# Quality Mesh Generation in Three Dimensions

Extended Abstract

Scott A. Mitchell* Stephen A. Vavasis†

## Abstract

We show how to triangulate a three dimensional polyhedral region with holes. Our triangulation is optimal in the following two senses. First, our triangulation achieves the best possible aspect ratio up to a constant. Second, for any other triangulation of the same region into $m$ triangles with bounded aspect ratio, our triangulation has size $n = O(m)$. Such a triangulation is desired as an initial mesh for a finite element mesh refinement algorithm. Previous three dimensional triangulation schemes either worked only on a restricted class of input, or did not guarantee well-shaped tetrahedra, or were not able to bound the output size. We build on some of the ideas presented in previous work by Bern, Eppstein, and Gilbert, who have shown how to triangulate a two dimensional polyhedral region with holes, with similar quality and optimality bounds.

## 1 Introduction

Triangulation of polyhedral regions is a fundamental geometric problem for numerical analysis. In particular, if one wishes to solve an elliptic boundary value problem on a three-dimensional domain, it is necessary to discretize the domain with a mesh. If the domain is sufficiently complicated, then the method of *finite elements* is commonly used. In this method the domain is divided into small convex polyhedral regions with a fixed number of faces called *elements*. A common choice for an element in three dimensions is the tetrahedron. Thus, a *triangulation* of the domain is required.

For numerical stability in the finite element method, it is necessary that the tetrahedra have bounded *aspect ratio*. This means that the angle between any adjacent pair of edges of the tetrahedron, or between any edge and a 2-dimensional face not containing it, is bounded below by a constant. For information about aspect ratio bounds in numerical analysis, see Babuška and Aziz [1976].

Our algorithm generates a triangulation for a nonconvex bounded polyhedral domain with holes. In addition, the triangulation is optimal in two respects. In particular, the best possible aspect ratio is achieved for the tetrahedra, and the number of tetrahedra is within a constant factor of the best possible for any triangulation with bounded aspect ratios.

Our work is closely based on earlier work by Bern, Eppstein and Gilbert [1990], who solved the corresponding problem for two-dimensional polyhedral domains. Our running time bounds are probably not the best possible; this is discussed further at the end of the paper. Our optimality condition is slightly stronger than Bern et. al.'s condition. In particular, they show that overall the number of triangles they generate is no more than a constant above optimal. The triangles used in a particular region of the domain, however, could be arbitrarily smaller than what is needed for the triangulation to achieve good aspect ratio. In our triangulation, no tetrahedron is ever more than a constant factor smaller than the tetrahedron at the same location in the best possible triangulation.

Other authors have considered three-dimensional regions, but, to our knowledge, no previous work has simultaneously addressed the problems of optimality, aspect ratio and of complicated nonconvex regions. Indeed, as far as we know, there is no previous algorithm to triangulate a nonconvex three-dimensional re-

*Center for Applied Mathematics, Cornell University, Ithaca, NY 14853. Supported in part by Hughes Research Laboratories, Malibu, CA, and by DARPA under ONR contract N00014-88K-0591, ONR contract N00014-89J-1946, and NSF grant IRI-9006137, by a Xerox summer internship and an NSF PYI award.

†Department of Computer Science, Cornell University, Ithaca, NY 14853. Supported by an NSF Presidential Young Investigator award.



8th Annual Computational Geometry, 6/92, Berlin, Germany
©1992 ACM 89791-518-6/92/0006/0212 ..... $1.50

gion with guaranteed aspect ratio (regardless of the optimality of the triangulation). Our technique is based on an octree partitioning of the domain. This technique has been used before, for example, by Carey, Sharna and Wang [1988]. Chazelle and Palios [1989] consider optimality but are not interested in angle bounds. Chew [1989b] presents an algorithm for two-dimensional polyhedral regions, but does not address the problem of optimal number of triangles. Moore and Warren [1990] address the problem of adaptive mesh generation for box-shaped regions. Mitchell [1987] and [1988] consider the problem of adaptive mesh generation of complex regions in regard to finite element error bounds. Because of the importance of mesh generation, the literature on this problem is very extensive; see for example, the conference proceedings edited by Hauser and Taylor [1986].

Recently, Bern and Eppstein [1990] have surveyed the literature on triangulations, giving more details about earlier work.

Our optimality proof is based on characteristic length functions. Similar functions have been used before by Miller and Thurston [1990], Miller and Vavasis [1990], and Miller, Teng and Vavasis [1991]. We have not seen them used, however, to analyze the construction of a triangulation.

The remainder of this paper is organized as follows. In Section 2 we discuss the formation of the octree. In Section 3 to Section 5 we we discuss the steps necessary to produce a triangulation given an octree. In Section 6 to Section 7 we provide the optimality proof. For the remainder of this introduction, we describe our assumptions.

Our triangulation is denoted $\Delta^{OCT}$. We assume we are given a three-dimensional polyhedral region as follows. There is a list of vertices with coordinates specified. There is also a list of edges and faces. The three lists are mutually linked. The list of edges for each vertex is ordered as they occur around the vertex. Similar orderings are assumed for the other lists. We assume that $P$ is connected; if not, each component could be triangulated separately.

A *face* of $P$, or another polyhedral region, can have either zero dimensions, called a *vertex*, one dimension, called an *edge*, or two dimensions, called a *facet*. In some circumstances we want to regard $P$ itself as the three-dimensional face. Note that the containment relation induces a partial order on faces.

We assume that the polyhedral region is nondegenerate in the following senses. Every two-dimensional face of $P$ has the interior of the polytope on exactly one side. Every one-dimensional edge is incident on exactly two two-dimensional faces. For every zero-dimensional face (vertex) $v$, for every small enough open neighborhood $N$ of $v$, the set $(N \cap P) - \{v\}$ has exactly one connected component. These assumptions may be dropped in future work.

