

# Compatible Triangulations of Spatial Decompositions

William J. Schroeder<sup>1</sup>

Berk Geveci  
Kitware, Inc.

Mathieu Malaterre

## ABSTRACT

We describe a general algorithm to produce compatible 3D triangulations from spatial decompositions. Such triangulations match edges and faces across spatial cell boundaries, solving several problems in graphics and visualization including the crack problem found in adaptive isosurface generation, triangulation of arbitrary grids (including unstructured grids), clipping, and the interval tetrahedrization problem. The algorithm produces compatible triangulations on a cell-by-cell basis, using a modified Delaunay triangulation with a simple point ordering rule to resolve degenerate cases and produce unique triangulations across cell boundaries. The algorithm is naturally parallel since it requires no neighborhood cell information, only a unique, global point numbering. We show application of this algorithm to adaptive contour generation; tetrahedrization of unstructured meshes; clipping and interval volume mesh generation.

**CR Categories:** I.3.5 [Computing Methodologies]: Computer Graphics—Computational Geometry and Object Modeling; Algorithms

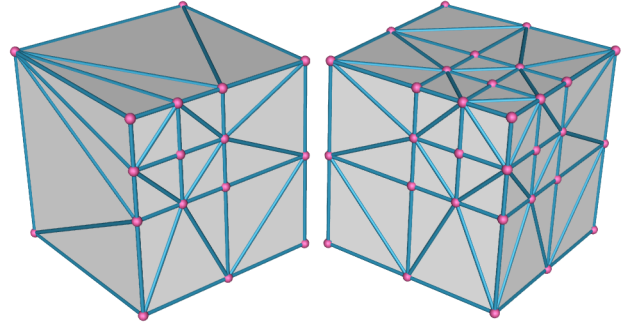
**Keywords:** triangulation, tetrahedrization, adaptive grid, clipping, contouring, template, Delaunay, parallel

## 1 INTRODUCTION

Three-dimensional spatial decompositions are routinely used in graphics and visualization to accelerate computation, model physically-based behavior, and represent sampled data. Adaptive spatial structures such as octrees, KD-trees, adaptively sampled distance fields [10], and adaptive mesh refinement data [2][1] are often employed to generate implicit surfaces or perform multi-resolution visualization of large data sets. Various types of grids, including regular volumes, topologically regular structured grids, and unstructured grids, are used in physically-based modeling applications and to simulate complex physical processes [8][32]. Volume data are also used to represent sampled data, such as that produced from laser digitizing systems [9], or digital sampling processes such as CT, MRI, and confocal microscopy.

Use of spatial decompositions introduces several problems into graphics and visualization applications. The so called crack problem occurs when neighboring cells in an adaptive decomposition exist at different levels of subdivision. Naive contour generation of such a decomposition results in cracks in the isosurface. The cracks may be geometric, where an actual hole exists in the surface, or topological, where T-junctions are created. Cracks may cause rendering artifacts, or introduce undesirable surface features that can adversely affect subsequent modeling operations such as decimation, connectivity analysis, or surface subdivision. Cracks

e-mail: {will.schroeder,berk.geveci,mathieu.malaterre}@kitware.com



**Figure 1.** Compatible triangulations across neighboring octant faces in an octree at different levels of subdivision. An ordered Delaunay triangulation guarantees a unique, compatible triangulation between planar faces of convex cells.

can be prevented by creating compatible triangulations between spatial decompositions as shown in Figure 1, and then operating on the triangulation.

Several rendering and visualization algorithms require the existence of tetrahedral meshes, e.g., [26][18][13][14]. Structured and unstructured grids often consist of hexahedron, prism, wedge, and pyramid cell topologies and require tessellation if such tetrahedral-based algorithms are to be used. A related problem is the triangulation of higher-order basis functions (e.g., quadratic, cubic or other higher-order basis used in finite element analysis) into linear tetrahedra for processing by conventional visualization algorithms. The triangulation of grids must insure that the 1D triangulation of each cell edge is compatible with the corresponding edge neighbor, and the 2D triangulation of each cell face is compatible with the corresponding face neighbor. While template-based approaches for cell tessellation work well for simple spatial decompositions with uniform cell types (e.g., volumes with voxel cells), heterogeneous meshes or higher-order meshes are harder to triangulate. Furthermore, conventional methods require a triangulation of the entire domain, which can bloat memory consumption or require a separate pre-processing step. The algorithm presented here is fast enough so that compatible cell tetrahedrizations can be produced on the fly as each cell is processed. The tetrahedra are then discarded after they are processed to reduce memory consumption.

Modeling operations applied to spatial decompositions introduce additional challenges. Clipping a 3D mesh using an implicit function or scalar contour value introduces complex cuts across cells. Clipping is useful for viewing the interior of structures, or to generate finite element meshes of anatomic structures (i.e., generating an interval mesh between two isocontour values). The faces of neighboring elements must be triangulated to insure inter-cell compatibility. Forming compatible triangulations is very difficult when the boundary of each clipped cell must be compatible with each of its neighbors.

This paper addresses these problems by describing a simple, deterministic algorithm that produces compatible triangulations

between the cells of spatial decompositions. The algorithm requires minimal knowledge of neighboring cells, only the coordinates and a unique id (i.e., point id) for each point defining a cell in the decomposition. In this paper we show how to modify the 3D Delaunay triangulation to generate unique triangulations even when degenerate points are present, thereby guaranteeing compatibility across cell faces. We will also show results for adaptive iso-contouring, mesh triangulation, clipping and mesh generation. Source code for this work is also available on the web.

