

# High-Quality Extraction of Isosurfaces from Regular and Irregular Grids

John Schreiner, Carlos E. Scheidegger, and Cláudio T. Silva, *Member, IEEE*

**Abstract**—Isosurfaces are ubiquitous in many fields, including visualization, graphics, and vision. They are often the main computational component of important processing pipelines (e.g., surface reconstruction), and are heavily used in practice. The classical approach to compute isosurfaces is to apply the Marching Cubes algorithm, which although robust and simple to implement, generates surfaces that require additional processing steps to improve triangle quality and mesh size. An important issue is that in some cases, the surfaces generated by Marching Cubes are irreparably damaged, and important details are lost which can not be recovered by subsequent processing. The main motivation of this work is to develop a technique capable of constructing high-quality and high-fidelity isosurfaces.

We propose a new advancing front technique that is capable of creating high-quality isosurfaces from regular and irregular volumetric datasets. Our work extends the *guidance field* framework of Schreiner et al. to implicit surfaces, and improves it in significant ways. In particular, we describe a set of sampling conditions that guarantee that surface features will be captured by the algorithm. We also describe an efficient technique to compute a minimal guidance field, which greatly improves performance. Our experimental results show that our technique can generate high-quality meshes from complex datasets.

**Index Terms**—Isosurface Extraction, Curvature, Advancing Front

## 1 INTRODUCTION

Implicit surfaces are a powerful and effective technique with a wide range of applications in many scientific areas. In particular, they are used in many modeling and geometric tasks in scientific visualization and computer graphics. An *isosurface*  $\mathcal{S}_a$  is defined as the *preimage* of a function  $f: \mathbf{R}^n \rightarrow \mathbf{R}$  and value  $a$ ; it is the set of points in the domain that map to  $a$ , i.e.  $\mathcal{S}_a = \{x \in \mathbf{R}^n : f(x) = a\}$  [17]. We then say  $f$  is the *implicit function* and  $a$  is the *isovalue*. Part of the popularity of implicit surfaces comes from their representation power, solid theoretical foundation, and relative ease of manipulation [9]. Their acceptance has partly been driven by the number of successful techniques for their computation and rendering. Following the discovery of Marching Cubes algorithm [33], the last two decades have seen a constant stream of work on effective techniques for the computation and visualization of isosurfaces (see, e.g., [6, 16, 32, 39, 49]).

Although a number of techniques have been proposed to solve the isosurface generation problem, the Marching Cubes (MC) algorithm [33, 39] forms the basis for many widely used methods. MC works by sampling the function  $f$  at a grid of fixed resolution, and uses a table of possible configurations of range signs to create a triangulated surface out of those samples. The main strengths of MC are its generality, simplicity and robustness, which have made it one of the most used meshing algorithms in practice (e.g., 3D photography [29]). The main problem with MC is the inherent bias caused by only placing vertices on edges which intersect the surface. This also implies that the sampling density is proportional to the grid resolution, and not to any intrinsic surface properties. Meshes generated by MC (and variants) are typically over-tessellated, and contain many bad triangles (1). The low quality makes them unsuitable for further geometry processing, and typically they require some form of post-processing (e.g., smoothing) before being used in applications.

There are a number of ways to improve the quality of MC meshes. One approach is to apply a remeshing technique [3] to optimize sampling, grading, size, and shape of elements, while keeping the overall

geometry the same. This can be challenging, since MC meshes are often large, complex, and contain many geometric degeneracies. To make the meshes simpler, and overall of better quality, a geometric simplification (or at least a smoothing) step is applied to them [21, 30]. We note that this is currently the preferred way to deal with MC meshes. Existing simplification techniques are very robust and scalable, and they are able to simplify even the largest isosurfaces. One issue with this approach is that the use of multiple steps makes it hard to control the approximation accuracy throughout the process.

Another way to solve this problem is to develop a direct technique for isosurface extraction which avoids the intermediate step of generating a MC mesh. One such example is the algorithm of Wood et al. [51], which first creates a coarse mesh with the correct topology and iteratively refines it with a force based solver. This produces an adaptively sampled mesh, but the transition regions are created with right triangles which leads to abrupt changes in the triangle sizes. An alternative approach is the advancing front paradigm, whose basic idea is to tile the surface one triangle at a time by performing a sequence of local decisions [7, 26]. Our work fits in this category. In particular, we build upon a recently proposed class of advancing front methods by Schreiner et al. [45] to develop a new general technique for isosurface generation.

The main contributions of our paper are:

- A new technique for extracting high-quality triangulated meshes of isosurfaces defined from functions sampled on regular and irregular grids. Our algorithm is efficient and guarantees bounded distance from the isosurface while being adaptive to curvature.
- An extension of the theoretical foundations of the guidance field approach of [45] that shows the conditions under which the reconstructed triangulated surface accurately reflects the implicit surface.
- An efficient way to cull redundant information from the guidance field, improving memory usage by a factor of ten or more, and speed by up to a factor of five.
- A set of detailed experimental results on both regular and unstructured grids that shows the effectiveness of our approach.

## 2 PREVIOUS WORK

Since the introduction of the classical Marching Cubes algorithm [33], there has been an immense amount of work in the polygonization of implicit surfaces. The original algorithm contained ambiguities, where

• John Schreiner, Carlos E. Scheidegger and Cláudio T. Silva are with the SCI Institute at the University of Utah, E-mail: {jmschrei, cscheid, csilva}@sci.utah.edu.

Manuscript received 31 March 2006; accepted 1 August 2006; posted online 6 November 2006.

For information on obtaining reprints of this article, please send e-mail to: [tcvg@computer.org](mailto:tcvg@computer.org).

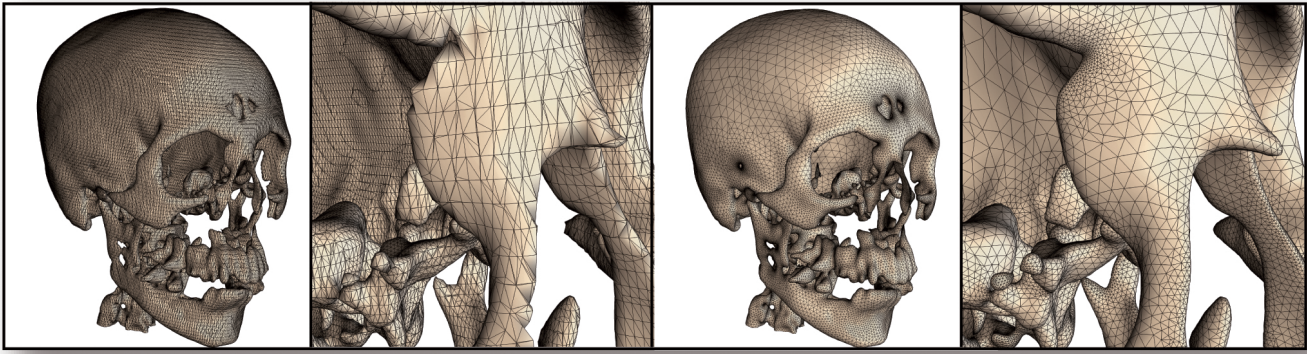


Fig. 1. Triangle meshes generated from Marching Cubes have inherently biased sampling, which produces badly shaped triangles. Our advancing front algorithm ensures appropriate mesh grading and curvature adaptation, and generates triangle meshes with excellent triangle shape. The triangulation produced by Marching Cubes is shown on the left side, and our technique is shown on the right side.

some configurations could be interpreted in more than one way, possibly generating non-manifold meshes. Nielson and Hamann's asymptotic decider [39] was among the first to address the topic using bilinear interpolation to resolve face ambiguities, and now the issue of ambiguity in MC has been thoroughly examined [14, 35, 37]. There are many variants of MC. For instance, the Marching Tetrahedra (MT) [40, 47] class of algorithms work well for tetrahedral meshes, and does not have ambiguous cases. Dual methods generate the topological dual of the MC surfaces, and reproduce sharp features by extrapolating the sampled normals [25].

