Dual/Primal Mesh Optimization for Polygonized Implicit Surfaces

Yutaka Ohtake† and Alexander G. Belyaev†,‡
† Computer Graphics Group, Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany
‡ University of Aizu, Aizu-Wakamatsu 965-8580 Japan
{ohtake,belyaev}@mpi-sb.mpg.de, belyaev@u-aizu.ac.jp

Figure 1: Left: initial 30^3 marching cubes mesh, 2.9K triangles. Middle: mesh optimized without adaptive subdivision (2.9K triangles), see Section 2 for details. Right: mesh optimization with adaptive subdivision was used (20K triangles), see Section 3 and Table 1 for details.

ABSTRACT

A new method for improving polygonizations of implicit surfaces with sharp features is proposed. The method is based on the observation that, given an implicit surface with sharp features, a triangle mesh whose triangles are tangent to the implicit surface at certain inner triangle points gives a better approximation of the implicit surface than the standard marching cubes mesh [11] (in our experiments we use VTK marching cubes [16]). First, given an initial triangle mesh, its dual mesh composed of the triangle centroids is considered. Then the dual mesh is modified such that its vertices are placed on the implicit surface and the mesh dual to the modified dual mesh is considered. Finally the vertex positions of that "double dual" mesh are optimized by minimizing a quadratic energy measuring a deviation of the mesh normals from the implicit surface normals computed at the vertices of the modified dual mesh. In order to achieve an accurate approximation of fine surface features, these basic steps are combined with adaptive mesh subdivision and curvature-weighted vertex resampling. The proposed method outperforms approaches based on the mesh evolution paradigm in speed and accuracy.

SM'02, June 17-21, 2002, Saarbrücken, Germany.

Categories and Subject Descriptors

I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*curve, surface, solid, and object representations.*

General Terms

Algorithms.

Keywords

Mesh optimization, polygonized implicit surfaces, dual meshes, adaptive subdivision.

1. INTRODUCTION

Fast and accurate polygonization of implicit surfaces remains to be an area of active research [3, 19, 6, 14, 9] (see also references therein). Implicit surfaces generated using Boolean operations usually contain sharp edges and corners. An elegant approach for fast feature-sensitive surface extraction from volume data described by directed distance fields was recently proposed in [9]. Unfortunately the most implicit surface modelers still work with the traditional one function representation. A dynamic mesh approach for accurate polygonization of implicit surfaces with sharp features was studied in our previous work [14]. The approach resembles the explicit Euler method for a surface evolution governed by partial differential equations and, therefore, is relatively slow. Moreover,

it is hardly combined with adaptive mesh subdivision because of stability problems.

In this paper, we propose a fast method for improving polygonizations of implicit surfaces with sharp features. The method explores the following observation: given an implicit surface with sharp features, a triangle mesh whose triangles are tangent to the implicit surface at certain inner triangle points gives a good approximation of the surface. A preliminary version of the method was developed in [12].

The core of the method consists of two steps. First, given an implicit surface and its initial triangulation, we consider the dual mesh composed of the triangle centroids and modify the dual mesh by projecting its vertices onto the implicit surface. Next, we consider the mesh dual to the modified dual mesh and optimize its vertex positions by minimizing a quadratic energy.

The quadratic energy associated with each vertex of that "double dual" resembles the Garland-Heckbert quadratic error metric (QEM) [4, 5]. It is equal to the sum of the squared distances from the vertex to the planes tangent to the implicit surface at the corresponding vertices of the modified dual mesh.

In order to achieve an accurate approximation of fine surface features, the two basic steps described above are combined with curvature-weighted vertex resampling and adaptive mesh subdivision procedures.

The proposed method has several limitations. The mesh optimization process we developed does not change the topology of the initial coarse mesh. Therefore, if fine topological details are not captured by the initial mesh, the method may produce a wrong reconstruction of the implicit surface. Another drawback of the method is a large number of calls of a function which defines the implicit surface. If the function is very complex, the method becomes computationally expensive. A possible remedy may consist of a voxelization of the function: precomputing function values for each voxel, as it was done in [13] where a dynamic mesh approach was used.

The method developed in this paper is extremely good in reconstruction of sharp features of implicit surfaces. Fig. 1 and Fig. 2 demonstrate how the proposed mesh optimization approach improves initial low-resolution marching cubes meshes of implicit surfaces with sharp features.

2. DOUBLE DUAL MESH OPTIMIZATION

Given an implicit surface, conventional implicit surface polygonizers generate meshes whose vertices are optimized according to their distances to the implicit surface: they either are situated on the implicit surface or lie very close to the implicit surface. It may produce poor polygonal approximations even for very simple implicit surfaces: a well-known example is Schwarz's polygonal cylinder [10], see the left image of Fig. 3. A better polygonization of the cylinder can be achieved if the mesh triangles are tangent to the cylinder, see the right image of Fig. 3.

A similar example in 2D is schematically illustrated in Fig. 4 where a curve is approximated by a polyline whose vertices lie on the curve (left image) and by a polyline tangent to the curve at inner edge vertices (right image).

If an implicit surface has sharp edges and corners, a mesh optimization procedure which modifies an initial mesh such that the optimized mesh becomes tangent to the implicit surface may greatly improve the approximation quality, as seen in Fig. 5.

Thus an optimization of mesh normals is more important than an optimization of mesh vertices, especially for polygonized implicit surfaces with sharp features.

Our approach to optimizing mesh normals is based on the concept of the dual mesh used often for FEM mesh generation purposes and rapidly gaining popularity in the geometric modeling community [18].