## 2 RELATED WORK

Several authors have encountered the isosurface crack problem and offered various methods to solve it. As early as 1988 Bloomenthal [4] described an adaptive octree method for generation of implicit surfaces. Edge tracking on the more divided octant face is used to eliminate cracks, requiring complex topological structures to navigate the net of edges on octant faces and on the isosurface. [24] address the problem by aligning the intersection edge generated from the higher resolution octant with the edge generated from the lower resolution octant. While this resolves the crack geometrically, it leaves topological cracks (T-junctions) in the surface that can adversely affect later operations on the mesh, including introducing rendering artifacts. [30] approaches the problem in a similar way, except that the low-resolution edge is subdivided by introducing points from the higher resolution edge segments. The triangle attached to the low resolution edge is also subdivided, creating a fan of triangles. The resulting mesh is watertight with no T-junctions. [29] describe a method to create compatible tessellations of adaptive mesh refinement (AMR) grids by constructing a dual grid to the original. This shrinks the original grid by one level and leaves gaps between blocks of different resolutions. The gaps are then triangulated to produce tetrahedra and wedges, a non-trivial process. The authors mention an approach based on Delaunay triangulation, but reject it in 3D due to the issue of triangulation degeneracies, which this paper addresses.

Generation of meshes from scalar fields is another important operation in modeling and visualization. [17] treats interval tetrahedrization of volumetric data. The idea behind this work is to generate a tetrahedral mesh in the volume between two isosurfaces. In application this can be used to create computational meshes from volume data, such as finite element meshes of bone structure. A principle challenge addressed by this method is to maintain compatibility between adjacent voxel triangulations. Face triangulations require control of face diagonals. Nielson's approach creates compatible triangulations using voxel templates, and is specialized for volume data. Clipping as defined by [23] separates  $n$ -dimensional data into two  $n$ -dimensional parts: one below a specified isosurface value, and one above. Clipping is a specialized form of interval volume tetrahedrization, where one of the isosurface values is set at the extreme range of the scalar data.

Many algorithms require the existence of tetrahedral meshes. Direct volume rendering techniques of non-regular data such as rectilinear grids, unstructured grids, or scattered points may project triangle faces [25][21], or integrate rays through tetrahedra [26]. Tetrahedra are also used to form multiresolution frameworks for volume visualization [33], including methods to decimate tetrahedral meshes [20]. Visualization of irregular grids, such as streamline generation [14] and detection of separation and attachment lines [13] are often most efficient on tetrahedral meshes, or benefit from the simplicity of their linear interpolation functions.

In many cases these algorithms require a preprocessing step that converts the grid into tetrahedra. While this is trivial for volumes or structured grids, for unstructured grids of mixed types (e.g., hexahedra, wedge, pyramid, and tetrahedra cells) the requirement of face compatibility is difficult to satisfy. Meshes based on higher-order basis such as  $p$ -order finite element meshes are also difficult to tessellate in a compatible manner. Such meshes are used to analyze curved geometry and provide high-rate numerical convergence to the solution of partial differential equations [15].

## 3 ALGORITHM

The algorithm described here creates compatible triangulations of spatial decompositions. The algorithm is simple, general, and scalable. It creates unique triangulations on the planar faces of convex cells using a modified Delaunay triangulation algorithm. Non-planar faces, and non-convex cells can be treated by mapping to parametric space, followed by triangulation in that space. The Delaunay triangulation is modified by using a globally unique point id to resolve degenerate cases.

We begin this section by providing some terminology and mathematical background. We then show how these properties can be used to generate compatible triangulations in various applications.

### 3.1 Terminology

We refer to the spatial decomposition of a domain in  $R^3$  as  $D$ , which is a discretization of  $D$  into the closed subdomains, or cells,  $c_i$ , where

$$\bigcup_i c_i = D \quad (1)$$

The cells  $c_i$  are bounded by a finite set of boundary entities  $f_j^m$  of dimension  $m$ , with  $0 \leq m \leq 2$ . A compatible spatial decomposition is one in which the intersection of any two cells  $c_i$  and  $c_j$  is a boundary entity common to both  $c_i$  and  $c_j$

$$c_i \cap c_j = f_k^m \text{ with } f_k^m \subset c_i \text{ and } f_k^m \subset c_j \quad (2)$$

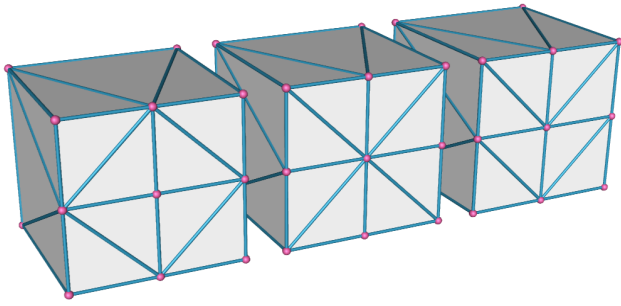
In application  $c_i$  are typically convex and of simple topological structure: a tetrahedron, hexahedron, or octant (possibly with subdivided faces).

A compatible triangulation is also a compatible decomposition, except that all  $c_i$  are simplices  $s_i$  (tetrahedra in  $R^3$ ). A compatible triangulation, or simply a triangulation, can be produced by tessellating each  $c_i$  in  $D$  in such a way as to preserve the properties of Equations (1) and (2).