Much effort has been put into making MC more efficient, with particular emphasis on avoiding empty cells [15, 31, 49]. There has also been work on generating high-quality triangulated isosurfaces. Physically based simulation offers a way of computing a good grid in which to perform the MC tests [20], and using repulsion forces to sample a surface to be triangulated [48]. Hybrid approaches are also possible. For example, Wood et al. [51] first create a coarse mesh with topology equivalent to the MC mesh, then use a physically based repulsion solver to semi-regularly tile the surface.

Curvature is an important geometric concept that is useful in processing isosurfaces. Kindlmann et al. [27] introduced a framework for computing differential properties of a volume through carefully constructed spatial filters. We use their formulation for curvatures of a level set to guarantee an accurate reconstruction of the isosurface. Witkin and Heckbert [50], and later Meyer et al. [34] use curvature information to appropriately sample implicit surfaces with particles. Crossno and Angel combine curvature information with physically based simulation to sample implicit surfaces with particle systems [18]. However, when particles are used as triangle vertices, triangles tend to be poorly shaped wherever there are abrupt changes in curvature. Curvature has also been essential in the development of advancing front algorithms. Scheidegger et al. [44] introduced an advancing front algorithm for triangulating point set surfaces which made use of a *guidance field* to both provide curvature adaptability and restrict the change in triangle sizes. Schreiner et al. [45] (reviewed in the next section) builds a solid theoretical advancing front framework on top of these initial ideas. In this paper, we provide a condition which guarantees that the entire surface is accurately triangulated. We also describe a procedure to build a minimal guidance field that contains no irrelevant samples, making the triangulation more efficient.

### 3 AN ADVANCING FRONT FRAMEWORK FOR MESHING IMPLICIT SURFACES

Advancing front (also referred to as surface tracking and continuation) algorithms are a class of surface meshing algorithms that begin with a *seed* point, and grow the triangulation across the surface. The *front* is the boundary of the mesh that has already been generated. It is *advanced* by choosing an edge in the front and creating a new triangle attached to it. If the new triangle causes a front to intersect itself, it is

split into two separate fronts. If it intersects a different front, the two are merged. There has been a substantial amount of previous work on using advancing front techniques for mesh generation [7, 46], and in particular, for implicit surface triangulation [8, 23, 24].

The meshing research community has for a long time used advancing fronts for generating high-quality meshes suitable for numerical simulation [41]. In that context, a meshing algorithm starts with well-defined domain representations and creates a piecewise linear representation, progressively increasing the dimension of the primitive: edges, triangles, and then tetrahedra. The essential difference in our setting is the absence of a well-defined description of the components of the volume. We assume a much more general model — any surface defined as an isosurface whose gradient and Hessian is non-zero everywhere. In this case, it would be a challenge in itself to find *any* charts for the reconstructed surface, let alone one that is suitable for use in meshing. Some works have tried to create a mesh under more general conditions. To the best of our knowledge, these seem limited to *ad hoc* steps like smoothing a mesh generated from preclassified images [12]. Some other problems we discuss have previously been noticed in that community as well. Borouchaki et al. discuss methods for achieving gradation control: enforcing that the length ratio of two consecutive edges is bounded [10]. As section 3.1 points out, we achieve gradation by careful construction of the guidance field.

In order to accurately mesh a surface, triangle sizing should be inversely proportional to the maximum curvature of the surface. This adaptivity to the local surface also prevents the output mesh from becoming unreasonably dense. Most early work did not adapt triangle size to geometric features, and thus depended on an *a priori* compromise between the size of the resulting mesh and the representation of small features of the isosurface. These algorithms were quickly followed by adaptive variants [1, 13, 26]. While these help reduce the size of the resulting mesh, they still have several drawbacks. These algorithms essentially replace the assumption of maximum curvature (embodied by the predetermined triangle size) by an assumption of *small curvature changes* along the surface. Since these algorithms sample curvature locally, it is still possible to step over important geometrical and topological features of the isosurface. Because of these limitations, previous advancing front algorithms relied on heuristics for checking front interference [26].

In itself, adapting the surface to curvature introduces another problem: sharp changes in curvature across the surface lead to sharp changes in triangle size and poorly shaped triangles [10]. Additionally, robust front interference tests are highly non-trivial. This is important, since the test is critical to the correct behavior of the algorithm. If the test fails to detect front interference, there will be two or more sheets of triangles over portions of the surface. The following figure from [44] shows a common geometric situation where usual adaptive advancing front algorithms fail to reconstruct the surface correctly. The inset at the left shows the consequences of determining

triangle sizes completely locally, while the right inset shows the desired correct behavior: triangles need to be made smaller *before* they reach the geometric feature.



The advancing front algorithm of Schreiner *et al.* [45] solves these issues robustly, by introducing a solid theoretical foundation for advancing front algorithms. In their algorithm, they use a *guidance field* to help edge and triangle size selection. The guidance field is a scalar function  $g$  defined on the surface, which determines how long the edges incident to a point may be. They show how to construct  $g$  in such a way that it has two important properties. First,  $g(x)$  is bounded above by  $2\sin(\rho)/\kappa_{\max}$ , where  $\kappa_{\max}$  is the maximum absolute curvature of the surface at point  $x$ , and  $\rho$  is a user defined parameter. This ensures that all of the features in the surface are captured accurately to the degree defined by  $\rho$ , independently of their scale. Second, the gradient magnitude of  $g$  is never greater than  $1 - \eta^{-1}$ , where  $\eta$  is another user defined parameter. This prevents the triangle sizes from changing too rapidly and poorly shaped triangles from being formed. They also show that their guidance field construction can be used to bound both the aspect ratio of most of the triangles, and the Hausdorff distance between the original surface and the output mesh. This in turn can be used to create a front intersection test procedure which can be proven to be robust.

