Efficient Delaunay Mesh Generation From Sampled Scalar Functions

Samrat Goswami[1], Andrew Gillette[2], and Chandrajit Bajaj[3]

[1] Institute for Computational and Engineering Sciences, University of Texas at Austin `samrat@ices.utexas.edu`
[2] Department of Mathematics, University of Texas at Austin `agillette@math.utexas.edu`
[3] Department of Computer Sciences and Institute for Computational and Engineering Sciences, University of Texas at Austin `bajaj@cs.utexas.edu`

Abstract: Many modern research areas face the challenge of meshing level sets of sampled scalar functions. While many algorithms focus on ensuring geometric qualities of the output mesh, recent attention has been paid to building topologically accurate Delaunay conforming meshes of any level set from such volumetric data.

In this paper, we present an algorithm which constructs a surface mesh homeomorphic to the true level set of the sampled scalar function. The presented algorithm also produces a tetrahedral volumetric mesh of good quality, both interior and exterior to the level set. The meshing scheme presented substantially improves over the existing algorithms in terms of efficiency. Finally, we show that when the unknown sampled scalar function, for which the level set is to be meshed, is approximated by a specific class of interpolant, the algorithm can be simplified by taking into account the nature of the interpolation scheme so as to circumvent some of the critical computations which tend to produce numerical instability.

1 Problem and Motivation

A wide variety of science and engineering applications rely on accurate level set triangulation. This is especially true for multiscale models in biology, such as macromolecular structures extracted from reconstructed single particle cryo-EM (Electron Microscopy), cell-processes and cell-organelles extracted from TEM (Tomographic Electron Microscopy), and even trabecular bone models extracted from SR-CT (Synchrotron Radiation Micro-Computed Tomography) imaging. Computational analysis of these models for estimation of nano, micro, or mesoscopic structural properties depends on the mesh representation of the contour components respecting their topological features.
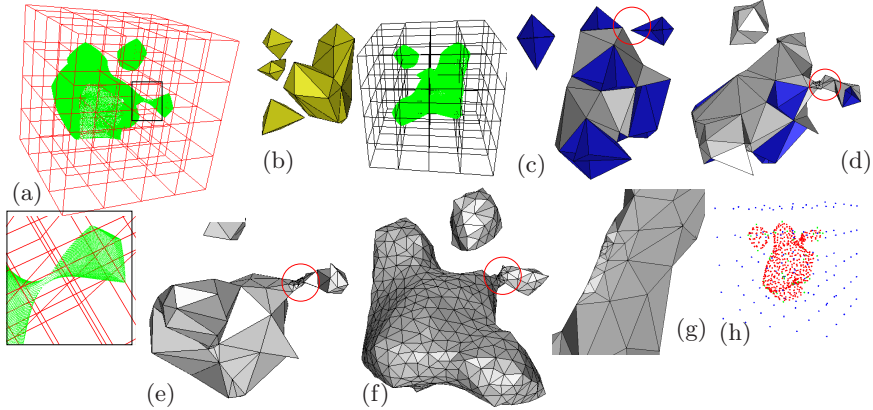
Fig. 1. Various stages of our algorithm. (a) A rectilinear grid with sample values of an unknown function at the grid points. Within cells, the function is approximated with a trilinear interpolant. For the purpose of visualization only, we collect a set of points (green) on the surface and display them. A narrow region of the surface is magnified below. (b) Another view of the data (right) and the same view of the mesh generated by Marching Cubes [22]. Note that the mesh is disconnected in the thin region. (c) The mesh generated by the restricted Delaunay triangulation of only edge and grid points. Blue facets have a grid point as at least one of their vertices. This point set is still not sufficient to produce a Delaunay-conforming mesh. (d) The mesh generated by addition of sample points of Σ. The topology is now recovered (Property I). (e,f) Geometrical refinement for progressively smaller value of ϵ. (g) Even in the magnified portion of the thin region, the triangles approximate the geometry nicely (Property II). (h) All the points involved in construction of the mesh including grid points (blue), edge points (green), and new points added by the algorithm (red). Observe that in order to recover the topology and reduce the geometric error in the approximation, many surface sample points are added to the point set.

Our goal is to find an algorithm to solve the following problem. The input to the algorithm is a rectilinear sampling of a bounded domain of an unknown scalar function F. The rectilinear grid need not be uniform; it may be adaptive as in the case of an octtree. The user then specifies a local interpolant to generate a level set approximation for any isovalue v; we use Σ to denote this level set of the interpolating function. Since the function F is unknown, we must assume that the local interpolant produces a good approximation of the function F within each cell of the grid. Our algorithm is general enough to use any local interpolant, however, in our experience, a trilinear interpolant is the most natural choice.

Our goal is construct a mesh in an efficient manner such that the following properties hold:

I **Topological Guarantee:** M is homeomorphic to Σ.

II **Geometrical Guarantee:** The Hausdorff distance from M to Σ is within a user-specified bound ϵ.

III **Delaunay Conformity:** M is a subcomplex of the Delaunay triangulation of the vertex set of M.

IV **Adaptivity:** The user can decimate part of the volumetric data and still preserve properties I, II, and III.

Once a surface M is generated that is Delaunay conforming, it is possible to improve the mesh quality by applying any Delaunay refinement algorithm. We detail such an algorithm in Section 4.1. Figure 1 visually illustrates a toy data set (a), the failure of a typical isocontouring method, in this case Marching Cubes, in reconstructing the level set (b), generation of the correct topology by our algorithm (c-d), geometric refinement (e-g), and the final Delaunay-conforming surface sample (h).

At this point, we emphasize the novelty of our approach. Although there exist algorithms which can be applied to solve the problem as stated, such algorithms are devised in a more general setting and thus do not exploit the natural structure of the volumetric data. Our approach has a number of unique advantages over its predecessors. As part of the algorithm, we collect some of the grid points around Σ and use them to build a Delaunay conforming mesh efficiently. Once the surface mesh is created and forced to conform to the Delaunay triangulation of the point set, these grid points can either be removed or used to construct an interior or exterior tetrahedral volume mesh.

