potentially great limitations: large use of disk space, bandwidth problems when the camera moves quickly, at close distances visual artifacts can become noticeable. However, being an image-based rendering approach, it also possesses one great quality: the performance of a cell is independent of the number of primitives it contains.

Some of the limitations can be lessened: disk space usage and bandwidth issues can be lessened by the use of compression and caching. Other issues, like visual artifacts at short distances are not so easy to correct, requiring a significant amount of computational power (whether in CPU or GPU). However, if we limit the use of impostors to large distances, the bandwidth and artifact problems are greatly reduced: if an impostor is far away, one would have to move the camera at a very high speed in order to force a large number of texture loads from disk. Also, at great distances the visual artifacts are minimal or non-existing, since the camera "rays" are nearly parallel, like the orthographic projection which gave origin to the image used on the billboard.

We have also concluded that for a multi-cell approach, an orthographic projection produces less error than a perspective projection, as visible in Figure 14.

7. FUTURE WORK

In terms of future work, there are quite a few avenues of possible improvement. In no particular order:

- implementation and analysis of the effects of compression;
- implementation of a caching mechanism;

- researching and implementing some form of correction that is less computationally expensive
- implementing a current form of correction while exploiting future new capabilities given by APIs, like the range-limited Early-Z mentioned;
- integrate a complementary method to the solution described in order to see how well the technique performs in a more ideal situation;
- research some more compact representation of the contents of the cell (like the view-dependent voxel in Far Voxels [Gobbetti08].

8. ACKNOWLEDGMENTS

We would like to thank INESC-ID and its VIMMI research group for the resources made available in the writing of this paper.

We would also like to thank Instituto Português do Património Arquitectónico (IPPAR), now incorporated in Instituto de Gestão do Património Arquitectónico e Arqueológico (IGESPAR) and "Artescan, Tridimensional Digitization" for the scanned model of the Batalha monastery. For the David scanned model, we also thank the Stanford University. The work presented in this paper was funded by the Portuguese Foundation for Science and Technology (FCT, which we also thank), VIZIR project grant (PTDC/EIA/66655/2006).

9. REFERENCES

[Aliaga99] Aliaga, D. Automatically reducing and bounding geometric complexity by using images. PhD thesis, University of North Carolina at Chapel Hill, 1999.

[Baboud06] Baboud, L., Décoret, X. Rendering geometry with relief textures. Proceedings of Graphics Interface 2006, 2006, pages 195-201.

[Cignoni04] Cignoni, P., Ganovelli, F. et al. Adaptive TetraPuzzles – efficient out-of-core construction and visualization of gigantic polygonal models. SIGGRAPH '04: ACM SIGGRAPH 2004 Papers, 2004, 796-803.

[Gobbetti08] Gobbetti, E., Kasik, D. et al. Technical strategies for massive model visualization. SPM '08: Proceedings of the 2008 ACM symposium n Solid and physical modeling, 2008, 405-415.

[Jackins80] Jackins, C., Tanimoto, S. Oct-trees and their use in representing three-dimensional objects. Computer Graphics and Image Processing, 1980, 249-270.

[Jeschke02] Jeschke, S., Wimmer, M., Schuman, H. Layered environment-map impostors for arbitrary scenes. Proc. of Graphic Interface, 2002, 1-8.

[Kaneko01] Kaneko, T. Takahei, T. et al. Detailed Shape Representation with Parallax Mapping. Proceedings of the ICAT 2001, 2001, 205-208.

[McMillan95a] McMillan, L., Bishop, G. Plenoptic Modeling: An Image-Based Rendering System. Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, 1995, 39-46.

[McMillan95b] McMillan, L., Bishop, G. Head-tracked stereoscopic display using image warping. Proceedings of SPIE, 2409 (1995), 21-30.

[McMillan97] McMillan, L. An image-based approach to three-dimensional computer graphics. PhD thesis, Universityof North Carolina at Chapel Hill, 1997.

[Policarpo05] Policarpo, F., Oliveira, M. et. al. Real-time relief mapping on arbitrary polygonal surfaces. Proceedings of the symposium on Interactive 3D graphics and games 2005,155-162.

[Policarpo07] Policarpo, F., Oliveira, M. Relaxed cone stepping for relief mapping. 2007,GPU Gems3.

[Rusinkiewicz00] Rusinkiewicz, S., Levoy, M. Qsplat: a multiresolution point rendering system for large meshes. SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, 2000, 343-352.

[Sajadi09] Sajadi, B., Huang, Y. et al. A Novel Page-Based Data Structure for Interactive Walkthroughs. Proceedings of the symposium on Interactive 3D graphics and games, 2009, 23-29.

[Saraiva09] Saraiva, C., Oliveira, J., Pereira, J., Araújo, B. A Comparison of Orthographic and Perspecive Projections in the Generation of Textures for Billboards. Proceedings of Encontro Português de Computação Gráfica, (17), 2009.

[Schaufler98] Schaufler, G. Image-based representation by layered impostors. Proceedings of the ACM symposium on Virtual reality software and technology, 1998, 99-104.

[Tatarchuk06] Tatarchuk, N. Dynamic Parallax Occlusion Mapping with Approximate Soft Shadows. Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, 2006, 63-69.

[Yoon05] Yoon, S.-E. Interactive visualization and collision detection using dynamic simplification and cache-coherent layouts. PhD thesis, University of North Carolina at Chapel Hill,2005.

Automatic Single Person Composition Analysis

Claudio Cavalcanti, Herman Gomes and Odilon Lima
Departamento de Sistemas e Computação
Universidade Federal de Campina Grande
Campina Grande, PB, Brazil
{csvcc,hmg,oflj}@dsc.ufcg.edu.br

João Carvalho and Luciana Veloso
Departamento de Engenharia Elétrica
Universidade Federal de Campina Grande
Campina Grande, PB, Brazil
{carvalho,veloso}@dee.ufcg.edu.br

ABSTRACT

This paper presents an approach to image analysis based on three photographic composition rules: Rule-of-Thirds, Zoom Rule and Integrity Rule. These rules are commonly used by experienced photographers as an important step for creating attractive photos. The proposed approach assumes there is only one person in the image and considers the use of a face detector to locate the photograph's main subject. The composition analysis computes a set of numerical measures from an input image. These measures are then combined to produce an estimate for the overall composition quality. Experiments involving a subjective evaluation by human observers have demonstrated promising results, given that some correlation has been observed between labelling by expert users and the proposed automatic analysis in up to 85% of a test set of images.

Keywords: Photographic Composition Rules, Image Analysis, Image Classification.

1 INTRODUCTION

Photography is one of the most known and accessible forms of art [Hed03a]. One consequence of the digital era is the drastic increase of the number of amateur photographers, due to the ever decreasing costs of digital equipment, storage, publishing and home printing. However, this phenomenon naturally leads to a need of selecting the better photographs within larger sets.

Automatically identifying attractive or appealing photographs in large image sets is a task that, according to our literature review, is not extensively researched. This is explained by the fact that it is not easy to objectively detect elements that allow distinguishing between attractive and non-attractive photographs. Thus, the goal of this work is defined towards the implementation of rules to assess the quality of a photograph based on composition.

Although basic photographic techniques and principles have not changed much over the years, knowledge of these techniques is not widespread. Photographic composition rules can be seen as heuristics used by experienced photographers to emphasize the subject of a photograph [Gri90a]. The subject is "what" or "who" the photographer wants to show to the viewer. Previous work show that, when evaluating the quality or appeal of a photograph, people usually consider composition as the most determinant feature [Sav00a]. Theoretical support has been provided for some composition rules. The best known is the Gestalt's Theory [Kof55a], that explains how the human brain interprets the visual patterns perceived by the eyes. Those patterns influence perception and some may be used (in the case of Photography) to draw the viewer's attention to the subject.

The goal of this work is to analyze images in order to assess their overall quality in an objective way, according to a set of pre-defined photographic composition rules. Three important rules are considered: Rule-of-Thirds, Zoom and Integrity. Assuming that we are dealing with images of people only, the relevant information needed can be derived from the head and eyes positions, located using existing detectors. Moreover, within the scope of this work, we only consider images with a single person as subject. Extensions of the proposed approach to deal with group photographs is left as future work.

A heterogeneous set of images was crawled from the World Wide Web to be used in the experiments, so that the rules could be tested with a large variety of faces, poses, backgrounds and photographers. Those images were labeled by volunteers as either attractive or non-attractive according to their overall feeling about the photograph. This set has been used in one of the two experimental evaluations presented in the paper. A second evaluation involved analyzing image collections belonging to specific users, experts in photography, and obtaining their opinion about the automatic classification results.

Results show that a global measure based on photographic composition and obtained using the proposed approach is aligned with human labeling of attractive photographs in up to 85% of the images used for experimentation, selected from the user's own set of images.

WSCG 2010 conference proceedings,
WSCG'2010, February 1 – February 4, 2010
Plzen, Czech Republic.

Next we describe some assumptions adopted in this work. First, when we refer to the quality of a photograph, this is not related to its low-level features such as compression artifacts, resolution, among others. Moreover, the images used in the experiments have been chosen so that they did not exhibit such low level quality problems. The scope of this work is centred on the analysis of the visual aspect of a photograph. The term attractiveness is related to the way one may find a photograph more appealing than others. Finally, the quality of the composition is related to the conformity to photographic composition rules, not necessarily to the more abstract concept of an attractive photograph.