Their algorithm addresses all of the major issues hindering advancing front algorithms, but their robustness arguments depend critically on two assumptions. First, they assume that the surface being triangulated is smooth (even though it is certainly not the case for remeshing applications, the focus of their work). For the case of implicit surfaces, we need only assume that the gradient of the implicit function is defined and non-zero at all points in the surface. This is a reasonable assumption to make. Second, they assume that the guidance field has been sufficiently sampled so that the first property of  $g$  will hold. In their work, they simply sample the mesh densely, but have no means to check for sufficiency. Features smaller than the sampling density can easily be missed by the triangulation. In Section 4.1 we present a sampling condition which can be used to determine how many samples are truly required. Additionally, there typically will be many more samples in the guidance field than are actually required, leading to increased memory usage and execution times. In Section 4.2 we present a procedure for culling out unnecessary samples, effectively computing a guidance field of minimal size. The following pseudocode is a summary of our algorithm:

```

TRIANGULATE( $f, \nabla f, H(f), k, \rho, \eta$ )
1  let  $\mathcal{S}$  be defined by all  $x$  such that  $f(x) = k$ 
2   $g \leftarrow$  GENERATE-SAMPLES( $\mathcal{S}, \nabla f, H(f)$ ) (see Section 4.1)
3  CULL( $g, \rho, \eta$ ) (see Section 4.2)
4   $Active \leftarrow$  SEED-FRONT( $\mathcal{S}$ ) (see Section 3.2)
5  while  $|Active| > 0$ 
6    do  $front \leftarrow$  GET-ANY-FRONT( $Active$ )
7    if OK-TO-ADD-TRIANGLE( $front, \rho, \eta$ )
8      then ADD-TRIANGLE-TO-FRONT( $front, \rho, \eta$ )
9    else  $other \leftarrow$  GET-INTERFERING-FRONT( $front$ )
10     if  $other = front$ 
11       then ( $f1, f2$ )  $\leftarrow$  SPLIT( $front$ )
12        REMOVE-FRONT( $Active, \{front\}$ )
13        ADD-FRONT( $Active, \{f1, f2\}$ )
14     else  $new-front \leftarrow$  MERGE( $front, other$ )
15        REMOVE-FRONT( $Active, \{front, other\}$ )
16        ADD-FRONT( $Active, \{new-front\}$ )

```

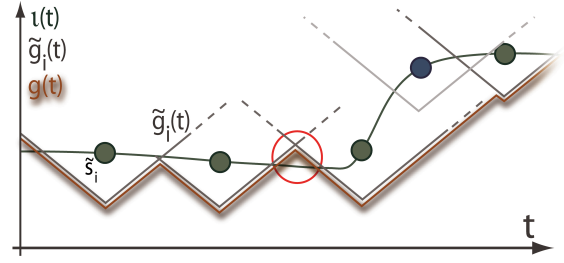


Fig. 2. The guidance field  $g(t)$  on a parametric curve  $C = f(t) : \mathbf{R} \rightarrow \mathcal{S}$ . Note that the functions are plotted against the parameter  $t$ , not the curve itself.  $g(C)$  is the lower envelope of all  $\tilde{g}$ . At the sample points  $\tilde{s}_i$ ,  $\tilde{g}_i$  is minimum, and it grows linearly as the distance from  $\tilde{s}_i$  increases. Note that if the sampling is too coarse,  $g(C)$  might not bound  $u(C)$  (as shown in the region inside the red circle). Section 4.1 shows how to provably prevent this. Finally, some samples will not influence  $g(C)$  (in the figure, the sample shown in blue). Section 4.2 shows how to efficiently remove these points.

### 3.1 Guidance field construction

Due to the central importance of the guidance field, we summarize the construction method of Schreiner *et al.* [45]. We start defining a function which gives the *ideal* edge length at a point  $s$  on the surface:

$$\iota(s) = \frac{2\sin(\rho/2)}{\kappa_{\max}(s)}$$

This function essentially determines the ideal size to be the length that subtends a constant angle  $\rho$  on the minimum osculating circle of the surface at  $x$ . Amenta and Bern [4] show that the *local feature size* (LFS) bounds the curvature by below.  $\iota(x)$  could be defined in terms of the LFS, and every property of the guidance field will still hold. We use curvature because computing the medial axis is much costlier. We want edge lengths to smoothly grade as triangles come closer to  $s$ . To achieve this, we create, for each point  $s$  in the isosurface, another function, induced by  $\iota(s)$ . This function is defined over the entire embedding space of the isosurface:

$$\tilde{g}_s(x) = (1 - \eta^{-1}) \cdot |x - s| + \eta^{-1} \cdot \iota(s)$$

This is just a cone centered at  $s$ , with slope  $1 - \eta^{-1}$ . If  $s$  was the only point sampled from the surface,  $\tilde{g}_s$  would tell us exactly how large an edge we could create at any other point on the surface. Note that  $\tilde{g}_s$  is conservative:  $\tilde{g}_s(s) < \iota(s)$ . This will be important for our sampling condition described in Section 4.1. Since we want all the constraints to be satisfied, we define the guidance field  $g$  to be the lower envelope of all of the  $\tilde{g}_s$  induced at each point on the surface:

$$g(x) = \min_{s \in \mathcal{S}} \tilde{g}_s(x)$$

Figure 2 illustrates  $g(x)$  for a low-dimensional case. We restrict ourselves to isotropic reconstruction: triangle edge lengths are independent of their orientation on the surface. The algorithm behavior depends on only two parameters,  $\rho$  and  $\eta$ , which control respectively the reconstruction accuracy and well-shapedness of triangles.  $\rho$  controls the approximation error [44], and  $\eta$  determines the rate of *triangle expansion*: every two adjacent edges  $e_1$  and  $e_2$  such that  $|e_1| > |e_2|$  should respect  $|e_1|/|e_2| \leq \eta$ . There are some noteworthy features of the constructed guidance field. As shown in [45],  $g(x)$  is Lipschitz-continuous, and so triangles will have good grading [43]. Additionally, its exact value can be evaluated efficiently by *kd-tree* queries, requiring no splatting of values in a regular grid or other, similar strategies [2].

### 3.2 Application to isosurfaces

Kindlmann *et al.* [27] show how to compute the geometry tensor  $G$  for implicit surfaces.  $G$  encodes all curvature information, and so we use



Silicium	# samples	# remaining	before	after
$\rho = 0.5$	818058	11190	1m28s	0m50s
$\rho = 0.3$	818058	28522	1m45s	1m08s
$\rho = 0.2$	818058	61443	2m07s	1m36s
Skull	# samples	# remaining	before	after
$\rho = 0.5$	12551234	50538	31m01	5m46s
$\rho = 0.3$	12551234	108931	27m05	7m27s
$\rho = 0.2$	12551234	198378	26m13	9m52s

Table 1. Sample of results of guidance field culling. “before” refers to running time without culling, and “after” refers to running time after culling.

it to compute  $\kappa_{\max}$ . We assume the isosurface does not pass through critical points of  $f$ , and we denote  $n = (\nabla f)/|\nabla f|$ . Then,

$$\begin{aligned}
 P &= I - nn^T \\
 H &= \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial y \partial z} \\ \frac{\partial^2 f}{\partial x \partial z} & \frac{\partial^2 f}{\partial y \partial z} & \frac{\partial^2 f}{\partial z^2} \end{bmatrix} \\
 G &= PHP/|\nabla f|
 \end{aligned}$$

$\kappa_{\max}$  is given by the spectral radius of  $G$ , which is easily computable from matrix invariants. From  $\kappa_{\max}$  evaluated at  $s$ , we compute  $\iota(s)$ , which determines a sample on the guidance field  $\tilde{g}_s$ .