**Interior Angle.** We define the *interior angle* or *angle* between two faces $F$ and $G$ in the cases where $F$ and $G$ are a facet and an edge, two facets, or two edges that have a common intersection $v$. We additionally require that one face is not a subset of the other. We say that two rays $r_1, r_2$ with a common endpoint $v$ can *see* each other if there are points $v_1, v_2$ on $r_1, r_2$ each at distance $t > 0$ from $v$ such that the sector $vv_1v_2$ lies in $P$.

The interior angle between two faces $F$ and $G$ meeting at a vertex $v$ is the minimum over all rays $r_1, r_2$ from $v$ of the angle of the sector $vr_1r_2$. Here we require that $r_1 \in F$ and $r_2 \in G$ and $vr_1r_2 \in P$ in a small neighborhood about $v$. The interior angle between two facets meeting at an edge is the dihedral angle between those facets interior to $P$. A more detailed definition appears in the full paper.

An important constant describing a nonconvex polyhedron $P$ is $\alpha$, the minimum interior angle between any pair of faces of $P$.

A useful measure of the shape of a tetrahedron is its *aspect ratio*, which we define to be the ratio of the radius $R$ of the smallest containing sphere, to the radius $r$ of the largest inscribed sphere. The aspect ratio of a three dimensional triangulation is the maximimum aspect ratio of a tetrahedron in the triangulation.

It can be proved that the aspect ratio of a tetrahedron is bounded above and below by constant multiples of the reciprocal of the sharpest interior angle (as defined above) in the tetrahedron. We can also show that the aspect ratio of any triangulation of $P$ is bounded by $k/\alpha$, where $k$ is a constant.

# 2 Subdivision of $P$ into cubes, the octree

The main data structure we use for our algorithm is an octree. We commonly refer to each node of the tree as a *box*. We associate with each box, $b$, a polyhedral region of $\mathbb{R}^3$ called the embedding of the box and denoted $I(b)$. During the generation of the octree, $I(b)$ is exactly a three dimensional cube. Later boxes are warped and triangulated, changing their geometric structure.

An octree node is either a leaf, or has eight children. The embedding of the eight children of a node are the eight cubes obtained by dividing the embedding of the box in half in each of the three dimensions. We say a node is *split* if it is not a leaf. The process of creating the eight children of a node is called *splitting*.

**Duplicate.** Some boxes in the octree will be *duplicated* into the original node and several new nodes, called duplicates or duplicate boxes. We create duplicates whenever the intersection of the box with $P$ has more than one connected component. Each duplicate

represents the same geometric cube in $\mathbb{R}^3$, but is associated with one connected component of $P \cap I(b)$. We use the notation $P \wedge b$ to denote the component of $P \cap I(b)$ assigned to a particular box $b$.

Whenever we split a box $b$, if $P \wedge b$ is nonconvex a child box $b'$ may have more than one component (i.e., $(P \wedge b) \cap b'$ might have more than one component even though $P \wedge b$ consists of one component). Whenever a box is split, we immediately determine whether any of its children contains more than one component of $P$, and if so, we make duplicates of the child box.

If a $P$ face is incident upon $P \wedge b$ for a box $b$, we say that a box *contains* the face. We maintain pointers between each box and the faces it contains.

**Extended Box.** For a given box $b$, we define the extended box of $b$, $\mathrm{ex}(b)$, such that $I(\mathrm{ex}(b))$ is the cube concentric with $I(b)$ and with each dimension expanded by a factor of 5. We use the notation $P \wedge \mathrm{ex}(b)$ to denote the component of $P \cap I(\mathrm{ex}(b))$ that contains $P \wedge b$. The extended box contains only the $P$ faces of $P \wedge \mathrm{ex}(b)$. Note that $\mathrm{ex}(b)$ is not a box of the octree, but may be constructed from boxes of the octree.

**Adjacent.** Two boxes are called *neighbors* if their embeddings intersect non-trivially, and one is not a duplicate of the other. The *size* of a box $b$ is the length of an edge, which we denote by $h(b)$. We say that box $b_1$ is *adjacent* to box $b_2$ if they are neighbors and there is a point of $P$ common to both, i.e. if $(P \wedge b_1) \cap (P \wedge b_2) \neq \emptyset$. In certain cases such as when a box contains faces meeting at a reflex angle, a box may be adjacent to two duplicates of a second box. We say $b_1, b_2$ are *balance-adjacent* if they are neighbors and there is a point of $P$ common to one of the boxes and the extended box of the other, i.e. if $(P \wedge \mathrm{ex}(b_1)) \cap (P \wedge b_2) \neq \emptyset$ or if $(P \wedge \mathrm{ex}(b_2)) \cap (P \wedge b_1) \neq \emptyset$. Boxes that are adjacent are balance-adjacent, but not all balance-adjacent boxes are adjacent .

**Balance Condition.** If we split a box, we immediately split other boxes to maintain the following invariant called the *balance condition*: No box is balance-adjacent to a box more than twice its size. Certain boxes containing vertices or edges of $P$ are called *protected* boxes, and are exempt from being split by the balance condition later in the algorithm. Nonetheless, the ratio of the size of an adjacent box to a protected box is bounded below by a constant that depends linearly on $\alpha$, as we observe below.

We generate the octree by selectively splitting and duplicating nodes. The goal of splitting and duplicating boxes is to make the boxes small enough so that the intersection of $P$ with the embedding of any box is as simple as possible. However, boxes should not be made too small, as this would lead to an excessive number of tetrahedra in the final triangulation.

It is easy to state the conditions under which we duplicate $b$, namely, whenever $P \cap I(b)$ has more than one component. Nonetheless, the box duplication process is actually the most computationally complicated part of the octree generation algorithm, because determining components of $P \cap I(b)$ is a nontrivial computational task. The duplication process is built upon standard algorithms from computational geometry including point-in-polygon testing and planar sweeps. See below.