Additionally, as a result of noise in the input data or a poor choice of the isovalue v, there may exist topological anomalies in the surface Σ. In mesh processing literature, such anomalies have been referred to as "topological noise" [26, 4]. The term "noise" indicates undesirable features of small geometric size that prevent the mesh from being used for further processing. Methods have been developed to remove such artifacts provided that the point sample of Σ is sufficiently dense near the anomalies; our method guarantees this density. Therefore, the output mesh M can be applied without any further refinement to any point processing algorithm that uses prior Delaunay-based reconstruction of geometry to detect and selectively remove these topological features.

Finally, algorithms involving volumetric data and meshes often become computationally demanding due to the large size of data sets. The adaptivity of the algorithm (property IV) provides one way to ease calculations by down-sampling less important regions of the volumetric data.

2 Prior Work

Mesh generation techniques have received significant attention in the past two decades. Two works in particular, one by Chew [12] and one by Edelsbrunner

and Shah [15], have spawned important and relevant results in the field. We will address the prominent successors of each of these works and compare the relative advantage of our approach.

Chew provided one of the first meshing algorithms for curved surfaces with provably good geometry in [12], although the algorithm as stated in the paper could not guarantee topological correctness. Boissonnat and Oudot showed how Chew's algorithm can be applied to produce a dense sample of Σ and subsequently mesh it. Oudot, Rineau and Yvinec recently improved that method by including sliver exudation, that is, the process of removing tetrahedra with very small dihedral angles from a mesh [24]. Alliez, Cohen-Steiner, Yvinec and Desbrun have also adapted the method to produce nicely graded meshes, i.e. meshes where the tetrahedra vary in size gradually based on their distance to the surface [1].

Separately, Edelsbrunner and Shah established a criterion called the *closed ball property* for ensuring that a mesh is Delaunay conforming [15]. We explain the closed ball property in Section 3. Recent work by Cheng, Dey, Ramos, and Ray uses this property to provide a method for constructing Delaunay conforming meshes that avoids the need to estimate local feature size [9]. This is a significant development as approximating local feature size is computationally expensive and not always numerically robust. Cheng, Dey and Ramos have extended this strategy for piecewise smooth complexes [10]. Very recently, Dey and Levine [14] have given a two phase algorithm to mesh isosurfaces from imaging data.

All of these approaches to mesh generation are useful, however, they all rely on an oracle to know whether an arbitrary ray intersects the surface Σ. The implementation of such an oracle becomes computationally prohibitive when applied to piecewise interpolated surfaces. For example, a common data set size is 100^3 vertices, meaning there exist 100^3 functions in the piecewise decomposition. Thus, a single ray may pass through over a hundred separate function domains, making intersection calculations expensive. As we detail in Section 4.2, a major advantage of the algorithm presented in this paper is that we take advantage of the original rectilinear scaffolding from the data to substantially reduce the computational overhead required.

Our work also improves upon existing methods for isosurface construction. Many well-known techniques exist for isosurfacing including marching cubes [22], active snakes, dual contouring [28], and higher order interpolation [6]. A variety of approaches have also been developed to provide hierarchical isosurface extraction and interior meshing [27, 17, 18]. Shewchuk and Labelle recently provided a straightforward isosurfacing and stuffing algorithm with good quality tetrahedra [19]. Relatively few works, however, take into account the effect of a trilinear interpolant within each grid cell [23, 11, 21]. Attali and Lachaud gave an algorithm for construction of Delaunay conforming isosurfaces [2]. Their method, however, relies on a specific rule established by Lauchaud in [20] that does not accommodate interpolation within grid cells.

Further, none of these techniques generalize easily to data that is sampled adaptively or to arbitrary interpolants.

3 Background

The Voronoi and Delaunay diagrams of a point set P, denoted $\mathrm{Vor}\, P$ and $\mathrm{Del}\, P$ respectively, play an important role in the computations involved in this paper. Due to page limitations, we do not go into the detail of their construction and refer the reader to any standard computational geometry textbook, e.g. [13].

Given a point set P chosen from the same space in which Σ is embedded, the *restricted Delaunay triangulation of P*, denoted $\mathrm{Del}\, P|_{\Sigma}$, is defined to be the set of Delaunay objects of $\mathrm{Del}\, P$ whose dual Voronoi objects have non-zero intersection with Σ. If P is chosen in a way that respects the structure and local feature size of Σ, $\mathrm{Del}\, P|_{\Sigma}$ will be a mesh with the desired properties.

Edelsbrunner and Shah gave a sufficient criterion called the *closed ball property* [15], sometimes referred to as the *topological ball property*, for $\mathrm{Del}\, P|_{\Sigma}$ to be homeomorphic to Σ. A Voronoi object V of dimension k satisfies the closed ball property if $V \cap \Sigma = \emptyset$ or $V \cap \Sigma$ is homeomorphic to a closed ball of dimension $k-1$. Accordingly, a point set P is said to satisfy the closed ball property if every Voronoi object of $\mathrm{Vor}\, P$ satisfies the closed ball property. Using the notion of *transversality* as defined in [16], their criterion can now be stated precisely.

Theorem 1. *[15] If Σ intersects each Voronoi object of $\mathrm{Vor}\, P$ transversally and $\mathrm{Vor}\, P$ satisfies the closed ball property, then $\mathrm{Del}\, P|_{\Sigma}$ is homeomorphic to Σ.*

We will show in Section 4 that our algorithm produces a mesh satisfying the closed ball property. This ensures that the mesh is Delaunay conforming (property III from Section 1) and, by the theorem, that the mesh is homeomorphic to Σ (property I).

4 Algorithm

In this section, we describe the algorithm, analyze its efficiency and present some simplifying results for a specific choice of local interpolant. Figure 2 shows an overview of the process in two dimensions.