### 3.2 Delaunay Properties

The Delaunay triangulation  $T(P)$  is a simple method to produce an unique triangulation of the convex hull of a set of non-degenerate points  $P$  [11]. One property of the Delaunay triangulation is that for any  $n+1$  points forming a simplex of dimension  $n$ , the circumsphere of the simplex contains no other points of the set  $P$ . Degenerate cases occur when this property does not hold, for example the eight vertices of a cube all lie on the same circumsphere. In such cases, the triangulation is not unique and can be created by choosing one triangulation of several alternatives, as long as the triangulation remains compatible.

An important property relative to the work presented here is as follows. Given a set of points  $P(c_i)$  that form the triangulation  $T(P(c_i))$ , and a set of points  $\partial P(c_i) \subset P(c_i)$  which lie on a  $m$ -dimensional hyperplane  $H^m$  with  $m \leq n$ , then  $T(\partial P(c_i))$  is a  $m$ -



**Figure 2.** Different orderings of degenerate points on octant faces (divided 2:1) produces different triangulations.

dimensional Delaunay triangulation on  $H^m$ . This property follows because the intersection of a  $n$ -dimensional Delaunay triangulation with  $H$  preserves the property of circumsphere containment [22]. Note that it is possible for edges to pass from points on one side of  $H$  to the points on the other side. However, if the  $\partial P(c_i)$  lies on the convex hull of  $c_i$ , then no such edges can exist, since all points are on one side of  $H$ .

Given this property, a compatible triangulation can be generated from a spatial decomposition using the Delaunay triangulation, assuming that each cell  $c_i$  is convex, each face  $f_k^m$  lies on some  $H^{m-1}$ , and the points  $P(c_i)$  are non-degenerate. Note also that each cell  $c_i$  can be triangulated independently and will generate a compatible triangulation since the faces of each  $f_k^m$  form a unique triangulation.

The ability to create compatible triangulations in an independent manner from cell to cell forms the basis of our algorithm. This approach works well as long as the spatial decomposition consists of convex cells with planar faces and non-degenerate point sets. However, these conditions are rarely met in practice; especially the requirement for non-degenerate point sets. Volumes and octrees are composed of cubical cells (i.e., voxels and octants) that have degenerate points, but are convex with planar faces. Therefore, to extend the applicability of the algorithm, we must address situations when these properties do not hold.

### 3.3 Degenerate Points

As stated earlier, the Delaunay triangulation is unique when the generating point set  $P$  is non-degenerate. Degenerate points introduce ambiguities into the triangulation, requiring an arbitrary choice in the triangulation process. The resulting triangulation has equivalent Delaunay properties as compared to other triangulation representing different choices, and is therefore not unique with respect to the Delaunay property.

To form a unique triangulation, we introduce an additional condition regarding the choice of which triangulation to pick. This condition is simple: we use the order of the point in  $P$ , or equivalently, a unique point id that supports a mathematical ordering operation (e.g., the operations  $<$  or  $>$ ). As Figure 2 shows, the order in which points on a subdivided 2:1 octant face are inserted can be used to control the resulting triangulation. As long as the same order is used in the neighboring octants, the triangulation will be a compatible triangulation.

A variety of point ordering rules are possible. The octant on the left in Figure 2 was triangulated with corner vertices first, followed by edge vertices. Finally, the center face vertex was added to the triangulation. The middle octant in Figure 2 had its central face

vertex inserted prior to its edge vertices. However, the simplest rule is to sort the points according to a unique integer id (or any other unique, sortable id). This will insure that the points on a face  $f_k^m$  are always inserted into the triangulation in the same order. This is true whether  $c_i$  or  $c_j$  (the cells on either side of the face) is triangulated, since the sorted order on  $f_k^m$  will always be the same.

### 3.4 Non-Planar Faces and Concave Cells

Many important spatial decompositions such as volumes and octrees are composed of cells that are convex with planar faces. The ordered Delaunay triangulation technique produces compatible triangulations for these types. However, there are decompositions that may consist of cells with non-planar faces such as finite element meshes (e.g., warped hexahedron or higher-order elements). Note that cells with non-planar faces implies non-convex cell shapes, since a non-planar, convex face on one cell implies a non-planar, concave face on the face neighbor.

The simplest way to treat such situations is to triangulate such cells in parametric space. Many cells such as finite elements are typically defined in an orthogonal  $r$ - $s$ - $t$  coordinate system that ranges from  $-1 \leq r, s, t \leq 1$  or  $0 \leq r, s, t \leq 1$  depending on the formulation [8]. This inevitably introduces degenerate points, since an element like a hexahedron with vertices at  $(\pm 1, \pm 1, \pm 1)$  in parametric space has eight vertices all of which lie on the same circumsphere. The point ordering property is used to disambiguate the triangulation; as a result it will be compatible with its  $f_k^m$  neighbor. Note that for this to be true, the mapping into parametric space must preserve point degeneracies; that is, a degenerate face in parametric space must remain degenerate independent of which face it is mapped to in parametric space. (For example, cells with tetrahedral topology should be mapped into a regular tetrahedron, not into the right-angle tetrahedron found in finite element analysis.)

In principle concave cells with planar faces may exist. This is rare in practical application. Such cells require the use of a constrained Delaunay triangulation, since the Delaunay triangulation is always convex. While simple techniques such as rejecting a simplex whose center is outside of the cell may work in many situations, there are potential cases where the triangulation may not form a valid geometric triangulation [22] with the cell, and simplex deletion cannot resolve the situation. While constrained Delaunay methods are available [5], a better alternative may be to decompose the cells of the spatial decomposition into convex pieces first, followed by triangulation. In this paper we assume that cells are convex or can be mapped into a convex cell in parametric space.