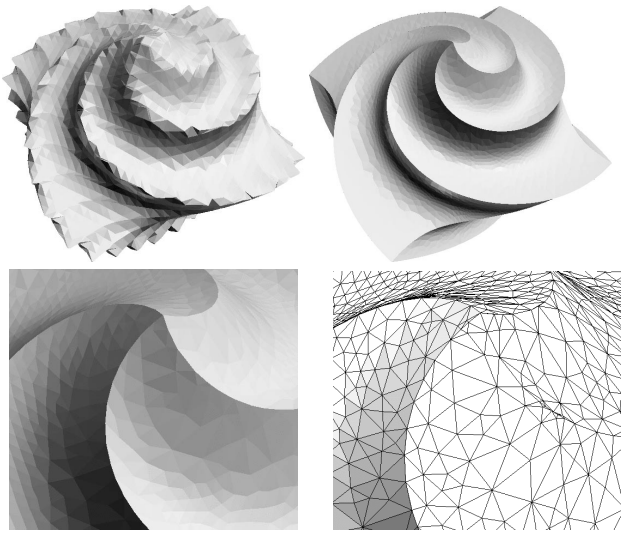


Figure 2: Top-left: an initial low-resolution marching cubes mesh of a model with sharp features. Top-right: mesh is optimized by the method developed in this paper. Bottom: a magnified view of a part of the optimized mesh; sharp features are very well reconstructed.

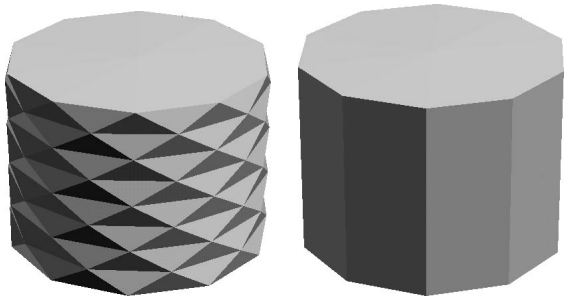


Figure 3: Left: Schwarz's polygonal cylinder. Right: Cylinder is approximated by a triangle mesh tangent to the cylinder.

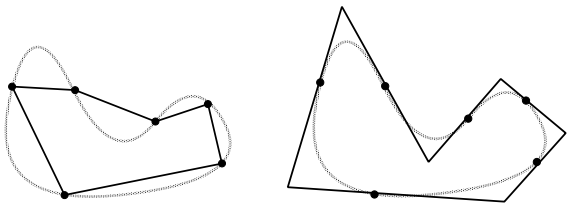


Figure 4: Left: A curve is approximated by a polyline whose vertices lie on the curve. Right: The same curve is approximated by a polyline tangent to the curve at inner edge vertices.

Given an implicit surface $f(x, y, z) = 0$ and its initial polygonization, our mesh optimization procedure consists of the following two steps:

1. Construct the dual mesh consisting of the centroid of the original mesh, modify the dual mesh by projecting its vertices onto the implicit surface, and find the tangent planes at the vertices of the modified dual mesh;

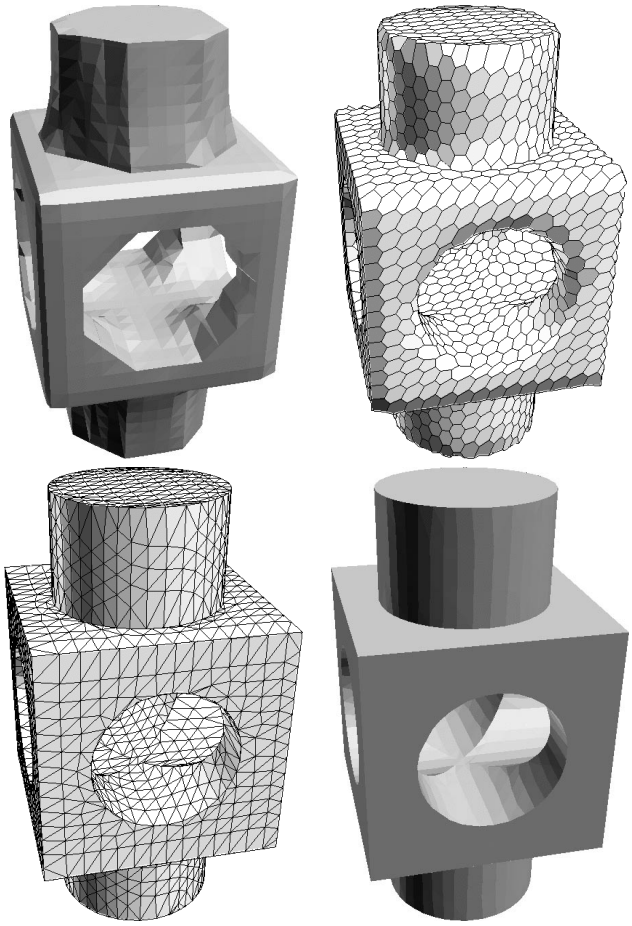


Figure 5: Top-left: initial (primal) marching cubes mesh. Top-right: optimized dual mesh whose vertices are placed onto the implicit surface. Bottom: optimized primal mesh.

2. for each vertex, update its position by minimizing an error function equal the sum of squared distances from the vertex to the tangent planes at the neighboring vertices of the modified dual mesh.

Fig. 6 sketches the procedure in 2D and details are explained in the rest of this section.

Optimizing dual mesh. Consider the dual mesh formed by the triangle centroids C. We are looking for an optimized dual mesh whose vertices are obtained by projecting the centroids C onto the implicit surface.

The projection P of a triangle centroid C onto the surface $f = 0$ is estimated as follows.

1. Set $P = C$.
2. If $|f(P)| < \epsilon$, then terminate. Otherwise set $Q := P$.
3. Set $R := Q + \lambda\,\mathbf{d}$, where $\mathbf{d}$ is obtained from $-f(Q)\,\nabla f(Q)$ by normalization and λ is a small constant.
4. If $f(Q)\,f(R) < 0$, then search for $P \in QR$ satisfying $|f(P)| < \epsilon$ by the bisection method and terminate.
5. Otherwise set $Q := R$ and return to Step 3.