4.1 Algorithm Description

Our algorithm is motivated primarily by the work of Cheng et al. in [9] who build a Delaunay conforming approximation of the level set of any general
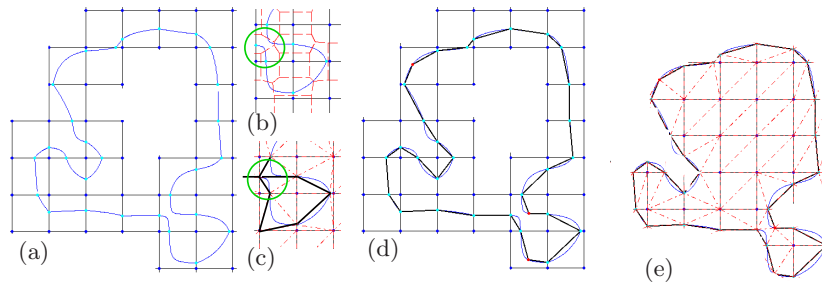
Fig. 2. A 2D example using bilinear interpolation of Σ demonstrating the importance of the closed ball property. (a) Grid points (dark blue) and edge points (light blue) relevant to our algorithm are shown. Note that in the 3D case we form a layer of grid points twice as thick. (b) A portion of the Voronoi diagram is shown in red and a location where the closed ball property is violated is circled. (c) Since the closed ball property is violated, the restricted Delaunay diagram (black) has incorrect topology in the circled region. (d) Red points are inserted where the closed ball property is violated and the restricted Delaunay graph is formed (black). (e) By including those grid points interior to Σ, we efficiently produce an interior mesh that does not alter the Delaunay conforming surface mesh.

implicit function. Since our problem is focused on locally interpolated functions, we take advantage of the natural scaffolding of the input grid to substantially improve the computational efficiency of the algorithm. Moreover, we also show that once the Delaunay conforming surface mesh is extracted, it is quite straightforward to build a tetrahedral volumetric mesh of good quality. We call the algorithm for extracting a surface mesh DELSURFMESH and the extension to build a Delaunay tetrahedral interior or exterior mesh DELVOLMESH.

In [9], the authors start with a small sample of points lying on the surface to be meshed. They keep refining the mesh until its vertex set satisfies the closed ball property, thereby providing a Delaunay conforming mesh homeomorphic to Σ. To ensure that a point set P satisfies the closed ball property, it is necessary to check the intersection of Σ with each Voronoi edge, facet, and cell of Vor P. While checking intersections is a computationally expensive task for a general implicit function, it is even more burdensome for the case of a piecewise locally interpolated function as given in our problem. For example, to determine if a certain Voronoi edge intersects Σ more than once, it is necessary to search all voxels containing any subset of Σ which are stabbed by the Voronoi edge, as Voronoi edges may be incident upon many voxels. Matters become worse for Voronoi facets or cells which may touch large regions of the domain.

It is in regards to this difficulty that our approach becomes significant. We exploit the "gridded" nature of the input data set to put $O(1)$ bounds on our intersection calculations. To start, we construct an initial sampling by

computing all the points where a grid edge intersects Σ. These points serve as the initial sampling of Σ. However, if we compute the Voronoi diagram of these E points alone, the Voronoi cells can intersect an arbitrary number of voxels, meaning we will still have trouble verifying and enforcing the closed ball property. To circumvent this problem, we compute a protective layer of grid points near Σ which we denote G. The selection of G traps Voronoi cells of E points into a few voxels. As the algorithm progresses, it adds more points to the existing samples and the nature of the insertion process ensures that the Voronoi cells of these new points are also trapped in a constant number of voxels. We derive bounds on the size of Voronoi cells of the initial samples and the new points for uniform and non-uniform rectilinear gridding (Octtree) in Section 4.2.

We now give the pseudocode of the algorithm DELSURFMESH and describe the specifics of the steps subsequently (Figure 3).

DELSURFMESH(Σ)
  1   Compute the point set E sampling Σ.
  2   Compute the protective layer G of grid points.
  3   Compute the Voronoi and Delaunay diagrams of $E \cup G$.
  4   Insert new sample points (N) repeatedly until Vor($E \cup G \cup N$) satisfies
          closed ball property.
 5   Output the Restricted Delaunay triangulation Del($E \cup G \cup N$)$|_\Sigma$.

Fig. 3. Pseudo-code of the DELSURFMESH algorithm.

We have already described how we choose the initial set of points on (E) and near (G) the surface Σ. The next task is to ensure that the closed ball property holds for the set of points. For a general interpolant within every grid cell, we employ the method given in [9]. For completeness, we briefly describe how the closed ball property can be violated and what measures are to be taken. This process is thus divided into three sub-steps CBP_VE for Voronoi edges, CBP_VF for Voronoi facets, and CBP_VC for Voronoi cells. As we show in Section 4.3, one can simplify this process considerably further if the typical trilinear interpolant is used, thereby improving the robustness and efficiency of the algorithm.

- CBP_VE: A Voronoi edge VE violates the closed ball property if it intersects Σ in more than one point. If this occurs, the intersection point which is farthest from the Delaunay triangle dual to VE is inserted into the triangulation.
- CBP_VF: A Voronoi facet VF violates the closed ball property if it intersects Σ in more than one component or if the intersection includes a closed loop in the interior of VF. In either case, the intersection point farthest from the Delaunay edge dual to VF is inserted into the triangulation.
- CBP_VC: A Voronoi cell VC violates the closed ball property if it intersects Σ in more than one component, if the intersection includes an