This paper is divided in the following sections. Section 2 reviews some previous work related to automating photographic composition analysis. Section 3 describes the three rules that are part of the approach. Section 4 discusses some processes for obtaining image classification thresholds. Section 5 presents the experiments performed to validate the proposed approach and their corresponding results. Finally, Section 6 brings final considerations and proposals for future work.

2 RELATED WORK

Banerjee and Evans [Ban04a] have developed a system for automating some aspects of photography. Working on a static image, the system implements a transformation that shifts the subject to a place that agrees with the Rule-of-Thirds. They also considered embedding their method into the camera, and defined automatic methods for background blurring and merger mitigation [Ban07b].

The Robot Photographer by Byers et al. [Bye03a] is one interesting example of autonomous photography. This robot can detect subjects in a large indoor area, where there are people. The composition rules used are the Rule-of-Thirds, the empty space (requiring that faces must occupy at least a third of the image), the no-middle rule (not allowing a person at the image center) and the edges rule (subject should not cross the photograph edges). After performing composition, the photograph is taken and stored by the robot until it is transferred to a computer. This approach can be applied to standing people only. According to the authors, the acceptance rate of the photographs autonomously taken in an experiment was around 35% for a set of human viewers.

Zhang et al. [Zha05a] present an approach for *auto cropping*, with the main goal of correcting failures detected in an image. The system works with three evaluation models: the composition model, the penalty model and the conservative model. The subject is located using face detectors. It proposes and implements 14 models of composition, to which each image is compared. A voting scheme was used to analyze the final corrected images, which gives the participants 3 options: good, acceptable and bad. A set of 100 images is considered in the work. Results show that the rate of images considered better than the original is 41%. The work proposed by Zhang et al. differs from our approach in the sense that they aim to enhance existing photographs by means of image cropping whereas our approach is designed to perform image analysis or classification. Moreover, their image evaluation was performed differently in the sense it gave three options to human viewers, what might have polarized the votes to the middle option.

A recent effort by Santella et al. [San06a] also emphasizes the idea of image cropping, using the viewer gaze movement to decide what are the image main points of interest, with no need to maintain image ratio. This approach is compared with a method that uses the points obtained using visual attention mechanisms. According to a voting experiment, the acceptance rate is 58% greater when compared to the traditional method of visual attention.

Datta et al. [Dat06a] present a computational approach to analyze the aesthetics of an image, using 15 metrics, from 56 initially considered. When using SVM (Support Vector Machines) [Vap99a] to classify an image by those metrics, a rate of 70.12% of agreement with a subjective evaluation is reached. The approach of Datta et al. [Dat06a] is similar to the one proposed in this paper, since, in both cases, metrics are used to evaluate an image. However, Datta's approach applies the composition rules by analyzing image pixels, whereas our approach uses higher-level information (face position), which is more effective to images containing people.

Ke et al. [Ke06a] propose a two-class photo classification. Photographs are classified as professional and snapshot based on some semantic features, such as the spatial distribution of edges, color distribution, hue count, blur, contrast and brightness. The aim is classifying images according to its quality in low quality or high quality.

In another work, Song et al. [Son04a] propose a face appeal model based on the size and position of each face in the image. The face appeal was represented using a Gaussian Mixture Model, where each Gaussian in the mixture corresponds roughly to one type of photographic style. In the approach proposed by Song et al., the rules are extracted from a large dataset which corresponds to the most of photographer's style.

A negative aspect of the two latter approaches is that they consider the spatial distribution of the ground truth data. This might force the images look the same way and allow common mistakes being considered as a desired behavior. They may also require a meticulous construction of the data set for dealing with social and cultural issues.

3 COMPOSITION RULES

Three rules were used in this work to analyze composition of a photograph: Rule-of-Thirds, Zoom Rule and Integrity Rule. The first analyzes how close the center-of-interest, i.e. the subject of the photograph, is to any of the "points-of-thirds". The second analyzes how close to the subject the camera is. Finally, the last rule verifies if the subject's head is preserved in the photo.

Those rules are commonly used by most photographers, both amateur and professional. There are many other possible rules, but we have chosen the three which are considered the most relevant by the specialized photographic literature [Hur04a, Hed05b]. We believe that the addition of more rules into our system will progressively have a positive impact on the quality of the results, since each rule analyzes a photograph according to different aspects.

The three chosen rules require detection of the subject position. We used the Rowley-Baluja-Kanade's face detector [Row98a, Row98b] to obtain the face coordinates of the subject, from which the approximate body position can be determined (assuming the subject is not in any unusual pose).

To obtain a more precise evaluation of the Rule-of-Thirds, we also used an eye detector, as eyes are the center-of-interest of most portraits. The eyes coordinates are detected using the approach proposed by Leite et al. [Lei07a], which achieved an accuracy of 99.63% using the Caltech image database [Weba]. The employed eye detection method outperformed other well known approaches, such as the Machine Perception Toolbox [Fas05a] and the Rowley-Baluja-Kanade eye detector [Row98a].

3.1 Rule-Of-Thirds

The Rule-of-Thirds is a well known rule that is used to verify how close to the "lines-of-thirds" (or their intersection, the points-of-thirds) is to subject. The lines of thirds are two lines that divide the image in three equal parts, both horizontally and vertically. Thus, the y coordinates of the two horizontal lines of thirds are $\frac{w}{3}$ and $\frac{2\times w}{3}$, whereas the x coordinates of the two vertical lines of thirds are $\frac{h}{3}$ and $\frac{2\times h}{3}$, where w and h are the image width and height, respectively.

Photographers apply the Rule-of-Thirds to avoid centralization of the subject, which may result in a too static photograph. The term static photograph means that it does not possess features that stimulate the viewer to search the image for regions of interest, thus resulting in an non-attractive image.

In portraits, eyes are the most adopted center-of-interest for human subjects. It is accepted that the ideal positioning of the subject center-of-interest is on any of the points of thirds. Figure 1 shows a portrait where the eyes are correctly positioned, according with the Rule-of-Thirds.

Figure 1: Rule-of-Thirds: One eye is on one of the four points of thirds, while the other is on a line-of-third.

In order to verify the Rule-of-Thirds, eight distances are calculated, forming the set $\Im$, which corresponds to the Euclidean distances of each subject eye to each of the four points-of-thirds. The smallest of those eight distances (s) is then selected as shown in Equation 1.

$$s = d \in \Im \wedge \nexists c \in \Im \mid c < d \tag{1}$$

The maximum distance (m) from any image point to the nearest point of thirds is given by Equation 2. This value corresponds to the distance from any image corner to its nearest point of thirds. In Equation 2, w is the width of the image and h is its height.

$$m = \frac{\sqrt{w^2 + h^2}}{3} \tag{2}$$

A normalized distance (called n) is defined in Equation 3, assuming values within the interval [-1..1].

$$n = 1 - (2 \times \frac{s}{m}) \tag{3}$$

If $n = 1$ in Equation 3, then a perfect match between the center-of-interest and one of the points-of-third is achieved, while $n = -1$ indicates a poor subject positioning.

3.2 Zoom Rule

The Zoom Rule analyses the relative face size within an image, with regards to the total image size. Once face position and dimensions are known, the face to image ratio can be computed.

We use Equation 4 to evaluate the parameter f which expresses the face height (k) to image height (h) ratio:

$$f = -1 + (2 \times \frac{k}{h}) \tag{4}$$

Although the face detector used in this approach returns equal values for face height and width, face height should be used for other face detectors since heads are taller than wider. Other approaches are also possible, like the ratio between face area and image area, as well as the face to image widths ratio.

The proportions between head height and head width can be obtained from anthropometric measures. Although not every measure is formally proved (since

they only represent the proportions of part of the population), they are widely used by industrial designers, artists (such as painters) and anatomists to estimate the size of human body parts from known proportions. In this work, some anthropometric measures have been adopted, e.g. the ratio between the human head height and width is approximately 1.5. Once the eyes have been located, the distance between them allows obtaining the head width and, consequently, the head height.

The Zoom Rule, evaluated using Equation 4 should produce values within the interval -1 to 1. However, too much zoom may cause the face coordinates to exceed image borders, resulting in values greater than 1. In those cases the value of f is clipped to 1.

It is not easy to define adequate values of zoom. For many situations, a face close-up photograph is desirable and produces a result as good as a full body shot. Therefore, there is not, roughly speaking, an inadequate zoom value. In this work, we consider that if the subject is too far from the camera (consequently small in the photograph) he or she becomes part of the background. Excessive zoom may also be harmful to composition as it restricts the photograph to the subject face expression only. It also may show face imperfections, which is not desirable in most cases. Thus, the interval used -1 to 1 is not proportional to zoom quality, being just a measure.

3.3 Integrity Rule

The Integrity Rule is used to detect if body parts of the subject have been chopped out of the photographs. Although this chopping may be tolerated to some degree, it is usually desirable that parts like face, eyes and sometimes the arms and hands be preserved, otherwise the composition of the photograph cannot be considered good.

