MESH GENERATION AND OPTIMAL TRIANGULATION

MARSHALL BERN
Xerox Palo Alto Research Center, Palo Alto, California 94304, U.S.A.

and

DAVID EPPSTEIN
Dept. of Information and Computer Science, University of California
Irvine, California 92717-3425, U.S.A.

ABSTRACT

We survey the computational geometry relevant to finite element mesh generation. We especially focus on optimal triangulations of geometric domains in two- and three-dimensions. An optimal triangulation is a partition of the domain into triangles or tetrahedra, that is best according to some criterion that measures the size, shape, or number of triangles. We discuss algorithms both for the optimization of triangulations on a fixed set of vertices and for the placement of new vertices (Steiner points). We briefly survey the heuristic algorithms used in some practical mesh generators.

1. Introduction

Computational geometry claims the two aims of solving practical problems and producing beautiful mathematics. There is a natural tension between these goals: the most elegant formulation of a problem rarely occurs in practice. But surprisingly often the aims complement each other. This chapter discusses the interplay between an important practical problem—finite element mesh generation, and a flourishing theoretical area—optimal triangulation algorithms.

Finite element methods have proved indispensable for physical simulation. These methods discretize the simulated domain—for example, the air around a wing—by dividing it into many small "elements", typically triangles or quadrilaterals in two dimensions and tetrahedra or hexahedra in three. The complex of elements is the mesh.

A triangulation is a partition of a geometric input, typically the region defined by a point set or a polytope, into simplices that meet only at shared faces. So in two dimensions, a triangulation consists of triangles that intersect only at shared edges and vertices. An optimal triangulation is one that is best according to some criterion that measures the size, shape, or number of simplices.

An example illustrates the complementary relationship of mesh generation and optimal triangulation. Numerical-analysis folklore had long held that finite element meshes should avoid elements with sharp angles. Remarkably, Lawson found that the Delaunay triangulation, a geometric construction with a long history, maximizes the minimum angle in a two-dimensional mesh [128]. At about the same time, work by other numerical analysts implicated the flattest—not the sharpest—angle as the critical factor in convergence [5, 112]. This problem did not have a solution waiting for it, and in developing an algorithm to minimize the maximum angle [78], computational geometers discovered a very interesting algorithmic paradigm (discussed in Section 2.2.2).

1.1. Background

The differential equations arising in physical simulation require numerical solution. Most numerical methods—there are exceptions—assume that the domain of interest is divided into a mesh of small, simple elements. There are two major types of meshes: structured and unstructured. This chapter studies only unstructured meshes, but here we briefly sketch the larger context.

A *structured mesh* in two dimensions is most often simply a square grid deformed by some coordinate transformation. Each vertex of the mesh, except at the boundaries, has an isomorphic local neighborhood. In three dimensions, a structured mesh is usually a deformed cubical grid. An *unstructured mesh* is most often a triangulation with arbitrarily varying local neighborhoods.

Structured meshes offer certain advantages over unstructured. They are simpler, and also more convenient for use in the simpler finite difference methods. They require less computer memory, as their coordinates can be calculated, rather than explicitly stored. Finally, structured meshes offer more direct control over the sizes and shapes of elements.

The big disadvantage of a structured mesh is its lack of flexibility in fitting a domain with a complicated shape. A number of techniques have been developed to find appropriate coordinate transformations: conformal mapping, algebraic methods, and numerical methods that themselves solve differential equations [39, 213, 214]. Even armed with these techniques, it may be impossible to find a transformation that fits a complicated domain acceptably well. Faced with this problem, some practitioners cut out a region of the grid, without any transformation, to give a "stair-case approximation" to the domain. But then the computed solution will be quite inaccurate near the boundary of the domain, an area that is often of special interest. Other practitioners break up the domain into simpler regions, perhaps overlapping, each of which can be more nearly matched by a deformed grid. This method and its associated numerical analysis make up "domain decomposition", a large area of study in its own right.

Because of the need to fit complicated domains, such as aircraft and machine parts, the trend in simulation has been towards unstructured meshes [8, 144, 212], although both types of meshes will continue to be important for some time to come. Indeed, there are methods that combine many small structured meshes into an overall unstructured mesh [221].

As we shall see in this chapter, unstructured meshes can fit arbitrarily complicated domains. Simply fitting the domain, however, is not enough. A finite element mesh must also use elements of appropriate size and shape, and these quantities may vary over the domain. Multiple requirements lead to many interesting and difficult triangulation problems; these computational geometry problems define the subject of this chapter.

1.2. Formulating the problems

In the formulation of triangulation problems, we see the tension between applicability and elegance. This tension pervades the major choices in the formulation: the type of input assumed, the type of triangulation desired, and the optimality criteria.

The most theoretically attractive inputs are polygonal regions in two dimensions and polyhedral regions in three, without any auxiliary information. Curved domains occur in practice, but usually the assumption of a flat-sided domain is not too limiting. The more severe restriction is the assumption of no auxiliary information. In practical mesh generation, some foreknowledge of the solution to the finite element computation usually guides the choice of element size, shape, and orientation.

A major binary choice arises in determining the type of triangulation. The vertices of the triangulation may be exactly the vertices of the input, or extra vertices—called *Steiner points*—may be allowed. In practice, mesh generation invariably allows Steiner points, although the placement of Steiner points is often separate from the subsequent triangulation process. Computational geometers have traditionally disallowed Steiner points, and thus their theorems have relevance only for the second stage of practical mesh generation, but recently a small number of geometers (especially ourselves) have considered problems allowing Steiner points. Steiner points change the character of optimal triangulation problems. Where previously the goal was an exact algorithm, it now must be an approximation algorithm that uses a modest number of Steiner points.

For practitioners, the ultimate optimality criteria are speed and accuracy of the finite element computation. These in turn impose a number of somewhat conflicting criteria on the mesh: element shape (such as bounds on angles), reasonable complexity (not too many elements), and element orientation (such as aligned with fluid flow). Computational geometers usually work on only one or two optimization criteria at a time, although certain triangulation algorithms such as Delaunay triangulation and quadtree triangulation optimize several criteria simultaneously. A typical computational geometry problem disallowing Steiner points is one mentioned above: triangulate a point set while minimizing the maximum angle [78] (Section 2.2.2). A typical problem allowing Steiner points is: triangulate an n-vertex polygonal domain using no angles larger than 90° and only a polynomial (in n) number of Steiner points [24, 29] (Section 2.3.2).

In the optimal triangulation algorithms we present, watch for a recurring theme: local versus global optimization. For many problems that disallow Steiner points, local optimizations can lead to global solutions (Sections 2.2.1 and 2.2.2). For problems that allow Steiner points, a typical approximation algorithm performs some initial global steps, such as defining a grid on the input, followed by some local optimizations (Sections 2.3.1, 2.3.2, and 2.3.4).

1.3. Organization

We organized this chapter at the topmost level into sections on two- and three-dimensional problems, with Section 2.5 serving as a bridge. The major sections are then subdivided by the type of output desired: triangulation without any optimization criteria, optimal triangulation without Steiner points, and finally optimal triangulation with Steiner points. To keep ourselves honest, we include sections surveying the heuristics devised by practitioners.

1. Introduction
   - 1.1. Background
   - 1.2. Formulating the problems
   - 1.3. Organization
2. Two-dimensional Triangulations
   - 2.1. Triangulation without optimization
   - 2.2. Optimal triangulation
     Delaunay triangulation, flip, edge insertion, dynamic programming.
   - 2.3. Steiner triangulation
     No small or large angles, maxmin height, min weight, conforming DT.
   - 2.4. Heuristically generated meshes
     Mesh improvement, quadtrees, polygon decomposition, advancing front.
   - 2.5. Two-and-a-half-dimensional problems
     Interpolation, surfaces for three-dimensional models.
3. Three-dimensional Triangulations
   - 3.1. Tetrahedralization without optimization
   - 3.2. Optimal tetrahedralization

3.3. Steiner tetrahedralization
Reducing Delaunay, nonobtuse dihedrals, bounded aspect ratio.
3.4. Heuristically generated three-dimensional meshes
Mesh improvement, octrees, polyhedron decomposition, advancing front.
4. Conclusions

2. Two-dimensional Triangulations

In this section, we consider triangular meshes in two dimensions. We distinguish four types of input domains, all of which can be viewed as polygonal regions of the plane, because their boundaries consist of straight (perhaps degenerate) line segments. The task is to partition the domain into triangles, that meet edge to edge, and that may be required to satisfy some other optimality properties. For all four types of domains, a single parameter n, the number of vertices, suffices to measure the input complexity.

- **Simple polygon.** The domain is a polygonal region of the plane, and its boundary forms a simple, polygonal, closed curve. The triangulation must use the edges of the boundary as edges in the triangulation. In a *Steiner triangulation* problem, extra vertices may be added to the interior or on the polygon; hence, the edges of the boundary may be subdivided to form several collinear edges in the triangulation.

- **Polygon with holes.** This differs from the previous case in that the boundary may form several disjoint polygonal Jordan curves. These curves surround *holes* within the polygon.

- **Point set.** The input is a set of points in the plane. Without Steiner points, the vertices of the triangulation are exactly the input points, and the boundary of the triangulation is the convex hull. With Steiner points, the vertices of the triangulation are a superset of the input points, and the boundary of the triangulation is a convex region that may be larger than the convex hull.

- **Planar straight line graph (PSLG).** The input is a set of vertices and *non-crossing* (that is, intersecting only at endpoints) line segments in the plane, which must be used as edges (or for Steiner problems, unions of edges) in the triangulation. This most general input occurs in practice for "multiple domains", that is, domains that include boundaries between different materials. One could imagine the segments as holes in a polygon, and for non-Steiner triangulation this is a correct view of the problem. For Steiner triangulations, however, the Steiner points must be identical on both sides of a segment, which would not necessarily happen if we treated each segment as a degenerate hole.

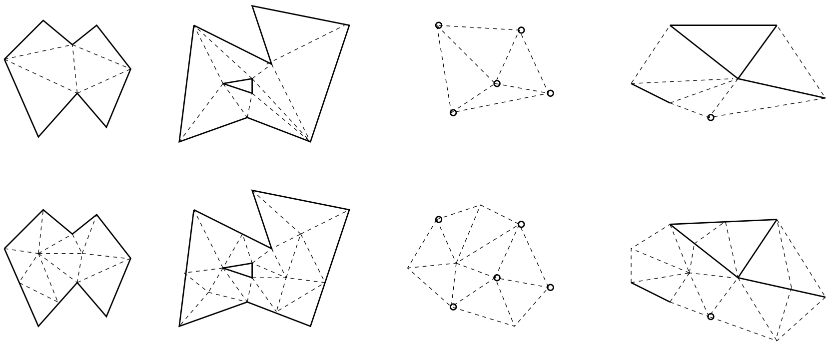


Figure 1. Triangulations with (below) and without Steiner points (above) of a polygon, a polygon with holes, a point set, and a PSLG. Solid lines show input; dotted show triangulation.

2.1. Triangulation without Optimization

In this section we review triangulation without Steiner points and without optimality criteria. The most basic question about polygon triangulation asks whether a triangulation always exists. A *diagonal* of a simple polygon is a line segment between two vertices, that lies inside the polygon and does not intersect the polygon's boundary except at its endpoints.

Lemma 1. *Every polygon with more than three sides has a diagonal.*

Proof: Let b be the vertex with minimum x-coordinate and ab and bc be its two incident edges. If ac is not cut by the polygon, then ac is a diagonal. Otherwise there must be at least one polygon vertex inside triangle abc, as in Figure 2. Let d be the vertex inside abc furthest from the line through a and c. Now segment bd cannot be cut by the polygon, since any edge intersecting bd must have one endpoint further from line ac. ∎

Once we have found a single diagonal, we can split the polygon in two, and recursively triangulate each part. The proof of the lemma implies a linear-time algorithm for finding a diagonal, so a triangulation can be found in time $O(n^2)$.

Now the following question arises: how quickly can we compute a triangulation? This problem attracted more than a decade of intensive research. It is not difficult to improve the running time to $O(n \log n)$ [93, 174]. For many geometric problems there are matching $\Omega(n \log n)$ lower bounds, but none was known in this case. There were also many faster special case algorithms [45, 93, 108]. After Tarjan and Van Wyk [211] broke through to $O(n \log \log n)$, Chazelle [43] ended the quest by announcing a linear-time triangulation algorithm for simple polygons.

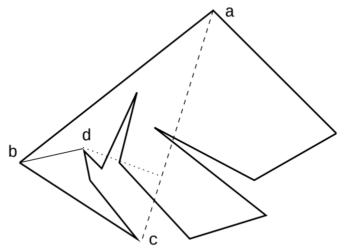


Figure 2. Finding a single diagonal.

We now discuss the other types of input domains. It turns out that any point set can be triangulated in time $O(n \log n)$, and conversely a point set triangulation algorithm can sort real numbers, so there is a matching lower bound in the algebraic decision tree model of computation (see [174]).

Lemma 1 applies to any polygon with holes, and hence to each face in a PSLG. (A *face* is a connected component of the plane minus the PSLG.) Hence every PSLG can be triangulated. It is easy to show by induction that every triangulation of a given PSLG has the same linear number of triangles, $n-2$ in the case of a simple polygon. Triangulating a PSLG can be accomplished in time $O(n \log n)$. Using Chazelle's algorithm, the time for polygons with holes can be improved to $O(n \log h)$, where h is the number of holes.