isolated component of Σ inside VC, or if the intersection includes a surface of positive genus with one or more disks removed.
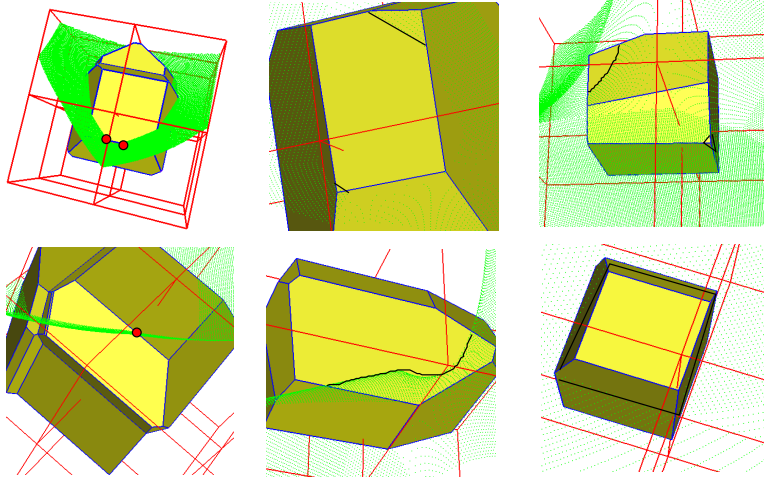
Fig. 4. Top row shows three samples scenarios where the closed ball property is violated for a Voronoi Edge (left), Voronoi Facet (middle) and Voronoi Cell (right). The bottom row shows the cases where the closed ball property is satisfied for Voronoi objects of the corresponding dimension (in the top row).

Note, the above properties are to be checked in the order given. Once a violation is detected, a new point is inserted into the existing Delaunay triangulation, the triangulation is updated, and the processes must begin again. Figure 4 shows different situations that can arise in this context.

Mesh Refinement

Although the topology of M is now correct, it may be possible that the geometry of M is not approximated sufficiently for an application purpose. Hence, we allow a user input ϵ and refine M as follows. For each Voronoi edge VE that intersects Σ, we compute the unique point $p \in VE \cap \Sigma$ and the circumcenter c of the dual Delaunay face to VE. If the distance between p and c is more than ϵ, we add p to the vertex set and regenerate the restricted Delaunay mesh. Every Voronoi edge dual to a restricted Delaunay facet is a normal approximation to Σ locally meaning the distance between p and c is an upper bound for the Hausdorff distance from Σ to the restricted Delaunay mesh. Hence, this process will yield a mesh satisfying property II.

In our current implementation, we maintain the 3D Vor/Del diagram of the point set throughout the process. Very recently, it was shown by Dey and Levine that this is not necessary [14]; one can recover the geometry while manipulating only the 2D mesh data structure of the surface, as long as there are enough points sampling a topologically correct approximation of Σ.

Tetrahedral Meshing of Interior/Exterior

At this stage, we have an accurate mesh approximation of the level set both topologically and geometrically. Since the mesh is already embedded in a Delaunay mesh that includes some grid points, we already have a tetrahedral mesh of both the interior and exterior of Σ. In order to improve the quality of the mesh elements, we use the algorithm DELVOLMESH defined as follows.

Without loss of generality, we describe how DELVOLMESH is used to generate a tetrahedral mesh of the interior of Σ. The input to DELVOLMESH is Del $(G \cup E \cup N)$, the volumetric mesh generated from DELSURFMESH. Note that Del $(G \cup E \cup N)$ has the output of DELSURF MESH, Del $(G \cup E \cup N)|_\Sigma$, as a subcomplex. We form a set G' of all grid vertices of the original rectilinear scaffolding which have function values less than the isovalue and do not belong to G. These points are distributed through the interior of Σ evenly (or evenly relative to an adaptive gridding) and we add them to the Delaunay mesh of the volume.

Here, our protective layer of grid points is crucially important. As we add the points of G', some triangles of the Delaunay mesh will necessarily change but the Delaunay triangles among surface points (E and N vertices) will be unaffected. This is a direct consequence of the fact that a point of G' is, by construction of G, closer to points of G than to points of $E \cup N$. Therefore, Del $(E \cup N \cup G \cup G')$ will still have M as the restricted Delaunay diagram. By throwing out the points of G exterior to Σ, we are left with a tetrahedral mesh of the volume with good quality tetrahedra and a Delaunay conforming surface mesh. The 2D analogue of DELVOLMESH is shown in Figure 2 (d) and (e).

4.2 Efficiency

Since our algorithm uses the natural structure of the rectilinear input data to construct Voronoi and Delaunay diagrams, we are able to provide two important results that reduce the computational burden. We state and discuss the significance of each one.

Theorem 2. *There is an $O(1)$ bound on the number of voxels that a Voronoi cell of an E or N point may intersect. In the case of uniform rectilinear gridding with voxels that are cubes, this bound is four voxels.*

Proof. We prove the following lemma for the simplest case in 2D which will be the crux of the argument for the proof of the more general cases.

Lemma 1. *For data given on an equally spaced 2D grid, the Voronoi cell of an E point is bounded within two pixels.*

Proof. Let $e \in E$ lie on the edge ε be between grid points $g_1, g_2 \in G$ and let $VC(e)$ be the (2D) Voronoi cell defined by e. There exist two lines perpendicular to ε, one passing through the midpoint of g_1 and e and one passing

through the midpoint of g_2 and e. By definition, $VC(e)$ is necessarily contained between these two lines which bounds $VC(e)$ to a single column of pixels.

Now consider one of of the two pixels containing both g_1 and g_2. Denote its other vertices as g_3 and g_4. Let s denote the side length of a pixel. Then both $||e - g_3|| > s/2$ and $||e - g_4|| < s/2$. For any point f on the edge between g_3 and g_4, observe that $||f - g_3|| \leq s/2$ or $||f - g_4|| \leq s/2$. Therefore, each point on the edge between g_3 and g_4 is closer to one of those grid points than it is to e, and hence cannot belong to $VC(e)$. This bounds $VC(e)$ to the two pixels containing ε, proving the lemma. A picture version of this proof appears in Figure 5a.   □
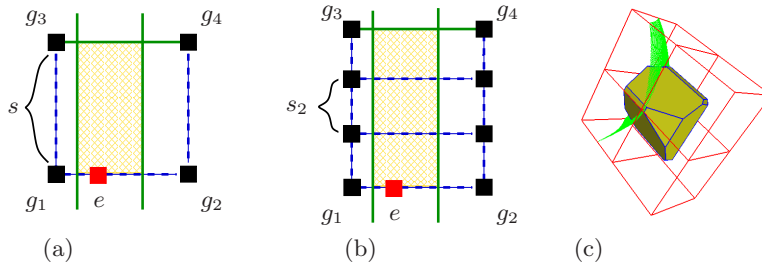
Fig. 5. (a) A picture proof of Lemma 1. The Voronoi cell of the edge point e must lie in the shaded region bounded by the solid lines (green). By symmetry, this restricts the Voronoi cell to a two pixel range. (b) A picture proof of Theorem 2 in the general 2D case. Here, each pixel has dimensions s_1 by s_2 with $\lceil s_1/s_2 \rceil = 3$. By symmetry, the Voronoi cell of e is restricted to a six pixel range. (c) A Voronoi cell of an edge point in 3D, visibly contained within four voxels.