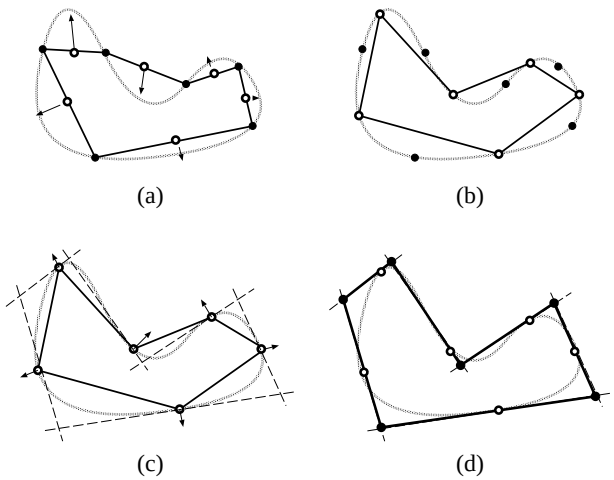


Figure 6: (a) An initial mesh usually delivers a poor approximation of a given implicit surface, so the dual mesh is considered. (a),(b) The vertices of the dual mesh are projected onto the implicit surface and form an optimized dual mesh. (c) the planes tangent to the implicit surface at the vertices of the modified dual mesh are determined. (d) For each vertex of the primal mesh, its optimal position is found by minimizing the sum of squared distances from the vertex to the tangent planes at the neighboring vertices of the modified dual mesh.

Here ϵ defines the precision with which we place the centroids onto the implicit surface.

Typically we use $\epsilon = 10^{-3}$. However, since the algebraic distance $f(P)$ from P to the implicit surface $f = 0$ may produce a very poor estimation of the Euclidean distance, it seems better to use the Taubin distance $\frac{f(P)}{||\nabla f(P)||}$ [17]. On the other hand, computation of ∇f increases the number of calls of the function f and may be computationally expensive. A possible remedy consists of voxelizatioin of the function $w = f(x, y, z)$ and precomputing ∇f for each voxel [13].

We have to choose λ carefully because too small λ leads to a computationally expensive procedure while too large λ causes jumps through thin parts of the solid bounded by the implicit surface. Consider a triangle of the initial polygonization and denote by e the averaged length of the triangle edges. Initially we set $\lambda = e/2$. However if the number of iterations is too large, we set λ to half its previous value in order to catch thin components of the implicit surface.

Furthermore, if the distance between the triangle centroid C and Q is greater than e, the search for the projection P is terminated and the original position of C is used for the optimized dual mesh.

Optimizing vertex positions. Consider the optimized dual mesh whose vertices P are placed onto the implicit surface $f = 0$. The unit surface normal at P and the plane tangent to the surface at P are given by

$$\mathbf{m}(P) = \nabla f(P)/||\nabla f(P)||, \qquad \mathbf{m}(P) \cdot (\mathbf{x} - P) = 0,$$

respectively, where $\mathbf{x}$ is a space point.

The gradient ∇f is equal to zero at the sharp features of the implicit surface $f = 0$. However the vertices P of the optimized dual mesh usually are not located at the sharp features and the unit normal $\mathbf{m}(P)$ is defined correctly. If however $\nabla f(P) = 0$, the

unit normal of optimized dual mesh at P can be used instead of $\mathbf{m}(P)$.

We want to optimize the vertex positions of the primal mesh using a quadric error metric similar to that introduced in [4, 5].

Let C_i be the primal mesh centroids of the triangles surrounding a primal mesh vertex $\mathbf{x}$ and P_i be their projections onto the implicit surface. Thus points P_i belong to the optimized dual mesh. We measure the distance error at the primal mesh vertex $\mathbf{x}$ by

$$E_{\mathrm{dist}}(\mathbf{x}) = \sum \left(\mathbf{m}(P_i) \cdot (\mathbf{x} - P_i)\right)^2$$

which is the sum of squared distances to the tangent planes at P_i.

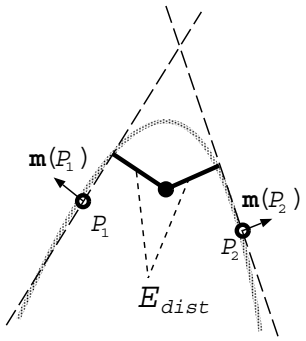


Figure 7: Geometric meaning of E_{dist}.

Since $E_{\mathrm{dist}}(\mathbf{x})$ is quadratic, its minimization is simple. The optimal position of the primal mesh vertex x is obtained by solving

$$\frac{d\,E_{\mathrm{dist}}(\mathbf{x})}{d\,\mathbf{x}} = 0 \qquad (1)$$

which is a system of three linear equations $\mathbf{A}\mathbf{x} = \mathbf{b}$. Note that the matrix $\mathbf{A}$ is singular when $\mathbf{x}$ lies on the flat regions or the straight sharp lines, as it was observed in [4, 9]. Similar to [9] we use the singular value decomposition [15] to find a minimum-norm least squares solution to (1). The old position of the corresponding primal mesh vertex is chosen as the origin of coordinates. If the ratio between the biggest singular value of $\mathbf{A}$ and any other nonzero singular values of $\mathbf{A}$ is above $\tau = 10^3$, the that latter singular value is set to zero. The threshold τ is chosen experimentally. Fig. 8 demonstrates how different values of τ affect reconstruction of a spike.