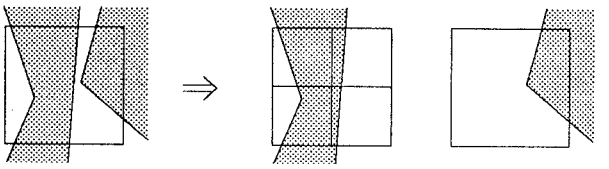

Figure 1: Here a box is duplicated for two components, and one duplicate box is split because it is crowded.

On the other hand, for splitting, the conditions under which a box is split are more complicated, but the computational tasks are straightforward. The octree generation algorithm is divided into three phases, the *vertex phase*, the *edge phase*, and the *facet phase*. We now describe the various steps of the splitting algorithm.

How finely we split boxes during the vertex phase depends on the following definition.

**Vertex Cone.** A *vertex cone* of a box $b$ is a set of $P$ faces, $F_1, F_2, \ldots, F_k$, that satisfy the following: $F_1$ is a vertex lying in $b$, and $F_2, \ldots, F_k$ are the superfaces of $F_1$. Moreover, $F_1, \ldots, F_k$ are exactly the $P$ faces incident upon $P \wedge \mathrm{ex}(b)$.

**Vertex Crowded.** We say that a box $b$ is *vertex crowded* if the following is satisfied: There is a $P$ vertex $v$ in $b$, but the superfaces of $v$ are not the only $P$ faces incident upon $P \wedge \mathrm{ex}(b)$. Equivalently, a box is vertex crowded if $P \wedge b$ contains a vertex that is not the apex of a vertex cone.

We recursively split and duplicate boxes, maintaining the balance condition at each step, until every box containing a vertex contains exactly one vertex cone. Clearly this procedure will terminate once the box sizes become a constant factor smaller than the minimum path length in $P$ between a vertex and a face that does not contain that vertex.

The details of the procedure for determining whether a box $b$ should be split are straightforward given an enumeration of the $P$ faces bounding $P \wedge \mathrm{ex}(b)$. Such an enumeration is obtained from the procedure for determining whether a box should be duplicated described below.

The conclusion of the vertex phase is a special one-time reorganization of the boxes so that each vertex of $P$ is far from the boundary of the box that contains it. The value of this property will become clear in Section 3 and later sections. In particular, after reorganization

the distance between a vertex and any box face is at least $h(b)/8$, where $b$ is the box containing the vertex. This step involves splitting every box at most one more time, and merging the box containing $b$ with some of its neighbors. This is called the centering step and its details are in the full paper.

Once the vertex is centered in its box $b$, this box is *protected*, meaning that it is never split again during the course of the algorithm. Also, protected boxes are never considered to be part of the extended box of another box.

The next phase of the octree generation algorithm focuses on edges of $P$, and is analogous to the vertex phase. The details of this case are in the full paper. An unprotected box contains an *edge cone* if it contains an edge, and if the edge and its two facets are the only $P$ faces contained in $P \wedge \mathrm{ex}(b)$. We define an unprotected box to be *edge crowded* if it contains an edge, and there is some face bounding $P \wedge \mathrm{ex}(b)$ that is not a superface of the edge. We recursively split edge crowded boxes and enforce the balance condition until the only unprotected boxes containing edges are boxes with edge cones. Then we protect all edge cone boxes, and boxes balance-adjacent to them. We do not reorganize in the same way as vertices, relying instead on Section 3.

The third phase of the octree generation algorithm focuses on facets of $P$. An unprotected box $b$ contains a *facet cone* if it contains a facet, and that is the only $P$ face incident on extended $P \wedge \mathrm{ex}(b)$. We say that a box is *facet crowded* if it contains a facet, and if the facet is not the only $P$ face incident upon $P \wedge \mathrm{ex}(b)$. We split unprotected boxes until any box containing a facet contains a facet cone. Boxes containing a facet are then protected. The details are in the full paper.

## 2.1 The duplication process

Recall that we duplicate $b$ whenever it is determined that $P \cap I(b)$ has more than one component. In this section we explain how to identify the components of $P \cap I(b)$. This same techniques allow us to determine the components of $P \cap I(\mathrm{ex}(b))$, which is necessary for determining whether boxes are crowded, and when to propagate the balance condition.

The first part of the duplication algorithm is a preprocessing algorithm. Before the octree generation begins, we identify the separate components of $\partial P$. One component is the *exterior* component; we call all the other components *floaters*. For each facet, edge, and vertex, we can determine which component contains it with a standard graph search. This takes total time $O(n)$, where $n$ is the total number of facets, edges and vertices of $P$.

Once the components are identified, we now perform a second procedure in which we identify a *tether* for each floater. First, for each floater $C$, we identify the point $v$ with the largest $x$ coordinate on $C$, which we call the *base* of the tether. From $v$ we shoot a ray in the positive $x$ coordinate direction and identify the first point $v'$ of $\partial P$ encountered, which we call the *head* of the tether. The head cannot be on the same floater as the base. A head exists, since the ray must eventually pass through the exterior component. The directed segment from $v$ to $v'$ is called the *tether* for $C$. See Figure 2 for a two dimensional example of the tethering structure of a polygon.

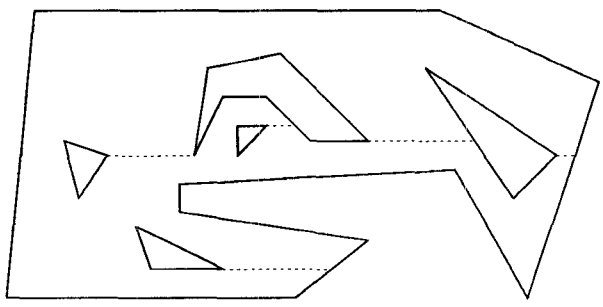

Figure 2: Floaters and tethers for a two dimensional polygon.

Note the following two properties of tethers: (1) Each tether is contained completely in $P$. (2) When regarded as a digraph on the components of $\partial P$, the tethers form an in-tree rooted at the exterior component.