Continuing with the 2D case and the same notation, suppose that each pixel is a rectangle with side lengths s_1 and s_2. Suppose ε has length s_1. As in the proof of the lemma, we can immediately restrict $VC(e)$ to a column of pixels. Consider only the range of $2\lceil s_1/s_2 \rceil$ pixels closest to ε. This range contains the two squares of side length s_1 with ε as one edge. Therefore, we may repeat the analysis in the proof of the lemma using the corners of this range instead of the corners of the pixel. Hence, $VC(e)$ is bounded within $2\lceil s_1/s_2 \rceil$ pixels. Figure 5b shows a simple example.

Now consider the 3D case where each voxel has dimensions s_1, s_2, and s_3 and again assume $e \in E$ lies on the edge ε with side length s_1. There are two planes perpendicular to ε which bounds $VC(e)$ to a flat grid of voxels. In either of the two rectilinear dimensions of this grid, we apply the same analysis as above to conclude that $VC(e)$ intersects at most $2(\lceil s_1/s_2 \rceil + \lceil s_1/s_3 \rceil)$ voxels. Note that this constant reduces to 4 when $s_1 = s_2 = s_3$.

The proof for points of type N is similar. Finally, we note that as we add more E and N points to the diagram, the Voronoi cells can only get smaller.

Therefore, the presence of other E or N points in the diagram is irrelevant to the bound. $\qquad\square$

Theorem 2 has significant implications for solving the ray intersection problem discussed in Section 2. A priori, Voronoi cells may touch an arbitrary number of voxels which cause an explosion in computational time when checking if the closed ball property is satisfied. With Theorem 2, the computational time can be bounded in advance.

Notably, the actual bound depends on the gridding scheme of the input, not the choice of interpolant. Accordingly, the width of the protective layer of grid points collected during the algorithm depends on the gridding scheme as well. For the case of adaptive gridding in an octtree construction, the user must require the "level difference" between two adjacent cells to be no more than some fixed number k. Given k, a loose bound on the number of voxels a Voronoi cell of an E or N point may intersect is $4k^2$, as an edge is incident on at most k^2 cells in each of the other two orthogonal direction. This bound may be improved to 4 for E points and $3k + 1$ for N points by considering limiting cases and using the convexity of Voronoi cells.

We have inserted the G points in order to decrease computations required to check the closed ball property, however we have increased the complexity of the Voronoi and Delaunay triangulations themselves. By Theorem 3 below, the new edges and facets formed by the addition of these G points do not add any significant burden to the closed ball property confirmation process. Further, these facets and edges are used in the algorithm DELVOLMESH described in Section 4.1.

Theorem 3. *Let VF be a Voronoi facet formed between two grid points and VE a Voronoi edge formed among three grid points at any point in the algorithm. Then $VF \cap \Sigma = \emptyset$ and $VE \cap \Sigma = \emptyset$.*

Proof. It suffices to prove the theorem for E points, since the addition of N points only makes existing Voronoi elements smaller. First we treat the 2D case. Consider a pixel with grid points g_1, g_2, g_3, g_4 and two edge points p and q. (If the pixel has more (four) or fewer (zero) edge points, the theorem is vacuous.) The edge points may occur on adjacent or opposite edges of the pixel, as shown in Figures 6 a and b, respectively. We consider the adjacent case first and suppose that the edges on which p and q lie intersect at g_4. Let R denote the region bounded by the rectangle formed with p and q as opposite corners (the shaded yellow regions of Figure 6). Note that $\Sigma \subset R$ since we have chosen a bilinear interpolant. Hence, it suffices to show that the Voronoi edges formed between g_1, g_2 and between g_1, g_3 do not intersect R. If x is a point on the Voronoi edge between g_1, g_2 then by definition $||x - g_1|| = ||x - g_2||$ and $||x - z|| \geq ||x - g_2||$ for all $z \in G \cup E - \{g_1, g_2\}$. However, if $x \in R$ then $||x - q|| < ||x - g_2||$, based on the labelling of Figure 6a. Hence, the Voronoi edge between g_1, g_2 cannot intersect R. Similarly, the Voronoi edge between g_1, g_3 cannot intersect R, proving this case. The case where p and q lie on opposite edges is similar.

The proof for the 3D case is analagous to the 2D case. It suffices to prove the claim for a closed Voronoi facet between two grid points g_1, g_2 as this includes the relevant Voronoi edges. Consider just one of the voxels into which this facet extends. In each of the rectinear directions away from the g_1, g_2 edge, there exists a closest point of E. Taking these two E points and the g_1, g_2 edge, we uniquely define a box. Using the definitions of the Voronoi diagram construction, we can similarly show that Σ lies outside of the box and the Vornoi facet lies inside it, thereby proving the theorem.   □
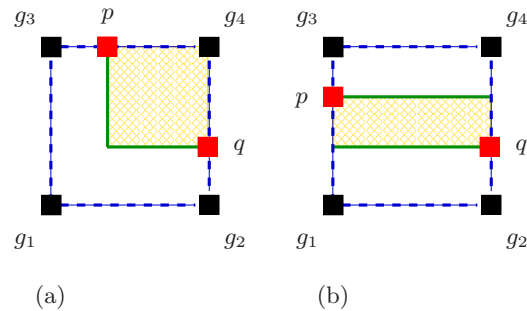
Fig. 6. Proof of Theorem 3 in the 2D case. We show that Σ lies entirely inside the shaded yellow region, while the Voronoi edges formed between two grid points necessarily lie outside of it.