In contrast to these positive results, determining whether a straight line graph (with crossing edges) contains a triangulation is NP-complete [140]. Also NP-complete is the problem of determining whether a given collection of triangles includes a triangulation of the triangles' vertex set. (These results tend to rule out greedy algorithms for optimal triangulation.)

2.2. Optimal Triangulation

We have seen that quite efficient algorithms exist for constructing a triangular mesh that covers a given domain. There may, however, be exponentially many triangulations with widely varying appearance. We now turn to the harder task of finding a triangulation that optimizes some measure of quality.

Since all non-Steiner triangulations of a two-dimensional input have the same number of triangles, reasonable quality measures depend upon the *shape* of triangles. Typical measures examine the angles, edge lengths, height, and area of a triangle. The measure of a triangulation is then taken to be the sum, maximum, or minimum of the measure over all triangles.

A number of quality measures find motivation in finite element methods. The numerical condition of matrices in a finite element computation is related to the minimum angle in the triangulation. The error of a finite element approximation is also related to the minimum angle [91, 209], and even more closely related to the maximum angle [5, 112]. The minimum height of a triangle relates to the quality of a curved-surface approximation [98], and to a key step in a new three-dimensional mesh generation algorithm [157]. Several other optimization criteria find application in interpolation [69, 178].

For a given element shape, approximation error grows with element size, which can be measured simply by maximum edge length [209] or by more complicated metrics. For some applications—such as modeling diffusion in a nonisotropic medium—size is appropriately measured by the area of the "min-containment circle", after the domain has been transformed by an appropriate affine map [55]. (Sizes of elements are typically used to weight finite element residuals in a-posteriori error estimates, in order to find regions of the mesh in need of refinement [6, 37].)

2.2.1. Delaunay Triangulation and the Flip Algorithm

A well-known construction, called the *Delaunay triangulation*, simultaneously optimizes several of the quality measures mentioned above: maxmin angle, minmax circumcircle, and minmax min-containment circle.

The Delaunay triangulation (DT) of a point set is the planar dual of the famous *Voronoi diagram*. The Voronoi diagram is a partition of the plane into polygonal cells, one for each input point, so that the cell for input point a consists of the region of the plane closer to a than to any other input point. So long as no four points lie on a common circle, then each vertex of the Voronoi diagram has degree three, and the DT, which has a bounded face for each Voronoi vertex and vice versa, will indeed be a triangulation. If four or more points do lie on a common circle, then these points will be the vertices of a larger face, that may then be triangulated to give a triangulation containing the DT.

Voronoi diagrams and Delaunay triangulations have been generalized in numerous directions. One important generalization, known as *power diagrams* and *regular triangulations*, replaces the distance to an input point by the squared distance minus a real-valued weight. For more information on Delaunay triangulations and Voronoi diagrams, see the surveys by Fortune [88] and Aurenhammer [3].

There is a nice relationship between Delaunay triangulation and three-dimensional convex hulls [34, 70]. Lift each point of the input to a *paraboloid* in three-space by mapping the point with coordinates (x, y) to the point $(x, y, x^2 + y^2)$. The convex hull of the lifted points can be divided into *lower* and *upper* parts; a face belongs to the lower convex hull if it is supported by a plane that separates the point set from $(0, 0, -\infty)$. It can be shown that the DT of the input points is the projection of the lower convex hull onto the xy-plane, as depicted in Figure 3.

Finally, a direct characterization: if a and b are input points, the DT contains the edge $\{a, b\}$ if and only if there is a circle through a and b that intersects

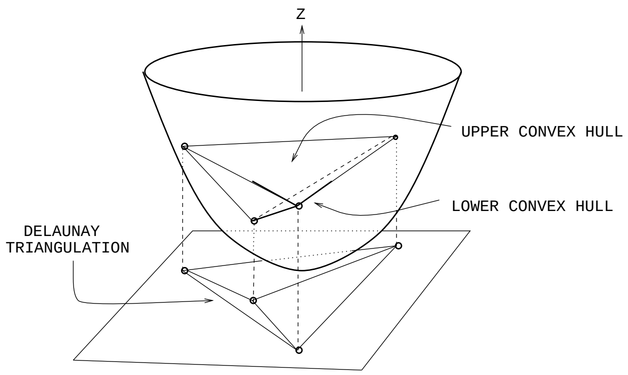


Figure 3. The lifting transformation maps the DT to the lower convex hull.

no other input points and contains no input points in its interior. Moreover, each circumscribing circle (*circumcircle*) of a DT triangle contains no input points in its interior.

Many $O(n \log n)$-time algorithms are known for computing the DT of a point set, the first being Shamos and Hoey's divide-and-conquer algorithm [200]. The relation between Delaunay triangulation and convex hulls allows the use of any $O(n \log n)$-time three-dimensional convex hull algorithm, such as that of Preparata and Hong [173]. Fortune [87, 88] invented an elegant sweepline algorithm. Guibas, Knuth, and Sharir [101] gave a simple, randomized incremental algorithm. (For a simpler analysis of this algorithm see [162, 199].) Maus [149] gave an algorithm for random points with expected linear running time; see also [27, 68, 152] for expected case analyses.

The typical domain for mesh generation, however, is not a point set, but a polygonal region. In some cases the DT of a nonconvex region's vertices contains a triangulation of the region, but in general DT edges cross the region's boundary, creating an invalid mesh. One approach to this problem adds more vertices to the boundary, so that the boundary edges are covered by the DT of the augmented point set. We consider this *conforming Delaunay triangulation* problem in Section 2.3.5. The alternative approach generalizes the definition of the DT in order to force certain edges (the boundary) into the triangulation. Such a triangulation is known as a *constrained Delaunay triangulation* (CDT) [49, 54, 56, 129, 131, 198].

For generality, we allow the domain to be a planar straight line graph. The CDT of a polygonal region may be obtained by treating the polygon as a PSLG, and then removing all CDT edges exterior to the polygon. We first define the notion of *visibility*. Given a PSLG, we say that point a is *visible* to point b if line segment ab does not cross any edge of the graph (ab may intersect an edge without crossing it). Point a is visible to line segment bc if it is visible to some point on bc.

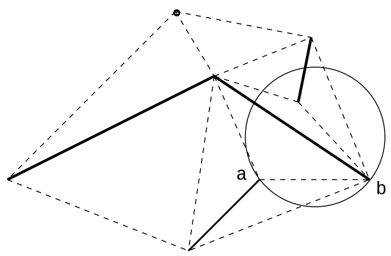


Figure 4. The CDT of a planar straight line graph.

Definition 1. *The* **constrained Delaunay triangulation (CDT)** *contains the edge $\{a, b\}$ between two input vertices, if and only if a is visible to b, and some circle through a and b contains no input point c visible to segment ab.*

For point sets, this definition reduces to the direct characterization of the DT given above. Definition 1 also implies that the circumcircle of a triangle abc in the CDT cannot contain an input point—other than a, b, or c—that is visible from the interior of abc. We now justify Definition 1 by showing that—like the DT—the CDT actually gives a triangulation for inputs in general position. We call the new edges, defined by the empty circle condition above, *Delaunay edges*.

Lemma 2. *If no four input vertices lie on a common circle, the CDT of a PSLG will be a triangulation.*

Proof: By assumption input edges intersect only at endpoints, and since endpoints of a Delaunay edge must be mutually visible, no Delaunay edge can cross an input edge. Suppose two Delaunay edges cross. But then all four endpoints are visible from the crossing point, and it is easy to show that two distinct circles satisfying the definition cannot exist. Therefore the CDT is a PSLG.

We now show that the CDT contains no face with more than three sides. Suppose that ab and ac are edges in the CDT, but that bc is not. Then the circumcircle of triangle abc must contain some other input point visible to bc.

Suppose there is an input point inside triangle abc. Such a point is visible to a unless blocked from view by an edge with an endpoint "closer" to a, meaning closer when projected onto a line perpendicular to bc. Thus the closest input point d to a (again measuring distance from bc) must be visible to a. Pass a circle through a and d, parallel to bc at d. Any input points contained in this circle cannot be visible to segment ad because they are blocked by ab and ac; hence ad is a Delaunay edge.

Suppose instead that there is an input point on the other side of bc from a, visible to bc and lying inside circle abc. Imagine a shrinking circle, that starts as circle abc, and then shrinks in a way that keeps it tangent to circle abc at a.

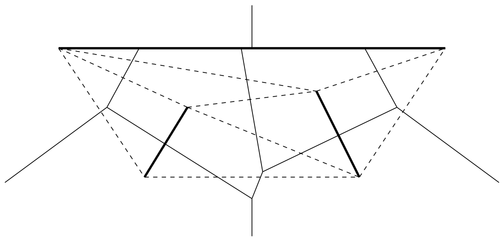


Figure 5. The CDT (dashed) is not the exact dual of the bounded Voronoi diagram.

Ultimately, only a single input point d visible to bc will lie in the shrunken circle, and so d must be visible to a. There can be no other input points in the circle that are visible to ad, so ad is a Delaunay edge.

By these two cases, we have shown that bac cannot be an untriangulated corner. This proves that every connected component is a triangulation. We now show by contradiction that the CDT is connected. Let a and b be nearest-neighbor input points from two different connected components. The diameter circle of ab cannot contain any other input points, so ab must be a Delaunay edge, giving a contradiction. ■

If the input is not in general position, some faces of the CDT may have more than three vertices, all on a common circle. To avoid this situation, the input can be slightly perturbed before computing the triangulation; Edelsbrunner and Mücke propose symbolic perturbations [72]. Alternatively, the triangulation may be completed after the computation. Mount and Saalfeld [159] show how to quickly complete a special-position DT or CDT to maintain the lexicographically maximum angle property. Dillencourt and Smith [63] prove that most completions can be realized as the DT of a small perturbation of the input. Unfortunately it is possible for the angle-maximizing completion to be one of the exceptions.

Just as the DT is dual to the Voronoi diagram, the CDT is related to the *bounded Voronoi diagram* [138, 198, 218], a division of the plane into cells, one for each input vertex, so that the cell for vertex a consists of the region of the plane for which a is the nearest visible input point. Contrary to several claims in the literature, the relation is not exact duality: the CDT may have an edge between a pair of vertices with bounded Voronoi cells that do not meet, but would meet if not cut off by an input edge (Figure 5).

The CDT can be computed directly from Definition 1 by directly testing each candidate edge, but such an algorithm is very inefficient. We now describe the *flip algorithm*, which computes the CDT of a PSLG using a simple local optimization technique; it is derived from a similar algorithm for the DT [128]. The worst-case running time of this algorithm, $O(n^2)$, is not optimal, but its ease of programming makes it quite practical for medium-sized input. Moreover, the algorithm is useful for proving a number of optimality properties.

The flip algorithm starts with any triangulation. For an edge e, not an input edge and not on the convex hull, we denote by Q_e the quadrilateral formed by the two triangles on each side of e. We say that Q_e is *reversed* if e is not in the CDT of the four outside edges of the quadrilateral. Equivalently, Q_e is reversed if the angles at its uncut corners sum to more than $180°$, or if e forms a smaller minimum angle with the outside edges than the other diagonal does. If Q_e is reversed, trivially the triangulation cannot be the CDT, because edge e violates the circle condition. The converse is also true, giving an example of a local optimization leading to global optimality.

Lemma 3. *If no quadrilaterals are reversed, the triangulation is the CDT.*

Proof: The idea behind this proof can be traced back to Delaunay [58]. We show that the circumcircle of each triangle abc contains no vertex d. Therefore each edge is a correct Delaunay edge. Suppose such a point d does exist for some triangle abc. The proof of Lemma 2 shows that we may choose d visible to a.

For a circle C with center at coordinates (x, y) and radius r, define the *power distance* of point $s = (x', y')$ to C, $p_C(s)$, to be $(x' - x)^2 + (y' - y)^2 - r^2$. Then $p_C(s)$ is positive, negative, or zero exactly when s is outside, inside, or on the boundary of C.

Now consider the sequence of triangles $t_1, t_2, \ldots, t_k$ crossing line segment ad, where $t_1 = abc$ and t_k includes d as a vertex. Construct a corresponding sequence of circumcircles $C_1, C_2, \ldots, C_k$. By assumption, each pair $t_i t_{i+1}$ forms a non-reversed quadrilateral. From this, it can be shown that for each i, $p_{C_i}(d) > p_{C_{i+1}}(d)$, and therefore that $p_{C_1}(d) > p_{C_k}(d) = 0$. But this contradicts the assumption that d lies inside circle C_1. ∎

The flip algorithm maintains a queue of edges whose quadrilaterals might be reversed. In the initial triangulation, the quadrilateral of any non-input edge might be reversed, so the queue initially contains all such edges. Then we repeatedly remove the first edge e from the queue. If Q_e is not reversed, we simply continue to the next edge. But if Q_e is reversed, we remove e from the triangulation, replacing it with the other diagonal of Q_e. This *flip* might change the status of some of the four outside edges of the quadrilateral, so we add the changed ones to the queue if not there already. When the queue is empty, we stop.

Lemma 4. *The flip algorithm terminates after $O(n^2)$ flips.*

Proof: We use the lifting relation between DTs and convex hulls. Under the mapping that takes (x, y) to $(x, y, x^2 + y^2)$, the DT of the input vertices lifts to the lower convex hull, and—due to the input edges—the CDT lifts to a surface above the lower convex hull. An arbitrary triangulation including the input edges lifts to a surface above the CDT surface. Each flip of a reversed quadrilateral corresponds to gluing a tetrahedron over a reflex edge in this surface. That tetrahedron is never removed, so the reflex edge never returns. After $O(n^2)$ flips, all non-CDT edges will be eliminated. ∎

Coupled with an $O(n^2)$-time algorithm for constructing an initial triangulation (see Section 2.1), we obtain an $O(n^2)$-time algorithm for constructing the CDT. The flip algorithm also gives a number of useful optimality properties of the CDT. The *min-containment circle* of a triangle t is the smallest circle containing t.