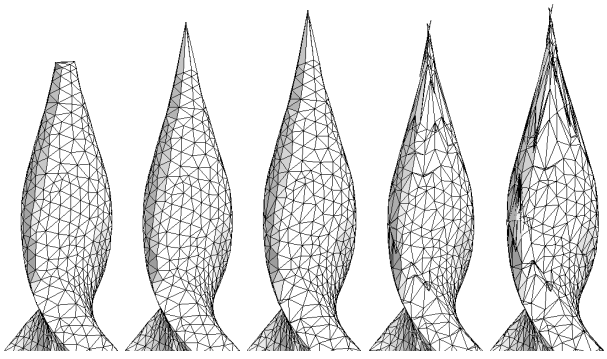


Figure 8: If τ is not large enough sharp features are not well reconstructed. Choosing too large values of τ leads to numerical instability.

3. RESAMPLING, SUBDIVISION, AND DECIMATION

In this section, we explain how the double dual mesh optimization can be combined with curvature-weighted vertex resampling and adaptive mesh subdivision. We also present a modification of Garland-Heckbert mesh decimation method for the polygonized implicit surfaces.

Curvature-weighted vertex resampling. The marching cubes often generate meshes with many thin triangles. That thin triangles may reduce numerical stability of our method described in the previous section. On the other hand, we need a triangulation which follows the surface geometry accurately, that is a triangulation with curvature-dependent triangle sizing. Very recently, a feature-sensitive remeshing technique was developed in [20] and a general approach for interactive geometry remeshing was proposed in [2]. In this paper, we use a simple resampling procedure which is easily combined with our mesh optimization described in the previous section.

Given an initial mesh with vertices $\mathbf{x}$, we improve the mesh by repeating three times the following double dual resampling procedure.

1. Vertices P of the optimized dual mesh are computed. (2)

2. Vertices $\mathbf{x}$ of the double dual (primal) mesh are updated via averaging vertices of the optimized dual mesh

$$\mathbf{x} = \sum w_i P_i \Big/ \sum w_i, \qquad (3)$$

where w_i are positive weights.

The averaging procedure (3) is similar to weighted Laplacian smoothing. Thus choosing equal weights leads to a uniform distribution of the vertices, as demonstrated in Fig. 9.

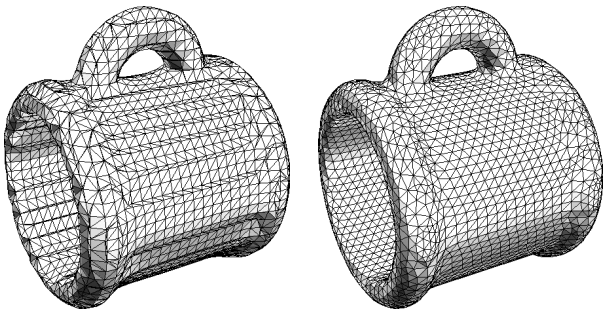


Figure 9: Left: initial mesh produced by marching cubes. Right: uniform mesh produced by three iterations of the double dual resampling process (2), (3) with equal weights ($\mathbf{c} = 0$ in (4)).

To produce a curvature dependent triangulation we set

$$w_i = 1 + c\,k_i, \quad k_i = \sum_j \frac{\arccos(\mathbf{m}(P_i) \cdot \mathbf{m}(P_j))}{\|P_i P_j\|}, \qquad (4)$$

where the sum is taken over three neighbors P_j of the optimized dual mesh vertex P_i and c is a user-specified threshold. Since $\mathbf{m}(P)$ is the unit normal vector at P of the isosurface passing through P, $\frac{\arccos(\mathbf{m}(P_i)\cdot\mathbf{m}(P_j))}{\|P_i P_j\|}$ estimates the absolute value of the directional curvature at P_i in the direction to P_j. Thus k_i in (4) measures a "curvedness" at P_i. Fig. 10 demonstrates advantages

of the curvature-dependent weights over the equal weights. The curvature-dependent resampling procedure described above generates a high quality mesh which is denser near curved regions and sharp features.

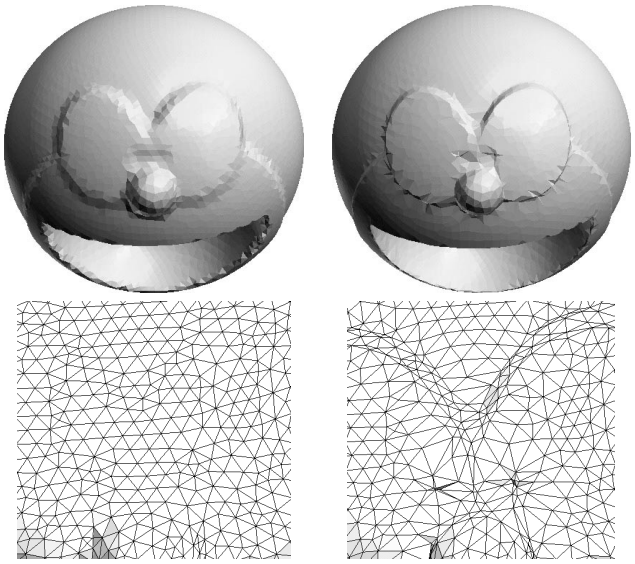
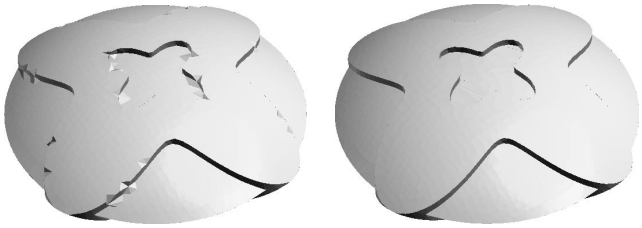


Figure 10: Left: equal weights are used. Right: curvature-dependent weights are used.

Adaptive resampling (2), (3), (4) is crucial for reconstruction of sharp features when the mesh is not dense enough, as it is demonstrated in Fig. 11. According to our experiments, choosing $c = 2$ leads to a good curvature depending triangulation.