In the present work, we opted for analyzing the integrity of the head, which is essential in any photograph containing people. Head width (o) is evaluated by Equation 5, where e is the distance between the points that define the center-of-eyes.

$$o = 1.2 \times e \tag{5}$$

Similarly, head height (p) is defined by Equation 6:

$$p = 1.5 \times o \tag{6}$$

The concept of integrity has been expressed as the ratio between the chopped out head area (t) and the total head area (u). The parameter t is evaluated by Equation 9, using the results of Equations 7 and 8. Two types of integrity violation are considered, one that chops out the top of the head (expressed by q, Equation 7) and the other that chops out the side of the head (expressed by r, Equation 8). The coordinates (x_1, y_1) and (x_2, y_2), represent the upper left and lower right vertices of the head bounding rectangle, respectively. This evaluation is possible, since most of face detectors return coordinates out of the bounds of the image (i.e. negative values or coordinates greater than image width or height) when faces are chopped out.

$$q = \begin{cases} l \times |y_1|, \\ \quad \text{if } y_1 < 0; \\ o \times (y_2 - h), \\ \quad \text{if } y_2 > h; \\ 0, \quad \text{otherwise.} \end{cases} \tag{7}$$

$$r = \begin{cases} n \times |x_1|, \\ \quad \text{if } x_1 < 0; \\ p \times (x_2 - w), \\ \quad \text{if } x_2 > w; \\ 0, \quad \text{otherwise.} \end{cases} \tag{8}$$

$$t = q + r \tag{9}$$

u, the total area of the head, is defined by Equation 10.

$$u = o \times p \tag{10}$$

Finally, Equation 11 shows how integrity is evaluated. The face detector used in this work [Row98a, Row98b] does not define a maximum allowed occlusion level, however, experiments show that up to 50% of face occlusion can be tolerated in extreme situations. That is why we consider only 50% of full head area (u) in Equation 11, obtaining a value between -1 and 1 for integrity. It is possible, although not common, that the face detector correctly detects a face that is more than 50% occluded. Therefore, the maximum allowed integrity value (i) is clipped to 1, to avoid values outside the defined range.

$$i = 1 - (2 \times \frac{t}{0.5 \times u}) \tag{11}$$

Figure 2 illustrates a situation where both the top and one side of the head have been chopped out. In this case, integrity violation results from y_1 being smaller than 0 and x_2 greater than the width of the image. In the latter situation (and in other similar cases) the value of head width o must be added to the value of the chopped areas intersection t, which is subtracted twice at the equations defined above. For simplicity, this sum will be omitted.

4 THRESHOLD DETERMINATION

In our literature review, we could not find an objective method to evaluate the overall quality of photographs. This kind of assessment is usually done subjectively.

Since there are several different ways to acquire and analyze a photograph, the problem scope needs to be defined. In this work we assume photographs that have

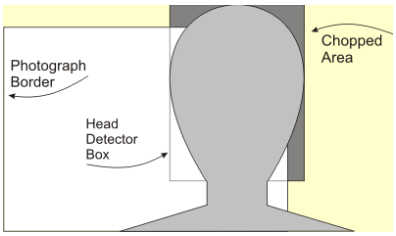


Figure 2: Integrity violation : top and side of the head chopped out.

one person only, whose face and eyes have been correctly detected by algorithms of face and eyes detection. The person is the photo's main subject. An incorrect detection happens when the face bounding box is greatly wider or narrower than the face or if any of the eyes is incorrectly detected. We did not consider paintings, drawings or post processed photographs in our experiments. Finally, only photographs with the subject at an upright position posing to the camera have been allowed in the image sets.

4.1 Building the Image Set

To the best of our knowledge, there is no publicly available image database labeled considering photographic composition rules. Therefore, in order to conduct our experiments an image set had to be acquired and labeled.

Firstly, images were downloaded from the World Wide Web, using a web crawler configured to download only photographs containing people. Several web sites of free photograph publishing were used as source of professional and consumer photographs, such as Flickr and DPChallenge.com. However, since part of the constructed image set is under copyright protection, it was not possible to provide a permanent link to it.

After downloading more than 2000 photographs, a face detector was used to filter those with only one face. Since the face detector sometimes fails (false detection, rejection or imprecision in location), the filtered images had to be manually examined and selected. The goal was to decouple the performance of the whole system from that of the face detector. Photographs which presented false detections or rejections were both discarded. The images that did not meet the scope requirements have also been discarded.

The decision to work with single person only has the benefit of considerably simplifying the problem, given the composition rules we are using are not well defined for groups in the photography literature. On the other hand, this imposes a restriction to the image set. We are currently developing additional rules for dealing with images containing more than one person.

In total, 417 images were used in the labeling process. Two photography experts participated in the labeling processes. A total of 65 out of 417 images were classified as attractive and the remaining 352 were considered non-attractive. The resulting image sets were carefully analyzed by a third expert.

4.2 Obtaining Thresholds

After the labeling process, two methods were used for obtaining classification thresholds: the first method is an automatic approach, whereas the second method needs user human interaction.

First Method This method uses a Neural Network to learn thresholds so that the test set images could be correctly classified as good or bad. A MLP Neural Network with the standard Back-Propagation algorithm was used to this goal.

A set of 126 images (half labeled as good and half labeled as bad) was used for the training The number of neurons in the hidden layer and the number of cycles has been systematically varied. For each set of parameters, the network has been trained 10 times (with different weight initializations) - the best network was then chosen from this set. The average correct classification rate for this best network was 95% with standard deviation of 4.11. This network had 64 hidden neurons, GL5 [1], 10,000 cycles as stop criteria and 0.001 as the learning step.

Second Method This approach consisted of classifying the photographs using the values obtained from the composition rules, according to all possible combinations of threshold values. All thresholds were varied from -1 to 1 in steps of 0.5. The step of 0.5 was chosen for minimizing the time needed for each user to evaluate an image set. In total, there are 250 possible combinations, when varying thresholds for all three composition rules and considering that the Zoom Rule requires two thresholds (lower and upper bounds). All 250 sets of images were analyzed, in order to choose the best one. The strategy chosen to ease this task was to analyze groups of ten sets of images at a time, choose the best group and finally select the best set from that group.

The resulting sets were analyzed to decide which of them contained the best images. The set chosen as the best indicates the thresholds that are most adequate to a specific user. The threshold values used were too restrictive in most cases, resulting in 191 sets with all images labeled as non-attractive, 12 groups where all images were classified as attractive, other 4 sets are extremly unbalanced. The first 207 sets are discarded resulting in 43 sets. Further analysis indicated the best image set and, consequently, the best set of thresholds. Some specific results are discussed in Section 5.2. In this setup, one set of thresholds was eventually determined as the best, but a possible approach is to select a

[1] Criterion to stop training when the objective function for a given set of weights produces error greater than 5% comparing to the best set of weights obtained so far.

small amount of acceptable sets from which the thresholds can be extracted. Table 1 shows the thresholds obtained for this method and the previous one. An image is labeled as good when obtained values for each rule are greater or equal the indicated threshold.

Method	Thirds	Zoom	Integrity
1 - Neural Network	0.70	-0.73	0.50
2 - Human Evaluation	0.00	-0.50	0.00

Table 1: Thresholds obtained.

4.3 Threshold analysis

The defined thresholds were analyzed by a group of volunteers to verify if the image set classified with those thresholds are representative. The first experiment presented 126 photographs to the participants without identifying which ones were classified as good composition aspect according to the defined threshold. Sixty people contributed to this experiment. The participants agreed with 60% of the image set labels. The labels were obtained by combining both methods using a logical AND.

This experiment considered an image set for which the contributors were not necessarily familiarized with. This fact might interfere with the capability of performing good judgment of some viewers. For instance, one may find a particular face, place or circumstance non-attractive and unconsciously vote the composition based on this feeling.

5 EXPERIMENTS AND RESULTS

The experiments described in this section were performed to analyze composition considering an image set known by the human participant. The goal is to validate both thresholds obtained in the previous section and evaluate the overall performance of the proposed approach.

5.1 Evaluating on Specific Image Sets

In order to evaluate the composition analysis in a different way and minimize the problem described in Section 4.3, some restrictions were imposed: (i) each contributor analyzed their own photographs, (ii) photographs are of a single event, (iii) photographs must contain exactly one person, and (iv) the face detector performed correctly.

By using only participants own photographs, it is possible to minimize the influence of subjective factors such as photogeny of unknown people, although a professional photographer is capable of well distinguishing among those (and many other) concepts. Another consequence of using experts photographs is the reduced number of non-attractive photographs. That allows our system to analyze, in a group of good photographs, which are the best ones.

By using photographs of a single event - a vacation, one weekend, or any well-defined period of time - photographs will look similar, thus minimizing the difficulties to analyze photographs from very different scenarios.

By using photographs containing only one person, a more acurate analysis is possible, once composition rules are not well-defined for group photography. In order to avoid an inaccurate analysis, images whose faces were not correctly detected by face detector detector did not performed correctly were also discarded.

Finally, by using only photographs without false face detections, the performance of the proposed analysis algorithm is decoupled from the face detection problem.

Although all those restrictions dramatically reduce the number of photographs to be analyzed, it better reflects the actual process of selecting photographs, where a photographer obtains dozens of photographs and only a few are analyzed and considered to be developed.

This experiment considered 55 images. Four observers, who declared themselves as specialists in photography, contributed with this experiment. The reduced number of participants is due to lack of experts in photography available for the experiment (since it is a time demanding experiment and those professionals are usually unable to stop their work for much time).