Theorem 1. *Of all triangulations of a PSLG, the CDT (1) minimizes the largest circumcircle; (2) minimizes the largest min-containment circle [55, 176]; and (3) maximizes the minimum angle in the triangulation [128].*

Proof: Each of these properties is improved by flipping a reversed quadrilateral. The optimal triangulation cannot be improved, so it has no reversed quadrilaterals, and hence by Lemma 3 must be the CDT. ∎

Result (3) is sometimes called "equiangularity" and, in fact, a stronger form holds: the CDT lexicographically maximizes the list of angles, from smallest to largest [129, 131]. An optimal interpolation property of the CDT appears in Section 2.5; there the input vertices include elevations. (The DT of a point set has some further properties. The DT contains the minimum spanning tree; the distance between vertices in the DT is $O(1)$ times their distance in the plane [48, 65, 122]; and, from any viewpoint, DT triangles are acyclically ordered by distance [57, 71].)

For very complex domains, an $O(n^2)$-time algorithm for computing the CDT may be too slow. Chew [49] and others [198, 218] have improved this bound to $O(n \log n)$.

Theorem 2 (Chew [49]). *The CDT can be constructed in time $O(n \log n)$.*

Proof Sketch: Chew gives a divide-and-conquer algorithm. Line segment endpoints are sorted by x-coordinates, and split by a vertical line into two subsets, each with at most $\lceil n/2 \rceil$ endpoints. CDTs are computed recursively for each subset and then merged.

In order to achieve the $O(n \log n)$ bound, the work at each level of the computation tree should be only $O(n)$. Hence the work in a single strip, bounded by two vertical lines, must be proportional to the number of endpoints in the strip, rather than to the number of line segments cutting the strip. To achieve this, the algorithm further divides a strip into regions bounded by segments that cut all the way across it. The algorithm computes the CDT only for those regions containing at least one endpoint.

The major remaining difficulty is merging two adjacent strips. This problem can be reduced to merging adjacent regions. As in some DT algorithms [200], the

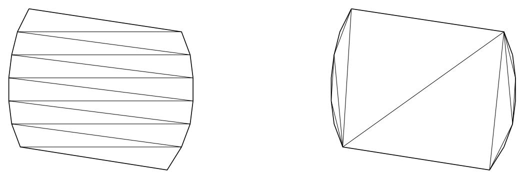


Figure 6. A local optimum for total edge length may be $\Omega(n)$ times the global optimum.

merger is performed by sweeping a circle along the boundary between the two regions. The work is proportional to the number of old edges removed and new ones added; since the sizes of the old and new triangulations are linear, so is the time bound. ∎

The CDT of a simple polygon, though not of a PSLG, can be constructed even more quickly. Aggarwal et al. [1] showed that, if the input is a convex polygon, the DT (which is also the CDT) can be found in linear time. The algorithm is based on the lifting transformation, and it also solves the problem for certain other types of point sets. In particular, it can update the DT after the removal of a vertex or edge. Djidjev and Lingas [64] devised a linear-time algorithm to compute the DT of a more general special case called a "monotone histogram". Finally, Klein and Lingas [126] gave a linear-time randomized algorithm for the CDT of a simple polygon.

2.2.2. Edge Insertion

The flip algorithm described above provides our first example of a local improvement algorithm. As we showed, the flip algorithm in fact produces a global optimum: the resulting triangulation maximizes the minimum angle, and optimizes several other criteria as well. The success of the flip algorithm for Delaunay triangulation has led to the use of edge flipping (with the appropriate definitions of reversed quadrilateral) for finding triangulations that approximately optimize other criteria, such as vertex degree [90], maximum angle [104], total edge length [220], or the ratio of the areas of inscribed circle and triangle [16]. Edge flipping to improve these criteria, however, will not usually compute a global optimum.

The problem is that the algorithm can get stuck in a *local optimum*, in which no flip improves the triangulation. A local optimum can be very far from a global optimum; for example, it may have total edge length $\Omega(n)$ times the true optimal length; see Figure 6.

One way to escape local optima is to allow local moves that do not improve the triangulation, as in simulated annealing [125, 216]. A different approach is to generalize the local improvement procedure. This reduces the number of local optima, as a triangulation without an edge to flip may still admit the generalized move.

This section describes one such generalization, called *edge insertion*, introduced by Edelsbrunner, Tan, and Waupotitsch [78] for the minmax angle problem.

Consider adding a new edge e to some existing triangulation $\mathcal{T}$ of a PSLG. Edge e crosses other edges in $\mathcal{T}$, causing them to be removed. At this point there will be two simple polygons without diagonals, one on each side of e. Optimal triangulation problems on simple polygons tend to be tractable; for now we may assume that these polygons are triangulated optimally using dynamic programming, as in Section 2.2.3. *Edge insertion* is the process of adding a candidate edge, incident to a vertex of a worst triangle (and cutting across that triangle); removing the crossed edges; and retriangulating the remaining regions. The added edge is rejected and the triangulation is returned to its previous state if the triangulation gets worse. Below we explain how to eliminate a possible edge on each insertion, so that the process terminates.

Notice that an edge insertion is more general than an edge flip, as a flip inserts a diagonal of a convex quadrilateral and removes the single edge it crosses. In fact, edge insertion is a significant generalization, as there may be $\Omega(n)$ edges that can be inserted to break a worst triangle, but only one flip. Because of the increased number of possibilities, intuitively edge insertion should reach better local optima than edge flipping, but take longer to do so.

Edelsbrunner et al. [78] showed that edge insertion can in fact compute a global optimum that edge flipping cannot: the triangulation minimizing the maximum angle. The correctness of edge insertion for this problem follows from an abstract property of the maximum angle measure, that also holds for several other natural quality measures, as shown in a subsequent paper by Bern, Edelsbrunner, Eppstein, Mitchell, and Tan [23]. Let f be a function measuring the badness of triangles, and assume that the quality of a triangulation $\mathcal{T}$ is defined to be the maximum (that is, worst) value of f over all triangles in $\mathcal{T}$, denoted $f(\mathcal{T})$.

Definition 2. *Let a, b, and c be vertices in some PSLG G. We say that a is an* **anchor vertex** *of triangle abc, if in any triangulation $\mathcal{T}$ of G, with $f(\mathcal{T}) < f(abc)$, there is an edge ad crossing bc. In other words, one cannot improve a triangulation containing triangle abc without cutting abc by an edge incident to the anchor a.*

For example, let $f(abc)$ be the measure of the largest angle in triangle abc. Then if the largest angle is $\angle bac$, a is an anchor vertex, because in any triangulation of quality better than $f(abc)$, there must be an edge subdividing $\angle bac$ and crossing bc. A triangle may have more than one anchor vertex, and all vertices in an optimal triangulation are anchors.

Definition 3. *Quality measure f has the* **weak anchor property** *if, for each triangulation $\mathcal{T}$, and each triangle abc in $\mathcal{T}$ with $f(abc) = f(\mathcal{T})$, there is an anchor vertex of abc. Similarly, f has the* **strong anchor property** *if all triangles of $\mathcal{T}$ (not just worst triangles) have anchors.*

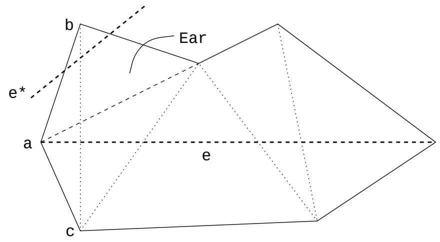


Figure 7. The ear triangle cut by e^* can be no worse than abc.

Lemma 5 (Bern et al. [23]). *Let f be a function with the weak anchor property. Then edge insertion finds a triangulation minimizing f.*

Proof Sketch: Assume $\mathcal{T}$ does not optimize f, abc is a worst triangle in $\mathcal{T}$, and abc has anchor vertex a. Let $\mathcal{T}^*$ denote a triangulation optimizing f; then some edge e of $\mathcal{T}^*$ is incident to a cutting bc. We show that inserting e cannot make $\mathcal{T}$ worse, and hence the edge insertion algorithm cannot get stuck. This follows if the simple polygons on each side of e can be triangulated with triangles of quality no worse than $f(abc)$. See Figure 7.

Call one of the simple polygons P. If P has more than three sides, some edges of $\mathcal{T}^*$ cut across P. It is not hard to see that an edge e^* of $\mathcal{T}^*$ that is "maximally far" from e must cut off a corner of P; that is, the side of e^* away from e supports a triangle in the union of P and $\mathcal{T}^*$. We now reduce P by removing the *ear* at that corner (the triangle defined by the endpoints of the sides of P incident to the corner). If this ear is worst in a new (nonoptimal) triangulation, its anchor cannot be the corner vertex, because there is no edge in triangulation $\mathcal{T}^*$ cutting from this vertex through the opposite edge. So its anchor must be one of the two side vertices. But there is no edge in $\mathcal{T}$ cutting from such an anchor through the opposite edge. Hence the ear triangle is either not a worst triangle or it has quality no worse than $f(\mathcal{T})$. Continuing the process of removing ears triangulates P. ∎

A successful insertion eliminates the crossed edge of the worst triangle in $\mathcal{T}$ (that is, the weak anchor property implies that the crossed edge need not be in an optimal triangulation); an unsuccessful insertion eliminates the edge inserted. Thus there are $O(n^2)$ insertions. Retriangulating by repeatedly removing ears at least as good as the worst triangle, rather than by dynamic programming, gives time $O(n)$ for each insertion. Thus the total time is $O(n^3)$. If the strong anchor property holds and insertions are performed in a certain order, then all crossed edges—not just the one in the worst triangle—can be eliminated in a successful insertion; then the running time can be improved to $O(n^2 \log n)$ [23, 78].

Theorem 3 (Edelsbrunner et al. [78], Bern et al. [23]). *A triangulation minimizing the maximum angle, or maximizing the minimum height, can be computed* *in time $O(n^2 \log n)$. A triangulation minimizing the maximum distance of a triangle from its circumcenter can be found in time $O(n^3)$.* ∎

The minmax angle triangulation has direct relevance to mesh generation due to the error estimate of Babuška and Aziz [5]. The distance from the circumcenter, or "eccentricity", is a measure of obtuseness, with larger triangles weighted more heavily. In Section 2.5, we show that edge insertion also finds an interpolating surface with minimum slope. All of these criteria are mentioned in [220].

2.2.3. Dynamic Programming

We have seen that a number of optimal triangulation problems admit efficient solutions for PSLGs (including polygons as a special case). We next show that many more such problems can be solved in polynomial time, for simple polygons only, using a dynamic programming approach usually attributed to Klincsek [127].

Let f be a quality measure of triangulations of simple polygons, that is a mapping from triangulations to the real numbers. Let P be a simple polygon. An arbitrary diagonal $\{a, b\}$ splits P into two simple polygons, P_1 and P_2. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be triangulations of P_1 and P_2, and let $\mathcal{T}$ be the triangulation of P that is the union of $\mathcal{T}_1$, $\mathcal{T}_2$, and edge $\{a, b\}$.

Definition 4. *We say f is* **decomposable** *if it meets the following conditions: (1) there is a "combining" function g such that $f(\mathcal{T}) = g(f(\mathcal{T}_1), f(\mathcal{T}_2), a, b)$, for all choices of polygon P, diagonal $\{a, b\}$, and triangulations $\mathcal{T}_1$ and $\mathcal{T}_2$; (2) g is computable in time $O(1)$ and monotonic in its first two arguments; and (3) if $\mathcal{T}$ is a single triangle, then $f(\mathcal{T})$ is computable in time $O(1)$.*

In other words, the measure of the entire triangulation can be computed quickly from the measures of the two pieces, together with the knowledge of how the pieces are glued together.

Lemma 6. *The following are decomposable measures: the minimum (maximum) angle in the triangulation, the minimum (maximum) circumcircle of a triangle, the minimum (maximum) length of an edge in the triangulation, the minimum (maximum) area of a triangle, and the sum of edge lengths in the triangulation.* ∎

Confirming this lemma is straightforward. For example, if $f(\mathcal{T})$ measures the minimum angle, then $g(f_1, f_2, a, b)$ is simply $\min\{f_1, f_2\}$. For the sum of edge lengths, $g(f_1, f_2, a, b) = f_1 + f_2 - |ab|$. This last criterion is especially important, as minimizing it for point sets (the well-known *minimum weight triangulation problem*) seems to be very difficult.

An example of a nondecomposable measure is the maximum degree of a vertex, as the maximum degree in $\mathcal{T}$ does not depend only on the maximum degrees in $\mathcal{T}_1$ and $\mathcal{T}_2$, but also on the degrees at a and b. A measure that fails the monotonicity requirement is the difference in areas between the largest and smallest triangles. (Incidentally, both of these measures can be optimized by slightly more complicated dynamic programming.)

