Fast Polygonization of Variational Implicit Surfaces

ALVARO CUNO, CLAUDIO ESPERANÇA, ANTONIO OLIVEIRA, PAULO ROMA CAVALCANTI

PESC-Programa de Engenharia de Sistemas e Computação
COPPE/Universidade Federal do Rio de Janeiro
{alvaro, esperanc, oliveira, roma}@lcg.ufrj.br

Abstract. This article presents a simple hierarchical adaptation of the Marching Cubes algorithm for polygonizing variational implicit surfaces used in modelling and reconstruction applications. The technique relies on placing the normal and boundary constraint points respecting a pseudo-Euclidean distance metrics. This procedure makes it possible to quickly prune the space and minimize the number of costly function evaluations and thus converge rapidly to the surface. Timings show that this technique tends to perform faster than Bloomenthal's continuation polygonizer [5].

1 Introduction

The process of creating implicit models has been intensively studied in the last few decades. This is attributed to the fact that implicit models enjoy several nice properties, such as definition of the surface by one analytical function, unification of the surface and volume modeling, easiness to perform complex edit operations, capacity to represent complex objects and inside/outside tests.

Recently, the use of radial basis functions (RBFs) for the creation of implicit models has been reported in quite a few publications [7, 20, 10, 11, 30, 15, 16, 8]. These models offer nice properties such as smoothness (the function is continuous and differentiable) and direct control over surface creation. Furthermore, the modelling process does not require previous knowledge of the surface topology.

Turk and O'Brien [26] introduced a new approach of creating implicit surfaces based in RBFs. These surfaces are described by so-called constraint points, i.e., locations in 3D through which the surface should pass and locations that are interior or exterior to the surface. A 3D implicit function is created from these constraint points using a variational scattered data interpolation approach. They coined the term *variational implicit surface* to refer to the zero-set of such a function. This implicit function is a sum of weighted radial basis functions (RBFs) centered on the constraint points (Turk and O'Brien used triharmonic splines as RBFs). The weights of the RBFs are determined by solving a linear system of equations.

The construction of the equations system requires $O(n^2)$ time and $O(n^2)$ space, where n is the number of constraint points, whereas solving the equations system using direct methods such as LU-decomposition or singular value decomposition has $O(n^2)$ and $O(n^3)$ computational and storage complexity, respectively.

In addition to the cost related with the solution of the equations system, a high quality visualization of the iso-surface requires the function to be evaluated at a very large number of points. Because of the global nature of variational implicit functions, all their terms must be used in computing the value function at any one point[1]. Thus, each evaluation of the interpolated function has $O(n)$ time complexity.

The visualization of surfaces of this type can be done by direct methods such as ray-tracing [13]. Also, Reuter et al. [22] present a point-based rendering developed specifically for RBF-based surfaces. Nevertheless, visualization is most frequently achieved by employing a polygonization schema such as Bloomenthal's continuation algorithm [5] or the Marching Cubes algorithm [18].

Most practitioners of implicit shape modelling based on RBFs favor the use of Bloomenthal's algorithm, since it is very efficient and has a fairly well known implementation. It also has the advantage of sampling the space only in the neighborhood of the desired iso-surface. Being a continuation method, however, it requires a "seed" point for every connected component of the iso-surface and providing a complete but non-redundant set of such seeds may prove to be hard in some cases.

On the other hand, methods which sample the space regularly (e.g., Marching Cubes) are guaranteed to produce a correct result. However, they are clearly not suitable for the task at hand since they pay a heavy performance penalty by sampling the space at irrelevant locations. In this context, a hierarchical method may prove useful, provided that it is capable of converging rapidly to space regions (cells) which straddle the desired iso-surface.

In this paper, we present an iso-surface extraction technique based in the hierarchical sampling of the implicit function domain. The technique minimizes the number of function evaluations and thus the efficient application of the

[1] It must be mentioned that approximate methods such as those described by [7] may help in reducing the number of terms in the summation.

Marching Cubes algorithm. The idea is based in interpolating a pseudo-Euclidean distance metric enforced by a careful choice of normal constraint points.

This paper is organized as follows. Section 2 outlines methods used for the construction and visualization of implicit models based in RBFs. Section 3 briefly introduces several concepts related to variational implicit surfaces. Section 4 explains the proposed technique. In Section 5, experimental results are presented. Section 6 presents concluding remarks and suggestions for future work.

2 Related work

2.1 RBF-based implicit function construction

The idea of using RBFs for modeling implicit surfaces was introduced by Savchenko [23] and Turk and O'Brien [26]. It consists of producing an scalar field in which the desired surface is a zero-set, whereas points inside/outside the surface are mapped to negative/positive values. Unfortunately, the global nature of this representation encumber its use in modelling surfaces described by a very large number of points.