Photographs are then analyzed and automatically classified by our algorithm. After this analysis is performed, photographs are presented in a web form. A color scheme has been used to help users identify the automatic classification. In this scheme, images whose composition aspect was considered as good or bad are bounded by a green or a red rectangle respectively. It was not informed to the participants what was the automatic selection criteria. Images are also sorted with respect to the global measure. This scheme is illustrated in Figure 3.

Figure 3: Voting scheme : Images that user would develop should be in green (represented by dark grey) while photos which user has no intention of developing must be in red (represented in light grey).

Each contributor is then invited to change the status of the photograph in the presented form until it correctly reflects their opinion about the following ques-

tion: “Which photographs should be developed and which should not”. In our point of view, an image labeled as “to be developed” was considered attractive once it was selected among others, while the remaining are considered “less attractive” but not necessarily “non-attractive” nor “bad” images.

When a observer clicks in a photograph, the bounding rectangle switches the color between green and red, representing user’s opinion about images attractiviness. There is no time requirement for the participant conclude their selection. At the end of the selection process, for each disagreement between automatic and manual classification (either good being classified as bad or bad being classificed as good), it was also asked to the participant the reason for their decision. The possible options were “Facial Expression”, “Brightness, Color of image”, “Photographic Composition” and “Beauty of the person”.

The result analysis has been done in two steps. The first step considered the images labeled as attractive. We verified that the users agree with the decision of the algorithm in 85% of the images. This is a promising result. Moreover, 75% of the 15% disagreement set were rejected by other factors not related to composition. The second step analyzed the images labeled as non-attractive. The participants agreed with 75% of these images.

5.2 Results Discussion

Despite some misclassified images, the obtained results look very promising. In the following paragraphs we analyze some photographs processed by our approach to substantiate this conclusion. Four images are shown in Figure 4. They were all classified as a good, according to the composition rules. In addition, they were classified as attractive by their owners. In another example, shown in Figure 5, images were considered bad according to their composition values. However, they were considered attractive by their owners, resulting in a disagreement.

Table 2 shows the values obtained for each of the example images (in Figure 4) for all photographic composition rules. For those images, there was agreement between manual and automatic classifications. Table 3 shows the values of the photographic composition rules obtained for images (in Figure 5) which presented divergence between manual and automatic classifications. In our experiment, the image was considered good by the algorithm when values are lower than the thresholds of any of the methods.

It is possible to verify in Table 2, that in all images Rule-of-Thirds and Integrity Rule were obeyed in both methods as can be seen in Table 1. Although there was a great variation in zoom values, they are still above the thresholds shown in Table 1.

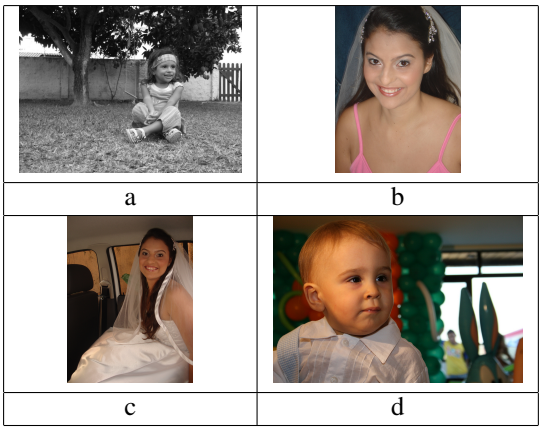


Figure 4: In these images all three composition rules were obeyed and they were classified as good according to the photographic composition aspect. They were also considered attractive by owners.

Figure 5: In these images, the analysis produced poor scores for the Rule-of-Thirds but good scores to other two rules, nevertheless, the photo was classified as bad according to the overall photographic composition aspect.

Image	Thirds	Zoom	Integrity
Figure 4a	0.86	-0.44	1.00
Figure 4b	0.90	0.00	1.00
Figure 4c	0.89	-0.42	1.00
Figure 4d	0.82	0.22	1.00

Table 2: The output values for each rule for images correctly classified. Images were considered good by both participants and automatic classification.

In the Table 3, the composition values were insuficient to result in an automatic labeling as good according to composition. However, the participants considered all images as attractive images. When asked them about their reasons, they reported factors different from composition. We consider this an expected result once an important composition rule is to allow “breaking the rules” in some occasions. If performed consciently, breaking the rules might improve the attractiviness of a photograph.

Although it was not possible to make the image set publicly available due to copyright reasons, we believe the experiments can be reproduced using other image sets and human subjects to label the images. Assuming these subjects have some knowledge in photography, the labeling process can be straightforward.

Image	Thirds	Zoom	Integrity
Figure 5a	-0.32	-0.32	1.00
Figure 5b	-0.19	-0.81	1.00
Figure 5c	-0.40	-0.55	1.00
Figure 5d	0.18	0.00	1.00

Table 3: The output values for each rule for images which there was divergence between automatic and manual labelings. Images were considered good by participantes whereas they were considered bad by automatic classification.

This work shows that there is a correlation between the proposed composition analysis and the observer's opinion with regards to the attractiveness of a photograph. That corroborates with the work developed by Savakis et al. [Sav00a] in which composition was considered by participants of a ground truth study the most important feature to be analyzed in a photograph. Our analysis shows that in a set containing mostly good photographs, the considered best was the one which the composition rules were correctly applied. It does not mean that a photograph with good composition aspect is attractive in a broad sense nor that images with bad composition aspect are unattractive, but if one has two photographs both attractive, the more attractive usually has a good composition aspect.

6 FINAL CONSIDERATIONS AND FUTURE WORK

After executing the described algorithms in 417 images, it is possible to state that composition rules play an important role in image quality. Compliance to those rules, based on the image database used, resulted in a significant number of photographs being correctly classified as attractive. It is also possible to say that not obeying to some or all rules does not necessarily result in non-attractive photographs. Moreover, in a final experiment, it was possible to verify that in up to 85% of the photographs the choice of the algorithm is aligned with the human preference. We believe this rate may be increased if other image analysis measures are included.

The next steps are to refine and extend the composition rules presented in this paper, e.g. define rules for more than one person, and add new rules to improve the quality of the analysis. It is also desirable to expand the image set and its labels. Other low-level image feature analysis, i.e. brightness, edges, etc., are also under investigation for producing a more precise classification.

ACKNOWLEDGEMENTS

This result was achieved in cooperation with Hewlett-Packard Brasil Ltda. using incentives of Brazilian Informatics Law (Law nº 8.2.48 of 1991). This work was also supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq - project 141329/2009-2).

REFERENCES

[Bye03a] Byers, Z., Dixon, M., Goodier, K., Grimm, C.M. and Smart, W.D. An autonomous robot photographer. In Conf.proc. of IROS, Location, pp. 2636-2641, v. 3, 2003.

[Ban04a] Banerjee, S. and Evans, B. Unsupervised Automation of Photographic Composition Rules in Digital Still Cameras. In Conf.proc. of IS&T SPIE Sensors, Color, Cameras, and Systems for Digital Photography, pp. 364-373, v. 5301, 2004.

[Ban07b] Banerjee, S. and Evans, B. In-Camera Automation of Photographic Composition Rules. IEEE Trans. on Image Processing, pp. 1807-1820, n. 7, v. 16, 2007.

[Dat06a] Datta, R., Joshi, D., Li, J. and Wang, J. Studying Aesthetics in Photographic Images Using a Computational Approach. Conf.Proc. of the ECCV, pp. 288-301, v. 3953, 2006.

[Fas05a] Fasel, I., Fortenberry, B., and Movellan, J.R. A Generative Framework for Real Time Object Detection and Classification. Journal of Computer Vision and Image Understanding, pp. 182-210, v. 98, 2005.

[Gri90a] Grill, T. and Scanlon, T. Photographic Composition. Amphoto Books. Watson-Guptill Publications, New York, 1990.

[Hed03a] Hedgecoe, J. The new manual of photography. D.K., London, 2003.

[Hed05b] Hedgecoe, J. The Book of Photography. D.K., London, 2005.

[Hur04a] Hurter, B. The Portrait Photographer's Guide to Posing. Amherst Media, New York, 2004.

[Kof55a] Koffka, K. Principles of Gestalt Psychology. Routledge and Kegan Paul Ltd., London, 1955.

[Ke06a] Ke, Y., Tang, X. and Jing, F. The Design of High-Level Features for Photo Quality Assessment. Journal of CVPR, pp. 419-426, v. 1, 2006.

[Lei07a] Leite, B., Pereira, E., Gomes, H., Veloso, L., Santos, C. and Carvalho, J. A Learning-based Eye Detector Coupled with Eye Candidate Filtering and PCA Features. In Conf.Proc. of XX SIBGRAPI, pp. 187-194, 2007.

[Row98a] Rowley, H. A., Baluja, S. and Kanade, T. Neural Network-Based Face Detection. IEEE Trans. on Pattern Analysis and Machine Intelligence, pp. 23-28, v. 20, 1998.

[Row98b] Rowley, H. A., Baluja, S. and Kanade, T. Rotation Invariant Neural Network-Based Face Detection. Journal on Computer Vision and Pattern Recognition, pp. 38-44, v. 20, 1998.

[San06a] Santella, A., Agrawala, M., DeCarlo, D. Salesin, D. and Cohen, M. Gaze Based Interaction for Semi Automatic Photo Cropping. In Conf.proc. of the SIGCHI Human Factors in computing systems, pp. 771-780, v. 5301, 2006.