Theorem 4 (Klincsek [127]). *A triangulation of a simple polygon optimizing any decomposable function can be computed in time $O(n^3)$.*

Proof: Assume we are trying to minimize $f(\mathcal{T})$. Number the vertices of polygon P by v_1, v_2, …, v_n, in order around the perimeter. If $v_i v_j$ is a diagonal of P, we denote by $P(i, j)$ the polygon formed by points v_i, v_{i+1}, …, v_j. Let $F(i, j)$ be the minimum value of f on a triangulation of $P(i, j)$. If $v_i v_j$ is not a diagonal, define $F(i, j) = +\infty$. We would like to compute $F(1, n)$.

Note that in any triangulation of $P(i, j)$, $v_i v_j$ must be a side of a triangle, say $v_i v_j v_k$, with $i < k < j$. Using the assumption that f is decomposable, we can compute the measure of the optimal triangulation of $P(i, j)$ by trying all choices for k.

$$F(i, j) = \min_{i<k<j} g\,(g(f(v_i v_j v_k), F(i, k), v_i, v_k), F(k, j), v_k, v_j).$$

We compute these values in increasing order of i, and for each i in increasing order of j; then each value of F will be computed before it is needed. This computes the measure of the triangulation; to compute the triangulation itself we maintain *back pointers* for each pair (i, j) to the k that supplied the minimum.

Each computation of the recurrence takes constant time per possible value of k, or $O(n)$ total. Testing whether a pair (i, j) forms a diagonal also takes $O(n)$ time. There are $O(n^2)$ such computations, for a time bound of $O(n^3)$. ■

In general this is the best time bound known. But the bulk of the work is done for pairs (i, j) that form a diagonal—the other pairs can be quickly ruled out. A sharper time bound is "input-sensitive", depending only on the number of diagonals in the polygon.

Definition 5. *The* **visibility graph** *of a polygon P has vertex set consisting of the vertices of P, and an edge between vertices a and b if a is visible to b in P.*

Let E be the number of edges in the visibility graph. The visibility graph can be computed in time $O(n^2)$ by traversing the boundary of the polygon once per vertex, performing a simple stack algorithm in each traversal [79, 116, 120, 130]. A more complicated algorithm reduces the time to $O(n \log n + E)$ [106].

We can easily improve the dynamic programming algorithm from $O(n^3)$ to $O(En)$, but we can even do a little better. The time bound depends on the number of triangles in the polygon; below we show that few edges imply few triangles, a fact previously observed by Chiba and Nishizeki [52].

Lemma 7. *A graph with E edges has at most $O(E^{3/2})$ triangles.*

Proof: Divide the vertices into two classes: *heavy* vertices with degree at least $\sqrt{E}$, and *light* vertices with smaller degree. If b is light we enumerate the triangles containing b by examining each pair of edges ab, bc, and testing if ac is also an edge. Each edge belongs to $O(\sqrt{E})$ pairs at its two endpoints, so this produces $O(E^{3/2})$ triangles overall. If b is heavy we enumerate triangles containing b by examining all edges ac, and testing if ab and bc are edges. There are $O(\sqrt{E})$ heavy vertices, so again the total is $O(E^{3/2})$ triangles. ■

Theorem 5. *A triangulation of a simple polygon optimizing any decomposable function can be computed in time $O(n^2 + E^{3/2})$, where E is the number of edges in the visibility graph.*

Proof: Recall that the dynamic programming algorithm tests diagonals (i, j) in order by i, and then for each i in order by j. When we start a new value of i, we enumerate the triangles containing v_i as in Lemma 7, and store for each j the list of k's that form triangles $v_i v_j v_k$. When we compute $F(i, j)$, we minimize only over the k's stored on this list. ■

It is curious that the worst case of this algorithm occurs when the polygon is convex (so that the visibility graph contains all possible edges). Typically, convexity makes optimization problems easier. Examples include geometric matching [147] and greedy triangulation (see below) [134, 135].

In the matching problem, one must connect the vertices in pairs by diagonals, minimizing the total edge length. The resulting graph has no crossings, and in this sense resembles the minimum weight triangulation. Matching can be solved by the same dynamic programming techniques as above; however, for convex polygons this can be improved to $O(n \log n)$ [147]. Another problem, construction of optimal binary trees, is also related to triangulation of convex polygons [205], and again the $O(n^3)$ dynamic program solves this problem. Yao [223] improved this to $O(n^2)$ using the *quadrangle inequality*, a relation that also holds for diagonal lengths in a convex polygon. In some cases this can be further improved to $O(n \log n)$ [110]. These results suggest that, at least for minimum weight triangulation of convex polygons, $O(n^3)$ is too slow.

Open Problem 1. *Can the minimum weight triangulation of a convex polygon be constructed in time $o(n^3)$?*

2.2.4. Other Optimal Triangulations

A few authors have considered problems of optimizing combinatorial properties of triangulations such as their degree [113] or connectivity [61]. Most of the remaining work on optimal triangulation has used edge length as a quality measure. As mentioned above, edge length can be used as a simple measure of triangle size, which in turn affects finite element approximation error.

Edelsbrunner and Tan [75] considered the triangulation minimizing the maximum edge length. They showed that this triangulation (like the DT) contains the edges of the minimum spanning tree. Therefore it can be found in time $O(n^3)$, by first computing the minimum spanning tree and then triangulating each remaining polygon using dynamic programming. They reduced this time to $O(n^2)$.

The *greedy triangulation* [92, 135] can be found by adding edges one at a time, always choosing the shortest edge that is not already crossed. This triangulation lexicographically minimizes the sorted vector of edge lengths. For arbitrary point sets the greedy triangulation can be computed in time $O(n^2)$ by dynamic maintenance of a CDT [135]. For convex polygons [135] or random point sets [62] the time bound can be improved to $O(n)$.

Eppstein [80] obtained another min-min triangulation result: the *farthest-point Delaunay triangulation* of a convex polygon minimizes the minimum angle. This triangulation can again be constructed in time $O(n)$ [1]. The farthest-point DT dualizes the farthest-point Voronoi diagram, a data structure for finding the farthest input point from a query point. The farthest-point DT can be defined for arbitrary point sets, but in general it is not a valid triangulation, as it only has edges connecting vertices on the convex hull.

Perhaps the most longstanding open problem in computational geometry is the complexity of the minimum weight triangulation (MWT) for arbitrary point sets [67]. (Recall that the MWT asks for the minimum total edge length.) Indeed, early authors called this the "optimal triangulation" problem. Garey and Johnson [92] included MWT in their list of famous problems neither known to be NP-complete nor known to be solvable in polynomial time. If the MWT problem is generalized slightly, so that the weight of an edge is an arbitrary function unrelated to its length, minimum weight triangulation becomes NP-complete [140]. Therefore, authors have concentrated on *approximating* the MWT.

Any triangulation achieves total edge length $O(n)$ times the minimum [124]. The DT, once claimed to be the MWT, can be as long as $\Omega(n)$ times the optimum [124, 146]. The same is true of the triangulation computed by edge-flipping for minimum length. It remains open how well edge insertion approximates the MWT, but it does not provide an exact solution. The greedy triangulation can be as bad as $\Omega(\sqrt{n})$ [133, 146]. For convex polygons, however, the greedy triangulation is an $O(1)$ approximation [134, 135]. The simple *ring heuristic*, that repeatedly connects every other vertex, gives a triangulation of length $O(\log n)$ times the boundary length of a convex polygon [170].

Lingas [137] suggested the following approach to MWT: start by adding the edges of the convex hull and the minimum spanning tree, and then compute the optimal triangulation within each polygonal region. Building on this approach, Plaisted and Hong [170, 165] gave what is currently the best MWT approximation. Instead of starting with the minimum spanning tree, they partition the convex hull into convex polygons. Then the optimal triangulation, greedy triangulation, or ring heuristic can be used to triangulate these polygons, achieving an $O(\log n)$ approximation. The Plaisted-Hong algorithm has recently been implemented with a running time of $O(n^2 \log n)$ [207].

Open Problem 2. *Is it possible to find the MWT (or an $O(1)$ approximation to the MWT) of a point set in polynomial time? Or is this problem NP-complete?*

2.3. Steiner Triangulation

In this section we discuss triangulations using Steiner points. As shown in Figure 1, a Steiner triangulation of a point set may add points outside the convex hull of the input. In a Steiner triangulation of a polygonal region or a PSLG, edges may be subdivided, but they must be *covered*, that is, each original edge must be a union of triangulation edges. Input vertices must be covered by triangle vertices; the remaining triangle vertices are the *Steiner points*.

Any two-dimensional input can be triangulated without Steiner points, so Steiner triangulation only makes sense in the context of some optimality criterion. We consider the following criteria: maxmin angle, minmax angle, maxmin height, and minimum total edge length. Finally, we discuss the conforming Delaunay triangulation problem.

We must exercise a little care in formulating Steiner versions of optimal triangulation problems. For both theoretical and practical reasons, we must concern ourselves with the number of Steiner points. Without any bound on the number of Steiner points, there may be no optimal triangulation—for example, the minimum angle in a triangulation of a point set can be brought arbitrarily close to 60°. And in practice, the number of Steiner points in a mesh directly affects the time to solve a finite element computation.

One might specify the desired number of Steiner points, and then find the optimal triangulation with that many points. Alternatively, one might specify the desired quality measure, and then minimize the number of Steiner points. Either of these formulations, however, results in seemingly intractable problems.

For these reasons, we turn to approximation algorithms. The algorithms we describe achieve quality that is within a constant of the best possible, while using a modest number of Steiner points. In some cases, the algorithms actually use a number of Steiner points that is within a constant factor of the minimum needed for the quality achieved. Allowing approximate optimality offers the additional benefit that we can sometimes simultaneously guarantee bounds on several different measures in the same triangulation.

2.3.1. No Small Angles

The first problem we consider is maximizing the minimum angle, solved in the non-Steiner case by Delaunay triangulation. In the Steiner version of this problem, we demand no small angles: every angle must be greater than some fixed bound.

The smallest angle of a triangle is related to two other quality measures: aspect ratio and height. If the smallest angle in a triangle is θ, the aspect ratio is between $1/\sin\theta$ and $2/\sin\theta$.

Definition 6. *The* **height** *of a triangle is the minimum distance from a vertex to a side. The* **aspect ratio** *is the ratio of the length of the longest side to the height.*

If there is any lower bound on the angles, the complexity of a triangulation of a polygonal domain may be nonpolynomial; indeed, inputs of constant complexity may need an unbounded number of triangles. Consider a rectangle with short side length one and long side length A. Then any bounded-aspect-ratio triangle contained in the rectangle has area $O(1)$; hence $\Omega(A)$ triangles are necessary to triangulate the rectangle with no small angles.

This lower bound holds in a different form for point set input. Consider a bounded-aspect-ratio Steiner triangulation of the vertices of the same rectangle. There must be an edge incident to the upper left corner of the rectangle that has length at most one. Now imagine walking from this corner to the most distant corner along edges of the triangulation. Each successive edge can be only a constant multiple longer than the previous edge, and the first edge must have length $O(1)$. Therefore there must be $\Omega(\log A)$ edges in the walk.

We now turn to upper bounds, meaning algorithms. Baker, Grosse, and Rafferty [12] first solved the problem of computing no-small-angle triangulations of polygonal regions and PSLGs. Their algorithm uses only triangles with angles between 13° and 90°, thereby also solving the nonobtuse triangulation problem described in the next section. They place a uniform square mesh over the input, fine enough that input vertices are several squares apart in the mesh. Using a number of special cases, they show how input edges may be incorporated into the mesh, by retriangulating the squares within a small distance of the edges. They also provide a similar set of special cases to handle the triangulation near each vertex.

Bern, Eppstein, and Gilbert [25] combined a similar approach to mesh generation with an innovative optimality analysis. Their algorithms use *quadtrees*, which had been used in heuristics since the early 80's [224] and suggested by Baker et al. [12] as a natural enhancement to their algorithm. Unlike a uniform mesh, a

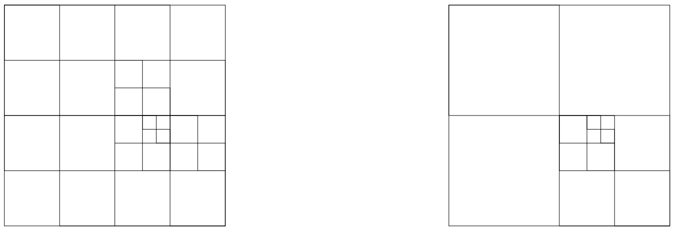


Figure 8. Balanced and unbalanced quadtrees.

quadtree can produce elements of widely varying sizes, resulting in greater accuracy and efficiency. For example, in fluid flow simulations small triangles can be used in turbulent regions for accuracy, and large triangles in smooth regions for efficiency.

Definition 7. *A* **quadtree** *[86, 187, 188] is a recursive partition of a region of the plane into axis-aligned squares. One square, the* **root***, covers the entire region. A square can be divided into four* **child** *squares, by splitting it with horizontal and vertical line segments through its center. The collection of squares then forms a tree, with smaller squares at lower levels of the tree.*

Bern et al. also maintain a *balance condition* in their quadtrees: the squares sharing a portion of a side with a minimal ("leaf") square B must be at most twice the size (side length) of B. Equivalently, each side of a leaf square is subdivided into at most two parts by neighboring squares. See Figure 8.

For simplicity, first assume that the input is just a point set. The triangulation algorithm builds a quadtree covering the set of input points. The quadtree squares are divided into smaller squares, until each input point is *well separated* from the other points: it must be in the center of a five by five grid of squares, all the same size, none containing another input point. Next the algorithm *warps* the quadtree to conform to the input. Each input point chooses its nearest quadtree vertex. No vertex can be chosen twice, because of the separation of input points. A chosen vertex is removed from the quadtree, and its incident edges are reconnected to the input point.