Morse et al. [20] have used compactly supported radial basis functions (CSRBFs), introduced by Wendland in [29], to confront this problem. Kojekine et al. [15] improved the method by organizing the sparse matrix produced by Morse into a band-diagonal sparse matrix which can be solved more efficiently. Because of the multiple zero-level sets created by this method, the resulting function has limited application in CSG, interpolation or similar applications [20].

Carr et al. [7] have used RBFs for reconstructing the surface of objects for which range data is available. In order to be able to cope with large amounts of data, they benefit from several optimizations reported by Beatson [3, 2]. Afterwards, Laga et al. [16] introduce the *Parametric Radial Basis Functions*, similar to the work of Carr, but adapted for fitting smooth objects as well as objects with sharp boundaries.

Recently, Tobor et al. [24] present a new approach to reconstruct large geometric datasets by dividing the global reconstruction domain into smaller local subdomains, solving the reconstruction problems in the local subdomains using radial basis functions with global support, and combining the solutions together using the partition of unity method [21].

2.2 Polygonization

Once modelled, an analytically-defined implicit function such as the kind produced by RBFs may be visualized in several ways. Most frequently, one wishes to produce a piecewise linear approximation of its zero-set. This procedure is known as polygonization. A detailed survey of polygonization methods is out of the scope of this paper. We refer the interested reader to comprehensive works in the area such as [1, 28].

Lorensen and Cline [18] presented the Marching Cubes algorithm for constructing iso-surfaces of 3D medical data. The basic principle is to reduce the problem to that of triangulating a single cube, which is intersected by a surface.

Bloomenthal [4] presented a polygonization algorithm in which the implicit function is adaptively sampled by subdividing space with an octree-like partitioning scheme, which may either converge to the surface (using octree cubes subdivision) or track it (by cell propagation). Terminal octree cells are then poligonized in constant time. Adaptive subdivision is also used to solve ambiguity problems.

In addition to these two basic approaches, many variants were proposed in order to deal with the inherent sampling problems (e.g., [25, 19]). In essence, these variants try to reduce the sampling rate and, at the same time, making sure that the resulting surface is geometric and topologically correct.

In general, any polygonizer developed for general implicit functions can be used to polygonize variational implicit surfaces. Turk and O'Brien [26, 27], Karpenko [14] perform iso-surface extraction using Bloomenthal's continuation method. Huong Quynh Dinh et al. [10, 11] extracted iso-surfaces using Marching Cubes [18]. Carr et al. [7] used a continuation method based on the marching tetrahedra algorithm [25].

Specific polygonizers for variational implicit surfaces were presented by [9, 16, 12]. The Crespin [9] algorithm performs an incremental Delaunay tetrahedralization of the constraints points. But it is very expensive and does not present visually good results.

Laga et. al. [16] use an octree scheme to find voxels near constraint points. The voxel classification procedure's main advantage is the fact that it does not require evaluation of the implicit function. After boundary voxels are found, they are polygonized by having their corners evaluated. The main shortcoming of this method is that it cannot be used when the point density is low.

Xiaogang et al. [12] present an approach which requires a coarse input mesh which approximates the desired iso-surface. This control mesh is recursively subdivided to a given level using a polyhedral subdivision scheme. Since the new added vertices usually do not lie on the implicit surface, they are mapped to the implicit surface using Newton's iteration method. The algorithm is efficient and produces high-quality meshes as a result of the interpolating subdivision scheme. The only drawback is that it requires a triangular mesh in order to supply the necessary connectivity information.

3 Variational implicit surfaces

Variational implicit surfaces are used in the context of the scattered data interpolation problem stated as follows:

> Given a set of n distinct constraint points $\{c_1, c_2, \ldots, c_n\}$, $c \in \Re^3$, and a set of function values for each of these points $\{v_1, v_2, \ldots, v_n\}$, $v \in \Re$, find the smooth function $f : \Re^3 \mapsto \Re$ such that $f(c_i) = v_i$, for $i = 1...n$. The smoothness of f is enforced by minimizing the second order energy functional given by

$$\int_{\mathbf{x}\epsilon\Re^3} \sum_{i,j} (\frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}_i \partial \mathbf{x}_j})^2 d\mathbf{x}. \qquad (1)$$

The solution to this problem is unique, and is known as the thin-plate interpolation of the points [17, 6]. There are several numerical methods that can be used to compute it, such as finite elements and finite differencing techniques. Alternatively, the solution can be expressed it in terms of a linear combination of RBFs:

$$f(\mathbf{x}) = \sum_{j=1}^{n} \lambda_j \phi(\mathbf{x} - c_j) + p(\mathbf{x}), \qquad (2)$$

where c_j are the positions of the known constraint points, λ_j are the weights of the radial basis functions centered at those points and $p(\mathbf{x})$ is a polynomial term.