[Sav00a] Savakis, A. E., Etz, S. P. and Loui, A. Evaluation of image appeal in consumer photography. In Conf.proc. of SPIE Human Vision and Electronic Imaging V, pp. 111-120, v. 3959, 2000.

[Son04a] Song, B., Li, M., Li, Z., Zhang, H., Liu, Z. Face Appeal Model Based on Statistics. , pp. 344-351, v. 3331, 2004.

[Vap99a] Vapnik, V. The Nature of Statistical Learning Theory, 2nd Ed.. Springer, , 1999.

[Weba] Weber, M. Caltech Frontal Image Database. http://www.vision.caltech.edu/html-files/archive.html, 1999.

[Zha05a] Zhang, M., Zhang, L., Sun, Y., Feng, L. and Ma, W. Auto cropping for digital photographs. In Conf.proc. IEEE ICME, pp. 4-8, 2005.

Geo-Spatial Hypermedia based on Augmented Reality

Sung-Soo Kim[†] Kyong-Ho Kim[†‡] Sie-Kyung Jang[§] Jin-Mook Lim[*] Kwang-Yun Wohn[‡]

[†]Electronics and Telecommunications Research Institute (ETRI)

[‡]Korea Advanced Institute of Science and Technology (KAIST)

[§]Sungkyunkwan University, [*]Samsung Electronics

South Korea

{sungsoo, kkh}@etri.re.kr skjang@gmail.com jinmook.lim@samsung.com wohn@kaist.ac.kr

ABSTRACT

In order to produce convincing geographic information, it is not only necessary to provide real-world videos according to the user's geographic locations but also non-spatial data in the videos. We present a novel approach for efficient linking heterogeneous data of the same objective nature, such as 2D maps, 3D virtual environments and videos with GPS data. Our approach is motivated by the observation that we can get the non-spatial data in a video by transforming the video search space into the 3D virtual world search space which contain the non-spatial attributes according to the remotely sensed GPS data. Our proposed system consists of two primary components: a client layer that implements the augmentation algorithm for geo-spatial video in exploiting server resources, and a server layer designed to provide the non-spatial attributes according to the client's requests. To implement attribute query in a video, we present an easily implementable data model that serves well as a foundation for point query in 2D. In order to apply this to telematics applications such as car navigation systems, we propose a live-video processing method using augmented reality technology according to the user's locations along navigation path. Experimental results demonstrate that our approach is effective in retrieving geospatial video clips and nonspatial data.

Keywords: Augmented Reality, Geographic Information Systems, 3D GIS, Geographic Hypermedia.

1 INTRODUCTION

Distributed GIS systems have been inter-operated with each other under ubiquitous computing environments, so-called, *Ubiquitous GIS*. This is one of the consequences of the evolution of computing environments over the past 30 years. However, there is no doubt that new developments in the fields of multimedia, hypertext/hypermedia, three-dimensional representations, and virtual reality technology will have a great impact on the type of research issues. An interesting application for geospatial video may be an image sequence analysis that follows a spatially related object and derives a trajectory of its movement. Surprisingly, few convincing systems have been implemented yet.

The challenging problems can be summarized in four points: *georeferencing of remotely sensed data*, *creating geo-spatial contents*, *linking among geo-spatial hypermedia* and *providing direct physical navigation*. The main objectives of our research are:

- Design and develop an efficient linking method for heterogeneous data of the same object property to provide various geographic information to the users.
- Study a novel method for live-video image registration with navigation information in augmented environments.

In this paper, we propose a novel approach to connect geospatial video material with the geographic information of real-world geo-objects. Our research problem is reduced to finding non-spatial attributes to augment these with the videos. The main idea is to transform the video search space into a three dimensional virtual world search space according to the remotely sensed GPS data for non-spatial data querying in a video. Specifically, we utilize two types of system components during navigation: geospatial video client and server components. Based on these system configurations, our approach can provide good geographic information accuracy of the augmented videos using the GPS data by a simple and inexpensive device. Overall, our approach offers the following benefits:

- **Generality:** Our algorithm is general and applicable to wide range of geospatial clients such as desktop, laptop, mobile devices and so on.
- **Applicability:** Our geospatial server does not require any modification of attribute query algorithms or the runtime applications even if the types of clients are changed.

2 RELATED WORK

In this section, we give a brief survey of related works on geo-spatial hypermedia and geographic information

services through a video.

Independent video clips-based methods: Historically, the *Aspen Movie Map* Project, developed at MIT in 1978, is the first project combining video and geographical information [LA80]. Using four cameras on a truck, the streets of Aspen were filmed (in both directions), taking an image every three meters. The system used two screens, a vertical one for the video and a horizontal one that showed the street map of Aspen. The user could point to a spot on the map and jump directly to it instead of finding the way through the city.

Many research projects have used video clips in a similar way. The most typical case is *multimedia atlases* where the user can find video clips of locations for providing a deeper definition of any geographical concept. Other applications with a geographical background have used video clips: a *collaborative hypermedia* tool for urban planning. Most systems simply link 2D vector maps with video clips.

Video clips with geographic information: Peng et al. proposed a method for video clip retrieval and ranking based on maximal matching and optimal matching in graph theory [PND03]. Toyama et al. proposed an end-to-end system that capitalizes on geographic location tags for digital photographs [TLRA03]. Navarrete proposed a method, which performs the image segmentation for a certain video frame through image processing procedures for combining video and geographic information [TJ01].

The main problem of this method when dealing with big sources of video is how to segment it, i.e. how to choose the fragments of video that will be the base of later indexing and search. On option is a handmade segmentation of video, but this is too expensive for huge archives. Moreover, manual indexing has other problems as Smeaton [SA00] points out :

- No consistency of interpretation by a single person over time
- No consistency of interpretation among a population of interpreters
- No universally agreed format of the representation, whether keyword, captions or some knowledge-based information.

Due to these reasons, automatic segmentation of video has been an intensive research field in recent years.

Augmented geo-spatial videos: Augmented reality techniques with geo-referenced information have been proposed in order to build context aware and mixed reality applications [RC03, RDD02]. Grønbæk et al. combined spatial hypermedia with techniques from GIS and location based services [GVO02]. Kim et al. introduced a novel method for integration and interaction of video and spatial information [KKLPL03]. In our work, we build on their idea to implement the direct physical navigation using the geo-spatial hypermedia. Hansen et al. introduced a method to create a highly distributed multi-user annotation system for geo-spatial hypermedia [HC05]. They introduced a number of central concepts to understand the relation between hypermedia and spatial information management. However, their geo-spatial hypermedia is limited to the application domain of architecture and landscape architecture. In this paper, we propose a novel *geo-spatial hypermedia* which is suitable for real-time GPS-based navigation system.

Contributions: This paper's contribution is on two levels. First, the paper describes a general framework for non-spatial data querying on geo-objects (e.g., buildings) in a video. The framework includes a data model and definitions of abstract functionality needed for non-spatial data querying. Second, the paper proposes a framework for linking among geographic hypermedia such as 2D maps, 3D virtual environments and videos. The framework is intuitive enough to provide bi-directional links between various hypermedia in a multimedia GIS.

3 DATA REPRESENTATION FOR GEOGRAPHIC HYPERMEDIA

We define the data representations for geographic hypermedia such as the 2D map, the 3D virtual world, the geo-spatial video and road networks.

3.1 2D Map Representation

A 2D representation enables to us to place georeferenced objects in the geo-feature infrastructure. The 2D representation of a geo-feature in the 2D map is given by a two-tuple $M^{2D} = (G, P)$, where G is a set of geometries and P is a set of nonspatial data (properties) for the geo-features.
The data instances of the set of nonspatial attributes are stored in database relations. Each tuple in the relation corresponds to one object. More specifically, G denotes $G = \{(\mathbb{P}_i, ..., \mathbb{P}_k) \mid \mathbb{P}_i \in \mathbf{R}^2,\ k \geq 3\}$. P is the set of nonspatial data, (*attribute*,*value*). The G is encoded as *well-known binary* (WKB) representation which provides a portable representation of a geometry value as a contiguous stream of bytes [OGC99].

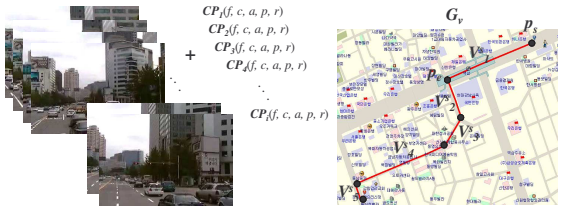
3.2 Virtual World Representation

The 3D representation of a geo-feature is given by a four-tuple $M^{3D} = (G, P, b, h)$, where b is a value of height on the ground and h is a value of a geo-feature's height (e.g., height of a building). The G and P are the

same as the 2D representation. We can create the 3D model for 2D map by extruding a 2D profile geometry with b and m [KKLPL03].