Finally, the quadtree is triangulated. Because of the balance condition, an unwarped (hence square) region in the quadtree has at most one subdivision point per side; it is easy to confirm that such a region can be triangulated into $O(1)$ triangles with no angles smaller than $\arctan(.5) \approx 26.5°$. A warped region is a quadrilateral, with no subdivisions. Three quadrilateral vertices are corners of a square, and the fourth must be fairly near a corner. A case analysis proves that such a quadrilateral can be triangulated with no angles smaller than 20° by adding a single diagonal. See Figure 9.

Along with this angle bound, Bern et al. prove that the number of Steiner points is small, using the following lemma.

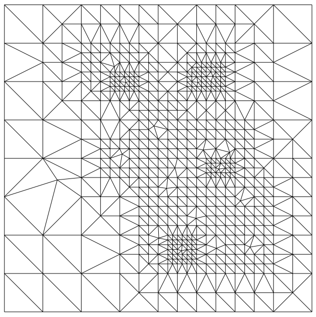


Figure 9. A quadtree triangulation of a point set.

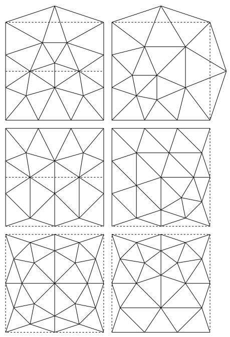
Lemma 8. *The number of squares produced by the algorithm above is $O(n \log A)$, where A is the maximum aspect ratio of a triangle in the DT of the input point set.*

Proof Sketch: The only nonlinear behavior occurs when a quadtree square B is split into children, and no input point becomes well separated by this split. In this case we can find a Delaunay triangle t connecting the input points in B to the rest of the DT, and charge the splits of B and its descendants to t. Since a balanced quadtree successively doubles square sizes, the number of splits charged to a single triangle is proportional to the logarithm of its aspect ratio. ∎

Theorem 6 (Bern et al. [25]). *The quadtree algorithm above uses $O(k)$ triangles, where k is the minimum number of triangles in a triangulation of the input with no angle smaller than $20°$.*

Proof Sketch: Let S be the set of input points along with the Steiner points used in an optimal triangulation with no angle smaller than $20°$. The DT of S has no angle smaller than $20°$, and hence no aspect ratio larger than 6. Then by the lemma above, the quadtree triangulation of S uses $O(k)$ Steiner points. Adding Steiner points to the input set only increases the complexity of the quadtree, so the triangulation of the original point set also has no more than $O(k)$ Steiner points. ∎

This theorem holds no matter what smaller constant replaces $20°$, but the constant hidden in the big-O notation increases as the angle bound decreases. By strengthening the balance condition in the quadtrees, and replacing quadtree squares by tiles with projections and indentations (Figure 10), Bern et al. can guarantee that all angles measure between $36°$ and $80°$. Perhaps bounds of $51°$ and $72°$ are achievable, but any further improvement would force the triangulation to be topologically equivalent to a mesh of equilateral triangles, which seems to require a much larger number of Steiner points.



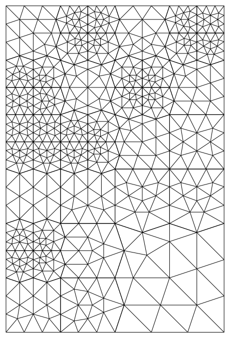


Figure 10. (a) Tiles for all acute triangulation. (b) Example triangulation.

A similar, more complicated, algorithm gives a triangulation of a polygon with holes with no *new* angles smaller than $\arctan(1/3) \approx 18.4°$. (A sharp input angle cannot be erased.) The complications arise primarily in the warping steps. See Figure 24 in Section 2.4 for an example. Theorem 6 extends to polygon input, guaranteeing mesh size within a constant of optimality, only the analog of Lemma 8 necessarily becomes slightly weaker, guaranteeing $O(nA)$ quadtree squares where A is now the maximum aspect ratio in the CDT.

Though not included in the original paper [25], this quadtree-based algorithm can be further extended to PSLGs. For this extension, an idea due to Mitchell and Vavasis [157] proves useful: duplicate each quadtree square that contains more than one face of the domain. Melissaratos and Souvaine [151] combine the algorithms of Baker et al. and Bern et al. to guarantee no obtuse angles as well. Bern, Eppstein, and Teng [26] consider parallel algorithms for quadtree-based meshes.

The algorithms just described are grid-based algorithms. Chew [50, 51] found a quite different, circle-based approach to no-small-angle triangulation of PSLGs. The following definition facilitates description of the algorithm.

Definition 8. *The* **input feature size** *of a point set is the minimum distance between input points. The input feature size of a PSLG is the minimum distance between a vertex and an edge not incident to that vertex. For a polygon or a polygon with holes, we only consider distances interior to the polygon.*

In a version of his algorithm that produces uniform meshes, Chew [50] assumes that all boundary edges have lengths between s and $\sqrt{3}s$, where s is the input feature size. This condition can be enforced by subdividing edges. Chew starts with the constrained Delaunay triangulation of the input, since it maximizes the minimum angle. Then, while there is a triangle with circumcircle of radius greater than s, he adds a Steiner point at the center of the circle, and recomputes the CDT. See Figure 11 for an example.

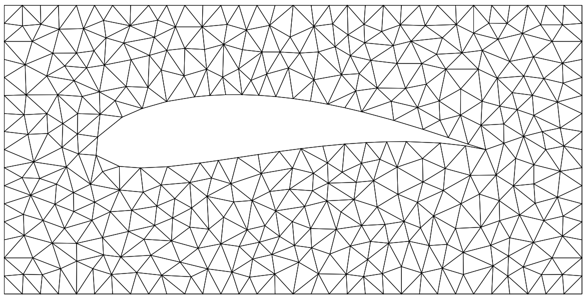


Figure 11. A uniform mesh computed by Chew's algorithm (Chew).

Theorem 7 (Chew [50]). *The circle-based algorithm computes a Steiner triangulation in which all angles are between* $30°$ *and* $120°$.

Proof Sketch: No point is ever added closer than s to another point. Hence the procedure terminates, as only a finite number of points can fit into the polygonal region. At the end of this procedure, all Delaunay circles have radius at most s, and all edge lengths are between s and $2s$. No triangle with edge lengths in this range can be more acute than $30°$ or more obtuse than $120°$, without having a circumcircle radius larger than s. ∎

Recently, Ruppert [183] built upon Chew's approach to give a remarkably simple PSLG triangulation algorithm with the theoretical guarantee of Theorem 6. This algorithm starts—as does the polygon algorithm of Bern et al.—by cutting off acute vertices of the input with isosceles triangles, to be patched in after the rest of the triangulation is complete. Then the domain is triangulated by the following loop: split an arbitrarily chosen skinny triangle (say one with an angle less than $20°$) by adding its circumcenter c, unless c lies within the diameter circle of a boundary edge e, in which case split e instead by adding its midpoint. Halt when all new angles are acceptable.

2.3.2. No Large Angles

Many practitioners have suggested largest angle as an important quality measure for both mesh generation [17, 104] and surface interpolation [98, 100]. Babuška and Aziz [5] (see also [112]) provide theoretical justification by proving convergence of the finite element method as triangle sizes diminish, so long as the maximum angle is bounded away from $180°$; they also give an example in which convergence fails when angles grow arbitrarily flat. An elementary example in which large angles spoil convergence is Schwarz's paradox [175]. A bound on the smallest angle implies a bound on the largest angle, but as we have seen, the elimination of small angles

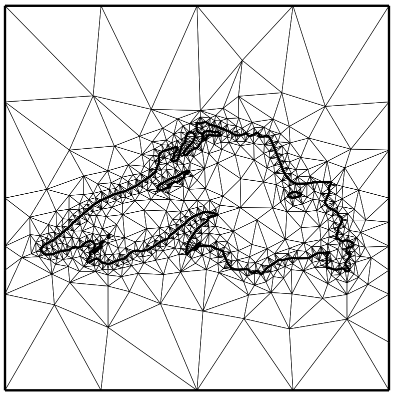


Figure 12. A mesh computed by Ruppert's algorithm (Ruppert).

requires a triangulation with complexity dependent not only on the number of input vertices n, but also on the geometry of the input. In this section, we allow small angles and achieve polynomial bounds on the number of triangles in a no-large-angle triangulation.

If the largest angle in a triangle is $90° - \epsilon$, then the smallest must be at least $\epsilon/2$. Thus the strongest bound on the largest angle, that does not imply a bound on the smallest angle, is $90°$; in other words, the triangulation must be *nonobtuse*. Nonobtuse triangulation claims a number of motivations. A nonobtuse mesh guarantees some desirable numerical properties related to diagonal dominance [12, 217]. Vavasis [217] recently proved that for problems with physical characteristics that vary enormously over the domain, a nonobtuse mesh implies faster convergence of a certain numerical method. A second motivation involves planar duality. Each (closed) triangle in a triangulation contains its circumcenter exactly when all angles measure at most $90°$. The planar dual of such a triangulation can be formed by simply adding perpendicular bisectors of edges (see [28] for more on constructing planar duals). Practitioners use dual meshes in the "finite volume" method in which each mesh vertex has an associated control volume; a perpendicular planar dual defines especially convenient control volumes, simplifying the calculation of flow or forces across element boundaries. Third, nonobtuse triangulation has application to computational learning theory [186]. Fourth, a nonobtuse triangulation simultaneously guarantees some optimality properties, as shown by the following lemma.

Lemma 9. *Any triangulation in which no triangle is obtuse must be a Delaunay triangulation or constrained DT. A nonobtuse triangulation also minimizes the maximum angle, and maximizes the minimum height.*

Proof Sketch: By Lemma 3, if a triangulation is not the CDT then some two

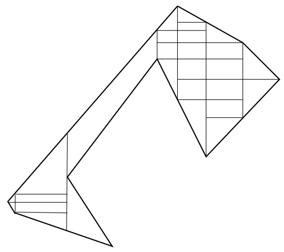


Figure 13. A polygon cut for nonobtuse triangulation.

adjacent triangles form a reversed quadrilateral. But no quadrilateral in which both triangles are nonobtuse can be reversed. Nonobtuse triangulations are unique, up to the choice of diagonal for points forming a rectangle. Any other triangulation would have to have an obtuse angle, and so could not minimize the maximum angle. The maxmin height triangulation is discussed below. An examination of cases shows that, if the triangle with minimum height is nonobtuse, then in any other triangulation there is an equal or smaller height triangle involving one of the same three vertices. ∎

As noted above, Baker et al. [12] give an algorithm for nonobtuse triangulation of polygons. Their algorithm also avoids small angles, and hence must have complexity that depends on geometry, as well as on n. There are also heuristic algorithms of this type [13, 161]. Is it possible to eliminate dependence on geometry?

Bern and Eppstein [24] devised the first polynomially bounded algorithm. The first step dices the polygon into rectangles, with nonrectangular portions left over at the boundaries. Start by passing a vertical line through each input vertex, stopping the line at the polyon boundary. Add a Steiner point at all intersections of input edges and vertical lines. This step divides the polygons into *slabs*—quadrilaterals with two vertical sides, possibly having subdivision points on the vertical sides. Next, draw a horizontal line segment through each vertex (input or Steiner), extending the line segment to the last possible vertical segment. In other words, each endpoint of a horizontal segment should lie either on a vertical segment, or on the vertex inducing the horizontal, and each horizontal segment should be as long as possible with this property. A polygon divided as above by horizontal and vertical lines is shown in Figure 13.

At this point, the polygon is divided into $O(n^2)$ regions of four types: (1) rectangles with unsubdivided sides; (2) right triangles with hypotenuse on the boundary of the polygon and vertical leg possibly subdivided; (3) obtuse triangles with two sides on the boundary of the polygon, and one leg vertical and possibly subdivided; (4) slabs with two sides on the boundary of the polygon, and two possibly-subdivided vertical sides, that cannot be simultaneously crossed by a horizontal line. The slabs can be divided by a diagonal into two obtuse triangles. The

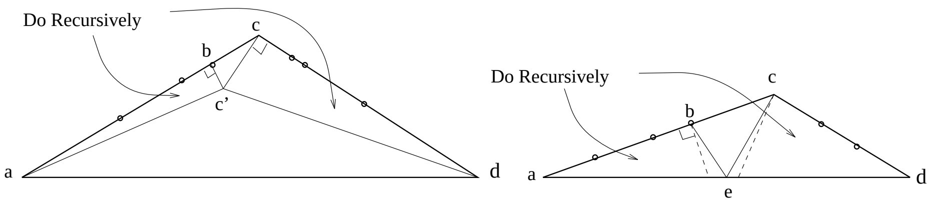


Figure 14. Reduction steps for nonobtuse triangulation of an obtuse triangle.

nonobtuse triangulation problem has now been reduced to triangulating right and obtuse triangles, that have subdivision points on one vertical side, while introducing new subdivisions only on the triangles' longest sides, which lie along the polygon boundary.

Lemma 10. *A right or obtuse triangle with m subdivision points on its two shorter sides can be triangulated into $O(m^2)$ nonobtuse triangles, without adding any new Steiner points to its shorter sides.*

Proof: The basic strategy is to replace the shorter sides by sides that form sharper angles with the long side. The region between the old and new sides is triangulated to reduce the number of subdivision points by one.