For the thin-plate solution in 3D, Turk and O'Brien [26] used $p(\mathbf{x}) = a + bx + cy + dz$ and the triharmonic spline[2] $\phi(\vec{r}) = \|\vec{r}\|^3$. They called the zero-set of the implicit function f a variational implicit surface.

To find the λ_j set and polynomial coefficients a, b, c, d we need satisfy the interpolation constraints $f(c_i) = v_i$. Substitute their left side, resulting in:

$$\sum_{j=1}^{n} \lambda_j \phi(c_i - c_j) + p(c_i) = v_i. \qquad (3)$$

Additionally, the set of λ_j have to satisfy the orthogonality conditions:

$$\sum_{i=1}^{n} \lambda_i = \sum_{i=1}^{n} \lambda_i c_i^x = \sum_{i=1}^{n} \lambda_i c_i^y, \sum_{i=1}^{n} \lambda_i c_i^z = 0, \qquad (4)$$

where $c_i = (c_i^x, c_i^y, c_i^z)$.

Since equations (3) and (4) are linear with respect to the unknowns λ_j, a, b, c and d, the problem can be formulated as a linear system. Let $\phi_{ij} = \phi(c_i - c_j)$. Then this linear system can be written as follows:

[2] Although here we focus on thin-plate radial basis functions, it should be noted that other RBFs may also be used.

$$\begin{pmatrix} \phi_{11} & \phi_{12} & \ldots & \phi_{1n} & 1 & c_1^x & c_1^y & c_1^z \\ \phi_{21} & \phi_{22} & \ldots & \phi_{1n} & 1 & c_2^x & c_2^y & c_2^z \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{k1} & \phi_{k2} & \ldots & \phi_{kn} & 1 & c_k^x & c_k^y & c_k^z \\ 1 & 1 & \ldots & 1 & 0 & 0 & 0 & 0 \\ c_1^x & c_2^x & \ldots & c_n^x & 0 & 0 & 0 & 0 \\ c_1^y & c_2^y & \ldots & c_n^y & 0 & 0 & 0 & 0 \\ c_1^z & c_2^z & \ldots & c_n^z & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \\ a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad (5)$$

This system is symmetric and positive semi-definite, and can be solved by direct or approximate methods.

4 The polygonization algorithm

4.1 Specification of normal constraint points

A variational implicit function is modeled by specifying a set of n constraint points $\{c_1, c_2, ..., c_n\}$, together with a set of function values $\{v_1, v_2, ..., v_n\}$ at the given points [30]. To control the implicit surface we may specify *boundary constraint points* which are the positions that take the value zero, and the created implicit surface will pass exactly through these points. In addition, we may generate the *normal constraint points* that are the positions that will define the orientation of the surface.

The choice and placement of constraint points is strongly dependent on the application. In reconstruction applications (e.g., [30]), the input data is frequently a polygonal mesh whose vertices q_i are used as boundary constraint points. Also, for each vertex q_i, a normal vector is estimated and a normal constraint point n_i is introduced by displacing q_i by a given small distance d along that vector. This normal constraint point is given a value of w. In summary, the constraints are given by $f(q_i) = 0$ and $f(n_i) = w$. Figure 1 illustrates the idea.

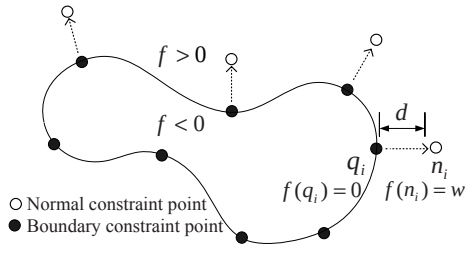

Figure 1: The normal constraint points n_i are placed along the estimated normal vector at a distance d from boundary constraint points q_i. The function f is such that $f(\mathbf{x}) < 0$ for $\mathbf{x}$ inside the curve and $f(\mathbf{x}) > 0$ outside the curve.

For reasons that will become clear later, we would like to ensure the following property for the interpolant:

$$|f(\mathbf{x})| \leq \delta(\mathbf{x}, S), \qquad (6)$$

where S is the the zero iso-surface of f and δ denotes the Euclidean distance metrics.

The primary means for enforcing this property is to adjust the values of w and d. A necessary condition is that $w < d$. This guarantees that if n_i is a normal constraint point, then $f(n_i) < d$. It is reasonable to assume that d is a good approximation of the Euclidean distance between n_i and S. In our implementation, this assumption is made stronger by ensuring that no other vertex $q_j, j \neq i$ is closer to n_i [7]. Furthermore, we may set w so that it is significantly smaller than d, and thus guaranteeing property (6) within a limited neighborhood of S.