Advancing front algorithms must start from a *seed point*. In these algorithms, there can only be as many connected components as there are seed points, so these are an important part of the process. There are efficient algorithms for computing these seed points, e.g., [6]. These points are then projected on the surface using gradient descent.

We sample the surface curvature by generating a jittered random set of sample points inside cells that neighbor the surface and then project each of them on the surface, again through gradient descent. Section 4.1 describes how to determine the number of points that are necessary to guarantee correct reproduction of features. Finally, each new triangle is created by determining the correct size as described in [45]. The point is projected on the surface by using the dihedral angle of the edge as the free parameter, and searching for the isovalue similarly to the technique described in [13].

#### 4 GUIDANCE FIELD ENHANCEMENTS

In this section, we describe two contributions to the theory of advancing front algorithms. The first one shows for the first time how to guarantee that all features of the surface will be represented in the triangle mesh, by determining an upper bound on the curvature of the isosurface. This gives a condition for sufficiency of the guidance field. The second contribution significantly improves the efficiency of the guidance field evaluation. We show that most samples do not affect the guidance field in any way, and we give an algorithm to compute the *minimal* set of samples that are required to define  $g$ .

##### 4.1 Creating a sufficient guidance field

The guidance field described in the previous section offers an ideal way to control the mesh gradation of the reconstruction. The fundamental problem, though, is how to determine whether a surface has been appropriately sampled. The guidance field is defined to be bounded above by  $\iota$  over all of the points on the mesh. However, with an arbitrarily sampled guidance field, this may not be the case as can be seen in Figure 2. The consequence of  $\iota$  not bounding  $g$  is that there will be points on the surface where the edge length created by the advancing front will be larger than the curvature at that point allows. Fortunately, since  $\tilde{g}_s(s)$  conservatively bounds  $\iota(s)$ , an infinite sampling density is not required to bound of  $\iota$  on  $g$ . In this section, we show how a sufficient sampling density can be computed. We call a guidance field *sufficient* when it is less than  $\iota$  at every point on the surface.

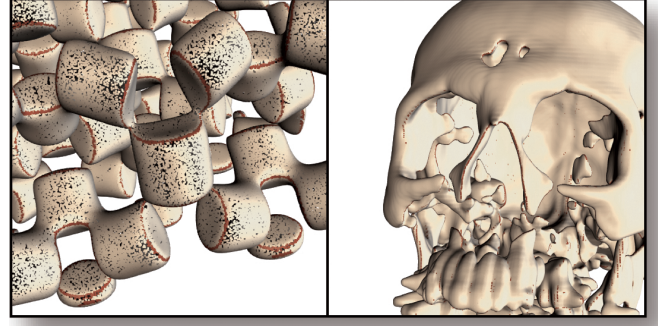


Fig. 3. Minimal guidance fields. The two figures are rendered using only the points in the guidance field — the black “regions” are simply the inside of the surface without any illumination. After trimming, all the tan-colored points are removed from the data structures, leaving only the red points for querying.

We begin by taking note of two aspects of the guidance field. First, we note that it can be made more conservative by simply reducing the radius of curvature for a sample point when it is inserted into the guidance field. This will never cause a sufficient guidance field to become insufficient, but it may cause an insufficient guidance field to become sufficient. Second, we note that given a guidance field sample  $s$ , and another point on the surface  $x$  where  $|x - s| \leq \iota(s)$ , then  $\tilde{g}_s(x) \leq \iota(s)$ . Assume that we can compute the minimum  $\iota$  over all of the points in the surface, and call it  $\lfloor \iota \rfloor$ . We can now redefine the guidance field samples to be in terms of  $\lfloor \iota \rfloor$ . This has the effect of making the guidance field more conservative, as in the first note above. It is now clear that the guidance field is sufficient if there are no points  $x$  in the surface such that there are no guidance field samples  $s$  with  $|x - s| \leq \lfloor \iota \rfloor$ . Stated conversely, the guidance field might not be sufficient if there exists a point  $x$  such that  $|x - s| > \lfloor \iota \rfloor$  for all guidance field samples  $s$ .

The final remaining issue is how to compute  $\lfloor \iota \rfloor$ . Since  $\iota$  is inversely proportional to  $\kappa_{\max}$ , this means that we must find an upper bound on  $\kappa_{\max}$ . In order to compute this bound, recall that  $\kappa_{\max}$  is defined as the absolute value of the largest eigenvalue of the geometry tensor  $G$ . Notice that this is exactly the spectral radius of  $G$ :  $\kappa_{\max} = r(G)$ . We will now bound the spectral radius of  $G$ . First, recall that the spectral radius is bounded by any consistent matrix norm:

$$r(M) \leq \|M\|$$

Recall, also, the *submultiplicative* property of matrix norms:

$$\|A \cdot B\| \leq \|A\| \cdot \|B\| \quad (1)$$

We use the Frobenius norm of a matrix,  $\|M\|^2 = \sum_{i,j} m_{ij}^2$ . We bound  $\|G\|$  by directly applying Equation (1) to the  $\kappa_{\max}$  definition.

$$\begin{aligned}
 r(G) &= \kappa_{\max} \\
 &\leq \|G\| \\
 &\leq \|PHP/|\nabla f|\|
 \end{aligned}$$

$$r(G) \leq \|P\| \cdot \|H\| \cdot \|P\| \cdot \|I/|\nabla f|\| \quad (2)$$

We denote the normal  $n = \nabla f/|\nabla f| = [n_x \ n_y \ n_z]^T$ , and we compute  $\|P\|$  by summing the diagonal terms and then the off-diagonal ones:

$$\begin{aligned}
 \|P\|^2 &= (1 - n_x^2)^2 + (1 - n_y^2)^2 + (1 - n_z^2)^2 + \\
 &\quad 2n_x^2 n_y^2 + 2n_y^2 n_z^2 + 2n_x^2 n_z^2 \\
 \|P\|^2 &= 3 - 2(n_x^2 + n_y^2 + n_z^2) + n_x^4 + n_y^4 + n_z^4 + \\
 &\quad 2n_x^2 n_y^2 + 2n_y^2 n_z^2 + 2n_x^2 n_z^2 \\
 \|P\|^2 &= 3 - 2(n_x^2 + n_y^2 + n_z^2) + (n_x^2 + n_y^2 + n_z^2)^2 \\
 &= 2,
 \end{aligned}$$



since  $|n| = 1$ , and so  $n_x^2 + n_y^2 + n_z^2 = 1$ . Another simple calculation shows that  $\|I/|\nabla f|\| = \sqrt{3}/|\nabla f|$ , and so

$$r(G) \leq \frac{2\sqrt{3}}{|\nabla f|} \|H\| \quad (3)$$

This inequality shows that  $\kappa_{\max}$  is intimately related to  $|\nabla f|$  and the Hessian  $H$ . More importantly, it shows us that to bound  $\kappa_{\max}$  above, it is enough to give a lower bound on  $|\nabla f|$  and an upper bound on  $\|H\|$ . Since bounding these values over all of the points in the isosurface is very difficult without a parameterization, a looser bound can be found by taking it over all of the points in the domain of the implicit function.