Now we discuss the procedure for determining when a box should be duplicated. Consider the $P$ faces contained in the box. A particular $P$ facet $F$ in the box may intersect the box in more than one component if the facet is nonconvex. We therefore must first determine all the components of each facet passing through the box. This may be done by sweeping a line segment across the facet, requiring $O(m \log m)$ steps if $m$ is the number of vertices of the facet. The total time for identifying all components of all facets for a box is $O(n \log n)$.

Then we perform a combinatorial graph search to identify the various components of $\partial P \cap I(b)$. Let these components be called *sheets*. There are two kinds of sheets: those that intersect the boundary of $I(b)$, and those that are completely internal to $I(b)$. Note that a sheet entirely inside $I(b)$ must be a floater; such a floater is called an *internal floater* for $b$. The other sheets are called *external* sheets. Note that a floater that intersects $I(b)$ is an internal floater if and only if it lies entirely in $I(b)$, otherwise it is one or more external sheets.

Clearly any sheet is incident upon a single connected component of $P \cap I(b)$ since the sheet itself is connected. Thus, the remaining task is to determine which sheets are connected to each other in $P \cap I(b)$.

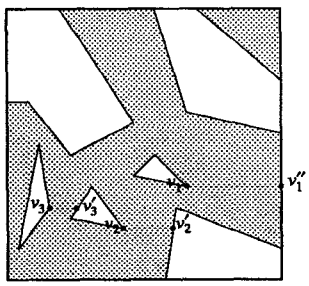


Figure 3: Determining the components of a box. Sweeping the boundary determines two components, and following tethers determines that all internal floaters belong to the same component.

For external sheets, the following algorithm exactly determines how they are connected. We consider the intersection of each sheet with the boundary of $I(b)$ (the surface of a cube). This intersection is a collection of disjoint simple polygons. We form the list of polygons for all sheets. We call these polygons the "rims." Note that these polygons break across edges of the cube. We then sweep over the boundary of I(b) by the sum of $x, y, z$ coordinates, forming a tree indicating which rim polygons are contained in others in $O(n \log n)$ time. From the sweep we can build a rooted tree indicating the containment hierarchy among the polygons. The reader unfamiliar with point-in-polygon tests and plane sweeps should consult Edelsbrunner [1987] or Preparata and Shamos [1985].

Thus, from an $O(n \log n)$ procedure we can determine exactly which external sheets are in the same components of $P \cap I(b)$. This leaves the problem of determining the component of $P \cap I(b)$ for an internal floater $C$. This is determined from the tether $(v, v')$ of $C$. If the head $v'$ is inside $I(b)$, then the floater lies in the same component as the facet containing $v'$ (because the entire tether is in $P$). This facet might be a facet of another internal floater, but because the tethers form an acyclic graph, eventually we will reach a point $v'$ on an external sheet. See Figure 3.

The other case is that the tether's head $v'$ lies outside $I(b)$. In this case we find the point $v''$ where the tether intersects the surface of $I(b)$. If we knew where in the polygon tree $v''$ lay, then we could determine the component containing $v''$ (which is the same component containing the floater). The polygon regions containing $v''$ are easily determined if, before starting the polygon containment sweep, we first insert $v''$ into the list of polygons as a singleton polygon. This should be done for all tethers. This does not change the running time bound of $O(n \log n)$.

Thus, in time $O(n \log n)$ we can determine all the components of $P \cap I(b)$. Moreover, it is easy to see that the rims allow us to determine which neighboring boxes are adjacent to a given duplicate box.

Finally, we need to determine the facets adjacent to $P \wedge \mathrm{ex}(b)$ for testing crowdedness. This is done by running the component-determination algorithm on $\mathrm{ex}(b)$, and then saving the component of $P \cap I(\mathrm{ex}(b))$ that contains a point in $P \wedge b$. This also allows us to determine which neighboring boxes are balance-adjacent to $b$.

We now consider the running time of constructing the octree. We believe that our current running time analysis is suboptimal, so we omit some of the details. First, there is a slight addition to the algorithm as described so far that gives us a better bound. If a box with vertex $v$ is vertex crowded, in time linear in the number of $P$ faces in the extended box, we determine the closest face to $v$ that is not a superface of $v$, and immediately split the box down to a size smaller than this distance. Similarly if the box is edge or facet crowded.

This allows us to claim that the total number of boxes constructed by the octree algorithm is bounded by a constant multiple of the number of leaf boxes that contain a point of $P$. From the warping and triangulating rules to follow, each such leaf box leads to at least one tetrahedron. In particular, the number of such leaf boxes is bounded by $\gamma$, the size of the output. Finally, the total amount of time spent on each box is at most $n \log n$, where $n$ is the number of faces of the polyhedral region $P$. Thus, a bound on the running time is $O(\gamma n \log n)$. Note that this subsumes the $O(n^2)$ preprocessing to find tethers, since $\gamma = \Omega(n)$. See Section 8 for more remarks on this bound.

We claim that after the octree algorithm is finished, we have the following result for relative sizes of boxes.

**Theorem 1** *Let $B$ be any box of the octree whose embedding contains a point of $P$, and $b$ an adjacent box. Then*

$$h(b) \geq k \cdot \alpha \cdot h(B),$$

*where $k$ is a constant.*

The proof of this theorem involves many lemmas and is contained in the full paper. The general idea is as follows. If a protected vertex box is adjacent to a box with an edge, $E$, then that edge box could be much smaller than the vertex box. However, if it is much smaller, this means that there is another face, $F$, not containing $E$, but very close to $E$ as it emerges from the protected vertex box. But $F$ is in the extended vertex box, and hence both it and $E$ must be a superface of the vertex in the protected vertex box. That is, we have an interior angle defined between $E$ and $F$. Recalling that their common vertex is far from the boundary of the protected vertex box, the edge box is only small when this interior angle is small. It is surprising that the result also holds for boxes with facets adjacent to a

protected vertex box. The proof for that case requires extensive analysis and appears in the full paper.