## 4 IMPLEMENTATION DETAILS AND APPLICATIONS

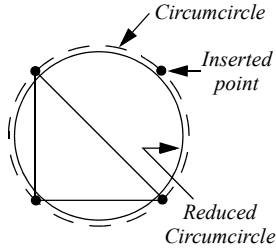
In this section we provide implementation details and show some results for the compatible triangulations algorithm. We apply the method to isocontour adaptive grids, triangulate unstructured grids and perform interval clipping. We begin by describing a simple algorithm for ordered Delaunay triangulation.

### 4.1 Ordered Delaunay Triangulation

The point insertion algorithm introduced by [3] and [28] is a simple  $n$ -dimensional algorithm that can be readily extended to support ordered triangulations. The essence of the algorithm is that it begins with an initial Delaunay triangulation (i.e., a bounding simplex) into which points are inserted one at a time. As each point is inserted, all simplices whose circumspheres contain the point are

deleted. This leaves a star convex insertion cavity with faces of dimension  $n-1$ . Each face of the cavity is connected to the point to form new simplices. The process repeats for each point until all points are inserted. The last step is to eliminate all simplices connected to the vertices of the original bounding triangulation. The remaining simplices form the Delaunay triangulation of the point set  $P$ .

To treat degenerate points via point ordering the basic in/out circumsphere test is modified as follows. Any point laying exactly on the circumsphere is rejected as being outside. In practice, this can be implemented by reducing the radius of the circumsphere by a small factor  $\epsilon$ , typically on the order of  $10^{-9}$  or smaller (assuming that the cell coordinates are in the unit cube parametric space). Thus points injected early take precedence over degenerate points inserted later in the triangulation process.

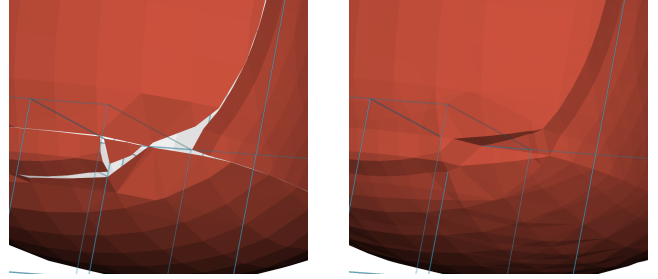
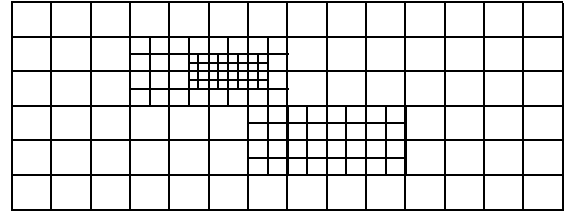


Delaunay algorithms based on floating point arithmetic such as the point insertion algorithm described above are numerically sensitive. Double precision is required, but even then small perturbations in computing the circumcenter/circumsphere can cause inconsistencies in the Delaunay triangulation, and eventual algorithm failure. However, in the application described here, the number of points in  $P(c_i)$  is small, typically on the order of tens of points. Moreover, as described later in Section 4.4, points close to one another and likely to cause numerical problems are merged. Thus the small number of points, combined with attention to point distribution, works well in practice. Further, in our implementation we use an additional check to insure that all faces of the insertion cavity are convex to the inserted point, and adjust the cavity as necessary to insure that this is the case.

The method of point ordering is a memory intensive algorithm that creates and destroys many tetrahedra, faces, and edges. To obtain maximum performance, it is necessary to carefully manage memory allocation and deletion. In our C++ class implementation, a pool of memory is pre-allocated and reused each time a cell is triangulated. Our implementation is less than 750 lines of executable code, reflecting the simplicity of the algorithm. (The source code is available in the VTK class `vtkOrderedTriangulator` available at <http://www.vtk.org>.)

## 4.2 Isocontouring Adaptive Grids

Adaptive grids such as the branch-on-need-octree (BONO) [31] or block-structured adaptive mesh refinement such as CHOMBO [1] can be used to represent data at varying levels of resolution and with low storage overhead. In such grids naive isosurface generation proceeds by visiting each terminal cell (i.e., octant or grid cell at the highest level of resolution at that location in the grid) using standard methods (e.g., marching cubes) to extract the isocontour in the cell, ignoring the difference in subdivision level across edges and faces. A typical result is shown in the lower left-hand side of Figure 3 where cracks are clearly seen between portions of the isosurface generated from different grid resolutions. In many applications such visual artifacts may be ignored; however, in situations where further processing of the isosurface is desirable (e.g., smoothing or decimation) such cracks pose a significant problem since the topology of the mesh is altered.



**Figure 3.** Crack-free isosurface generation in an adaptive mesh refinement (AMR) grid such as that depicted by the image at top. On the lower left, naive isocontouring algorithms produce cracks between differing levels of resolution. On the lower right, compatible triangulations produce crack-free, watertight surfaces. (The thin blue lines show the outline of different blocks in the AMR grid.)