3.3 Geospatial Video Representation

The *geospatial video* is a spatial data that has a remotely sensed data as well as a video data. The geospatial



(a) Geospatial video content (b) The segment-based graph

Figure 1: Geospatial video.

video representation is given by a two-tuple $M^V = (\mathcal{V}, \mathcal{CP})$, where $\mathcal{V}$ is a set of image sequences in a video stream and $\mathcal{CP}$ is a set of camera parameters of the remotely sensed data. If I_i denotes a i-th image frame in a video $\mathcal{V}$, then $\mathcal{V}_k$ which has i image sequences is defined as:

$$\mathcal{V}_k = \{I_1, I_2, ..., I_i\}$$

The $\mathcal{CP}_i$ denotes the i-th camera parameters which contains internal parameters such as focal length $f(f_x, f_y)$, center $c(c_x, c_y)$, aspect ratio a and external parameters such as position $p(c_x, c_y, c_z)$ and orientation $r(r_x, r_y, r_z)$.

We use an integrated GPS approach using the GPS-Van, the so-called *4S-Van* to get improved results for the georeferencing [SBE01]. The hardware architecture of the 4S-Van consists of a data store part and a sensor part. The sensor part has a global positioning system (GPS), an inertial measurement unit (IMU), a color CCD camera, a B/W CCD camera, and an infrared rays camera. The 4S-Van acquires the $\mathcal{CP}$ every second. The $\mathcal{V}$ and $\mathcal{CP}_i$ are obtained from the 4S-Van.

3.4 Road Network Representation

Generally, the road network database has a node table and link table. We use **DB** to denote the road network database. The relation schema of node and link to describe the association can be represented as the following.

DB = { NODE, LINK }
Node(ID, LINKNUM, ADJNODE, GEOMETRY),
ADJNODE = (ID, PASSINFO, ANGLE),
$0 \leq |\text{ADJNODE}| \leq 8$, GEOMETRY = $\{P | P_i \in \mathbf{R}^2\}$.
Link(ID, SN, TN, DIST, ROADCLASS, LANECNT, GEOMETRY),
GEOMETRY = $\{(P_i, ..., P_k) | P_i \in \mathbf{R}^2, k \geq 2\}$.

The ID would serve to identify nodes and links uniquely. The LINKNUM denotes the number of adjacent links ADJNODE. The LINK table has start node (SN), destination node (DN), distance (DIST), class of road (ROADCLASS), the number of traffic-lanes (LANECNT) and geometry (GEOMETRY). The ADJNODE includes pass information at the intersection (PASSINFO), adjacent angle (ANGLE).

Given a real-world road network r, the *graph road network* is a two tuple $G_r = (V, E)$ where V denotes a set of vertices and E denotes a set of edges. For a directed graph G_r, the *segmented-based line digraph* $G_s = \mathcal{L}(G)$ has vertex set $V(G_s) = E(G_r)$ and edge set

$$E(G_s) = \{ab : a, b \in V(G_s), \text{HEAD}(a) = \text{TAIL}(b)\}$$

In order to provide the link between M^{2D}, M^{3D} and M^V, it is necessary to create the G_s for the video indexing. The graph road network G_r is decomposed into line segments, which are then indexed.

Without spatial index structures, finding the video for arbitrary moving points on the road network is computationally expensive. Thus, we use the R-tree which approximates data objects by axis-aligned minimum bounding rectangles (MBRs). However, approximating segments using axis-aligned MBRs proves to be inefficient due to the large amounts of *dead space*. So, for all leaf nodes of the R-tree, we construct a buffer zone of a line segment instead of a axis-aligned MBR to process the proximity query efficiently. Buffering involves the creation of a zone of a specified width around a point, line or polygonal area. In our research case, line segment buffering is required and the process for buffering a single segment is as follows. Two endpoints $p_s(x_1, y_1)$ and $p_e(x_2, y_2)$ belong to $\mathbf{R}^2$ of parallel buffer lines which lie on either side of the line segment at perpendicular distance d are determined using the following formulae:

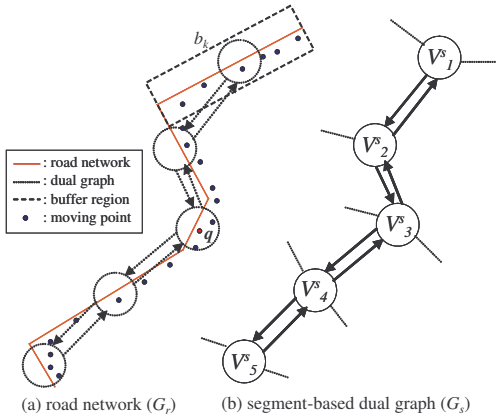
$$x_i \pm d \cdot \sin(\tan^{-1}(\frac{\Delta x}{\Delta y}))$$

$$y_i \pm d \cdot \cos(\tan^{-1}(\frac{\Delta x}{\Delta y})),$$

where Δx and Δy denote the difference between the two endpoints, p_s and p_e.

Here, we define a logical video segment $\mathcal{V}^s$ in the G_s as a three tuple $(p_s, p_e, \mathcal{V}^i)$. The first two elements belong to $\mathbf{R}^2$ and are the start and end points of the video segment. The last value is an index value ($\mathcal{V}^i$) as a three tuple $(f_s, f_e, \mathcal{V})$ of a video. The first two elements are starting frame number and ending frame number in a video. The last element is a video file location for browsing. There are more than one logical video segments in a video. However, the $\mathcal{V}$ is physically continuous. The details of the point query algorithm for video browsing is shown in **Algorithm 1**.

There are many geoobjects O_i in a geospatial video $\mathcal{V}$. However, $\mathcal{V}$ has no any geometries, G and attributes,



(a) road network (G_r) (b) segment-based dual graph (G_s)

Figure 2: The segment-based line digraph.

Require: The R-tree $\mathcal{R}_m$, the dual graph G_s
1: **procedure** FindVideoSegment
2: **input :** A query position q
3: **output :** A logical video segment $\mathcal{V}^s$
4:
5: $\text{MBR}_i = \text{findParentOfLeaf}(\mathcal{R}_m, q)$
6: **for** $\forall b_k \in \text{MBR}_i$ **do**
7: **if** $\text{isPointIn}(b_k, q)$ **then**
8: $\mathcal{V}^s = \text{getDualGraphNode}(G_s, b_k)$
9: **return** $\mathcal{V}^s$

Algorithm 1: Point query for video browsing.

P of O_i. Thus, it is necessary to propose a new algorithm for the backward linking of the geospatial video. The proposed algorithm will be presented in the next section.

4 LINKING GEO-SPATIAL HYPERMEDIA

We apply navigation concepts such as *direct physical navigation* and *indirect representational navigation* [GVO02]. Physical walking or driving is a example of direct physical navigation which includes interactive GPS based travel guides, location based tourist information, augmented reality based navigation. Indirect representational navigation is pointing and searching information on locations physically remote from the user's location. We propose a novel geo-spatial hypermedia based on these two navigation strategies.

4.1 Geo-Spatial Hypermedia for Indirect Representational Navigation

Two logical links are maintained between the spatial and nonspatial data instances of an object: *forward* and *backward* links. The linked instances and the links form what is termed a *spatial relation*. Forward links are used to retrieve the spatial information of an object given the object's nonspatial information. Backward links are used to retrieve the nonspatial information of an object given the object's spatial information. In order to improve the performance of search operations for these logical links in databases, it requires special support at the physical level. This is true for conventional databases as well as spatial databases, where typical search operations include the *point query* and the *region query*. Suitable index structures for the object ID (O_{ID}) are hash tables or B$^+$-trees. The hash table is particularly suitable for keys consisting of O_{ID} attributes since only exact match queries have to be supported. On the other hand, B$^+$-trees are advantageous for attributes that allow range queries–for example, int and float values.

The M^{2D} and M^{3D} have the spatial relation for the forward and backward links. However, M^V has only a video $\mathcal{V}$ and a remotely sensed data $\mathcal{CP}$ without the G and P. One possible approach to provide the attribute of geoobjects in the video is based on *MPEG-4 standard encoding*, which encodes spatial objects in every video frame according to MPEG-4 scene representation (BIFS) format [CL02]. This approach is a simple and intuitive method. However, this approach requires a lot of manual MPEG-4 authoring for every frame in the video. If the user wants to browse the video at the position q in the M^{2D}, the system finds the nearest neighbor segment(s) $\mathcal{V}^s_n$ in the G_s. Given a set of line segments L in the G_r, construct the modified R-tree $\mathcal{R}_m$ in $O(n\log n)$ time. Now, for a query point q, finding a nearest neighbor segment(s) q is reduced to the problem of finding in which buffer region(s) b_k it falls, for the sites of those buffer regions are precisely its nearest neighbors. The problem of locating a point inside a partition is called *point location*. We can perform the point location in $O(\log n)$ time to find the $\mathcal{V}^s$.

In order to find the attributes of geofeatures at user-selected window position in the video frame, we introduce a new approach, the so-called *search space transformation algorithm*. The main idea is to transform the search space in the M^V into the that of the M^{3D} according to the $\mathcal{CP}$ for non-spatial data querying in a $\mathcal{V}$. The problem to find the attributes of geoobjects (O_i) at image plane coordinate $p(x, y)$ in the video frame (f_n) is mapped into the problem of finding the attributes of ray-intersected objects (VO_i) at graphics plane coordinate $p(x, y)$ in the virtual world according to that video frame. The concept of search space transformation is shown in Fig.3. We proceed to design a software system that implements the search space transformation. A client-server architecture is natural for the problem considered: users work with the client software installed on their devices; and the server assists the clients through the http protocol in providing users with the geoobject query results.