Creating a sufficient guidance field in the way described above would have the effect of making  $g$  flat — the edge lengths of the triangulation would be almost uniform over the entire surface. Additionally, the bound on  $\kappa_{\max}$  may go to infinity as the gradient magnitude goes to zero. These two issues can easily be addressed by adaptively subdividing the domain of the implicit surface, and creating sufficient guidance fields in the separate regions independently. The guidance fields are merged by using the new sample points from all separate regions: this ensures that the resulting guidance field is sufficient and Lipschitz-continuous. Since we assume that the implicit function has non-zero gradient at all the points on the surface and the function is at least  $C^1$  continuous, this procedure is guaranteed to find a finite bound: there is always a tubular neighborhood of the surface where the gradient is non-zero.

## 4.2 Minimal guidance field

Creating the guidance field with sufficiently many samples to capture all of the details in the surface can produce a very large number of points. The size of the guidance field directly affects the memory usage and running time of our algorithm, so it is desirable to remove as many irrelevant samples as possible. Though many of the samples are necessary, a large portion of them provide no information to the guidance field. This happens when the edge length required for the curvature at a sample  $s_1$  in the guidance field is always smaller than that of another sample  $s_2$ . In this situation, we say that  $s_1$  *dominates*  $s_2$ , as illustrated in Figure 2.

A naïve procedure for culling the unnecessary samples is to compare each sample to every other sample of the guidance field. This is simple since determining if a single point dominates another is straightforward. Since our guidance fields often initially contain millions of samples, this  $O(n^2)$  algorithm is not practical.

Looking at Figure 2, notice how each of the samples define a cone in  $\mathbf{R}^{n+1}$ , where  $n$  is the dimension of the embedding space. This analogy translates perfectly into the case of 2-manifolds embedded in  $\mathbf{R}^3$ : each  $\tilde{g}$  defines a right cone in  $[x, y, z, r]$  space. By constructing a hierarchical data structure in this space, we can perform carefully constructed range queries and remove entire sets of unnecessary samples in a single query. The most important observation is this: if a point  $s_1$  dominates a point  $s_2$ ,  $s_2$  lies inside the cone defined by  $s_1$ . Also, the *dominates* relation is transitive: if  $s_1$  dominates  $s_2$  and  $s_2$  dominates  $s_3$ , then  $s_1$  dominates  $s_3$ . This means that if  $s_1$  culls a set of points  $\{s_i\}$ , the cones induced by all  $\{s_i\}$  need not be checked at all.

All of the points in the guidance field are initially inserted into a 4-dimensional kd-tree. The coordinates for each sample  $s$  are  $[s_x, s_y, s_z, s_r]$ , where  $[s_x, s_y, s_z]$  is the sample's location in  $\mathbf{R}^3$ , and  $s_r$  is the height of the function  $\tilde{g}_s$  at  $s$ . Note now that the set of points  $\{[x, y, z, \tilde{g}_s([x, y, z])]\}$ ,  $x, y, z \in \mathbf{R}^3$ , can be represented by a 4-dimensional cone with apex  $[s_x, s_y, s_z, s_r]$ , axis  $[0, 0, 0, 1]$ , and angle  $\tan^{-1}(\eta/\eta - 1)$ . This cone is completely defined by the sample's location and the user parameters  $\rho$  and  $\eta$ . Finding all of the samples that are dominated by  $s$  is now reduced to a kd-tree query to find all of the points that lie inside of this cone. Such a query relies on bounding box / cone intersection and point-in-cone tests, both of which are quite simple given that the cones are always aligned with the  $r$  axis. When doing the kd-tree query, we simply mark all of the samples that lie inside of the cone. If all of the children of a node have been marked, the node is also marked. This effectively prunes off branches of the

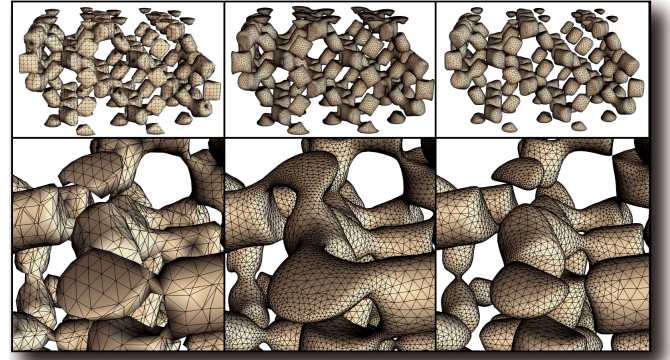


Fig. 4. Isosurfacing a structured grid of a silicon lattice simulation. From left to right: marching cubes output, and our method for  $\rho = 0.3$ , using respectively Catmull-Rom and B-splines for reconstruction.

kd-tree from subsequent queries. Ideally, we would query the cones of the samples that dominate the most samples first, thus pruning off large parts of the tree early in the process. However, since this information is not known or easily computed, we use a heuristic. We note that a given sample will never be dominated by another sample with lower curvature. By sorting the samples by their curvature and doing the queries in descending order, we can perform the culling in about ten seconds for a million samples. When more than this many samples are present in the original guidance field, we recursively subdivide until the culling can be performed on a smaller subset. The following pseudocode summarizes the algorithm.

```

CULL( $g, \rho, \eta$ )
1   $tree \leftarrow kd\text{-}tree(\tilde{s})$ 
2   $Sort(\tilde{s}, \kappa_{\max}(\tilde{s}))$ 
3  for  $0 \leq i \leq |\tilde{s}|$ :
4      do if  $\text{Not}(\text{Marked}(\tilde{s}_i))$ 
5          then  $c \leftarrow \text{Cone}(\tilde{s}_i, \rho, \eta)$ 
6               $\text{MarkIfInside}(tree, c)$ 
7  Discard all marked  $\tilde{s}_i$  from  $g(x)$ 

```

This procedure typically removes a large percentage of the sample points, resulting in much lower memory usage and significantly faster guidance field queries. Table 1 shows some of the results of culling the guidance field. Notice that even though culling requires additional processing, the total running time improves significantly. Also, notice how in some cases, less than half of a percent of the points remain in the guidance field after culling. Figure 3 shows the location of the guidance field samples in space. It is clear that the samples in the high curvature areas are the most significant, and tend to dominate the low curvature samples in their vicinity. Surprisingly, this effect is quite non-local, which accounts for the drastic reduction in the total sample count.