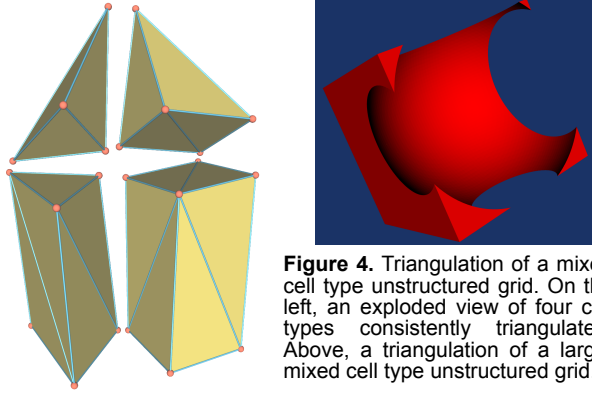
To address this problem, the method of compatible triangulation is employed by modifying the isocontouring algorithm as follows. Each cell through which the contour passes is tetrahedrized by injecting the corner, edge, and face vertices required to match the neighboring grid levels (e.g., Figure 1). The vertices are numbered with a unique integer id, which can be generated based on a logical subdivision of the domain, or can be synthetically generated according to a recursive tree traversal. Once the tetrahedra are generated, each tetrahedra is isocontoured using marching tetrahedra [27]. Because the triangulation is compatible, the isosurface is crack-free with no T-junctions. Note that the entire  $D$  does not need to be triangulated, only those cells through which the contour surface passes. The crack-free result is shown in the right hand side of Figure 3.

## 4.3 Triangulation of Unstructured Grids

Unstructured grids used in finite element analysis often consist of cells of mixed topology such as hexahedra, wedges and pyramids. Tetrahedrization of these meshes must produce triangulations that are compatible across face neighbors. The method of compatible triangulation works well for such situations as long as the triangulation is performed in parametric space. Figure 4(a) is an example of a heterogeneous finite element mesh that has been tetrahedralized using this method. Note that the parametric mapping of each face must be consistent between all cell types that may share that face (i.e., are face neighbors). That is, a triangle face mapped in parametric space for a given cell type must be similar to the matching triangle face in the neighboring cell otherwise the resulting triangulation may be incompatible.

The method of compatible triangulations has also been used to tessellate cells of higher-order basis. The  $p$ -method in finite element analysis approximates the solution field and/or cell geometry with quadratic, cubic, or even mixed/arbitrary order basis. In application, the order of the (typically polynomial) basis is modified in response to an error metric to accelerate numerical convergence. Unfortunately, visualizing the results on such cell types is poorly supported by current visualization tools since they typically pro-





**Figure 4.** Triangulation of a mixed cell type unstructured grid. On the left, an exploded view of four cell types consistently triangulated. Above, a triangulation of a large, mixed cell type unstructured grid.

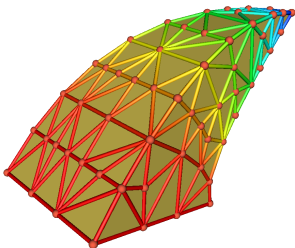
vide representations for up to quadratic basis (and occasionally cubic basis). Current ad-hoc approaches address this problem by subdividing higher-order cells into a pre-specified set of linear cells. This may result in the generation of too many or too few cells. We have successfully employed the method of compatible triangulations to adaptively subdivide the cells as described by the following.

Similar to the 2D method of [7], the process begins with an initial coarse tetrahedrization of the mesh similar to that of Figure 4. (Meshes that have cell types of non-tetrahedral topology are pre-tessellated into tetrahedra. The ordered triangulator can be used for this task if necessary.) Then for each initial tetrahedron, an error metric is evaluated on each of the six edges. If for a particular edge the error exceeds a user-specified criterion, the edge is marked for subdivision and a new point is generated at the mid-point of the edge. Then the ordered triangulator is used to generate a new tetrahedrization of the original tetrahedron (the generation is performed in parametric space). Up to ten points may be injected (four vertices and up to six mid-edge points). The new tetrahedra are then recursively processed and the algorithm continues until the error metric satisfies the convergence criterion. Figure 5 shows the result of this algorithm applied to a quadratic finite element using the method of compatible triangulations.

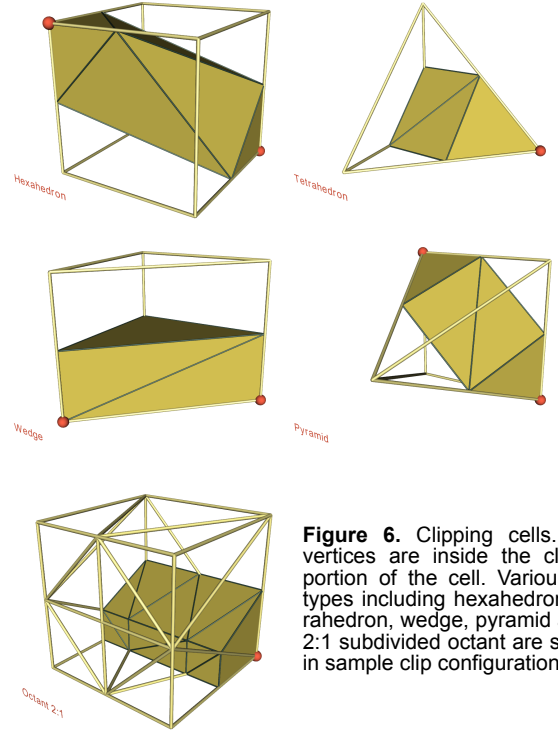
#### 4.4 Interval Tetrahedrization and Clipping

As defined by [17], an isosurface for the scalar field in three dimensional space is given by  $F(x, y, z) = c$  and the interval volume lies between two isosurfaces  $\alpha \leq F(x, y, z) \leq \beta$ . Clipping is defined by the half-sided operation  $C(x, y, z) \geq \alpha$ . It is possible to transform the volume interval into the half-sided clipping function  $F(x, y, z) \geq 0$  by using the equation:

$$C(x, y, z) = \begin{cases} F(x, y, z) - \alpha & \text{if } F(x, y, z) < \alpha \\ \beta - F(x, y, z) & \text{if } F(x, y, z) > \beta \\ \min(F(x, y, z) - \alpha, \beta - F(x, y, z)) & \text{otherwise} \end{cases}$$



**Figure 5.** The method of compatible triangulations used to tessellate a higher-order finite element basis (in this case a quadratic isoparametric element). The tessellation is driven by a function of the edge length and variation in solution values. The tessellation is performed in parametric space.