4.3 Simplifying Results

Thus far, we have addressed how the inclusion of the grid points G reduces computational overhead. We now examine how particular choices of the interpolant can further ease the calculations. First we consider the process CBP_VF. If $VF \cap \Sigma \neq \emptyset$, then the intersection is a compact 1-manifold by the transversality assumption. All compact 1-manifolds are homeomorphic to a finite collection of line segments and circles and we distinguish between the case of no circles and at least one circle. If no circles exist, the closed ball property is violated if and only if more than two edges of the facet intersect Σ which is easy to check.

If at least one circle occurs in the intersection of Σ and a Voronoi facet, calculations can become more subtle. In the case of trilinear interpolation, these types of intersections do arise in two fashions. First, there exist configurations of relative function values that produce a tunnel topology inside a single voxel, for example, Case 13.5.2 as defined in [21]. In some cases, a Voronoi facet will pass through the entire tunnel, causing an interior loop in its intersection with Σ. In this case, we can detect the topology of the cell by the function values and add the "shoulder points" as described in [21]. The shoulder points are positioned so that the Voronoi diagram breaks these interior loops on Voronoi facets, thereby easing calculations. Alternatively, it may

occur that the interior loop of a Voronoi facet passes through multiple voxels, which still provides some difficulties. We conjecture that this phenomenon happens only if the interpolated function is non smooth at a voxel edge or vertex and are working to simplify this problem.

Now we turn to CBP_VC. For this process, use of the trilinear interpolant provides much stronger results to simplify calculations. First, if Σ intersects a Voronoi cell, the intersection manifold must have a boundary component by the following Theorem.

Theorem 4. *If each facet of a Voronoi cell VC has empty intersection with Σ and Σ is defined by a trilinear interpolant, then $VC \cap \Sigma = \emptyset$. That is, there cannot be a component of Σ entirely inside a single Voronoi cell.*

Proof. The trilinear interpolation method precludes the existence of isolated surface components within a single voxel. Therefore, every component of Σ has a point of G in its interior and hence is sampled by at least six points of E, corresponding to the six rectilinear directions. Thus, each component of Σ intersects at least six Voronoi cells. A similar proof holds if the data is not given on a rectilinear grid. $\square$

Therefore, the process of checking the closed ball property for a Voronoi cell is reduced to checking if the intersection along the facets of a cell is a single closed curve (or empty). A priori, the possibility remains that the intersection surface is a manifold of positive genus with a disc removed. Detection of such cases has been discussed in detail in [9]. However, if our algorithm is applied in the common case of trilinear interpolation with voxels that are cubes, these difficult cases can also be excluded, leading to a much simpler algorithm.

Theorem 5. *Let VC be a Voronoi cell whose facets and edges satisfy the closed ball property and let Σ be defined by a trilinear interpolant on a uniform grid with voxels that are cubes. Then the closed ball property is satisfied for VC if and only if $VC \cap \Sigma$ has a single boundary component.*

Proof. It suffices to exclude the minimum case where $VC \cap \Sigma$ is homeomorphic to a torus with a disc removed. To generate such a surface by trilinear interpolation on a grid of cubes requires more than four voxels since the intersection of Σ with a voxel face cannot contain closed loops. By Theorem 2, the surface therefore intersects more than one Voronoi cell. $\square$

5 Implementation and Results

We have used CGAL [8] to build and maintain the Voronoi and Delaunay diagram of the point set. We also needed to compute the intersection of a ray with the surface Σ inside certain voxels; for this task, we have used another publicly available library called SYNAPS [25].

As described in the problem statement in Section 1, only the parameter ϵ is needed to run the algorithm on a given data set. This input dictates the desired upper bound on the Hausdorff distance between the surface approximation M and the interpolated surface Σ. Given this parameter, we reduce the amount of computation needed by snapping some of the points of E in the following manner. If there exist points $g \in G$ and $e_0, e_1, e_2 \in E$ such that the e_i are on grid edges incident to g and $d(g, e_i) < \epsilon$ for $i = 0, 1, 2$, we snap the three e_i points onto the common point g. Since all e_i points are within ϵ distance of the grid point and Σ passes through each e_i, the closet point of Σ to g cannot be more than ϵ away, which is within the user specified limit of tolerance.
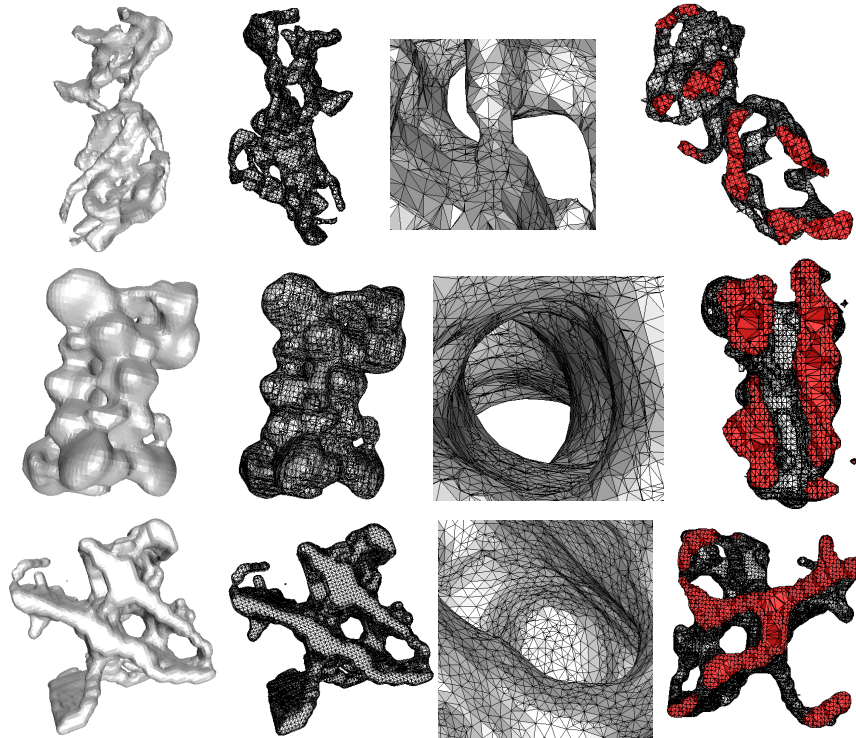
Fig. 7. Performance of the meshing algorithm. The top row shows the geometry, surface mesh, a closeup on the surface mesh and a cut-away of the volumetric tetrahedral mesh for 1CID. The second and the third rows show the same for 1MAG and BONE, respectively.