$c = 0$ $c = 2$

Figure 11: Meshes optimized by the method proposed in Section 2 after three preliminary rounds of the vertex resampling procedure. Left: $\mathbf{c = 0}$ produces a uniform mesh which is not dense enough to catch sharp features. Right: adaptive remeshing with $\mathbf{c = 2}$ leads to a good reconstruction of sharp features.

If an initial mesh is not dense enough, another remedy to catch sharp features consists of adaptive mesh subdivision.

Adaptive mesh subdivision. If the initial (marching cubes) polygonization is not dense enough, small surface features cannot be well reconstructed. So we use linear one-to-four subdivision of those mesh triangles T where the mesh normals $\mathbf{n}(T)$ have large deviations from the implicit surface normals $\mathbf{m} = \nabla f / ||\nabla f||$.

Consider a triangle T and its imaginary one-to-four subdivision into four triangles T_i with centroids C_i, $i = 1, 2, 3, 4$. We measure the deviation of $\mathbf{n}(T)$ from the implicit surface normals by

$$e_n(T) = A(T) \sum_{i=1}^{4} \left(1 - |\mathbf{n}(T) \cdot \mathbf{m}(C_i)|\right),$$

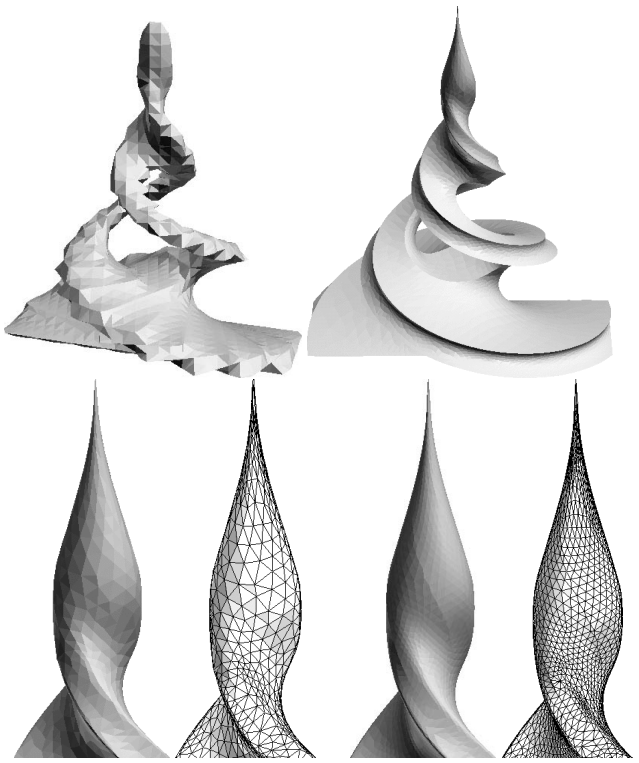


Figure 12: Top-left: initial marching cubes mesh. Top-right: the optimized mesh. Bottom: influence of the subdivision parameter ε; a magnified view of a spike part of the mesh optimized with $\varepsilon = 10^{-3}$ (two left images) and with $\varepsilon = 10^{-4}$ (two right images).

where $A(T)$ is the area of triangle T. If $e_n(T)$ is greater than a user-specified threshold ε, triangle T is subdivided. The parameter ε controls the accuracy of the polygonization. When ε is decreased, the accuracy is improved at the cost of an increase of the number of mesh triangles. Typically we choose ε in the range between 10^{-4} and 10^{-3}. The bottom images of Fig. 12 demonstrate how the optimized mesh depends on the parameter ε.

To eliminate T-junctions, the triangles sharing either two or three edges with one-to-four subdivided triangles are also one-to-four subdivided, the triangles sharing only one edge with an one-to-four subdivided triangle are one-to-two subdivided. This can produce thin triangles and vertices of high degrees, if the mesh subdivision procedure is applied several times. A better approach would consist of using the Kobbelt $\sqrt{3}$-subdivision technique [7].

Gathering all together. Given an initial marching cubes mesh, we optimize it combining the following mesh processing operations:

[I] the curvature-weighted resampling procedure (2), (3), (4);

[II] the dual/primal mesh optimization procedure (see Section 2);

[III] the adaptive mesh subdivision.

A complete mesh optimization procedure consists of the following combination of the above operations

$$([\,\mathrm{I}\,]^m + [\,\mathrm{II}\,] + [\mathrm{III}])^n + ([\,\mathrm{I}\,]^m + [\,\mathrm{II}\,]).$$