The tasks of client and server are shown in Fig.4. First, the client passes the current video frame number f_n in the selected video and image plane coordinate

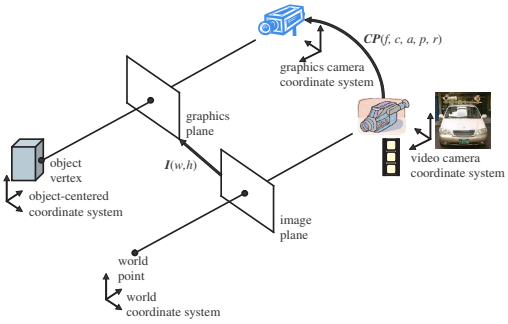


Figure 3: Search space transformation.

1: **procedure** FindObjectAttribute
2: **input :** $\mathcal{V}_i$, f_n, w, h, p
3: **output :** $prop_k$
4:
5: resizeVRView(w,h)
6: $\mathcal{CP}_j$ = getCameraParam($\mathcal{V}_i$, f_n)
7: locateVRCamera($\mathcal{CP}_j$)
8: O_{ID} = computeRayIntersection(p_x, p_y)
9: $prop_k$ = getAttribute(O_{ID})

Algorithm 2: Attribute query in a video.

$p(p_x, p_y)$ to the server. The server gets the $\mathcal{CP}$, position $P(c_x, c_y, c_z)$ and orientation $O(r_x, r_y, r_z)$ from the database according to the video ($\mathcal{V}$) and f_n. The server locates the camera to P with O in the 3D virtual space and then calculates the ray-intersection at $p(p_x, p_y)$ to get the identification (ID) of the geoobjects. Finally, the server passes the attributes of selected geoobjects in the 3D virtual world to the client according to the selected ID.

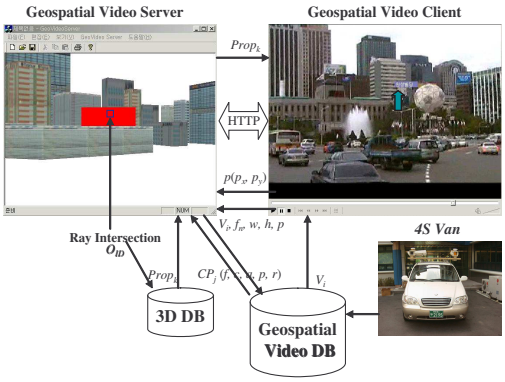


Figure 4: The client/server tasks

The details of our search space algorithm for backward linking is shown in **Algorithm 2**. The first four input parameters of the procedure are the video $\mathcal{V}_i$, current video frame number f_n, width/height of the $\mathcal{V}_i$, w, h and image plane $p(p_x, p_y)$. The return parameter of the procedure is the attribute $prop_k$ of the geo-object ID O_{ID} . The geo-object attribute search in the video is the key function of the software system previously presented.

The links between the mentioned search space transformation tasks and the sub-procedures are as follows. First, the call of the resizeVRView procedure in line 5 corresponds to the task of M^{3D} view resizing. Second, the getCameraParam procedure returns the $\mathcal{CP}_j$ from the database and the server locates the virtual camera to $\mathcal{CP}_j$ by using the locateVRCamera procedure in line 6 and 7. Then, the computeRayIntersection procedure in line 8 finds O_{ID} of the intersected geo-object in M^{3D}. Finally, the getAttribute procedure returns the attribute $prop_k$ for the O_{ID}. The implemented computeRayIntersection procedure in line 9 which computes *ray-box intersection* requires $\Theta(\log n)$ time, where n denotes the total number of the geo-objects.

4.2 Route Determination using Linear Dual Graph

The route determination procedure (simply, routing) has an important role in the navigation system. In our work, the aim of the routing is to minimize the search space for video browsing without exhaustively considering all links on the road network. In this section, we introduce the route determination algorithms under the turn-restrictions.

In most of traffic networks in South Korea, the route planning for car navigation or public transport should consider no-left-turn, P-turn, U-Turn and other turn problems to find a minimal travel time cost. In particular, in urban environment, turning left is often forbidden in order to minimize congestion and, when allowed, traffic lights and counter flows cause an extra travel cost itinerary between two nodes.

The common approaches to handle these problems are *node expansion* and *linear dual graph* methods [SW01B]. The node expansion approach builds up the expanded network, G_e, which is obtained by highlighting each movement at an intersections by means of dummy nodes and edges, where the costs of the dummy edges are penalties. The major disadvantage of this approach is that the resulting network G_e is significantly larger than the original graph G.

In determining the performance of the approaches, it is useful to summarize their storage requirements. The node expansion method requires $\delta_{max}|N_G|$ nodes for the primal graph G since any node n expands into $\delta(n)$ nodes (where, δ_{max} denotes maximum degree of node). This method also requires the number of edges in G, $|E_G|$, plus the number of paths of length 2, $\frac{\delta_{max}}{2}|E_G|$ [SW01]. The *boundary node* is a node which has a single-source and single-target in G. The linear dual graph (D) method requires the number of nodes which $|E_G|$ adds to the number of boundary nodes in G, $|N'_G|$. The storage requirement for the edges in D is the number of edges which $\frac{\delta_{max}}{2}|E_G|$ plus $|N'_G|$ (see Table 1).

Method	$\|N\|$	$\|E\|$
G_e	$\delta_{max}\|N_G\|$	$(1+\frac{\delta_{max}}{2})\|E_G\|$
D	$\|E_G\|+\|N'_G\|$	$\frac{\delta_{max}}{2}\|E_G\|+2\|N'_G\|$

Table 1: Estimation of upper limits for node expansion and linear dual graph. (N'_G denotes boundary nodes of a primal graph G and δ_{max} denotes maximum degree of node)

Based on these facts, we adopted the linear dual graph technique as a conceptual model for our route specification on G_s.

4.3 Geo-Spatial Hypermedia for Direct Physical Navigation

Augmented reality (AR) provides an especially rich medium for experimenting with location-aware and location-based applications, in which virtual objects are designed to take the environment into account [SS03]. In order to have efficient telematics systems all the information must be geo-referenced and the underlying support system must handle some of the functionalities usually supported by Geographic Information Systems (GIS), namely the mapping between the virtual and the real world. Thus, we should handle real-time videos according to user's locations along the navigation path. The run-time approach exploits live-time video streams rather than preprocessed video streams of the 4S-Van. It is useful to provide live-video with navigation information such as, building's name, current speed and turn information for users. To augment live-video with these information, we perform a projection of the 3D world onto a 2D image plane for correct registration of the virtual and live-video images. Fig. 5 shows the overall proposed system architecture for real-time approach.

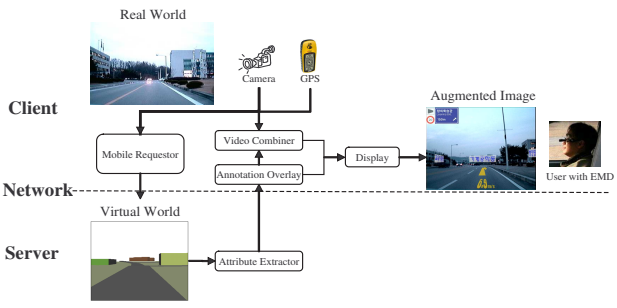


Figure 5: Overall proposed system schematic

The major procedure for our run-time approach is as follows.

1. *Extracting 3D information according to the GPS data.* The client passes the GPS data, position $P(c_x, c_y, c_z)$ and orientation $O(r_x, r_y, r_z)$, to the server. Then the server locates the virtual camera in 3D space according to the GPS data. The server find all attributes of the buildings from 3D database in given view frustum. We also extract the four points which compose a rectangular face of the 3D building model. This information will be used for registration of the annotated information with live-video images at the following step.

2. *Overlaying the building's name and the guide information to the video.* Every building in the virtual world from 3D database is represented as a hexahedron. Each face of the building is a rectangular face. So, this face has *coplanar* and *convex* properties. And, this face keeps these properties even if it is transformed by a *affine transformation* such as, translation, rotation, scaling and shear). Therefore, we can augmented building's annotation with live image by using these four points of rectangular face. More specifically, we use two overlay methods; orthogonal method and projective method to overlay an annotation of a building. We use *inverse mapping by bi-linear interpolation* which is the most popular practical strategy for texture mapping in order to perform the *projective transformation* [WP01]. For inverse mapping it is convenient to consider a single (compound) transformation from two-dimensional screen space (x, y) to two-dimensional texture space (u,v). This is just an image warping operation and we can write this in homogeneous coordinates as:

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} u' \\ v' \\ q \end{bmatrix}$$

where $(x, y) = (x'/w, y'/w)$ and $(u, v) = (u'/q, v'/q)$. This is known as a rational linear transformation. The inverse transform–the one of interest to us in practice is given by:

$$\begin{bmatrix} u' \\ v' \\ q \end{bmatrix} = \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix} \begin{bmatrix} x' \\ y' \\ w \end{bmatrix}$$

$$= \begin{bmatrix} ei-fh & ch-bi & bf-ce \\ fg-di & ai-cg & cd-af \\ dh-eg & bg-ah & ae-bd \end{bmatrix} \begin{bmatrix} x' \\ y' \\ w \end{bmatrix}$$

Moreover, the transmission data size for annotation overlaying is constant since we need these four points for each building in order to overlay the annotation.