Figure 2 illustrates how the interpolating function and the Euclidean distance function vary along a line intersecting a reconstruction of the "Stanford Bunny" (Fig. 2.a). The model was created based on a 800-vertex polygonal mesh using $d = 0.015$ and $w = 3d/4$. We may observe that the value of the interpolating function is less than or equal to that of the Euclidean distance function in the neighborhood of the model (Fig. 2.b). As the sampled point gets farther from the model, however, the interpolating function will eventually catch up with the Euclidean distance function (see Fig. 2.c).

Fortunately, our polygonization method does not requires property (6) to be valid everywhere, but only within a limited neighborhood of the model.

4.2 Construction of the interpolating function

Once the constraint points are determined, the linear equation system 5 is constructed and solved. We use a standard LU-decomposition method for this purpose.

It should be noted, however, that due to the use of triharmonic splines as basis functions, the resulting matrix is dense and entries tend to assume larger values in positions which are more distant from the main diagonal. In particular, entries on the main diagonal have zero values. Such ill-conditioned matrices can be solved by direct and approximate methods only at a high computational cost. Recently, some mathematical advances were proposed which help coping with this problem. Among these, we may cite, the Beatson-Faul-Goodsell-Powell (BFGP) method, Fast Multipole methods and pre-conditioning based methods. See [6] for details of this classification.

4.3 Adaptive space sampling

In this step the space is sampled using progressively finer grids in order to find which cells of a given target size straddle the surface. The idea is to refine a grid cell only if it contains a part of the surface. At the end we obtain a set of non-overlapping cells of equal size, each of which is guaranteed to overlap the surface. These are then triangulated using the standard Marching Cubes method.

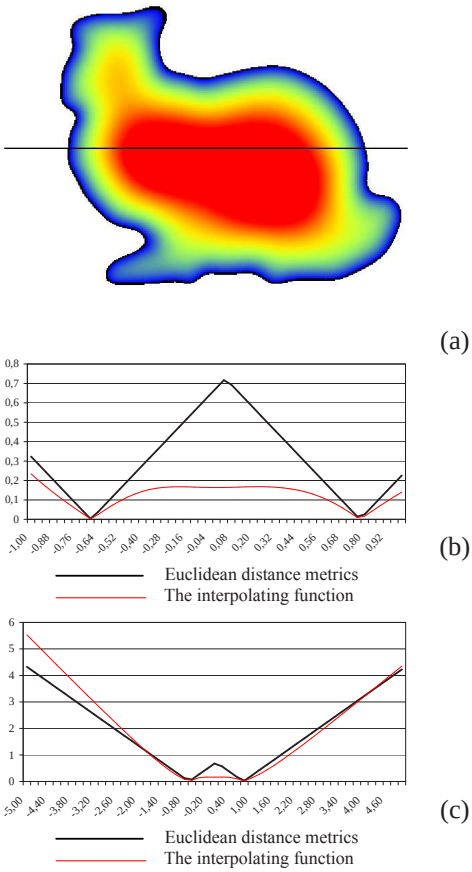

Figure 2: (a) A cross section of the interpolating function of Stanford Bunny.(b) Plot of the interpolating function f and the Euclidean distance metrics evaluated along a line in the neighborhood of the model. (c) A zoom-out of the chart shown in (b).

Let C be a cubic cell of the grid. In order to determine if C straddles the surface, we evaluate the interpolating function at the cube's center s. If r is the radius of the smallest sphere which contains C, then the iso-surface S may intersect C only if $\delta(s, S) \leq r$. We shall assume that property (6) is valid within a given neighborhood of S, say, all points $\mathbf{x}$ such that $|f(\mathbf{x})| \leq \Delta$. This is stated below:

$$|f(\mathbf{x})| \leq \Delta \Rightarrow \delta(\mathbf{x}, S) \geq |f(\mathbf{x})|. \quad (7)$$

Thus, as long as s is inside that neighborhood, we may substitute $f(s)$ for $\delta(s, S)$ in our distance test (see Fig. 3). In other words, cell C need not be subdivided if $r < |f(\mathbf{s})| \leq \Delta$.