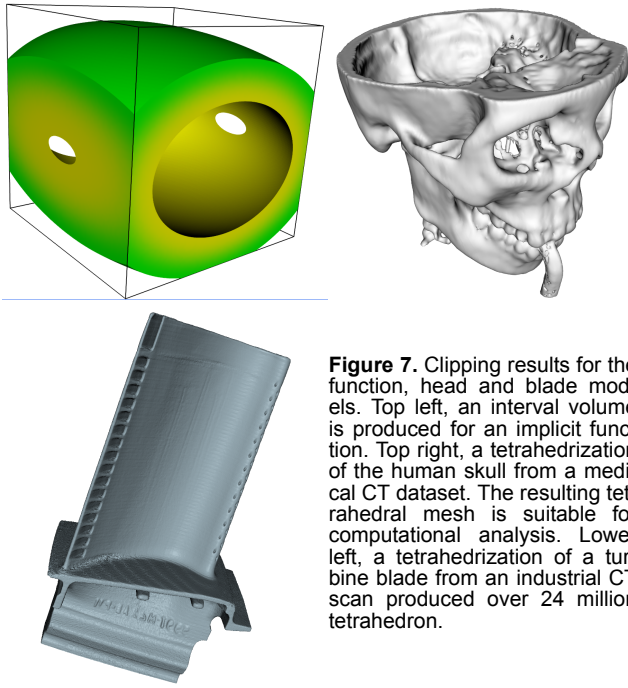


**Figure 6.** Clipping cells. Red vertices are inside the clipped portion of the cell. Various cell types including hexahedron, tetrahedron, wedge, pyramid and a 2:1 subdivided octant are shown in sample clip configurations.

There are two approaches to clipping cells to produce an interval volume using the method of compatible triangulations. The first is to triangulate each cell, and then clip the resulting tetrahedra using standard techniques. The second, which we use in practice, is to generate cell edge intersections from the clipping function and add them to  $P(c_i)$  prior to ordered Delaunay triangulation. Intersection points are given a unique id based on which  $D$  edge was used to generate them. Then all tetrahedra with vertices marked “outside” the clip function are deleted. (As vertices are inserted into the triangulation they are marked “outside”, “inside”, and “on” the clip surface.) This leaves those tetrahedra inside the clipped region. To avoid numerical problems with the Delaunay triangulation due to tetrahedral slivers or pancakes, edge intersection points that are close to nearby cell points are merged with the cell points. Typically a merge tolerance of one one-hundredth of the edge length is used.

The process produces valid tetrahedrizations because the cell faces are convex and clipping produces at most one intersection per edge that is necessarily connected to other edge intersection points by straight lines. Cutting a convex face with one or more lines leaves convex pieces, which are in turn are triangulated correctly because the ordered Delaunay triangulation will always triangulate convex pieces in a unique, and compatible manner.

Figure 6 shows an example of clipping various cell types including tetrahedron, hexahedron, triangular prism (wedge), pyramid and an octant with one face subdivided 2:1. In Figure 7 clipping is applied to three volume datasets—a synthetic implicit function of resolution  $100^3$ , a CT scan of a human head of resolution  $256^2$  by 94 slices, and an industrial CT scan of a turbine blade at resolution  $512^2$  by 300 slices—to produce tetrahedral meshes. Table 1 shows the number of tetrahedra produced in each model and the time to produce the tetrahedrization.



**Figure 7.** Clipping results for the function, head and blade models. Top left, an interval volume is produced for an implicit function. Top right, a tetrahedrization of the human skull from a medical CT dataset. The resulting tetrahedral mesh is suitable for computational analysis. Lower left, a tetrahedrization of a turbine blade from an industrial CT scan produced over 24 million tetrahedron.

#### 4.5 Template Generation for Performance

The ordered triangulation technique is inherently slower than techniques based on table lookup such as that described by [17]. However, it is possible to obtain nearly the same performance as these faster, templated techniques by using the ordered triangulator to generate templates on the fly, caching each template after generation, and then reusing the cached template. Indeed, we have found that generating templates is often easier to implement than manual, a priori creation of templates. For example, the octant configuration of Figure 1 requires  $18!$  distinct combinations (here eighteen points define the convex point set  $P(c_i)$ —a large number of templates to manually generate and store in a pre-computed template table). Using template generation, it is possible to specify the number of stored templates and retain the most frequently used ones to minimize memory resources devoted to template storage.