3. *Displaying the overlaid video on the screen.* We use the eye-mounted display (EMD) as a display device based on video-see through approach. The screen resolution of the EMD is 640×480 pixels.

The mobile requestor sends GPS data to the server to obtain the building's name in a live-video, the same as geospatial video client in Fig.4. The video combiner

(a) Orthogonal overlay (b) Projective overlay

Figure 6: Annotation overays for geo-objects

plays an important role for augmenting a live-video with information from virtual world. Fig.6 (a) is a result of annotation overlay using parallel projection and Fig.6 (b) is a result using perspective projection.

5 EXPERIMENTAL RESULTS

The proposed algorithm is implemented in C++ and OpenGL. We tested the implemented system on several datasets which were obtained from Jung-Gu, Seoul, Korea by using the 4S-Van. We have evaluated our system on a PC running Windows XP operating system with Intel Xeon 3.0GHz Quad-core dual CPUs, 4 GB memory and an NVIDIA QuadroFX 5600 GPU. The client is implemented in C++ and ATL/COM as a component software.

Figure 7: Hardware components of client system.

The mobile client system consists of global positioning system (GPS), camera, eye-mounted display (EMD) as video see-through device, and reconfigurable computer (Laptop computer) as shown in Fig. 7. The details of the current hardware setup are the following:

- Garmin's eTrex GPS : used to capture the GPS data every second, such as current position, altitude, current speed, average speed and moving distance. The average position accuracy of GPS receiver is ±6m and the altitude accuracy is ±30m.

- Creative Labs Video Blaster's WebCam : used to capture the image according to the real road network.

- Eye Mounted Display (EMD): used to show the composed image to user and its screen resolution is 640×480 pixels.

In our experiments for the preprocessing approach, we perform a picking operation with at least 70 random points in the geospatial video client to measure the overall accuracy of geo-object query. By using the proposed method, we are able to achieve average 85.8 % accuracy for the benchmark scene which contains seven buildings. We report the accuracy of geo-objects query for our benchmark data in Fig. 8.

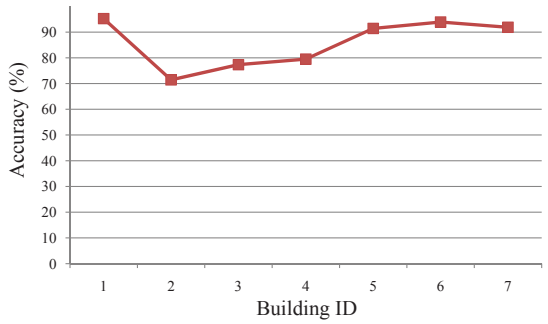


Figure 8: Accuracy of geo-objects query: The test scene contains seven buildings.

In order to evaluate the routing performance, we have tested the performance of route determination algorithm with our route queries in 38 different benchmarks. Fig. 9 shows the performance testing result according to the route length (path length). Performance is measured by executing workloads, each of them consisting of 38 sample queries. Generally, the number of path computations depends on a route length. If there are many intersections between source and destination, the number of path computations will increase significantly. However, the response time receives little effect from the Euclidean distance between source and destination. Therefore, the response time depends on the number of path computation.

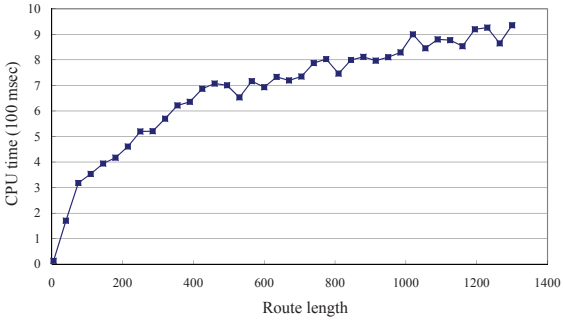


Figure 9: Performance of route determination according to the route length.

Limitations. Our approach works well for our current set of benchmarks. However, it has a few limitations. Our search space transformation algorithm has

the assumptions for *rectangular buildings*; especially extruded models from 2D maps. Therefore, there is no guarantee that our algorithm can handle general *polygonal buildings*. Moreover, our current direct physical navigation approach requires more *user experience* testing and *cognitive* experiments in terms of human computer interaction.

6 CONCLUSION

Geographic hypermedia offers a new opportunity to navigate with heterogeneous geographic contents by using remotely sensed data. We have identified four research themes for geographic hypermedia navigation, provided some background on significant achievements in those areas, as well as highlighted some of the remaining challenges.

We have proposed a new approach to linking among geospatial hypermedia by using the remotely sensed data obtained from the 4S-Van. The main idea of the search space transformation is to transform the search space of the video into that of the 3D virtual world according to the remotely sensed parameters for non-spatial data querying in a video. In conjunction with the proposed method for direct physical navigation systems, we can provide convincing geospatial hypermedia. We believe that this work is a first but important step towards an important research area in ubiquitous LBS. The experimental evaluation confirms the applicability of the proposed approach.

In summary, location-based geospatial hypermedia will play a central role in numerous mobile computing applications. We expect that research interest in such geographic hypermedia will grow as the number of multimedia mobile devices and related services continue to increase.

ACKNOWLEDGEMENTS

We would like to thank Jong-Hun Lee at GEOMania and research students at Virtual Reality Research Center (VRRC) at KAIST for useful discussions and feedback, and anonymous reviewers for helpful comments. This work was supported in part by the IT R&D program of MCST/KEIT [2006-S-045-1].

REFERENCES

[CL02] Chiariglione, L., *MPEG-4: Jump-Start*, Prentice Hall PTR, pp. 143-213, 2002.

[GVO02] Grønbæk, K., Vestergaard, P. P., and ørbæk, P., Towards geo-spatial hypermedia: Concepts and prototype implementation. In *Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia (HYPERTEXT '02)*, pp. 117-126, June 2002.

[HC05] Hansen, F. A., Christensen, B. G., and Bouvin, N. O. 2005. RSS as a distribution medium for geo-spatial hypermedia. In *Proceedings of the Sixteenth ACM Conference on Hypertext and Hypermedia (HYPERTEXT '05)*, pp. 254-256, Sep. 2005.

[KKLPL03] Kyong-Ho Kim, Sung-Soo Kim, Sung-Ho Lee, Jong-Hyun Park and Jong-Hun Lee, The Interactive Geographic Video, In *Proceedings of IEEE Geoscience and Remote Sensing Symposium, 2003 (IGARSS'03)*, vol.1, pp. 59-61, 2003.

[LA80] Lippman, A., Movie Maps: An Application of the Optical Videodisc to Computer Graphics, In *Proceedings of SIGGRAPH'80*, pp. 32-43, July 1980.

[OGC99] Open GIS Consortium, OpenGIS Simple Feature Specification for OLE/COM, *OpenGIS Implementation Specifications, Revision 1.1*, 1999.

[PND03] Yu-Xin Peng, Chong-Wah Ngo, Qing-Jie Dong, Zong-Ming Guo, Jian-Guo Xiao, Video Clip Retrieval by Maximal Matching and Optimal Matching in Graph Theory, In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME) 2003*, Vol.I, pp. 317-320, 2003.

[RC03] Romero, L. and Correia, N., HyperReal: a hypermedia model for mixed reality. In *Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia (HYPERTEXT '03)*, pp. 2-9, August 2003.

[RDD02] Romao, T., Dias, E., Danado, J., Correia, N., Trabuco, A., Santos, C., Santos, R., Nobre, E., Camara, A., and Romero, L., Augmenting reality with geo-referenced information for environmental management. In *Proceedings of the 10th ACM international Symposium on Advances in Geographic information Systems (GIS '02)*, pp. 175-180, Nov. 2002.

[SA00] Smeaton, A.F, Browsing and Searching Digital Video and Digital Audio Information, In *Proceedings of the 3rd European Summer School on Information Retrieval (ESSIR2000)*, Sep. 2000.

[SBE01] Seung-Young Lee, Byoung-Woo Oh, Eun-Young Han, A Study on Application for 4S-Van, In *Proceedings of International Symposium on Remote Sensing (ISRS) 2001*, pp. 124-127, Oct. 2001.

[SS03] Sinem Güven, Steven Feiner, Authoring 3D Hypermedia for Wearable Augmented and Virtual Reality, In *Proceedings of International Symposium on Wearable Computers(ISWC'03)*, pp. 118-126, 2003.

[SW01] Stephan Winter, Weighting the Path Continuation in Route Planning, In *Proceedings of 9th ACM International Symposium on Advances in Geographic Information Systems*, pp. 173-176, 2001.

[SW01B] Stephan Winter, Modeling Costs of Turns in Route Planning, *GeoInformatica*, Vol. 6, No. 4, pp. 345-360, 2002.

[TLRA03] Kentaro Toyama, Ron Logan, Asta Roseway, P. Anandan, Geographic Location Tags on Digital Images, In *Proceedings of ACM Multimedia 2003*, pp. 156-166, Nov. 2003.

[TJ01] Toni Navarrete, Josep Blat, VideoGIS: Combining Video and Geographical Information, *Research Report, Dept. of Computer Science, University of Pompeu Fabra*, 2001.

[WP01] Watt, A., Policarpo, F., *3D Games: Real-time Rendering and Software Technology*, ACM Press New York, Addison-Wesley, pp. 218-220, 2001.