Let us now consider a cell C which is relatively distant from S, more specifically, a cell whose center s is such that $|f(s)| > \Delta$. In this case, it is safe to assume that the

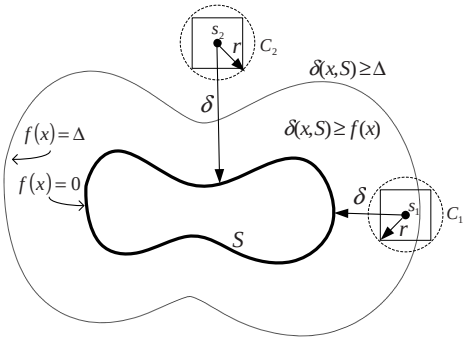

Figure 3: A cell C_1 "near" from S may be rejected if $f(s_1) > r$ since in this case $\delta(s_1, S) > r$. A cell C_2 "distant" from S satisfies $f(s_2) > \Delta$ and thus also satisfies $\delta(s_2, S) > \Delta$. C_2 can be discarded provided that $r < \Delta$.

distance between s and S is also greater than Δ, since all points p which are at a distance smaller than Δ to S necessarily satisfy $|f(\mathbf{p})| < \Delta$. Thus, we may state

$$|f(\mathbf{x})| > \Delta \Rightarrow \delta(\mathbf{x}, S) > \Delta. \quad (8)$$

We may take advantage of this by making sure that the radius of C is never greater than Δ since in this case $|f(s)| > \Delta$ will necessarily imply $\delta(\mathbf{x}, S) > r$, meaning that the cell cannot intersect S (see Fig. 3). This assumption permits us to cover both cases (cell "near" or "far" from S) with a single simple rejection test:

$$|f(s)| > r \Rightarrow C \text{ does not straddle } S. \quad (9)$$

One last issue remains to be addressed: how is Δ established? In practice, there is no need to compute a fixed value for Δ, but merely to ensure that it is large enough, i.e., that the largest cell radius used in the polygonization process is never greater than Δ. This is accomplished by using a sufficiently large ratio between d and w (see Fig. 4). Nonetheless, too large a ratio will have a detrimental effect on the algorithm performance. To see this, notice that in this case, $f(\mathbf{x})$ will return too low an estimate for $\delta(\mathbf{x}, S)$ in many cases, thus requiring many cells to be subdivided needlessly. We conducted several experiments (see Section 5) that indicate that $d = 4w/3$ is usually a large enough ratio without incurring in heavy performance penalties.

The space sampling algorithm used in our implementation is summarized in the following pseudo-code:

procedure *SpaceSample*(f, C, *maxlevel*, *level*)

1. **If** (*level* == *maxlevel*) **then**
   (a) Evaluate f at those corner points of C that were not yet evaluated;
   (b) *MCcellPolygonize*(C);
2. **Else If** (*Straddle*(f, C)) **then**
   (a) Subdivide C into 8 sub-cubes C_i of equal size;
   (b) **For** $i = 0..7$ **do**
   SpaceSample(f, C_i, *maxlevel*, *level*+1);

Function *Straddle* performs the rejection test (9). It returns true if C straddles the surface $f(\mathbf{x}) = 0$ and false otherwise. Unit (leaf) cubes correspond to the maximum subdivision level *maxlevel*. These are passed to function *MCcellPolygonize*, which performs the cell triangulation process [18]. The rejection test on these cubes is implicitly performed by the Marching Cubes standard procedure, i.e., by examining the signs of the function at the cube's corners.

A crucial consideration when implementing this algorithm is to avoid reevaluating the implicit function at points already visited. For instance, the "if" clause in step 2 evaluates the function at the center of cube C; this value may later correspond to a cube corner in step $1.(a)$. Our implementation employs a cache of evaluated points so that f is computed only once for each space point.

5 Experiments

The experiments were performed on a PC equipped with an AMD-Duron processor running at 1.3 GHz and 256 MB of main memory.

All models were simplified down to 800 vertices and placed inside a cubical space with (-1,-1,-1) and (1,1,1) as minimum and maximum points, respectively.

In our implementation the cubical space is uniformly divided into a 3D grid of identical cubes whose size is $s \times s \times s$, where s is a power of two.

We first conducted experiments aimed at finding "optimal" values for the maximum cell radius r and for the w/d ratio. For this purpose, we performed the polygonization of the "Stanford Bunny" data set using six values of w/d (0.2, 0.5, 0.667, 0.75, 0.8 and 0.83) and six values for r. Since we use an octree-like decomposition scheme, the six values of r correspond to the octant radii of consecutive refinement levels of the initial cubic space, i.e., $\sqrt{3}/2$, $\sqrt{3}/4$, and so forth. In Table 1 we observe the direct relation between different values of r and w/d . A small w/d ratio implies a large Δ size, making it possible to use large initial sizes for r while still yielding a correct polygonization. However, large Δ's have a detrimental effect on the algorithm performance.

Figure 4 illustrates how the value of Δ depends on the w/d ratio. Recall that Δ denotes the "size" of the region where f can be used as a lower bound for the Euclidean distance.