Figure	Computational time for MC	MC mesh #triangles	Optimized mesh comp. time	Optimized mesh #triangles	Optimization procedure (combination of [I], [II], [III])	Subdivision threshold ε
Fig. 1 right	0.08 sec.	2.9K	2.9 sec.	20K	$([\,\mathrm{I}\,]^3 + [\,\mathrm{II}\,] + [\,\mathrm{III}\,])^2 + ([\,\mathrm{I}\,]^3 + [\,\mathrm{II}\,])$	0.0005
Fig. 2	0.39 sec.	9.1K	3.5 sec.	19K	$([\,\mathrm{I}\,]^3 + [\,\mathrm{II}\,] + [\,\mathrm{III}\,])^2 + ([\,\mathrm{I}\,]^3 + [\,\mathrm{II}\,])$	0.0005
Fig. 5	0.08 sec.	4.3K	0.12 sec.	4.3K	$[\,\mathrm{I}\,] + [\,\mathrm{II}\,]$	–
Fig. 12 bottom-left	0.34 sec.	4.7K	3.1 sec.	12K	$([\,\mathrm{I}\,]^3 + [\,\mathrm{II}\,] + [\,\mathrm{III}\,])^3 + ([\,\mathrm{I}\,]^3 + [\,\mathrm{II}\,])$	0.001
Fig. 12 bottom-right	0.34 sec.	4.7K	9.5 sec.	34K	$([\,\mathrm{I}\,]^3 + [\,\mathrm{II}\,] + [\,\mathrm{III}\,])^5 + ([\,\mathrm{I}\,]^3 + [\,\mathrm{II}\,])$	0.0001
Fig. 15 middle	6.9 sec.	64K	80 sec.	75K	$([\,\mathrm{I}\,]^3 + [\,\mathrm{II}\,] + [\,\mathrm{III}\,])^2 + ([\,\mathrm{I}\,]^3 + [\,\mathrm{II}\,])$	0.0005
Fig. 16 middle	2.6 sec.	27K	19 sec.	27K	–	–
Fig. 16 right	2.6 sec.	27K	1.1 sec.	27K	$[\,\mathrm{I}\,]^3 + [\,\mathrm{II}\,]$	–
Fig. 17 left	3.1 sec.	40K	–	–	–	–
Fig. 17 right	0.39 sec.	9.7K	1.2 sec.	34K	$([\,\mathrm{II}\,] + [\,\mathrm{III}\,]) + [\,\mathrm{II}\,]$	0.0005
Fig. 19	0.47 sec.	3.2K	1.3 sec.	3.8K	$([\,\mathrm{I}\,]^3 + [\,\mathrm{II}\,] + [\,\mathrm{III}\,]) + ([\,\mathrm{I}\,]^3 + [\,\mathrm{II}\,])$	0.0005

Table 1: Sizes, timings, computational procedures, and subdivision thresholds for models considered in the paper. Timing results are measured on 1.2 GHz Mobile Pentium III PC.

First $[\,\mathrm{I}\,]$ is applied m times, then sequentially $[\,\mathrm{II}\,]$ and $[\,\mathrm{III}\,]$ are applied, and the combination $([\,\mathrm{I}\,]^m + [\,\mathrm{II}\,] + [\,\mathrm{III}\,])$ is repeated n times. Finally $[\,\mathrm{I}\,]^m$ and $[\,\mathrm{II}\,]$ are successively applied again.

According to our experiments, good results are obtained when $m = 3$ and $n = 2, 3, 4, 5$. See Table 1 for exact combinations of operations $[\,\mathrm{I}\,]$, $[\,\mathrm{II}\,]$, and $[\,\mathrm{III}\,]$ for each optimized model considered in the paper.

Fig. 12 shows mesh optimization for an implicit surface with sharp edges and a thin spike.

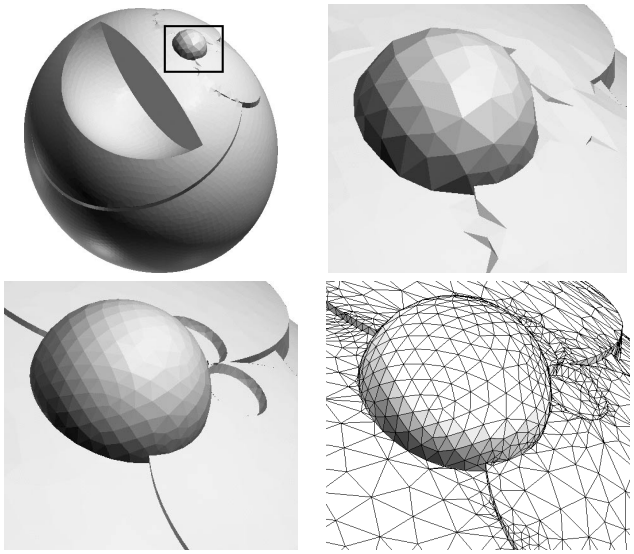


Figure 13: Top: optimized without adaptive subdivision. Bottom: optimized with adaptive subdivision.

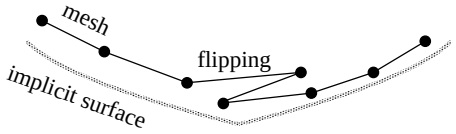


Figure 14: Mesh flipping.

Fig. 13 demonstrates advantages of the adaptive mesh subdivision procedure. The top images show a triangulation optimized two times without adaptive subdivision and the bottom images present a triangulation optimized with adaptive subdivision. In the latter case, all fine features are very well reconstructed.

The above mesh optimization procedure may produce a small number of flipped triangles where the optimized mesh bends over itself, as seen in Fig. 14, or few degenerate very thin triangles situated along sharp features. Such artifacts do not affect the visual appearance of the optimized mesh. Their detection and elimination is simple and the resulting mesh defects can be easily fixed by a simplified mesh optimization procedure consisting of the two basic steps described in Section 2.

Decimation. The adaptive subdivision technique proposed above may produce redundant triangles. We use a simple modification of the Garland-Heckbert mesh decimation method [4, 5] with the error function E_{dist} introduced in Section 2. The use of E_{dist} seems more appropriate for the simplification of polygonized implicit surfaces.

Fig. 15 demonstrates processing the Doraemon model.[1] A marching cubes mesh of the Doraemon model is shown in the left image, the twice optimized mesh is presented in the middle image, and 90%-decimated mesh is demonstrated in the right image. The decimated mesh preserves all fine features of the original implicit surface model.

4. DISCUSSION

The mesh optimization method developed in this paper outperforms the dynamic mesh optimization method [14] in speed and accuracy. Fig. 16 compares the methods.

The developed method works better than the high-resolution marching cubes even for smooth implicit surfaces, as demonstrated in Fig. 17.

Surprisingly, our method does not improve Schwarz's cylinder, as seen in Fig. 18. A combination with a dynamic mesh connectivity approach [8] may be useful for such kind of problems.

We would like to point out again that the developed approach is extremely good when dealing with implicit surfaces with sharp features, as it is demonstrated by Fig. 19.

Table 1 presents sizings, timings, computational procedures, and subdivision thresholds for the most of models considered in the paper.