# 3 Warping box faces

*Warping* consists of moving box vertices. We warp the faces of a box away from $P$ faces to achieve good aspect ratio when we triangulate. It is only necessary to warp box faces that are close to $P$ faces, where the definition of "close" is below. Note that $P$ vertices are already guaranteed to be well separated from box faces because of the special reorganization at the end of the vertex phase. Thus, we only need to warp for $P$ edges and $P$ faces.

The warping is done in two passes. The first warping pass is for box edges close to $P$ edges. We say that a box edge $E$ is *close* to a $P$ edge $F$ if the distance from $E$ to $F$ is less than $h/8$. Here, $h$ is the size of the smallest box sharing the box edge $E$. Note that the boxes containing $E$ are all edge-protected, since they are all adjacent to a box containing $F$. This means that the sizes of these boxes are all within a factor of two from each other.

For every close edge $E$, we move it away from $F$ as follows. We move every point on the edge, including the endpoints, by distance $h/8$ in the direction that is orthogonal to $E$ and $F$, oriented away from $F$. (If $E$ and $F$ happen to be parallel, we move the points of $E$ in the direction orthogonal to $E$ and coplanar with $F$.)

Even though the boxes after warping have complicated shapes, we still let the "size" of a box $b$ be its prewarped edge length, and we continue to use notation $h(b)$ for this size.

The second warping pass is for box vertices close to $P$ facets. We say that a box vertex $v$ is *close* to a $P$ facet $F$ if the distance between them is at most $h/16$, where $h$ is the size of the smallest box containing $v$. For such a vertex, we move it distance $h/16$ away from the facet in the direction orthogonal to the facet.

An unfortunate consequence of moving box vertices is that for some box facets, the vertices of the facet are no longer coplanar. This inconsistency is removed when we triangulate box facets in the following section.

# 4 Two dimensional triangulation

After warping, we triangulate facets of boxes. This is a preliminary step to the three dimensional triangulation algorithm of the next section. Suppose two boxes $b, b'$ are adjacent, and suppose their intersection is a facet $S$. If we assume that $h(b) \geq h(b')$, then $S$ is a facet of $b'$. Under these circumstances, we always triangulate $S$ by considering it a facet of $b'$, not $b$. (If $b, b'$ are the same size, we break ties arbitrarily). This means that we can always assume that when triangulating a square $S$, there is no box vertex in the interior of $S$. There are four box vertices at the corners of $S$, and there may be additional box vertices at points along edges of $S$.

Note that a protected vertex box (after the centering step described in the full paper) is always adjacent to boxes the same size or smaller. This means that, without loss of generality, we do not have to triangulate and warp facets of a protected vertex box; instead we can triangulate and warp the facets of the boxes adjacent to it as in the last paragraph.

We now describe the triangulation of a box facet $S$. Note that after warping, the points of $S$ are not necessarily coplanar, although the points are nearly coplanar because the warping distances are small.

The triangulation of a box facet $S$ breaks down into four cases, and cases C and D have two subcases. The cases depend on which $P$ faces pass through the facet. Note that we only have to triangulate the portion of $S$ interior to $P$, which we call $\pi$. Note that $\pi$ is bounded by a closed path of line segments. We can think of $\pi$ as a "perturbed polygon" (since the vertices are nearly coplanar). The six subcases are illustrated in Figure 4; the regions $\pi$ are shaded. Each vertex of $\pi$ is the intersection of a $P$ edge with a box facet, or the intersection of a $P$ facet with a box edge, or a box vertex.

*Case A, no P faces pass through the facet S.* In this case, we put in a new point $v$ at the center of the prewarped box facet, and we connect $v$ to every segment along the boundary of $S$.

*Case B, two P facets F, G and their common edge E pass through facet S.* Let $v$ be the $\pi$ vertex where $E$ passes through the prewarped version of facet $S$. Then we connect $v$ to all segments on the boundary of $\pi$, triangulating $\pi$.

*Case C, one P facet F passes through (two edges of) the facet S.* Let $x, y$ be the two points where $F$ passes through two edges of $S$, and let $v$ be the midpoint of $x, y$. Let $c$ be the center of $S$ (before warping). Let $h$ be the edge length of $S$ (before warping).

*Subcase C1, v is within h/4 of c.* Then we connect $v$ to every segment along the boundary of $\pi$. This divides the region into triangles.

*Subcase C2, v is further than h/4 from c.* Then we connect $c$ to every segment along the boundary of $\pi$. This divides polygonal region $\pi$ into triangles and quadrilaterals. Then we insert a diagonal (arbitrarily) into each quadrilateral, triangulating $\pi$.

*Case D, two P facets F, G pass through facet S, but no P edge.* Let $x, y$ and $x', y'$ be points where $F, G$ respectively cross through the boundary of $S$. Let $v, v'$ be the midpoints of these segments. Let $c$ be the center of $S$ (before warping). Let $h$ be the edge length of $S$ (before warping).

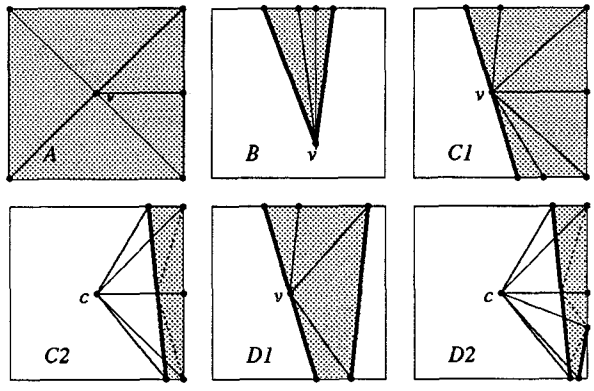


Figure 4: The six cases for triangulating a box facet.