We prove the lemma inductively, while also showing that the number of Steiner points added to the base is at most $m + 1$. An obtuse triangle without subdivision points can be triangulated simply by dropping a perpendicular to its hypotenuse. Obtuse triangles with subdivision points are triangulated in two cases as shown in Figure 14. In the first case, the apex c of the triangle (that is, the vertex with the obtuse angle) has been "merged" with the first subdivision point b by drawing perpendiculars to cd and ab. In the second case, if the meeting point c' does not lie inside acd, we split acd by adding a point e to ad, taking care that triangle bce is nonobtuse. These reduction steps are due to Chew (personal communication); the original paper included a couple more cases. ∎

Theorem 8 (Bern and Eppstein [24]). *The grid-based algorithm above triangulates an n-vertex polygon (with holes) with $O(n^2)$ nonobtuse triangles.* ∎

Recently, Bern, Mitchell, and Ruppert [29] improved the bound on nonobtuse triangulation to $O(n)$ by inventing a more intricate, circle-based, algorithm. This algorithm packs the domain with $O(n)$ non-overlapping disks, until each region not covered has at most four sides (line segments or arcs), as shown on the left in Figure 15. The first disks to be added connect holes to the outer boundary, so that each uncovered region is simply connected, and each generalized Voronoi diagram defined by proximity to sides of an uncovered region has the topology of

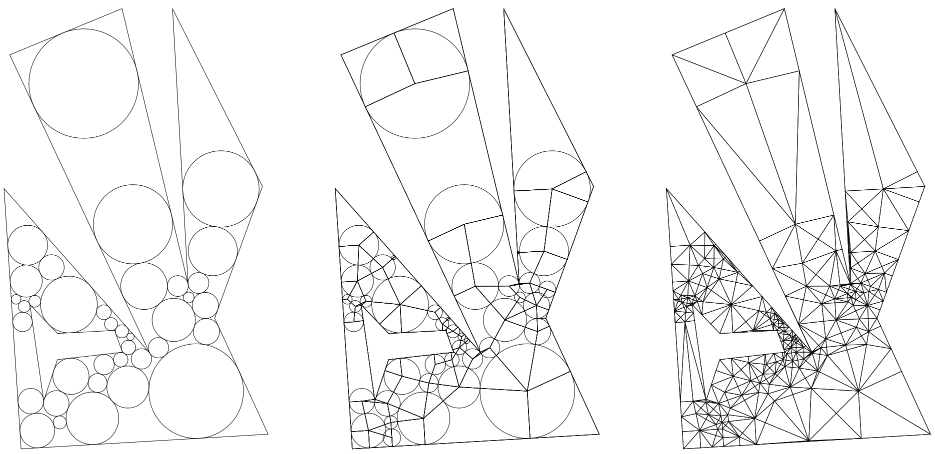


Figure 15. Steps in the circle-based nonobtuse triangulation algorithm.

a tree. Later disks are centered at vertices of these generalized Voronoi diagrams, contacting three sides and breaking uncovered regions into three simpler ones.

The disks themselves do not appear in the triangulation, but rather act as guides for the placement of Steiner points and edges. After the disk-packing stage, the algorithm adds edges (radii) between centers of disks and points of tangency, dividing the domain into small polygons as shown in the middle in Figure 15.

Finally, the algorithm triangulates the small polygons using Steiner points located only interior to the polygons or on the domain boundary, thereby ensuring that triangulated small polygons fit together, as shown in the rightmost figure. The triangulation step depends critically on two properties of tangent disks: (1) a three-sided uncovered region induces a triangular polygon, whose inscribed circle passes through the points of tangency, and (2) a four-sided uncovered region induces a quadrangular polygon that similarly admits a circle passing through the four tangencies. Figure 16 shows two of the patterns used to triangulate small polygons. There are also patterns for small polygons bounded by sides of the domain. Certain misshapen four-sided regions—for example, one in which the "inscribed" circle center falls on the wrong side of a chord—present more difficult cases. Luckily these can be reduced to the illustrated cases by packing in a few more disks, such as the second disk along the bottom in Figure 15 (middle).

Theorem 9 (Bern et al. [29]). *The circle-based algorithm just sketched triangulates an n-vertex polygon (with holes) with $O(n)$ nonobtuse triangles.*

We now turn to nonobtuse triangulation of more complicated domains. Nonobtuse triangulations of PSLGs have application to mesh generation for multiple domains, such as a domain composed of more than one material. Bern and Eppstein

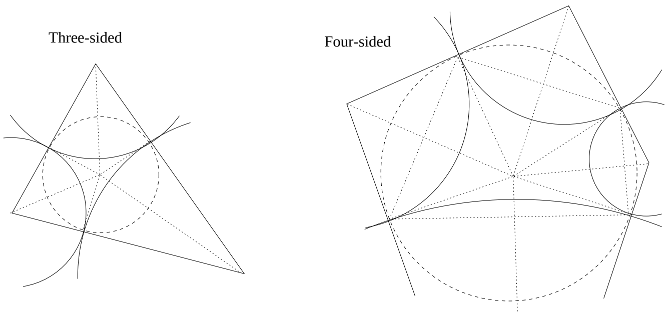


Figure 16. Triangulating the induced small polygons.

extend the grid-based algorithm described above to solve a special case of this problem; they show how to compute an $O(n^4)$-complexity nonobtuse refinement of a given triangulation of a simple polygon. It is currently unknown how to extend the circle-based algorithm.

An example due to M.S. Paterson (personal communication) shows that $\Omega(n^2)$ is a lower bound for this problem. From any vertex in a nonobtuse triangulation, there must be a downwards edge with slope at least 45° from the horizontal (unless the vertex lies on the bottom boundary of the domain). In Figure 17 the vertices on top are sufficiently far apart that their downwards paths remain disjoint through all $\Omega(n)$ skinny triangles. Notice that this example works for any constant maximum angle bound smaller than 180°.

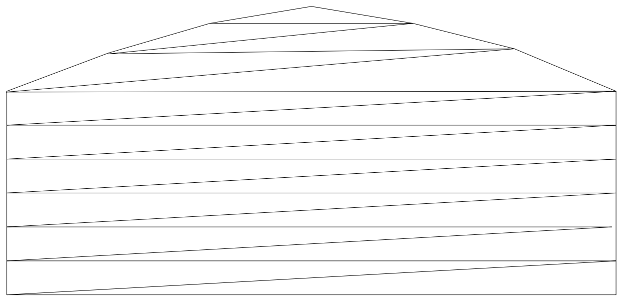


Figure 17. An $\Omega(n^2)$ example for no-large-angle triangulation.

Apparently, neither the grid- nor circle-based algorithms solves a more difficult special case: nonobtuse triangulation of both the interior and exterior (up to the convex hull) of a polygon. This problem arises in computational learning theory [186]. Due to Lemma 9, a solution to this problem would also solve the conforming Delaunay triangulation problem described below. The hard part of the problem is getting "exterior" and "interior" Steiner points to match at the polygon

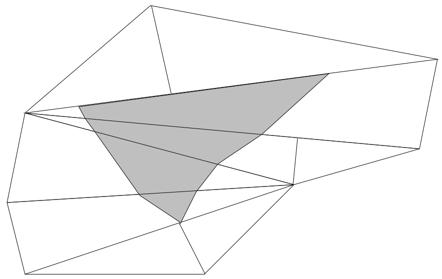


Figure 18. The horn of a subdivision point is the union of all possible "curative" paths.

boundary. There are however nonpolynomial algorithms [12, 151], and an $\Omega(n^2)$ lower bound [186], similar to Paterson's lower bound example.

Open Problem 3. *Is there an algorithm for polynomial-complexity nonobtuse triangulation of both the interior and exterior of a polygon? Of a PSLG?*

It is also natural to ask about no-large-angle requirements less stringent than nonobtuse triangulation. Bern, Dobkin, and Eppstein [22] gave a grid-based algorithm to triangulate simple polygons using $O(n \log n)$ triangles—and polygons with holes using $O(n^{3/2})$ triangles—so that all angles measure at most 150°; both of these results are superceded by the linear nonobtuse triangulation result outlined above. No-large-angle triangulations, however, are known for some types of input for which nonobtuse triangulation remains open.

Recently, Mitchell [155] showed how to triangulate any planar straight-line graph using $O(n^2 \log n)$ triangles with maximum angle 157.5°. Tan [210] improved the maximum angle bound to 132° and the complexity to the optimal $O(n^2)$. These two results use a similar global approach. Starting from any triangulation of the PSLG, Mitchell and Tan split over-large angles by dropping perpendiculars to opposite sides. This produces a partition into well-shaped triangles, but one in which triangles do not meet face to face. To refine the partition into a triangulation, they conceptually extend polygonal paths from each subdivision point to a boundary point or interior vertex. Whenever a path crosses the edge of a triangle, it may continue at any legal angle away from the edge. The union of all (the infinite number of) possible paths thus sweeps out a cone-shaped region of the PSLG, called a *horn*. The horn of a subdivision point grows one triangle at a time, until it first encloses a vertex or boundary point (or, in Mitchell's version, intersects itself or another horn), and a single path can be chosen. See Figure 18.

In Mitchell's version, two intersecting horns produce a new Steiner point, so $O(\log n)$ successive phases of horn-growing are required to cure all Steiner points. In Tan's improvement, horns act independently of each other, after a preprocessing stage in which certain obstacles are set up in the PSLG. The obstacles prevent spiraling, so that each of the $O(n)$ horns propagates through only $O(n)$ triangles. After growing all the horns, Tan erases the path edges—leaving just the Steiner points—and shows that each original triangle can now be retriangulated with no large angles.

2.3.3. Maxmin Height

We now consider maximizing the minimum height of a triangle. This problem arises in Mitchell and Vavasis's three-dimensional mesh generation algorithm [157].

As we saw above, a minimum-height triangulation without Steiner points can be computed using the edge-insertion paradigm [23]. It is not immediately obvious that Steiner points can improve the height. Consider a regular n-gon, however, with sides of unit length. Any triangulation without Steiner points must contain an ear, which has height $\sin(\pi/n)$, but a Steiner point at the center allows all triangles to have nearly unit height. This example is due to Mitchell (personal communication).

For point set input, the linear-complexity nonobtuse triangulation algorithm described above already solves the maxmin height problem. Notice that the minimum distance s between two points, that is, the input feature size defined in Section 2.3.1, gives an upper bound on the minimum height achievable. The nonobtuse triangulation algorithm defines quadtree squares of side length $\Omega(s)$, and each triangle has height proportional to the smallest quadtree square it touches.

For polygons, the input feature size s again gives an upper bound on the minimum height achievable. The no-small-angle quadtree algorithm produces a triangulation with minimum height $\Omega(s)$, but with nonpolynomial complexity. The nonobtuse triangulation algorithms described above do not help, as both the grid- and circle-based algorithms can produce triangles with small height.

Theorem 10 (Bern et al. [22]). *A polygon with holes can be triangulated into $O(n)$ triangles of height $\Omega(s)$.*

Proof Sketch: We give a very rough sketch. The first step starts by cutting off acute corners of the polygon with isosceles triangles; these triangles are triangulated at the very end without adding any new Steiner points. Then the remaining polygon P is cut with vertical and horizontal lines. Imagine the plane divided into an infinite square grid with spacing $s/3$. Now erase all of the grid, except portions of lines bounding squares that either contain—or are adjacent to a square that contains—an input vertex. See Figure 19.

The second step carefully warps some segments of the horizontal and vertical cutting lines to conform to the boundary of P. A case analysis shows that all resulting faces can be triangulated with height $\Omega(s)$. At this point, there are $O(n)$ triangles of height $\Omega(s)$ along the boundary of P, and $O(n^2)$ rectangles of

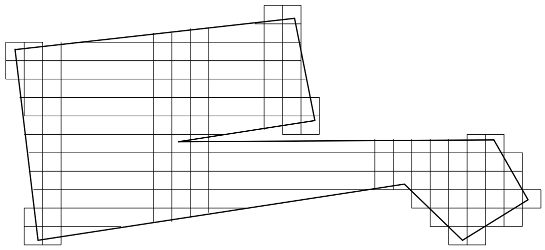


Figure 19. A polygon cut for maxmin-height triangulation.

height $\Omega(s)$ interior to P. The third step merges interior rectangles into rectilinear polygons to reduce the complexity. Each rectilinear polygon Q will have input feature size $\Omega(s)$ and satisfy a certain matching condition: a horizontal or vertical line extended from a vertex of the polygon across the interior first intersects Q at another vertex.

A sweep algorithm now triangulates such a rectilinear polygon with linear complexity. First draw in all interior vertical line segments between vertices of Q. Then sweep from left to right, adding Steiner points to vertical segments. The key idea is to project only every other vertex from the current vertical line onto the next vertical line. Horizontal edges are added between corresponding points. When the left-to-right sweep has ended, perform a similar sweep from right to left. After both sweeps, each newly-created face interior to Q has at most one subdivision point on each vertical side; such faces can be triangulated into at most four triangles of height $\Omega(s)$. The total number of Steiner points added in the sweeps is $O(n)$, as it is the sum of a geometric series. ∎

This result can be combined with a grid-based no-large-angle triangulation algorithm [22] to triangulate simple polygons with $O(n \log n)$ Steiner points, with maximum angle at most $150°$ and minimum height $\Omega(s)$.

2.3.4. Minimum Weight

As in maxmin height triangulation, it is not obvious that adding Steiner points can reduce the total edge length of a triangulation of a point set. But it is not hard to make examples (say, $n-1$ points on an arc that bows towards one distant point) to show that the minimum weight Steiner triangulation (MWST) can have total edge length $\Omega(n)$ times smaller than the minimum weight triangulation (MWT).