## 5 DEFINING THE IMPLICIT FUNCTION

An important advantage of our advancing front technique is its generality. The algorithm itself is entirely oblivious to how the implicit function is defined. We only require within a band of the desired isosurface, that the function be continuous (at least  $C^1$ , preferably higher order continuity), and that its value, first, and second order partial derivatives can be evaluated. We also require that the gradient is non-zero at all points in the isosurface. These requirements are quite mild and allow our algorithm to be applied to many different implicit function definitions. Choosing the definition can be left up to the user. This allows the implicit function being used by our advancing front algorithm to exactly match the function used to generate the data. For example, if the user wishes to extract the isosurfaces of a high order finite element simulation which assumes a specific polynomial interpolation scheme, the same interpolation could be plugged into our algorithm. We have been careful in our implementation to make this an

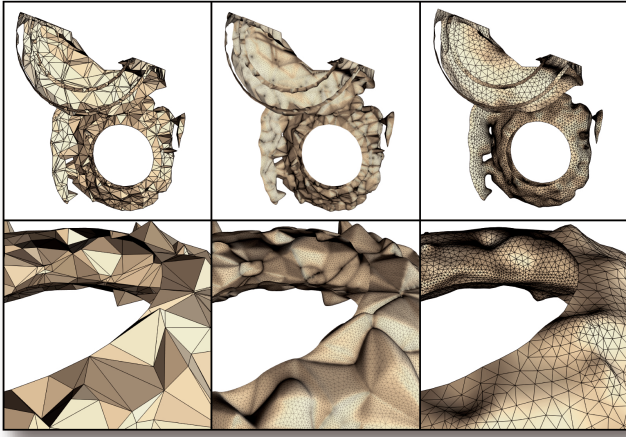


Fig. 5. Isosurfacing unstructured grids. From left to right: MT output, and our method for  $\rho = 0.5$ , using respectively Nielson interpolation and Moving Least Squares for reconstruction.

easy step for the user. In this paper we show examples generated with four different surface definitions — two for structured grids and two for unstructured grids. See Table 2.

### 5.1 Structured Grids

To define an implicit function over a regular grid of sample points, we use piecewise cubic trivariate polynomials generated by two different spline types. To generate an implicit function that interpolates the data points, we use Catmull-Rom splines [11]. These splines can be thought of as using finite differences at each data point to generate gradients. The function is then extended to the interior of each cell by Hermite interpolation. This results in cubic polynomials which are  $C^1$  continuous across cell boundaries. B-splines [42] are another popular spline. These produce polynomials in each cell that maintain  $C^2$  continuity across the boundaries, but do not interpolate the input data. This may be desirable if the data is noisy. Since the Catmull-Rom splines interpolate, they tend to produce functions with high curvature when noise is present. Though we can triangulate these surfaces without problem, the high curvature isosurfaces require more triangles to be accurately captured (see Figure 4). These spline representations are ideal because they define analytic, piecewise polynomials, which can easily be differentiated to compute gradient and curvature information.

### 5.2 Unstructured Grids

Extending the implicit function from the data points to the interior of the cells of unstructured meshes is much more challenging. Here, we have a mesh defined by a set of connected tetrahedra with values assigned at each vertex. A scheme for defining the function on the interior of each tetrahedra that is at least  $C^1$  continuous across faces must be created. In this paper we tried two such schemes. Splines cannot easily be applied to this situation because of the irregular nature of tetrahedral meshes. Similar to regular grids, we use one implicit function definition that is  $C^1$  and interpolates the data values, and one that has higher order continuity but only approximates the data.

The irregular mesh interpolation scheme that we use is that of Nielson et al. [38]. We have chosen this method for its simplicity, though other techniques are available, such as A-patches [5] and DMS-splines [19]. Nielson interpolation requires that the gradients of the implicit function are known at the vertices. Since this is generally not the case, we substitute an approximation. To estimate the gradient at vertex, we fit a trivariate quadratic polynomial to the 2-ring neighborhood of the vertex. We define the gradient of the implicit function to be the gradient of this polynomial at the vertex. The Nielson scheme first uses the function and gradient information of the vertices to define the function and gradient along all of the edges in the mesh. It then uses this edge information and extends it to be defined across all of

	Interpolating	Approximating
Structured	Catmull-Rom (CR)	B-Splines (BS)
Unstructured	Nielson (NI)	MLS

Table 2. The four surface definitions implemented in this paper.

the faces. Finally, the function is extended from the faces to the interiors of the tetrahedra. Each of the extension procedures (vertices to edges to faces to tetrahedra) is based on Hermite interpolation, and is constructed in a way that maintains  $C^1$  continuity across the cell boundaries. The resulting implicit function definition for the interior of the tetrahedra is a complicated rational function that is not practical to differentiate analytically. Instead of resorting to finite differencing, we use C++ metaprogramming to automatically compute the function and its derivatives by encoding the chain rule for all of the primitive functions used.

We use the popular moving least squares (MLS) method [28] as an implicit function definition that approximates the input data. The idea of the MLS method is to compute a low degree polynomial that best approximates the input data, weighted by a function of the distance from the evaluation point to the data points. The function value at the evaluation point is then simply the value of the polynomial at that point. This has a smoothing effect on the data, so it is especially useful when noise is present in the input data. A nice property of MLS is that the smoothness of the resulting function is exactly that of the chosen weighting function. We use rapidly decaying gaussians as the weighting functions, with widths determined by the local point density. Though the function is very smooth, it is not possible to directly compute its derivatives because every evaluation involves the solution of a linear least squares problem. We have found that using quadratic polynomials for the fitting and using their differential properties as an approximation of the function performs well in practice.

## 6 EXPERIMENTAL RESULTS AND DISCUSSION

We have implemented the algorithm in C++, and in this section we describe the results of a series of experiments. All timings were performed on a PC with two dual-core AMD Opteron processors running at 2.25GHz, with 4GB of main memory, running SuSE Linux 10.0. Advancing front algorithms are mostly local, allowing decisions and actions to be taken independently of the reconstruction of distant surface parts. Our implementation is multi-threaded, and it tries to take advantage of this and distribute the work of projecting points on the fronts to a set of worker threads. The guidance field computation and culling is also performed entirely in parallel. We use four threads for all shown experiments.

Table 3 shows a summary of the experimental results of our implementation. The running times for our algorithms are typically in the order of several minutes for the larger datasets we used. The histogram column shows the distribution of triangle circle ratios (the ratio of the incircle to the circumcircle). Notice that the distribution is excellent, and also uniform across different models and parameter settings. For many of the rectangular grids, the number of triangles generated is significantly less than the output of MC.

The triangle quality generated by our algorithm is very good, as the circumcircle ratio histograms of Table 3 show. For all models generated, 99% of the triangles have quality at least half of optimal, and the triangle with median quality is within 3% of optimal. These results hold across different models,  $\rho$  and  $\eta$  values, and different surface definitions. In terms of running time, our algorithm is not as fast as MC or MT implementations. However, it should be noted that the triangulations created by our technique will tend to require negligible or no downstream processing, unlike the results of MC or MT.

The surfaces generated by our algorithm are dependent on the intrinsic geometry of the implicit function, and not on the grid on which they were sample. As a consequence, we are able to generate high-quality meshes for high-curvature surfaces. Naturally, Catmull-Rom splines tend to generate surfaces with higher curvatures, while B-splines offer an overall smoother result. We show both schemes, together with MC, on Figure 4. Notice that the topology of the Catmull-

















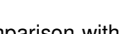

Model	Alg.	$\rho$	$\eta$	time	# tris	Histogram	Model	Alg.	$\rho$	$\eta$	time	# tris	Histogram
Aneurism	MC	—	—	0:07	133.5K		SPX	MT	—	—	0:00	2.3K	
	BS	0.2	1.2	5:18	461.7K			NI	0.5	1.2	14:06	645.9K	
Silicium	MC	—	—	0:00	29.8K		Torso-1	MLS	0.5	1.2	1:48	26.7K	
	CR	0.3	1.2	1:30	192.1K			MT	—	—	0:01	3.1K	
Engine	CR	0.5	1.33	0:58	92.1K		Torso-2	NI	0.5	1.2	2:28	72.8K	
	MC	—	—	0:09	592.1K			MLS	0.5	1.2	2:04	702	
Skull	BS	0.3	1.2	12:16	304.4K			MT	—	—	0:02	24.2K	
	MC	—	—	0:06	393.2K			NI	0.5	1.2	12:48	656K	
	CR	0.5	1.2	5:50	259.2K			MLS	0.5	1.2	4:24	2.4K	

Table 3. Sample of results of our implementation, and comparison with MC and MT. The histograms show the distribution of circumcircle ratios, where the three lines represent the minimum, the first half percentile and the median. The legend for the second column follows the conventions of the rest of the paper. Dataset sizes: Aneurism, 256<sup>3</sup> bytes; Silicium, 98 × 34 × 34 bytes; Engine, 256 × 128 × 128 bytes; Skull, 256 × 256 × 226 bytes; SPX, 13k tetrahedra; Torso, 1.1 million tetrahedra.