Although we have reduced the complexity of the ray-surface intersection problem as describe in Section 4, we still have to compute it for a constant number of voxels for every Voronoi edge. This was accomplished by parameterizing the Voronoi edge segment between two Voronoi vertices p_0 and p_1 with a real number t. Inside every candidate voxel, we then detect the in-

tersection points of the segment with the interpolating polynomial function using the Algebraic solver available in the library SYNAPS. If an intersection point lies inside the corresponding voxel, we consider it as a valid intersection.

The performance of the algorithm on biological entities at various scales is shown in Figure 7.

The 1CID data set was obtained via blurring the coordinates of the atoms available from the Protein Data Bank (PDB) [7]. The resulting 3D scalar volume represents the electron density of the protein molecule T Cell Surface Glycoprotein CD4 . The goal in this case is to extract a well-sampled Delaunay conforming surface mesh so that one can analyze the secondary structure using the unstable manifolds of the index 1 and index 2 saddle points of a suitable distance function [5].

1MAG, shown in the second row of Figure 7, is the PDB entry of the ion channel Gramicidin A, obtained from soil bacteria Bacillus brevis. The molecular surface has many tunnels as shown in the left most sub-figure in the second row. However, only the tunnel in the middle is topologically significant as it is important to preserve the property of the ion channel. Once the isosurface has been extracted, it is therefore necessary to remove all the other tunnels and preserve the main tunnel. The algorithm for such removal of topological features has been described in [4] which relies on a Delaunay conforming isosurface in order to detect and remove unwanted topological features.

Finally, BONE data is shown in the third row of Figure 7. This data set is provided by our collaborator from the University of Rome. The goal here is to mesh the internal bone structure for stress-strain analysis.

The size of the 1CID volume data set is 128^3. It took 1 second to build the initial triangulation, 45 seconds to enforce the closed ball property, and 35 seconds to recover the geometry for $\epsilon = 0.00001$. The size of the 1MAG dataset is 128^3. It took 1.5 seconds to build the initial triangulation, 20 seconds to enforce the closed ball property, and 28 seconds to recover the geometry for $\epsilon = 0.00001$. The size of the BONE dataset is 64^3. It took 5 seconds to build the initial triangulation, 65 seconds to enforce the closed ball property, and 89 seconds to recover the geometry for $\epsilon = 0.0001$. Note, the time taken to mesh does not depend on the size of the volume data set since we were able to enforce that the ray-surface intersection calculation is of constant complexity. However, as the level set passes through more voxels, the number of points $(E \cup G)$ increases and that increases the time complexity of the algorithm.

6 Conclusion

We have presented an improvement over traditional level set meshing approaches. In particular, we claimed that the isosurfaces extracted via well-known approaches such as marching cubes or dual contouring are not suitable as they not only fail to capture the topology in a provable manner but also do

not produce sufficient samples for the extracted approximation to be embedded in the Delaunay triangulation of the sampled set of points. On the other hand, there are elegant approaches for meshing general implicit surfaces, yet these algorithms suffer from the computational overhead of computing the intersection of a ray and the level set. Typically, the level set of a sampled scalar function is approximated via a number of piecewise interpolating functions and therefore finding the intersection of a ray with the level set requires a large number of checks. In light of this, we have presented an algorithm which not only overcomes the sampling issue of traditional isosurfacing techniques but also adopts a well-known provable algorithm [9] for generating a good sample of the isosurface efficiently. We have also shown that for the commonly used trilinear interpolant, many of the difficult cases that arise in implementation of the algorithm can be avoided, thereby improving numerical robustness.
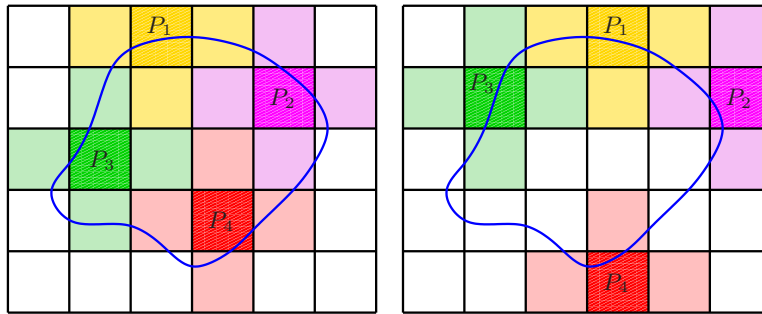
Fig. 8. Data may be partitioned to run the algorithm in parallel. In this 2D example, the voxels indicated by the P_i are to be processed. One can send data from each P_i voxel to a separate processor, along with the collar of protective voxels around it indicated by the same color shading. Two iterations are shown.

Scope

As noted earlier, the scope of this algorithm is immense. First, we have not assumed any particular interpolation scheme in order to prove that we need to check only a few voxels when detecting ray-surface intersection. Therefore, it is possible to extend the algorithm to any higher order interpolant, for example cubic A-patches [3]. Second, adaptive sampling of the scalar function poses no problem to this algorithm as explained previously. Finally, since we exploit the local nature of the interpolated surface Σ, the algorithm can easily be parallelized which is especially important for the large data sets often used in the multiscale biological models we have discussed. Figure 8 indicates how a data set might be partitioned to speed up computational time the analogous 2D case. The different colored pixels marked with P_i can be processed concurrently on separate processors as the Voronoi cells of the

E and N points in those pixels are guaranteed to be inside the protective collar of surrounding pixels. We note that not all pixels can be processed concurrently as the protective collars must be non-overlapping. However, by iterating the process, the most difficult computations can be done in parallel.