Lingas [137] suggested approximating the MWT of a point set by first choosing all edges from a minimum spanning tree. This partitions the plane into polygons, that can be triangulated optimally using dynamic programming. Lingas did not consider using Steiner points, and his algorithm does not yield a good approximation to the MWST. Clarkson [53] extended Lingas's approach, and gave the first nontrivial MWST approximation. (It can be shown that any triangulation approximates the MWST within a factor of $O(n)$.) Clarkson showed that any polygon has a Steiner triangulation with total edge length $O(\log n)$ times the polygon's perimeter. (The same result can be seen from the quadtree triangulation described below.) Combining this result with Lingas's use of the minimum spanning tree produces a Steiner triangulation of a point set with length $O(\log n)$ times that of the MWST.

Clarkson's result was improved by Eppstein [81], who gave a constant-factor approximation to the MWST. Eppstein showed that the quadtree triangulation algorithm of Bern et al. (Section 2.3.1), which was designed for angle bounds, actually gives such an approximation.

Eppstein first proves that, if $\mathcal{T}$ is any triangulation of the input point set, and B is a quadtree-algorithm square with side length ℓ, then there is an edge in $\mathcal{T}$ with length $\Omega(\ell)$, that has an endpoint within distance $O(\ell)$ of B. Therefore we can charge the length of the triangles associated with B to an edge in $\mathcal{T}$. Each edge is charged at most proportionally to its own length, and hence the quadtree triangulation approximates the MWT. Then, as in the optimality proof of the no-small-angle quadtree triangulation, Eppstein notes that adding Steiner points only increases the total length of the quadtree triangulation. So the quadtree triangulation of the input points has length less than the quadtree triangulation of the input along with an optimal set of Steiner points, which in turn approximates the MWST. This argument works both for the no-small-angle quadtree triangulation (Section 2.3.1), and for the linear-size nonobtuse quadtree triangulation (Section 2.3.2).

The same techniques can be used to construct an approximate MWST of convex polygons. One simply triangulates the vertices, cuts off the region of the triangulation outside the polygon, and adds diagonals to retriangulate cut polygons. This reduction requires convexity; otherwise the length of the point set triangulation may not approximate the MWST.

Open Problem 4. *Is there an efficient algorithm for approximating the MWST of an arbitrary nonconvex polygon?*

Using his quadtree characterization of the MWST length, Eppstein also proves some further properties of the MWST of a point set.

- The MWST has total edge length $O(\log n)$ times that of the minimum spanning tree [53]. For any n, there exist point sets for which the MWST has edge length $\Omega(\log n)$ times the minimum spanning tree length.
- For any n, there exist point sets for which the MWT has total length $\Omega(n)$ times the MWST length. As noted above, this is tight.

- For any n, there exist convex polygons for which the MWT has total length $\Omega(\log n)$ times that of the MWST. Again, this is tight, as both lengths are within $O(\log n)$ of the polygon's perimeter.
- Any convex polygon has a Steiner triangulation in which all Steiner points lie on the polygon boundary, with total length $O(1)$ times that of the MWST.

The last item suggests the question of whether Steiner points in the interior of a polygon really help. For nonconvex polygons, they do; otherwise the weight may be $\Omega(n/\log n)$ times the MWST length. But for convex polygons this question remains open. If we can assume that all Steiner points lie on the boundary, then it seems likely that a dynamic programming algorithm can compute an optimal MWST.

2.3.5. Conforming Delaunay Triangulations

In this section, the input is a PSLG. We consider the problem of finding a set of Steiner points inducing a Delaunay triangulation that is a Steiner triangulation of the input. In other words, each input edge must be a union of edges in the DT of the input vertices and the Steiner points. We call this the *conforming Delaunay triangulation* problem. As noted above, this problem is related to nonobtuse triangulation of the interior and exterior of a polygon. The $\Omega(n^2)$ lower bound for interior-exterior nonobtuse triangulation can be adapted to the conforming DT problem.

We first consider an approach to the conforming DT problem that uses a number of Steiner points dependent upon the geometry of the input as well as on n. The algorithm is a modification of one due to Saalfeld [185], also related to two previous algorithms [32, 190]. Start with a triangulation of the input, and let ϵ be the minimum height of any triangle. On each edge, add two Steiner points at distances $\epsilon/3$ from the endpoints. The diameter circles for the two segments of length $\epsilon/3$ on each edge will be empty, and hence these outer segments are now edges in the DT. Let δ be the minimum distance between any pair of points in the now augmented PSLG. No interior segment of any edge is within distance δ of any other edge. Hence if we partition these segments into subsegments of length less than δ, each subsegment will have an empty diameter circle, and the result will be a Delaunay triangulation that covers the input (Figure 20).

Nackman and Srinivasan [164] describe an alternative for polygons with holes. It is not necessary to subdivide input edges so finely that each segment has an empty diameter circle, as in the previous method; in fact any empty circle through the endpoints of a segment suffices. Thus Nackman and Srinivasan cover edges with a set of disks that do not overlap other edges, and place Steiner points at the intersections of circles and edges, as in Figure 21. The assumption that the

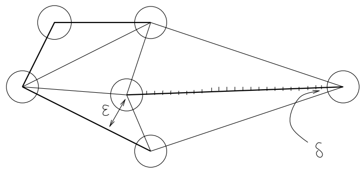


Figure 20. Example of Saalfeld's conforming DT algorithm.

input is a polygon implies the existence of a finite set of suitable disks, and it is even possible to minimize the number of disks [164]. But this is no guarantee that the algorithm uses a minimum number of Steiner points—indeed, Nackman and Srinivasan give an example of constant complexity for which their method requires a nonconstant number of Steiner points.

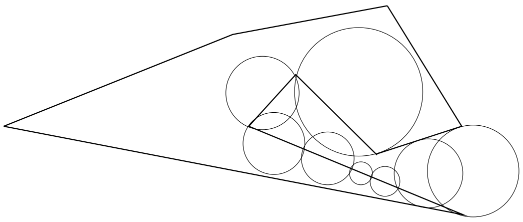


Figure 21. Covering each edge by circles not touching any other edges.

In a theoretical breakthrough, Edelsbrunner and Tan [76] gave an algorithm that uses only a polynomial number of Steiner points, regardless of the input geometry. Their algorithm places Steiner points in two stages, called the "blocking" and "propagation" phases. The blocking phase computes a set of $O(n)$ non-overlapping disks whose union is connected and spans the vertices of the input PSLG. A Steiner point is added wherever a disk crosses an edge or meets another disk, $O(n^2)$ Steiner points total. The propagation phase treats the segments of input edges that are not yet covered by empty circles. This phase takes any not-yet-covered segment ab and attempts to add its diameter circle. If the diameter circle encloses another vertex c visible to ab, the orthogonal projection of c is added to ab, forming two shorter segments. See Figure 22.

Theorem 11 (Edelsbrunner and Tan [76]). *The algorithm just sketched gives a set of $O(n^3)$ points for which (a completion of) the Delaunay triangulation conforms to all input edges.*

Proof Sketch: Correctness follows from the fact that each segment of an input edge has a circle with empty interior passing through its endpoints. For the complexity analysis, notice that the propagation phase does not add Steiner points

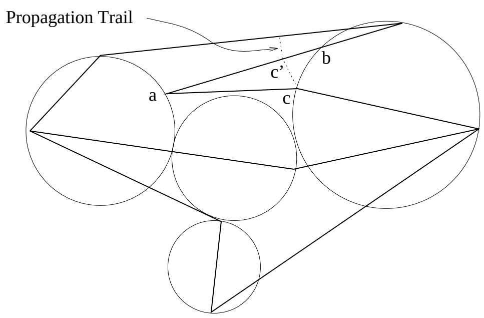


Figure 22. Propagation trails cannot spiral to hit the same edge twice.

inside disks, so a line segment connecting two consecutive vertices around the boundary of a disk will appear in the DT. These segments divide the plane into simple polygons—including the outer face—that propagation "trails" cannot cross. There are $O(n^2)$ trails each of length $O(n)$. ■

2.4. Heuristically Generated Meshes

In this section we describe two-dimensional mesh generation in practice. We do not attempt a thorough literature survey, rather we give informal descriptions of a few mesh generation approaches, chosen to illustrate some of the main ideas in the field. There are already a number of articles that survey and classify the literature on automatic mesh generation [109, 189, 202, 208, 212, 221], although we are not aware of a recent survey of two-dimensional mesh generation with an extensive bibliography. Overall, two-dimensional mesh generation seems fairly mature, and a number of different approaches give good results. As we mentioned in the introduction, we restrict our discussion to generation of unstructured triangular meshes; for structured meshes, such as quadrilateral meshes given by conformal mapping techniques, see [39, 213, 214].

2.4.1. Mesh Improvement Techniques

Before describing heuristic mesh generation methods, we mention some mesh *improvement* techniques, which can be used as post-processing steps after any of the heuristics.

One very useful technique, dating to the 1960s [222], is called *Laplacian smoothing* because its repositioning formula can be derived from a finite difference approximation of Laplace's equation [105]. In Laplacian smoothing, a vertex v in the interior of the mesh is moved to the centroid (center of mass) of its neighbors.

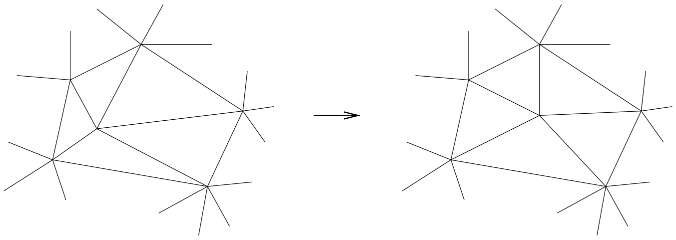


Figure 23. Moving a vertex to its neighbors' center of mass.

The vertex should not be moved if the centroid lies outside the polygon formed by its neighbors. This repositioning usually improves the size and shape of the triangles around v, but it is not guaranteed to do so. A variation weights neighbor vertices by the areas of their surrounding elements. Laplacian smoothing is applied successively to each interior node of the mesh, for several (four or five) rounds. See Figure 23.

A second technique, called *mesh relaxation* by Frey and Field [90], flips edges to regularize degrees. Removing high- and low-degree vertices makes subsequent Laplacian smoothing more effective. A quadrilateral diagonal is flipped if the sum of its endpoints' degrees exceeds the opposing diagonal's sum by more than two.

We may also view the flip algorithm for producing a constrained Delaunay triangulation (Section 2.2.1) as a mesh improvement technique. Notice that Delaunay flipping and mesh relaxation may disagree.

Finally, the problem of refining a given mesh occurs quite often in practice, for example when an initial finite element computation reveals a region that requires greater resolution. Refinement and its reverse—"derefinement" or coarsening—assume even greater importance when the solution varies with time, and small features move across the domain [121]. Bank [13, 14] refines by splitting selected triangles into four similar copies of themselves, and then splitting neighboring triangles into two triangles to correct subdivision points. This algorithm works well enough in practice–and is especially easy to derefine—but obliviously splitting a triangle and its descendants a number of times may make a neighboring triangle unacceptably sharp. Smoothing can fix sharp angles, but the resulting mesh will not then be a strict refinement of the original mesh.

Addressing this flaw, Rivara [179, 180] proposed the following recursive algorithm: bisect a triangle in need of refinement by adding a diagonal from the opposite vertex to the midpoint of the longest edge, then refine its neighbor the same way. Bisections may propagate for some way across the triangulation, but this algorithm—now called *Rivara refinement*—always terminates, since each bisection splits a longer edge. Based on a theorem of Rosenberg and Stenger [182],

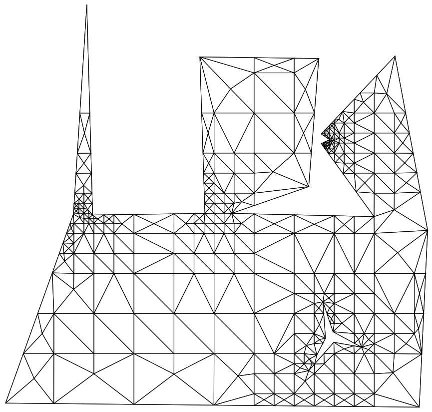


Figure 24. A mesh derived from a quadtree (S. Mitchell).

Rivara showed that arbitrary repetition of this algorithm never produces an angle smaller than half the minimum original angle. In practice Rivara refinement typically improves angles.

2.4.2. Quadtrees

Long before the theoretically good mesh generation methods described in the last section [25], quadtrees had been used in heuristic methods. Yerry and Shephard [224] construct a quadtree representation of a polygonal or curved domain by recursively splitting squares intersected by the boundary of the domain, until squares reach some minimum size bound. Splits may also result from an upper bound on size (that may vary over the domain) or from a balance condition: no quadtree square is adjacent to one more than twice its size. After the quadtree is constructed, each square containing a portion of the boundary is replaced with a shape chosen from a fixed set of patterns. Triangulating each face then yields a mesh that approximates the domain.

A more advanced version of the algorithm uses warping and trimming methods to produce a mesh that does not change the shape of the input [7]. The theoretical paper by Bern et al. [25], contributed warping rules guaranteed not to produce small angles, as well as an analysis of the number of triangles in the mesh. Figure 24 shows an example mesh computed by Mitchell [157], using a modification of the algorithm in [25]. This example demonstrates careful selection of quadtree square sizes around holes and “almost holes”.