Next, we performed other experiments to test the efficiency of the algorithm on the polygonization of some reconstruction functions. See results in Figure 5 and on

	$r = \frac{\sqrt{3}}{2}$	$r = \frac{\sqrt{3}}{4}$	$r = \frac{\sqrt{3}}{8}$	$r = \frac{\sqrt{3}}{16}$	$r = \frac{\sqrt{3}}{32}$	$r = \frac{\sqrt{3}}{64}$
$\frac{w}{d}=\frac{1}{5}$	26.38% ok	26.38% ok	26.38% ok	26.36% ok	26.68% ok	32.78% ok
$\frac{w}{d}=\frac{1}{2}$	12.36% ok	12.36% ok	12.36% ok	12.41% ok	13.26% ok	21.94% ok
$\frac{w}{d}=\frac{2}{3}$	9.89% ok	9.89% ok	9.89% ok	9.97% ok	10.93% ok	20.09% ok
$\frac{w}{d}=\frac{3}{4}$	9.01% ok	9.01% ok	9.01% ok	9.10% ok	10.10% ok	19.44% ok
$\frac{w}{d}=\frac{4}{5}$	8.57% hole	8.57% hole	8.58% hole	8.67% hole	9.69% ok	19.12% ok
$\frac{w}{d}=\frac{5}{6}$	8.29% hole	8.29% hole	8.30% hole	8.39% hole	9.42% hole	18.90% ok

Table 1: This table shows results of the proposed algorithm in polygonizing the "Stanford Bunny" reconstruction function modeled with different values of w/d. d was set to 0.015 in all tests. The percentage values correspond to the total number of function evaluations required by the algorithm compared with the number of evaluations required by a standard Marching Cubes. A "hole" value means that resulting polygonization is incorrect, i.e., the algorithm rejected one or more cells straddling the surface.

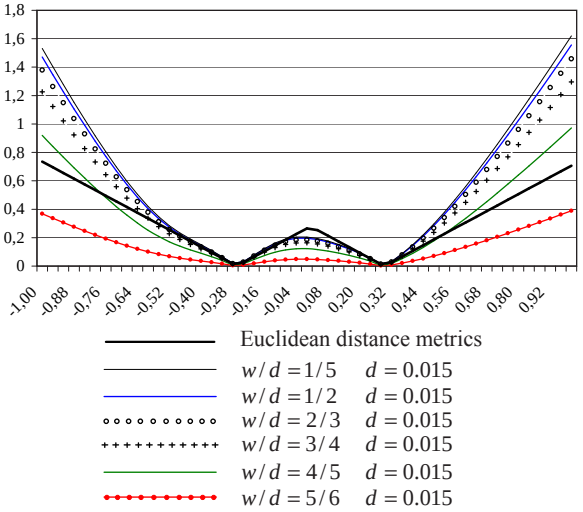

Figure 4: Plot of the various interpolating functions used in the experiments (see Table 1), evaluated on the line segment defined by (-1,-1,-1) and (1,1,1).

the left side of Table 2. All functions were built using 800-vertex polygonal meshes as input data and setting $d = 0.015$ and $w/d = 3/4$. The LU decomposition for solving the equations system took 42 s. in all experiments. The initial cube was initially subdivided in $128 \times 128 \times 128$ cells.

Finally, some experiments were performed to compare the proposed algorithm with the Bloomenthal's continuation implementation [5]. In order to obtain a "fair" comparison, we had to modify Bloomenthal's implementation so that its vertex computing function, which ordinarily uses binary subdivision, would employ linear interpolation instead. Moreover, the algorithm was configured to use cubical decomposition instead of the default tetrahedral decomposition. Also, a vertex of the polygonal mesh is used as the seed point. In both implementations, the cube sizes correspond to a $128 \times 128 \times 128$ decomposition of the world space. The comparison results are shown in Table 2.

The experiments indicate that the proposed technique tends to perform slightly faster than Bloomenthal's implementation. As one might expect, the number of triangles for both methods are almost identical. The slight variation is due to the fact that the cubical decompositions are not the same, since the "seed" point used in Bloomenthal's code determines the decomposition's origin. The improved performance can be attributed to the smaller number of function evaluations performed by the proposed technique.

We also tested the generated polygonizations for topological consistency with positive results. It should be noted, however, that the proposed technique is based on the Marching Cubes [18] algorithm and thus it inherits all its disadvantages such as the creation of an excessively large number of triangles and the introduction of some ambiguities in lower resolutions.

6 Conclusions and future work

We have presented a technique for the fast polygonization of variational implicit surfaces. This technique minimizes the number of the costly function evaluations using an hierarchical sampling of the space and thus permitting an efficient application of the Marching Cubes algorithm.

The proposed technique can also be applied to the polygonization of other classes of implicit objects, as long as their generating functions behave as lower bounds for Euclidean distance metrics within a given neighborhood (see the discussion in Section 4.3). One important observation is that this condition is considerably weaker than the well-known Lipschitz criteria exclusion [13].