Notice that the optimization of the Doraemon model is compu-

[1] Doraemon is a round cat-style robot, one of the most famous Japanese manga characters. The model was built within the HyperFun project [1].

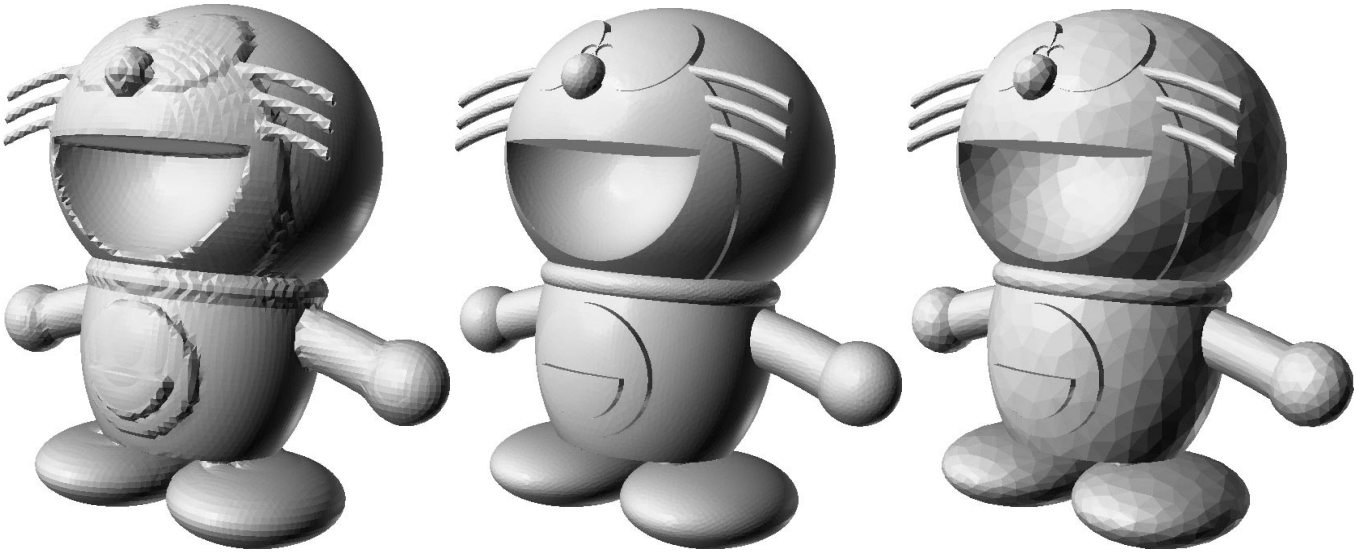


Figure 15: Left: Doraemon polygonized by marching cubes, 64K triangles, 100^3 grid was used. Middle: optimized mesh, 75K triangles. Right: 90%-decimated mesh, 7.5K triangles.

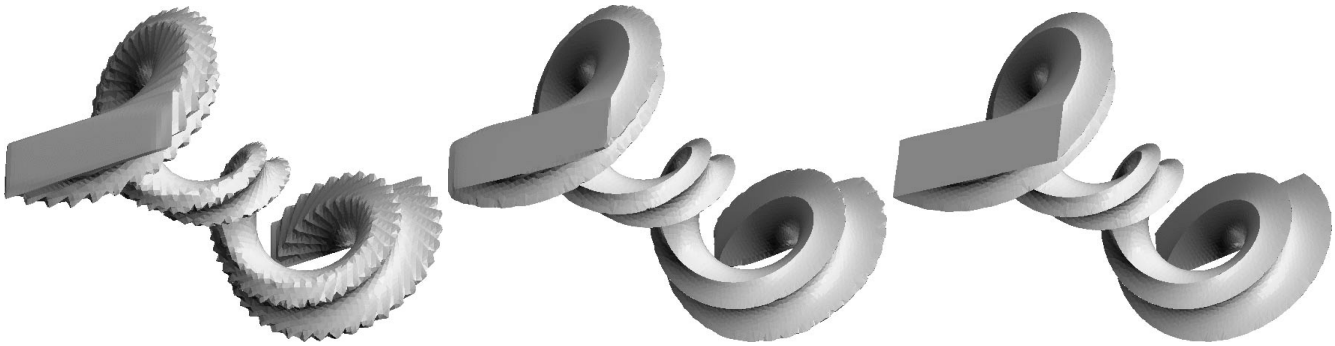


Figure 16: Left: Initial marching cubes mesh (27K triangles). Middle: dynamic mesh optimization (27K triangles, stabilization is achieved after about 20 sec.). Right: the marching cubes mesh optimized by the proposed method; adaptive subdivision was not used (1 sec.). Sharp features are very well reconstructed.

tationally expensive because the model is described by a very complex implicit function. As we mentioned early, one possible way to accelerate the optimization procedure for a model represented by a complex function consists of a preliminary voxelization of the function.

Acknowledgments

We are grateful to Alexander Pasko for drawing our attention to the problem of polygonizing implicit surfaces with sharp features and for the Doraemon model. We would like to thank the anonymous reviewers of this paper for their valuable and constructive comments.

REFERENCES

[1] V. Adzhiev, R. Cartwright, E. Fausett, A. Ossipov, A. Pasko, and V. Savchenko. HyperFun project: A framework for collaborative multidimensional F-rep modeling. In J. Hughes and C. Schlick, editors, *Implicit Surfaces '99, Eurographics/ACM SIGGRAPH Workshop*, pages 59–69, 1999. Visit also http://www.hyperfun.org.

[2] P. Alliez, M. Meyer, and M. Desbrun. Interactive geometry remeshing. In *Computer Graphics (Proceedings of SIGGRAPH 2002)*, July 2002.

[3] J. Bloomenthal. Chapter 4: Surface tiling. In J. Bloomenthal, editor, *Introduction to Implicit Surfaces*. Morgan Kaufmann, 1997.