To generate templates using ordered triangulations, two pieces of information are required: a template *type* and *index*. The template type refers to the topology of the template (e.g., an octant with one edge subdivided 2:1 for a total of nine points). The template index is a permutation index computed as a by product of the sorting operation. That is, when the initial point set  $P(c_i)$  is sorted to produce the ordered point set  $P^s(c_i)$ , a point  $p_i$  in position  $i$  in

the unsorted list  $P(c_i)$  will end up in position  $j$  in the list  $P^s(c_i)$ . The resulting swap from position  $i$  to its final position  $j$  is used to compute the template index as

$$index = \sum_i |i - j|^i \quad (3)$$

where  $|i - j|$  is the number of positions the point moved during sorting.

Once generated, the template index is stored with the list of tetrahedra that it generates in a data structure providing constant or logarithmic time lookup such as an STL map. (In some applications the large number of possible templates discourages the use of constant time data containers such as vectors or linear arrays.) Later, when a cell of the same type and template index is encountered, the template is retrieved and the tetrahedra are produced directly from the template definition.

Table 1 shows the relative timings to produce the interval tetrahedrizations of the implicit function, human skull and the turbine blade data sets.

#### 4.6 Parallel Implementation

The ordered triangulation method scales well in parallel application because minimal boundary information is required between cells. Each cell requires that its vertices have a unique, global point id. These ids can often be determined implicitly, for example a volume extent is enough to determine all the point ids for the voxels contained in that extent. This reduces the need to broadcast point ids across all processors.

Table 1 shows the results of the implicit function, head and turbine blade data sets run with two, four, and eight processors. The parallel system consists of eight dual processor, Windows 2000 nodes connected via gigabit ethernet and MPI communication. Each node was configured with 1 gigabyte memory with 800-MHz processors. A simple load balancing scheme subdivided the volume across the processors. As the table shows, the blade was particularly unbalanced due to extra work required to process the larger dovetail section, while the symmetric implicit function is well balanced and scales linearly with the number of processors. Note also as the models became larger the work required to manage the output (including some swap effects) overshadowed the benefits of templates versus the ordered triangulator. In general, templating improved performance up to an order of magnitude.

### 5 CONCLUSIONS AND FUTURE WORK

We have modified the 3D Delaunay triangulation to produce compatible triangulations of spatial decompositions. The modification

Example	# Tets	Time To Clip (Normalized by Fastest Elapsed Time)							
		1 CPU		2		4		8	
		OT	Template	OT	Template	OT	Template	OT	Template
implicit function ( $100^3$ )	2,001,311	94.69	9.17	48.38	4.56	24.46	2.11	12.32	1.08
				(46.75)	(4.22)	(22.32)	(2.00)	(11.47)	(1.00)
skull ( $252^2$ by 94)	3,315,906	167.44	31.79	83.66	12.44	44.05	5.44	32.83	3.4
				(82.24)	(12.09)	(38.49)	(4.94)	(6.78)	(1.37)
turbine blade ( $512^2$ by 300)	24,695,233	(too big)		747.46	388.95	476.22	161.59	295.32	65.02
				(739.84)	(381.72)	(209.15)	(76.11)	(10.95)	(10.67)

**Table 1.** Clipping examples with one and more processors. Different times (maximum elapsed time per processor) are presented depending on whether or not template generation and caching is used to accelerate the ordered triangulator (OT). Times are normalized by the fastest elapsed time to show relative speed (fastest time was 3.4377 seconds). Times in parenthesis are the minimum elapsed time on a processor to show relative load balance on processors.

requires the use of a unique, sortable labeling of each point in order to resolve ambiguous cases when the Delaunay property is incapable of distinguishing between triangulations. The method can be used in any spatial decomposition where the cells of the decomposition are convex with planar faces; or where cells can be mapped and triangulated in a canonical space with convex shape and planar faces. The method is general and simple: it can be easily employed by providing a list of point containing both coordinates and a unique id. Furthermore, minimal boundary information is required so the algorithm can be parallelized in a scalable manner.

The method described here is general although not as fast as algorithms based on case tables or templated triangulation rules. However, such rules are difficult to create and implement in all but the simplest spatial decompositions consisting of regular structure and a simple cell types (e.g., a volume or a structured grid with a homogeneity of hexahedral cells). Indeed, our method has used to generate case tables on the fly where the most common templates are generated and cached while the less common configurations are delegated to using the compatible triangulation algorithm.

This method can be extended to higher dimensions, since the Delaunay triangulation is inherently an  $n$ -dimensional algorithm. In addition, concave cells, or cells with non-planar faces, can be treated as well by using constrained Delaunay triangulations. However, this introduces a great deal of complexity into the algorithm, diminishing one of its key features, which is simplicity.

## ACKNOWLEDGEMENTS

Some of this work was supported by NSF SBIR Phase II Grant DMI-0238964. Kathleen Bonnell of Lawrence Livermore National Lab provided data and debugging help for this work.