Rom isosurface is strictly different than the one generated by the assumption of trilinearity imposed by Marching Cubes. This precludes using the MC mesh as a base mesh for refining isosurfaces defined by different filters. Finally, the excess topology eventually generated by noise could be removed using an approach such as the one suggested by Guskov and Wood [22]. For unstructured meshes, notice the geometry generated for the Nielson interpolation scheme on Figure 5. The tetrahedral mesh might be coarse, but the defined function has high curvature inside each cell, and so an accurate sampling requires more triangles. A more relaxed scheme such as MLS approximation creates isosurfaces with less curvature, and hence less triangles are needed.

All isosurfaces presented in this paper are smooth: both gradient and Hessian are well-defined. This is a reasonable assumption given the necessarily band-limited reconstruction of sampled volumes. Additionally, a surface with a sharp corner will not admit a regular isosurface representation (i.e., one where  $f(x) = k, \nabla f(x) \neq 0$ ). More advanced representations that keep the flexibility of implicit surfaces might be possible, but they are outside the scope of this paper.

Triangle counts are often less than MC triangle counts, and while one of the schemes generates much denser triangle meshes than seems necessary, overall they tend to be well within the applicability range of downstream processing methods. Additionally, the mesh has both bounded distance and (trivially, cf. Section 3.1) bounded normal error. If that proves to be too restrictive, a downstream tool might choose to relax any of these constraints and simplify the mesh accordingly. It should also be clear that the results generated in this paper are easily generalizable to different surface formulations. If a function admits gradient and Hessian, we can use our technique to extract a high-quality, high-fidelity triangulation of any of its regular isosurfaces. This is especially true for high-order meshes, where the image of each cell is dictated by a high-order (typically around degree 10 [36]) polynomial. In this case, pieces of the isosurface might have a much higher frequency than the one implied by the vertex spacing. Our model, however, is entirely oblivious of the underlying mesh, and given a function that computes gradient and Hessian, will still work as if it were dealing with a low-order mesh.

## 7 CONCLUSION AND FUTURE WORK

We have presented a robust algorithm for extracting surfaces from implicit functions. Our algorithm generates meshes that combine triangle quality, adaptiveness and fidelity, easing the process of making isosurface meshes more directly applicable for downstream processing. We believe this algorithm can be adapted for processing gigantic data with minimal changes, by combining the generation of the guidance field with the actual isosurface extraction and out-of-core techniques. We would also like to give triangle shape guarantees for the meshes generated by the algorithm. These are all intriguing avenues for future work.

## ACKNOWLEDGMENTS

Our research is funded by the National Science Foundation (grants CCF-0401498, EIA-0323604, OISE-0405402, IIS-0513692, CCF-0528201), the Department of Energy, Sandia National Laboratories, Lawrence Livermore National Laboratory, an IBM Faculty Award, and a University of Utah Seed Grant.

## REFERENCES

- [1] S. Akkouché and E. Galin. Adaptive implicit surface polygonization using marching triangles. *Comput. Graph. Forum*, 20(2):67–80, 2001.
- [2] P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun. Variational tetrahedral meshing. *ACM Trans. Graph.*, 24(3):617–625, 2005.
- [3] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene. Recent advances in remeshing of surfaces. In *State-of-the-art report of the AIM@SHAPE EU network*. Springer, 2005.
- [4] N. Amenta and M. W. Bern. Surface reconstruction by voronoi filtering. In *Symposium on Computational Geometry*, pages 39–48, 1998.
- [5] C. L. Bajaj, J. Chen, and G. Xu. Modeling with cubic a-patches. *ACM Trans. Graph.*, 14(2):103–133, 1995.
- [6] C. L. Bajaj, V. Pascucci, and D. R. Schikore. Fast isocontouring for improved interactivity. In *1996 Volume Visualization Symposium*, pages 39–46, 1996.
- [7] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999.
- [8] J. Bloomenthal. An implicit surface polygonizer. In P. Heckbert, editor, *Graphics Gems IV*, pages 324–349. Academic Press, Boston, 1994.
- [9] J. Bloomenthal, editor. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.
- [10] H. Borouchaki, F. Hecht, and P. J. Frey. Mesh gradation control. In *6th International Meshing Roundtable*, pages 131–141, 1997.
- [11] E. Catmull and R. Rom. *Computer Aided Geometric Design*, chapter A Class of Local Interpolating Splines. Academic Press, 1974.
- [12] J. R. Cebal and R. Löhner. From medical images to CFD meshes. In *8th International Meshing Roundtable*, pages 321–331, 1999.
- [13] M. Cermák and V. Skala. Adaptive edge spinning algorithm for polygonization of implicit surfaces. In *Computer Graphics International*, pages 36–43, 2004.
- [14] P. Cignoni, F. Ganovei, C. Montani, and R. Scopigno. Reconstruction of topologically correct and adaptive trilinear isosurfaces. *Computers & Graphics*, 24(3):399–418, 2000.
- [15] P. Cignoni, P. Marino, C. Montani, E. Puppo, and R. Scopigno. Speeding up isosurface extraction using interval trees. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):158–170, 1997.
- [16] P. Cignoni, C. Montani, E. Puppo, and R. Scopigno. Optimal isosurface extraction from irregular volume data. *1996 Volume Visualization Symposium*, pages 31–38, 1996. ISBN 0-89791-741-3.
- [17] R. Courant and F. John. *Introduction to Calculus and Analysis*, volume 2. John Wiley and sons, 1974.