*Subcase D1, $v$ or $v'$ is within $h/4$ of $c$.* Say $v$ is within $h/4$ of $c$. Then we connect $v$ to every segment along the boundary of $\pi$. This divides $\pi$ into triangles.

*Subcase D2, $v$ and $v'$ are both further than $h/4$ from $c$.* Then we connect $c$ to every segment along the boundary of $\pi$. This divides $\pi$ into triangles and quadrilaterals. Then we insert a diagonal (arbitrarily) into each quadrilateral, triangulating $\pi$.

In the full paper, a case analysis of the smallest possible distance between $\pi$ vertices and an analysis of the angles at $v$ or $c$ subtended by an edge of $\pi$ results in the following theorem.

**Theorem 2** *Let $B$ be a box. For any triangle $T$ on a facet of $B$, let $r$ be the radius of the largest inscribed circle. Then $r \geq k \cdot \alpha \cdot h(B)$, where $k$ is a constant.*

General results have been obtained for two-dimensional triangulations with guaranteed inscribed-circle radius bounds. Bern, Edelsbrunner, Eppstein, Mitchell, and Tan [1991] have a result concerning optimal two-dimensional triangulation of polygons, assuming new points cannot be introduced. Bern, Dobkin, and Eppstein [1991] have similar results in the case that new points can be introduced. We have not been able to incorporate these algorithms in the present version of our triangulation algorithm because we need to introduce new points, but the new points have to be introduced in a carefully controlled fashion.

# 5 Three dimensional triangulation

We now describe how to form tetrahedra from the triangles in the last section. We triangulate on a box by box basis. The details of how we triangulate depends on a case analysis of what is contained in a box, and how it intersects the box. However, the general principle is to find a central vertex, and then form one tetrahedron or prism for each triangle in the box by taking the convex hull of this vertex and the triangle. The organization of the argument is very similar to the two-dimensional triangulation in the last section. We cover two of the possible cases here; all the cases are enumerated in the full paper.

**Vertex box tetrahedra.** For a box $b$ containing a vertex of $P$, the three-dimensional triangulation is particularly easy. We take as the central vertex $v$ the $P$ vertex itself. For each of the triangles of $\pi$ on the surface of $b$, we form a tetrahedron by taking its convex hull with $v$.

**Facet box tetrahedra.** Consider a box $b$ containing one $P$ facet $F$. We introduce a central vertex $v$ at the centroid of the prewarped box. If $v$ is within distance $h(b)/4$ to $F$, then we move $v$ to the closest point of $F$ to $v$.

We now have three cases. The first case is if $v$ lies in the interior of $P$, but not on $F$. For each $\pi$ triangle on the boundary of the box we form a tetrahedron by taking its convex hull with $v$. We also triangulate the polygonal region of $F \cap (P \wedge b)$ and form a tetrahedron for each triangle of $F \cap (P \wedge b)$ by taking its convex hull with $v$. This polygonal region is nearly convex (it would be convex if the triangles on each the surface of the box were coplanar). To form a triangulation of $F \cap (P \wedge b)$, we make a new vertex centrally located in the region and take its convex hull with each edge of the region, as in Case A of the two dimensional triangulation.

The second case is that $v$ lies on $F$. We take the central vertex $v$, and proceed as if it were a vertex of $P$. That is, for each $\pi$ triangle of $I(b)$, we generate a tetrahedron by taking the convex hull of it and $v$.

The third case is if $v$ lies outside of $P$. For each $\pi$ triangle of $I(b)$ we form a tetrahedron by taking the convex hull of it with $v$. These tetrahedra are clipped by $F$ into "prisms" with nonparallel sides, and a triangular top and bottom. Zero, one or two of the vertices of the top may also be vertices of the bottom. If two vertices are shared, then a prism is a tetrahedron. If one vertex is shared, a prism is split into two tetrahedra by introducing a facet between the shared vertex, and one distinct vertex of the top and one distinct vertex of the bottom. If no vertices are shared between the top and bottom, a prism is split into three tetrahedra by introducing two facets. One facet is between two of the top vertices and the opposite bottom vertex. The other facet is between the opposite bottom vertex, one of the other bottom vertices, and the opposite top vertex.

# 6 Aspect ratio of tetrahedra

Our triangulation is optimal in two respects. First, the maximum aspect ratio among tetrahedra of our triangulation is optimal up to a constant multiple. Second, compared to all other triangulations of fixed aspect ratio, our triangulation has the minimum number of tetrahedra up to a constant multiple. In this section we establish the first optimality property, focusing on the optimality of the aspect ratio.

Recall that the tetrahedra we form are either the convex hull of a central vertex and a triangle on the surface of a box or a facet of $P$, or else a portion of a prism. We now want to argue about the aspect ratio of all tetrahedra arising in the algorithm of the previous section.

There are three types of tetrahedra arising in the triangulation. A *Type A* tetrahedron has one vertex $v$ centrally located in the box, and the other vertices on a box face. This type of tetrahedron arises in the cases of triangulating a box with a vertex, a box with an edge, or a box with a facet in which the vertex $v$ in the last section lies near the center of the box, is in $P$, and is not close to the second facet (if the box has two facets). Also some of the tetrahedra arising from the case of two facets in a box and $v$ on one of the facets are of this type.

A *Type B* tetrahedron arises only in the two-facet case described in the full paper.

A *Type C* tetrahedron arises from a vertex $v$ in the last section outside $P$. This happens only with boxes with one or two facets and no edges. This causes $P \wedge b$ to be divided into prisms, and then each prism is divided into tetrahedra. These tetrahedra arise in cases analogous to triangles in cases $C2$ and $D2$ in Figure 4.

Intuitively, all three types of tetrahedra have aspect ratio at most $1/\alpha$, because they involve a base (the base being a triangle) whose inscribed radius is between $k_1\alpha h(b)$ and $k_2 h(b)$, and the apex is well-centered over the base, and is distance between $k_3\alpha h(b)$ and $k_4 h(b)$ from the base. Here, $k_1, \ldots, k_4$ are constants.