## REFERENCES

- [1] M. Berger and P. Colella. Local Adaptive Mesh Refinement for Shock Hydrodynamics. *Journal of Computational Physics*, 82:64-84. Lawrence Livermore Laboratory Report No. UCRL-97196, 1989.
- [2] M. Berger and J. Olinger. Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations. *Journal of Computational Physics*, 53:484-512, 1984.
- [3] A. Bowyer. Computing Dirichlet Tessellations. *Computer Journal*, 24(2):162-166, 1981.
- [4] J. Bloomenthal. Polygonization of Implicit Surfaces. *Computer Aided Geometric Design*, 5(4), 1988.
- [5] F. Chin and C. Wang. Finding the Constrained Delaunay Triangulation and Constrained Voronoi Diagram of a Simple Polygon in Linear Time. *SIAM Journal On Computing*, 28(2):471-486, 1998.
- [6] P. Cignoni, C. Montani, and R. Scopigno. Tetrahedra Based Volume Visualization. In *Mathematical Visualization—Algorithms, Applications, and Numerics*, H.-C. Hege and K. Polthier (Eds.), Springer Verlag, ISBN 3-540-63991-8, pp. 3-18, 1998.
- [7] A. J. Chung and A. J. Field. A Simple Recursive Tessellator for Adaptive Surface Triangulation, *J. of Graphics Tools*, 5(3), pp. 1-9, 2000.
- [8] R. Cook, D. Malkus, M. Plesha and R. Witt. *Concepts and Applications of Finite Element Analysis, Fourth Edition*. John Wiley & Sons, New York. ISBN 0471356050, 2001.
- [9] B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. In *Proc. of SIGGRAPH 96*, 1996.
- [10] S. Frisken, R. Perry, A. Rockwood and T. Jones. Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics. In *Proceedings of SIGGRAPH 00*, pp. 249-254, 2000.
- [11] L. Guibas and J. Stolfi. Primitives For The Manipulation of General Subdivisions and the Computation of Voronoi Diagrams. *ACM Trans. Graphics*, 4:74-123, 1985.
- [12] M. Hall and J. Warren. Adaptive Polygonalization of Implicitly Defined Surfaces. *IEEE Computer Graphics & Applications*, 10:33-42, 1990.
- [13] D. Kenwright, C. Henze and C. Levit. Feature Extraction of Separation and Attachment Lines. *IEEE Transactions on Visualization and Computer Graphics*, 5(2):135-144, 1999.
- [14] D. Kenwright and D. Lane. Interactive Time-Dependent Particle Tracing Using Tetrahedral Decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):120-129, 1996.
- [15] M.S. Shephard, J.E. Flaherty, K.E. Jansen, X. Li, X. Luo, N. Chevaugneon, J-F. Remacle, M.W. Beall, and R.M. O'Bara. Automatic p-version Mesh Generation for Curved Domains. *J. for Applied Numerical Mathematics*, 2003.
- [16] J. Menon. An Introduction to Implicit Techniques. *SIGGRAPH Course Notes on Implicit Surfaces for Geometric Modeling and Computer Graphics*, 1996.
- [17] G. Nielson and J. Sung. Interval Volume Tetrahedrization. In *Proceedings of Visualization 97*, ACM Press, 1997.
- [18] M. Ohlberger and M. Rumpf. Hierarchical And Adaptive Visualization On Nested Grids. *Computing* 59:269-285, 1997.
- [19] F. Preparata and M. Shamos. *Computational Geometry*. Springer-Verlag, New York, 1985.
- [20] K. Renze and J. Oliver. Generalized Unstructured Decimation. *IEEE Computer Graphics and Applications*, 16(6):24-32, 1996.
- [21] S. Rottgear, C. Teitzel, R. Grossno and T. Ertl. Interactive Volume Visualization of Hierarchical 3D-Meshes by Cell-Projection. *EUROGRAPHICS 99*, 18(3), 1999.
- [22] W.J. Schroeder. *Geometric Triangulations: With Application To Fully Automatic 3-D Mesh Generation*. Ph.D. thesis, RPI, 1991.
- [23] W.J. Schroeder, K.M. Martin and W.E. Lorensen. *The Visualization Toolkit An Object-Oriented Approach To 3D Graphics, 3rd Edition*. Kitware, Inc., 2003.
- [24] R. Shekhar, E. Fayyad, R. Yagel and J. Cornhill. Octree-Based Decimation Of Marching Cubes Surfaces. In *Proceedings of Visualization 96*, IEEE Computer Society. pp 335-342, 1996.
- [25] P. Shirley and A. Tuchman. A Polygonal Approximation to Direct Scalar Volume Rendering. In *Proceedings San Diego Workshop on Volume Visualization, Computer Graphics*, 24(5):63-70, 1990.
- [26] C. Silva and J. Mitchell. The Lazy Sweep Ray Casting Algorithm for Rendering Irregular Grids. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):142-157, 1997.
- [27] G. Treece, R. Prager and A. Gee. Regularised Marching Tetrahedra: Improved Iso-surface Extraction. *Computers and Graphics*, 23(4):583-598, 1999.
- [28] D. Watson. Computing the n-Dimensional Delaunay Tessellation with Application To Voronoi Polytopes. *Comp. J.*, 24(2):167-172, 1981.
- [29] H. Weber, O. Kreylos, T. Ligocki, J. Shalf, H. Hagen and B. Hamann. Extraction of Crack-Free Isosurfaces from Adaptive Mesh Refinement Data. In *Proceedings of the EG+IEEE VisSym in Ascona*, pages 22-30, 2001.
- [30] R. Westermann, L. Kobbelt and T. Ertl. Real-Time Exploration Of Regular Volume Data By Adaptive Reconstruction Of Isosurfaces. *The Visual Computer* 15:100-111, 1999.
- [31] J. Wilhelms and van Gelder, A. Octrees for Faster Isosurface Generation. *ACM Transactions on Graphics*, 11(3):201-227.
- [32] A. Witkin. Physically Based Modeling: Principles and Practice. *SIGGRAPH '97, Course Notes*, 1997.
- [33] Y. Zhou, B. Chen and A. Kaufman. Multiresolution Tetrahedral Framework for Visualizing Regular Volume Data. In *Proceedings of IEEE Visualization 97*. IEEE Computer Society, 1997.