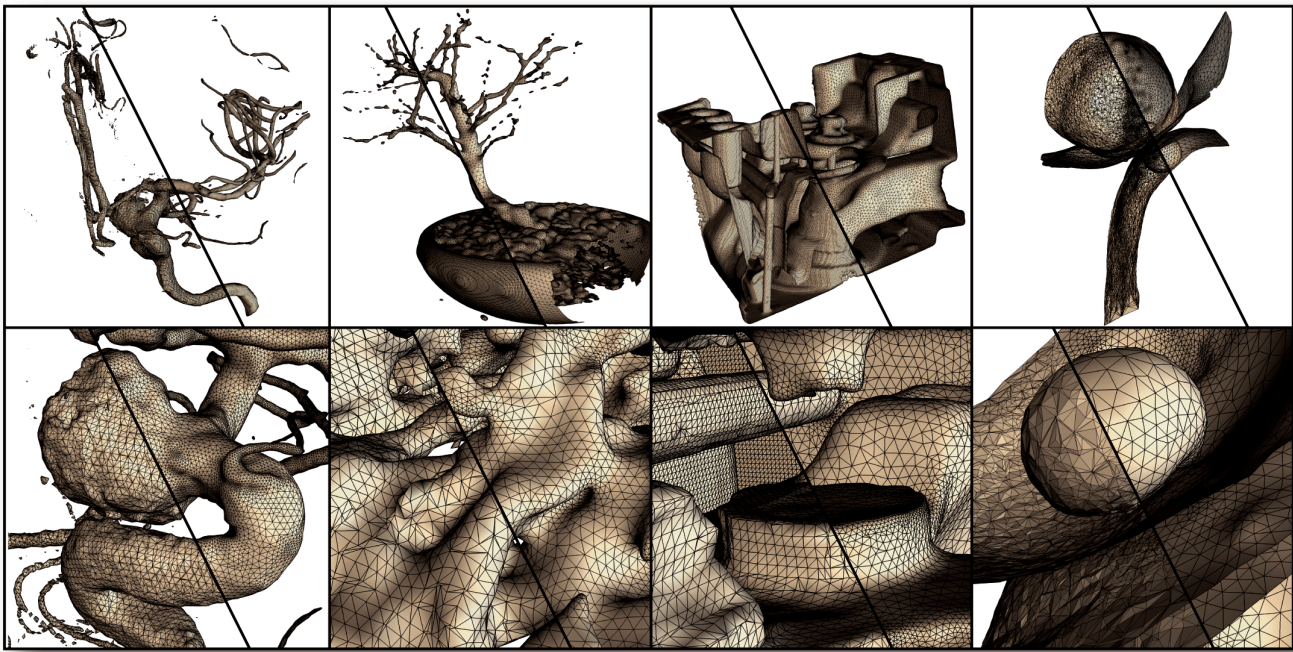


Fig. 6. Some results of our algorithm, compared to MC and MT. From left to right: CT scans of an aneurism, a bonsai and an engine block, and isopotential surfaces of a human torso simulation. The first three datasets are regular grids, while the last one is a tetrahedral mesh.

- [18] P. Crossno and E. Angel. Isosurface extraction using particle systems. In R. Yagel and H. Hagen, editors, *IEEE Visualization 1997*, pages 495–498, 1997.
- [19] W. Dahmen, C. A. Micchelli, and H.-P. Seidel. Blossoming begets bsplines built better by b-patches. *Math. Comp.*, 59(199):97–115, 1992.
- [20] L. H. de Figueiredo, J. Gomes, D. Terzopolous, and L. Velho. Physically-based methods for polygonization of implicit surfaces. In *Graphics Interface*, 1992.
- [21] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH 1997*, pages 209–216, 1997.
- [22] I. Guskov and Z. Wood. Topological noise removal. In *Graphics Interface*, 2001.
- [23] E. Hartmann. A marching method for the triangulation of surfaces. *The Visual Computer*, 14(3):95–108, 1998.
- [24] A. Hilton, A. Stoddart, J. Illingworth, and T. Windeatt. Marching triangles: Range image fusion for complex object modelling. *International Conference on Image Processing*, B:381–384, 1996.
- [25] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. *ACM Transactions on Graphics*, 21(3):339–346, 2002.
- [26] T. Karkanis and A. J. Stewart. Curvature-dependent triangulation of implicit surfaces. *IEEE Computer Graphics and Applications*, 21(2):60–69, 2001.
- [27] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Möller. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *IEEE Visualization*, 2003.
- [28] P. Lancaster and K. Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of Computation*, 87:141–158, 1981.
- [29] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3D scanning of large statues. In *SIGGRAPH 2000*, pages 131–144, 2000.
- [30] P. Lindstrom. Out-of-core simplification of large polygonal models. In *SIGGRAPH 2000*, pages 259–262. ACM Press/Addison-Wesley Publishing Co., 2000.
- [31] Y. Livnat, H. Shen, and C. Johnson. A near optimal isosurface extraction algorithm using the span space. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):73–84, 1996.
- [32] Y. Livnat and X. Tricoche. Interactive point-based isosurface extraction. In *IEEE Visualization*, pages 457–464, 2004.
- [33] W. Lorensen and H. Cline. Marching Cubes: A high-resolution 3D surface construction algorithm. In *SIGGRAPH 1987*, pages 163–169, 1987.
- [34] M. Meyer, P. Georgel, and R. Whitaker. Robust particle systems for curvature dependent sampling of implicit surfaces. In *International Conference on Shape Modeling and Applications*, pages 123–133, 2005.
- [35] B. K. Natarajan. On generating topologically consistent isosurfaces from uniform samples. *The Visual Computer*, 11(1):52–62, 1994.
- [36] B. Nelson and R. M. Kirby. Ray-tracing polymorphic multi-domain spectral/hp elements for isosurface rendering. *IEEE Transactions in Visualization and Computer Graphics*, 12(1):114–125, 2006.
- [37] G. Nielson. On marching cubes. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):283–297, 2003.
- [38] G. Nielson, H. Hagen, and H. Müller. *Scientific Visualization*, chapter 20. IEEE Computer Society, 1997.
- [39] G. M. Nielson and B. Hamann. The asymptotic decider: Removing the ambiguity in marching cubes. In *IEEE Visualization*, pages 83–91, 1991.
- [40] P. Ning and J. Bloomenthal. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, 13(6):33–41, 1993.
- [41] S. Owen. A survey of unstructured mesh generation technology. <http://www.andrew.cmu.edu/user/sowen/survey/index.html>.
- [42] L. Piegl and W. Tiller. *The NURBS book*. Springer, 1997.
- [43] J. Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. In *Symposium on Discrete Algorithms*, 1993.
- [44] C. E. Scheidegger, S. Fleishman, and C. T. Silva. Triangulating point set surfaces with bounded error. In *Symposium on Geometry Processing*, pages 63–72, 2005.
- [45] J. Schreiner, C. Scheidegger, S. Fleishman, and C. Silva. Direct (re)meshing for efficient surface processing. *Comput. Graph. Forum (Proceedings of Eurographics 2006)*, 2006. to appear.
- [46] C. T. Silva and J. S. B. Mitchell. Greedy cuts: An advancing front terrain triangulation algorithm. In *ACM-GIS*, pages 137–144, 1998.
- [47] G. Treece, R. Prager, and A. Gee. Regularised marching tetrahedra: Improved iso-surface extraction. *Computers & Graphics*, 23(4):583–598, 1999.
- [48] G. Turk. Re-tiling polygonal surfaces. In *SIGGRAPH 1992*, pages 55–64, 1992.
- [49] J. Wilhelms and A. V. Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227, 1992.
- [50] A. P. Witkin and P. S. Heckbert. Using particles to sample and control implicit surfaces. In *SIGGRAPH 1994*, pages 269–277, 1994.
- [51] Z. J. Wood, P. Schröder, D. Breen, and M. Desbrun. Semi-regular mesh extraction from volumes. In *IEEE Visualization*, pages 275–282, 2000.