Additionally, it should be possible to adapt the ideas presented in this paper to other visualization techniques. For instance, ray tracing of implicit surfaces could be efficiently performed by coupling the proposed voxel rejection criteria with standard octree-based speed-up techniques.

To conclude, we would like to explore this technique in the context of FastRBF [7], adaptive polygonization scheme and the support of sharp features.

Acknowledgments

Many thanks are due to Dr. Marco Aurélio P. Cabral for his helpful insights. We are also grateful to CNPq for providing financial support for the first author.

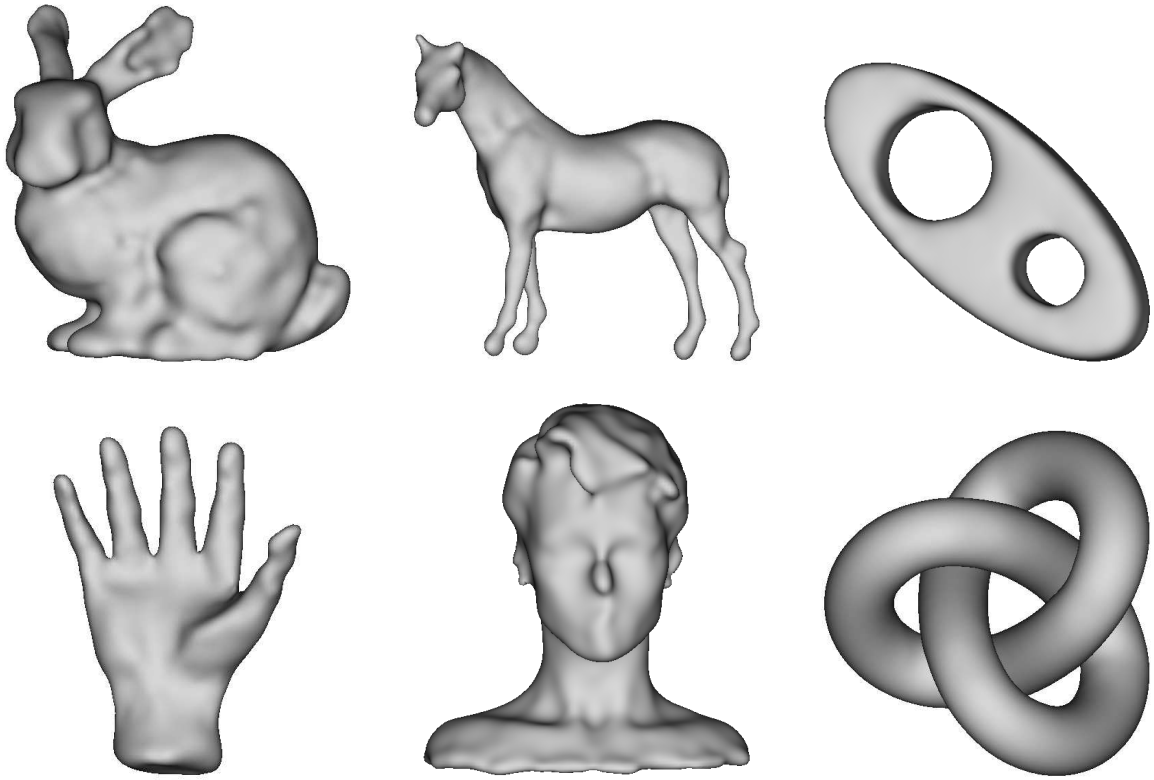

Figure 5: Different models polygonized with the proposed technique.

References

[1] C. Bajaj, J. Blinn, J. Bloomenthal, M. Cani-Gascuel, A. Rockwood, B. Wyvill, and G. Wyvill. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers, INC., San Francisco, California, 1997.

[2] R. K. Beatson, J. B. Cherrie, and D. L. Ragozin. Fast evaluation of radial basis functions: Methods for four-dimensional polyharmonic splines. *SIAM J. Math. Anal.*, 32(6):1272–1310, 2001.

[3] R. K. Beatson and W. A. Light. Fast evaluation of radial basis functions: Methods for two-dimensional polyharmonic splines. *IMA Journal of Numerical Analysis*, (17):343–372, 1997.

[4] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, pages 341–335, November 1988.

[5] J. Bloomenthal. An implicit surface polygonizer. *Graphics Gems IV*, pages 324–349, 1994.

[6] M. D. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge University Press, 2003.

[7] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In *Proceedings of SIGGRAPH 2001*, pages 67–76. ACM Press, 2001.

[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 *Proceedings of ACM Graphite 2003*, pages 119–126. ACM Press, 2003.