The aspect ratio computations are very technical and involved. In the full paper we prove aspect ratio bounds for all types of tetrahedra by constructing an inscribed and a containing sphere. These proofs are omitted here. The containing sphere has radius no more than the box size, and the inscribed sphere has radius at least a constant times the size of an adjacent box.

Hence the result is that we can prove a theorem showing that the aspect ratio of any tetrahedron in our triangulation is bounded by a constant multipled by the ratio between $h(b)$ and $h(b')$, where $b, b'$ are adjacent boxes. We already know from Theorem 1 that this ratio is bounded above by $k/\alpha$.

We can also show that any triangulation of $P$ must have a tetrahedron of aspect ratio at least $k/\alpha$ by considering the tetrahedron that must fit "inside" the sharpest interior angle of $P$. The details are in the full paper.

Let $T$ be a three dimensional triangulation. Let $A(T)$ be its aspect ratio, defined to be be the maximum over all tetrahedron $t \in T$, of the aspect ratio of $t$. We may now state the theorem concerning the optimality of the aspect ratio of our triangulation of $P$.

**Theorem 3 (Aspect ratio optimality)**

$$A(\triangle^{OCT}) \leq kA(\triangle^*)$$

*where $k$ is a true constant, independent of $P$ and $\alpha$, and $\triangle^{OCT}$ is the triangulation of $P$ arising from our algorithm, and $\triangle^*$ is any other triangulation of $P$.*

# 7 Optimality of the cardinality of $\triangle^{OCT}$

In this section we prove that the number of tetrahedra in our triangulation is no more than a constant factor larger than any other triangulation with a bounded aspect ratio. The reasoning in this section is as follows. We first prove that any triangulation with bounded aspect ratio, which we denote $\triangle^*$, has certain geometric properties concerning how fast the sizes of the tetrahedra can change. We use these geometric properties to derive upper bounds on how large the tetrahedra of $\triangle^*$ can be in terms of the boxes of our octree. We also give lower bounds on how small the tetrahedra of our triangulation $\triangle^{OCT}$ can be compared to the size of boxes in our octree. Comparing these two sets of bounds shows that our triangulation is within a constant factor of optimal.

**Conformal.** Any triangulation $\triangle^T$ of $P$, including our triangulation, that we consider in this paper must be *conformal* to $P$. This means that the following condition must hold. Let $x$ be a point on a $P$ face $F$. Let $F'$ be the lowest dimensional face of $\triangle^T$ containing $x$. Then $F' \subset F$.

**Characteristic length function.** We define $f_T : P \rightarrow \mathbb{R}$ to be the *characteristic length function* of a triangulation $\triangle^T$. This function is defined as follows. If $x$ is a point of $P$, then we define $f_T(x)$ to be the longest edge among all tetrahedra that contain $x$.

**Aspect ratio bound.** We let $A$ denote the maximum aspect ratio of all tetrahedra of $\triangle^*$. In rest of this section there will be many constant factors that depend on $A$, usually denoted $c_1, c_2, \ldots$.

The following is one of many lemmas that we prove in the full paper relating aspect ratio to how the characteristic length function of $\triangle^*$ may change over $P$.

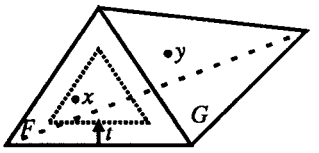


Figure 5: An illustration of case 2. in Theorem 4.

**Lemma 1** *Given a bounded aspect ratio triangulation $\triangle^*$, two points $x, y$, and any two faces $F, G$ containing $x, y$ respectively, define $H = F \cap G$. If $H \neq \emptyset$, then there exists a constant $c_1$ depending on $A$ such that $f_*(x) \geq c_1 \cdot f_*(y)$.*

**Geodesic distance.** We let $\text{dist}_P(F, G)$ denote the *geodesic distance* in $P$ between two closed subsets $F, G$ of $P$. In other words, this is the length of the shortest path in $P$ between a point of $F$ and a point of $G$. This distance is always at least as great as the Euclidean distance.

We now state the following theorem about the geodesic distance between points on faces in a triangulation. The theorem has two cases, depending on whether the faces have a common point or not. See Figure 5.

**Theorem 4** *Let $F, G$ be two faces of $\triangle^*$. Let $H = F \cap G$. Let $x, y$ be two points such that $x \in F$, $y \in G$. Then there exist constants $c_2$ and $c_3$ depending on $A$, such that*

1. *If $H \neq \emptyset$ and $\text{dist}_P(x, H) \geq t$, then $\text{dist}_P(x, y) \geq c_2 t$.*
2. *If $H = \emptyset$, then $\text{dist}_P(x, y) \geq c_3 f_*(x)$.*

Combining the previous theorem and lemma shows the following.

**Theorem 5** *For a triangulation $\triangle^*$ with aspect ratio bound $A$, there exist constants $c_4$ and $c_5$ dependent on $A$ such that for all $x, y \in P$,*

$$f_*(x) \leq \max(c_4 \cdot f_*(y), c_5 \cdot \text{dist}_P(x, y)).$$

An observation about our triangulation $\triangle^{OCT}$ is that the characteristic length function at a point in a box is always bounded below by a constant multiple of the size of an adjacent box. Thus from Theorem 1 and Theorem 3, we have that the characteristic length function at a point in our triangulation is always at least the size of the box containing that point times a constant multiple depending on $A$.

We can prove that any triangulation $\triangle^*$ with aspect ratio bounded by $A$ has its characteristic length function bounded above by a constant multiple of the box sizes in our octree, where this constant depends on $A$. We first note that if a box $b$ containing a point $x$ on a $P$ face $F$ was split to a certain size, then we may find another $P$ face $G$ that does not contain $F$, and a point $y \in G$ such that $\text{dist}_P(x, y) \leq 6\sqrt{3}h(b)$.