[4] M. Garland. *Quadric-Based Polygonal Surface Simplification*. PhD thesis, Carnegie Mellon University, May 1999.

[5] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Computer Graphics (Proceedings of SIGGRAPH '97)*, pages 209–216, August 1997.

[6] T. Karkanis and A. J. Stewart. Curvature-dependent triangulation of implicit surfaces. *IEEE Computer Graphics & Applications*, 21(2):60–69, March / April 2001.

[7] L. Kobbelt. $\sqrt{3}$-Subdivision. In *Computer Graphics (Proceedings of SIGGRAPH 2000)*, pages 103–112, July 2000.

[8] L. Kobbelt, T. Bareuther, and H.-P. Seidel. Multi-resolution shape deformations for meshes with dynamic vertex connectivity. *Computer Graphics Forum (Eurographics 2000 issue)*, 19(3):C249–C260, 2000.

[9] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Sei-

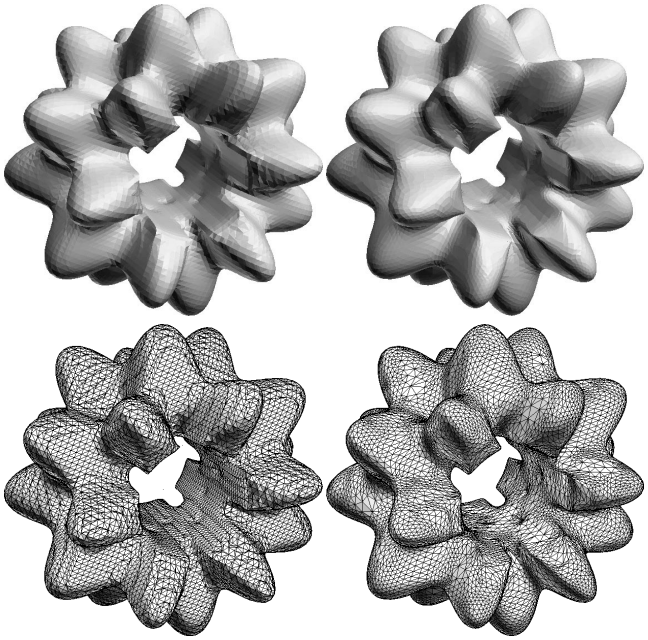


Figure 17: Left: 100^3 marching cubes mesh (40K triangles). Right: optimized 50^3 marching cubes mesh (9.7K triangles).

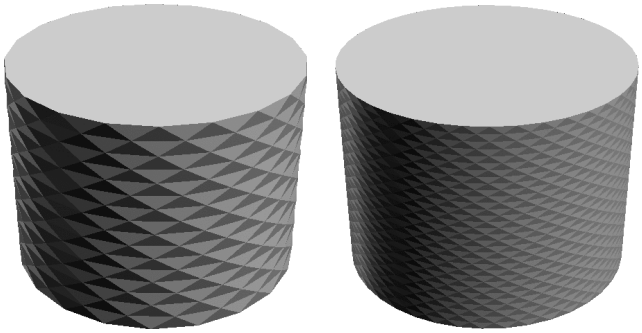


Figure 18: Schwarz's polygonal cylinder from Fig. 3 is optimized with one round of subdivision (left) and two rounds of subdivision (right).

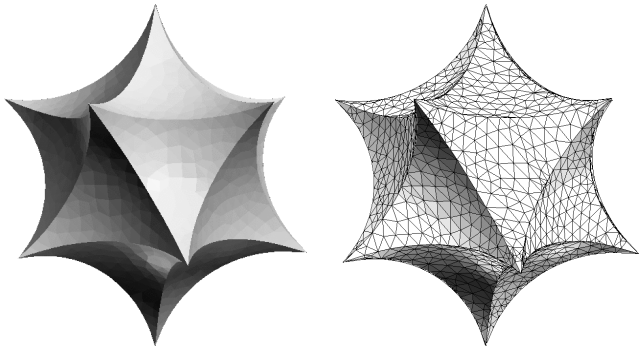


Figure 19: Optimized mesh for a curvilinear dodecahedron with sharp edges and corners.

del. Feature sensitive surface extraction from volume data. In *Computer Graphics (Proceedings of SIGGRAPH 2001)*, pages 57–66, August 2001.

[10] J. J. Koenderink. *Solid Shape*. MIT Press, 1990.

[11] W. E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3D surface construction algorithm. *Computer Graphics (Proceedings of SIGGRAPH '87)*, 21(3):163–169, 1987.

[12] Y. Ohtake. *Mesh Optimization and Feature Extraction*. PhD thesis, University of Aizu, March 2002.

[13] Y. Ohtake and A. G. Belyaev. Mesh optimization for polygonized isosurfaces. *Computer Graphics Forum (Eurographics 2001 issue)*, 20(3):368–376, 2001.

[14] Y. Ohtake, A. G. Belyaev, and A. Pasko. Dynamic meshes for accurate polygonization of implicit surfaces with sharp features. In *Shape Modeling International 2001*, pages 74–81, Genova, Italy, May 2001.

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1993.

[16] W. Schroeder, K. Martin, and W. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Prentice Hall, 1998.

[17] G. Taubin. Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1115–1138, 1991.

[18] G. Taubin. Dual mesh resampling. In *Proceedings of Pacific Graphics '01*, pages 180–188, October 2001.

[19] L. Velho. Simple and efficient polygonization of implicit surfaces. *Journal of Graphics Tools*, 1(2):5–24, 1996.

[20] J. Vorsatz, C. Rössl, L. P. Kobbelt, and H.-P. Seidel. Feature sensitive remeshing. *Computer Graphics Forum (Eurographics 2001 issue)*, 20(3):393–401, 2001.