[9] B. Crespin. Dynamic triangulation of variational implicit surfaces using incremental delaunay tetrahedralization. In *Proceedings of the 2002 IEEE Symposium on Volume Visualization and Graphics*, pages 73–80, Piscataway, NJ, 28–29 2002. IEEE.

[10] H. Q. Dinh, G. Turk, and G. Slabaugh. Reconstructing surfaces using anisotropic basis functions. In *Proceedings of the Eighth International Conference On Computer Vision*, pages 606–613, Los Alamitos, CA, July 2001. IEEE Computer Society.

[11] H. Q. Dinh, G. Turk, and G. Slabaugh. Reconstructing surfaces by volumetric regularization using radial basis functions. In *IEEE Transactions on Pattern Analy-*

Model	Constraint points (number)	The proposed algorithm				Bloomenthal's algorithm		
		Iso-surface extraction	Evaluations (number)	Triangles (number)	Evaluations (%)	Iso-surface extraction	Evaluations (number)	Triangles (number)
Bunny	1600	58s	193395	81340	9.01%	75s	244171	81420
Horse	1600	34s	111560	47028	5.20%	43s	141128	47088
Torus2	1600	27s	90654	33932	4.22%	31s	102867	34304
Hand	1599	26s	83202	34660	3.87%	32s	104085	34720
Head	1600	53s	176270	71528	8.21%	65s	214441	71508
Knot	1600	81s	269701	114704	12.56%	104s	343725	114592

Table 2: Comparison between the Bloomenthal's continuation algorithm and the proposed technique.

sis and Machine Intelligence, pages 1358–1371. IEEE Computer Society, October 2002.

[12] X. Jim, H. Sun, and Q. Peng. Subdivision interpolating implicit surfaces. *Computer Graphics*, pages 763–772, October 2003.

[13] D. Kalra and A. Barr. Guaranteed ray intersections with implicit surfaces. *Computer Graphics*, pages 297–306, July 1989.

[14] O. Karpenko, J. F. Hughes, and R. Raskar. Free-form sketching with variational implicit surfaces. *Computer Graphics Forum*, September 2002.

[15] N. Kojekine. *Computer Graphics and Computer Aided Geometric Design by means of Compactly Supported Radial Basis Functions*. PhD thesis, Tokyo Institute of Technology, 2003.

[16] H. Laga, R. Piperakis, H. Takahashi, and M. Nakajima. A radial basis function based approach for 3d object modeling and reconstruction. In *IWAIT2003*, pages 139–144, 2003.

[17] S. K. Lodha and R. Franke. Scattered data techniques for surfaces. In G. Nielson H. Hagen and F. Post, editors, *Proceedings of Dagstuhl Conference on Scientific Visualization*, pages 182–222, Dagstuhl-Germany, June 1999. IEEE Computer Society Press.

[18] W. Lorensen and H. Cline. Marching cubes: a high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987.

[19] S. V. Matveyev. Approximation of isosurface in the marching cube: Ambiguity problem. In *Proceedings IEEE Visualization*, pages 288–292. IEEE Computer Society, October 1994.

[20] B. S. Morse, T. S. Yoo, P. Rheingans, D. T. Chen, and K. R. Subramanian. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *Shape Modeling International*, pages 89–98, Genova, Italy, May 2001.

[21] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.P. Seidel. Multi-level partition of unity implicits. *ACM Transactions on Graphics*, 22(3):463–470, July 2003.

[22] P. Reuter, I. Tobor, C. Schlick, and S. Dedieu. Point-based modelling and rendering using radial basis functions. In *Proceedings of ACM Graphite*, pages 111–118, Held in Melbourne, Australia, 2003. ACM Press.

[23] V. Savchenko, A. Pasko, O. Okunev, and T. Kunii. Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum*, 14(4):181–188, 1995.

[24] I. Tobor, P. Reuter, and C. Schlick. Efficient reconstruction of large scattered geometric datasets using the partition of unity and radial basis functions. *Journal of WSCG*, 12(1-3):467–474, February 2004.

[25] G. M. Treece, R. W. Prager, and A. H. Gee. Regularised marching tetrahedra: improved iso-surface extraction. *Computers and Graphics*, 23(4):583–598, 1999.

[26] G. Turk and J. O'Brien. Variational implicit surfaces. Technical report, Georgia Institute of Technology, May 1999.

[27] G. Turk and J. O'Brien. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics*, pages 855 – 873, October 2002.

[28] L. Velho, J. Gomes, and L. H. Figueiredo. *Implicit Objects in Computer Graphics*. Springer Verlag, New York, 2002.

[29] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial basis functions of minimal degree. *AICM*, (4):389–396, 1995.

[30] G. Yngve and G. Turk. Robust creation of implicit surfaces from polygonal meshes. In *IEEE Transactions on Visualization and Computer Graphics*, pages 346–359. IEEE, 2002.