Limitations

One limitation of our algorithm is that it tends to generate more samples due to the layer of G points that we use to restrict the search space. Therefore it is necessary to apply adaptive sampling of the scalar function such that bigger voxels are automatically created in regions of less detail and smaller voxels in regions of high detail. By the results of Section 4.2, the efficiency of the algorithm remains the same.

Future Work

The extracted mesh typically has a "gridding" artifact because of the influence of the grid. Further, sometimes the uniformity of the grid causes it to be over-sampled. Therefore, ideally, one would decimate the grid so that the G vertices lie close to the medial axis after the isovalue is known. We plan to develop a scheme for such an optimal placement of the grid vertices. Also, the efficiency of the core computation requires a scaffolding structure. Hence, once the Delaunay conforming surface mesh is extracted, it is not clear how the surface can be decimated (if needed) by throwing away some of the grid vertices, while still efficiently checking that the resulting (decimated) point sample satisfies the closed ball property.

References

1. P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun. Variational tetrahedral meshing. In *SIGGRAPH 2005*, pages 617–625, 2005.
2. D. Attali and J.-O. Lachaud. Delaunay conforming iso-surface, skeleton extraction and noise removal. *Comp. Geom.: Theory and Appl.*, 19:175–189, 2001.
3. C. Bajaj, J. Chen, and G. Xu. Modeling with cubic A-patches. *ACM Transactions on Graphics*, 14(2):103–133, 1995.
4. C. Bajaj, A. Gillette, and S. Goswami. Topology based selection and curation of level sets. In *TopoInVis 2007*, Accepted.
5. C. Bajaj and S. Goswami. Automatic fold and structural motif elucidation from 3d em maps of macromolecules. In *ICVGIP 2006*, pages 264–275, 2006.
6. C. Bajaj, G. Xu, and Q. Zhang. Smooth surface constructions via a higher-order level-set method. In *Proc. of CAD/Graphics 2007*, Accepted.
7. H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Bourne. The Protein Data Bank. *Nucleic Acids Research*, pages 235–242, 2000.

 8. CGAL Consortium. CGAL: Computational Geometry Algorithms Library. *http://www.cgal.org*.
 9. S.-W. Cheng, T. Dey, E. Ramos, and T. Ray. Sampling and meshing a surface with guaranteed topology and geometry. *SCG '04: Proc. of the 20th Annual Symposium on Computational Geometry*, pages 280–289, 2004.
10. S.-W. Cheng, T. K. Dey, and E. A. Ramos. Delaunay refinement for piecewise smooth complexes. In *SODA*, pages 1096–1105, 2007.
11. E. Chernyaev. Marching cubes 33: Construction of topologically correct isosurfaces. *Technical Report. CERN CN/95-17*, 1995.
12. L. Chew. Guaranteed-quality mesh generation for curved surfaces. In *Proc. SoCG '93*, pages 274–280, 1993.
13. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.
14. T. Dey and J. Levine. Delaunay meshing of isosurfaces. In *Proc. Shape Modeling International [to appear]*, 2007.
15. H. Edelsbrunner and N. Shah. Triangulating topological spaces. *Intl. Journal of Comput. Geom. and Appl.*, 7:365–378, 1997.
16. V. Guillemin and A. Pollack. *Differential Topology*. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1974.
17. I. Guskov, A. Khodakovsky, P. Schroder, and W. Sweldens. Hybrid meshes: multiresolution using regular and irregular refinement. In *SCG '02: Proc. of the 18th Annual Symposium on Computational Geometry*, pages 264–272, 2002.
18. K. Hormann, U. Labsik, M. Meister, and G. Greiner. Hierarchical extraction of iso-surfaces with semi-regular meshes. In *SMA '02: Proc. of 7th ACM Symposium on Solid Modeling and Applications*, pages 53–58, 2002.
19. F. Labelle and J. Shewchuk. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. In *SIGGRAPH (to appear)*, 2007.
20. J.-O. Lachaud. Topologically defined iso-surfaces. In *DCGA '96: Proc. 6th Intl. Workshop on Discr. Geom. for Comp. Imagery*, pages 245–256, 1996.
21. A. Lopes and K. Brodlie. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. In *IEEE Transactions on Visualization and Computer Graphics*, volume 9, pages 16–29, 2003.
22. W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM SIGGRAPH '87*, pages 163–169, 1987.
23. B. K. Natarajan. On generating topologically consistent isosurfaces from uniform samples. *The Visual Computer*, 11:52–62, 1994.
24. S. Oudot, L. Rineau, and M. Yvinec. Meshing volumes bounded by smooth surfaces. In *Proc. 14th Intl. Meshing Roundtable*, pages 203–219, 2005.
25. G. D. Reis, B. Mourrain, R. Rouillier, and P. Trbuchet. An environment for symbolic and numeric computation. In *Proc. Internat. Conf. on Mathematical Software*, pages 239–249, 2002.
26. Z. Wood, H. Hoppe, M. Desbrun, and P. Schroder. Removing excess topology from isosurfaces. *ACM Transactions on Graphics*, 23(2):190–208, April 2004.
27. Z. Wood, P. Schroder, D. Breen, and M. Desbrun. Semi-regular mesh extraction from volumes. In *VIS '00: Proc. of the Conference on Visualization 2000*, pages 275–282. IEEE Computer Society Press, 2000.
28. Y. Zhang, C. Bajaj, and B.-S. Sohn. Adaptive and quality 3d meshing from imaging data. In *Proc. of 8th ACM Symposium on Solid Modeling and Applications*, pages 286–291, June 2003.