Consider a vertex $x$ of $P$. We may apply Theorem 4 Case 2 to the $P$ faces $F$ and $G$ of the last paragraph ($F$ is $x$, and $G$ is disjoint from $F$) to show that $f_*(x)$ is at most a constant factor larger than the size of the box containing $x$. As in Theorem 4, this constant factor depends on $A$. From Theorem 4 and Lemma 1 we can show that this result also holds for an arbitrary point in a protected vertex box.

For a point $x$ on a $P$ edge $F$, we use a similar argument. We find a distinct $P$ face $G$ and $y \in G$ as in the vertex case. By conformality, there are two faces of $\triangle^*$ contained in the two faces $F$ and $G$ of $P$, respectively, that contain our points $x$ and $y$. We use Theorem 5 and Theorem 4 applied to these triangulation faces, and the fact that we have proved the result for $P$ vertices, to conclude that $f_*(x)$ is at most a constant factor larger than the size of the box containing $x$. Again this constant factor depends on $A$. The result generalizes to an arbitrary point of $P$ in a protected edge box. Similar arguments prove the result for points in a protected facet box, and finally for an unprotected box (containing no $P$ faces). The full proof appears in the full paper.

Thus, we show that the tetrahedra in $\triangle^{OCT}$ are never more than a constant factor smaller than those in $\triangle^*$. A straightforward volume counting argument then gives the following result.

**Theorem 6** *For all polytopes $P$ and constants $A$, there exists a constant $c''$, dependent only on $A$ and $\alpha$, such that*

$$|\triangle^{OCT}| \leq c''|\triangle^*|,$$

*where $|\triangle^*|$ denotes the number of tetrahedra of $\triangle^*$ and similarly for $|\triangle^{OCT}|$.*

# 8 Conclusions

An important open question is the running time. We demonstrated a running time bound of $O(\gamma n \log n)$ for our algorithm, which is inferior to Bern, Epstein and Gilbert's running time of $O(n \log n + \gamma)$ for two dimensional regions. However, we have recently found out by private communication with those authors that this bound does not actually hold for the algorithm that they propose for two-dimensional nonconvex polygons. Therefore, it is not clear what is the best possible running time for triangulation of nonconvex polyhedral regions in either two or three dimensions.

It would be interesting to generalize our algorithm to work for higher dimensional regions. Also, the non-

degeneracy conditions on the input region could be relaxed.

Another open question concerns optimizing the tetrahedra for properties other than aspect ratio. For example, is there an algorithm for triangulating a three-dimensional nonconvex polyhedral region to maximize inscribed sphere radius of the tetrahedra?

Finally, we have plans to implement this algorithm. The two-dimensional analog of this algorithm has been implemented by the first author in C++.

## Acknowledgement

Thanks to Paul Chew of Cornell for discussing various triangulations with us. Thanks to Marshall Bern, David Eppstein and John Gilbert of Xerox for help with their earlier work. Thanks to Joe Mitchell of Cornell for helping with the plane sweep algorithm.

## References

I. Babuška and A. K. Aziz [1976], On the angle condition in the finite element method, *SIAM J. Num. Anal.* 13:214–226.

M. Bern, D. Dobkin, and D. Eppstein [1991], Triangulating polygons without large angles, preprint.

M. Bern, H. Edelsbrunner, D. Eppstein, S. Mitchell, and T. S. Tan [1991], Edge-insertion for optimal triangulations, preprint.

M. Bern, D. Eppstein, J. Gilbert [1990] Provably Good Mesh Generation, Xerox Palo Alto Research Center. Also, *Proc. 1990 Symposium on Foundations of Computer Science.*

M. Bern and D. Eppstein [1991], Mesh Generation and Optimal Triangulation, preprint.

G. Carey, M. Sharna, and K. Wang [1988], A Class of Data Structures for 2-D and 3-D adaptive mesh refinement, *Int. J. Numer. Meth. in Engr.* 26:2607–2622.

B. Chazelle, L. Palios [1989], Triangulating a Nonconvex Polytope, Proc. 5th ACM Symposium on Computational Geometry. 393–399.

L. P. Chew [1989a], Constrained Delaunay Triangulation, *Algorithmica* 4:97–108.

L. P. Chew [1989b], Guaranteed-Quality Triangular Meshes, C. S. Cornell, TR 89–983.

H. Edelsbrunner [1987], *Algorithms in combinatorial geometry,* Springer Verlag, Berlin.

J. Hauser and C. Taylor [1986], *Numerical Grid Generation in Computational Fluid Dynamics (Proceedings)*, Pineridge Press, Swansea, U.K.

G. L. Miller, S.-H. Teng, and S. A. Vavasis [1991], A Unified Geometric Approach to Graph Separators, *Proc. 32nd Symp. Foundations of Comp. Sci.*

G. L. Miller and W. Thurston [1990], Separators in Two and Three Dimensions, *Proc. 22nd Symp. on Theory of Computing*, 300–309.

G. L. Miller and S. A. Vavasis [1990], Density Graphs and Separators, Department of Computer Science, Cornell, Tech Report 90-1169, and to appear, Symposium on Discrete Algorithms.

W. F. Mitchell [1987], A Comparison of Adaptive Refinement Techniques for Elliptic Problems, Department of Computer Science, University of Illinois at Urbana-Champaign, Report No. UIUCDCS-R-87-1375.

W. F. Mitchell [1988], Unified Multilevel Adaptive Finite Element Methods for Elliptic Problems, Department of Computer Science, University of Illinois at Urbana-Champaign, Report No. UIUCDCS-R-88-1446.

D. Moore, J. Warren [1990], Multidimensional Adaptive Mesh Generation, Department of Computer Science, Rice University, Rice COMP TR 90-106.

F. Preparata and M. Shamos [1985], *Computational Geometry: An Introduction*, Springer Verlag, New York.