Quadtrees are the most convenient way to produce *graded* meshes, that is, meshes with small elements near complicated parts of the boundary that grade up to larger elements elsewhere. Another advantage is that the quadtree itself

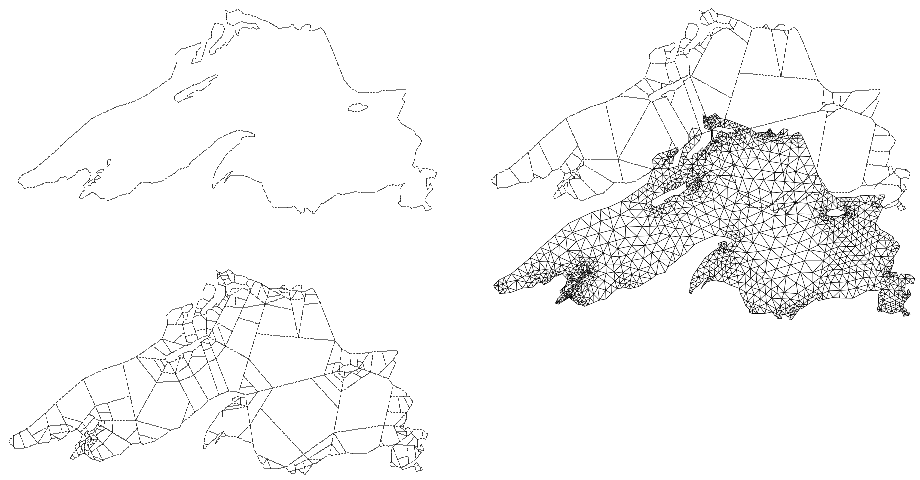


Figure 25. Lake Superior (Joe) (a) input, (b) decomposition, (c) refinement, (d) mesh.

may be computed entirely in integer arithemetic, so that floating-point operations are carried out only within small squares containing simple parts of the domain boundary.

Quadtree methods, however, have been criticized for occasionally producing poorly-shaped boundary elements, and for introducing artificial preferred directions (namely parallel to the x- or y-axes) [208]. The problem of poorly-shaped elements can be solved by the warping methods of Bern et al. [25]. The second problem may be more inherent, although non-square quadtree tiles, as used in [25, 158], or some sort of randomization procedure, coupled with Laplacian smoothing, may sufficiently break up the directionality.

2.4.3. Polygon Decomposition

The polygon decomposition approach also initially divides the domain (most generally a PSLG) into simple regions. This approach, however, attempts to find intrinsic dividing lines, rather than dividing lines from a rectilinear grid.

Joe and Simpson [119, 114] first divide the domain into convex polygons by cutting along lines extending from reflex vertices (that is, vertices at which the interior angle measures more than 180°). The resulting convex polygons are further subdivided into convex polygons with boundary edge lengths that do not vary too much. Cutting lines are chosen heuristically, attempting to avoid small angles. Finally each convex polygon is triangulated using triangles of approximately equal size, taking care to match Steiner points at the cutting lines. See Figure 25.

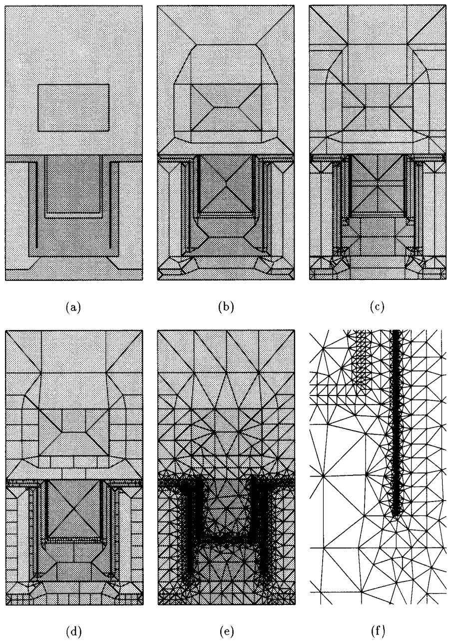


Figure 26. The steps in meshing a multiple domain (Srinivasan et al.).

In Joe and Simpson's method, two input parameters control the mesh: a target number of triangles (typically exceeded by a small amount), and a "smoothing parameter" that controls the allowable variation in size between two neighboring polygons. These parameters are combined with something dependent on *local feature size* to yield a "mesh distribution function", that gives a target triangle size at each point within the domain.

The concept of local feature size recurs in most mesh generation approaches. We may define the local feature size at point a to be the size of the quadtree box that contains a, as produced by the polygon version of the algorithm of Bern et al. [25], described in Section 2.3.2. This sets the local feature size at a vertex v of the polygonal boundary P to be proportional to the minimum distance (within the domain) to an edge of P not incident to v. Local feature size then varies fairly smoothly between vertices. Most of the mesh generators described in this section define their own versions of local feature size, but the definition just given is sufficient for understanding. Many of the mesh generators also allow the user to control the local feature size in some way, perhaps through input parameters. This extra control is important in applications in which the solution to the finite element computation is expected to show features smaller than the features of the domain.

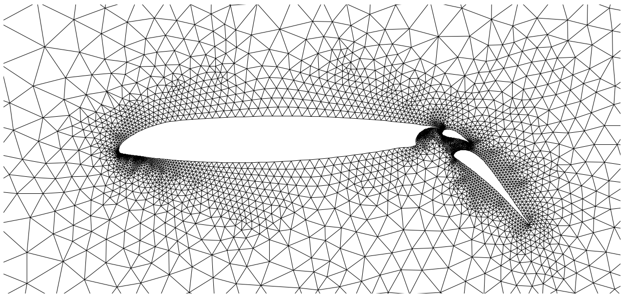


Figure 27. A Delaunay triangulation of points placed in layers (Barth and Jespersen).

Srinivasan et al. [208] recently developed an interesting polygon decomposition mesh generator, using the *symmetric axis transform*. The symmetric axis is the set of all centers of disks contained in P that contact P at two or more points; it consists of straight lines and parabolic arcs.

Figure 26 shows the operation of this mesh generator on a multiple domain. The input is shown in (a), with different materials shown by different shades. The first step computes the Voronoi diagram of the edges and vertices of the domain; this contains each face's symmetric axis. Parabolic arcs in each symmetric axis are then replaced by one or two straight edges (chords of the arc), as shown in (b). Each vertex on a symmetric axis is then joined to two or more points on the domain boundary by the touching radii of the disk centered at the vertex, resulting in a PSLG in which each face is a triangle or trapezoid, as shown in (c). A "sliver processing" step then removes or breaks up faces with bad aspect ratio (see (d)). Boundary edges and touching radii are then used to extract local feature size values. These values induce a node spacing at each vertex in the PSLG, and interpolation then gives a node spacing function over the entire domain. The node spacing function guides an iterative process that adds and deletes more Steiner points. Final triangulation is accomplished by the constrained Delaunay triangulation (shown in (e) and in a zoom in (f)).

2.4.4. Advancing Front

The *advancing front* approach to mesh generation [136, 141] is especially well-suited to fluid dynamics problems. In this approach, the domain's boundary P is first subdivided appropriately, and then Steiner points are placed in successive layers around each connected component of P. This yields triangles oriented with the flow field. Figure 27 shows a mesh computed by Barth and Jespersen [19]; this mesh is the Delaunay triangulation of vertices placed by the advancing front method.

Lo [143] has developed a mesh generator that places Steiner points along

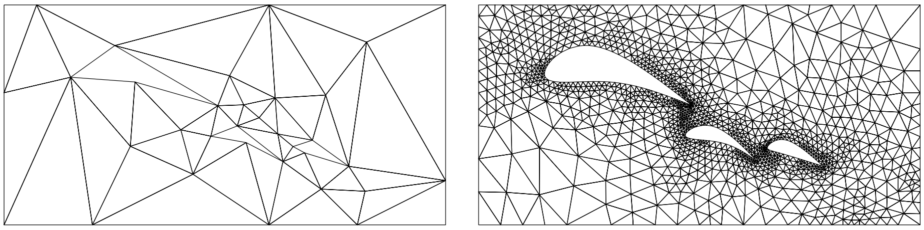


Figure 28. Input and output of a refinement-based generator (Shaw).

predefined contour lines, that need not follow the domain boundary. This generalization allows the output of an initial finite element computation to control the generation of the next mesh.

Mavriplis [150] has computed meshes for high Reynolds number flows using an idea related to contour lines. His method identifies "stretching" lines and places the first Steiner points along these curves. Local, structured meshes generate interior Steiner points. The method then computes Delaunay triangulations in locally transformed regions in order to generate long, thin—but not overly obtuse—triangles oriented with the flow. Such a triangulation is an especially efficient mesh for laminar flows. (See [55, 171] for other stretched triangulations.)

2.4.5. Mesh Refinement

A number of researchers have taken mesh refinement as the central step in mesh generation itself [14, 89, 179]. For example, the algorithms of Chew [49] and Ruppert [183], discussed in the last section, refine the constrained Delaunay triangulation by adding centers of circumcircles.

An earlier, somewhat related, heuristic method is due to Frey [89]. In Frey's method, Steiner points are initially added to the boundary according to a "spacing function" that approximates local feature size. After this step, for most practical inputs, a Delaunay triangulation of the input vertices and the Steiner points includes the domain boundary. (Using the recent results of [164, 185], this step can be made exact; see Section 2.3.5 above.) Next interior Steiner points are added. Satisfactory results were obtained by the following method: (1) find a triangle t containing its circumcenter; (2) generate a prospective Steiner point a partway between the incenter and circumcenter of t; (3) if a is not too close to the vertices of t (where close is defined by the local feature size), then add a and rebuild the Delaunay triangulation.

Shaw [201] has recently developed a simple, refinement-based mesh generator. The user inputs a very rough, triangular, initial mesh, with the correct number of boundary components in roughly correct locations. The user assigns a local feature size to each node of the initial mesh.

Triangles larger than their vertices' smallest feature size are then split as in [13, 14] into four similar copies of themselves by adding Steiner points at the midpoints of sides. A midpoint is assigned the average feature size of the endpoints. Neighboring faces are split in two to correct subdivision points. Splitting is followed by a few cycles of Delaunay flipping and weighted Laplacian smoothing, and the refinement cycle repeats. (Field [83] also interleaves Laplacian smoothing and Delaunay flipping.) As the boundary triangles refine, they are parametrically "pulled" to the correct geometry (which may include spline curves), thereby computing a valid boundary element mesh at the same time as the rest of the mesh. See Figure 28 for an example of an input mesh and the resulting refined mesh; the three nontriangular faces in the input mesh correspond to the holes in the domain. Final aspect ratios are quite insensitive to the quality of the input mesh.

2.5. Two-and-a-half-dimensional Problems

A 2.5-dimensional problem asks for a triangulated surface embedded in three dimensions. We first discuss interpolating surfaces for point set data with elevations, and then triangulated surfaces for three-dimensional models.

2.5.1. Interpolation of Bivariate Functions

The input is a set of points S in the plane, along with a real-valued *elevation* $f(a)$ at each point $a \in S$. Any two-dimensional triangulation $\mathcal{T}$ of the input points induces a piecewise-linear function $f_{\mathcal{T}}$ defined on the region R bounded by the convex hull of S. For each point d, $f_{\mathcal{T}}(d)$ is the weighted average of the elevations at the vertices, a, b, and c, of the triangle abc in $\mathcal{T}$ that contains d. Writing d as $c_1 a + c_2 b + c_3 c$, with $c_1 + c_2 + c_3 = 1$ and $c_1, c_2, c_3 \geq 0$, we have $f_{\mathcal{T}}(d) = c_1 f(a) + c_2 f(b) + c_3 f(c)$. We say that $f_{\mathcal{T}}$ *interpolates* S.

The question arises: which triangulations are good for interpolation? This question has been discussed in the literature [16, 56, 66, 128, 195]. Rippa [177] recently proved a surprising result. Regardless of the input elevations, the Delaunay triangulation gives an interpolating surface, or *elevated triangulation*, optimal in a certain least energy sense.

Theorem 12 (Rippa [177]). *Let $f_{\mathcal{T}}$ be a piecewise-linear function interpolating S induced by a triangulation $\mathcal{T}$. The piecewise-linear function $f_{\mathcal{DT}}$ induced by the Delaunay triangulation satisfies*

$$\iint_R (\nabla f_{\mathcal{DT}})^2 \leq \iint_R (\nabla f_{\mathcal{T}})^2.$$

We now explain the integrals above in terms more familiar to computational geometers. Let abc be a triangle of triangulation $\mathcal{T}$. Let the plane passing through $f(a)$, $f(b)$, and $f(c)$ have the equation $z = Ax + By + C$ in Cartesian coordinates. Then over abc, the gradient squared $(\nabla f_{\mathcal{T}})^2$ is simply the constant $A^2 + B^2$, and the contribution of abc is its area times this constant. Rippa's theorem states that the sum of these contributions over all triangles is minimized by the Delaunay triangulation. The proof of Rippa's theorem is an intricate calculation showing that the flip procedure cannot increase the integral. Hence the CDT is also an optimal interpolating surface. (De Floriani et al. [56] had previously proposed the use of the CDT for this purpose.)

Rippa and Schiff [178] show that the minimization above corresponds to the energy functional associated with the nonhomogeneous Laplace equation. We may think of the DT as giving the stretched membrane (a "drumhead") with least potential energy, among all elevated-triangulation membranes. Rippa and Schiff also discuss other energy functionals, and use the flip algorithm as a heuristic for their minimization. See [16, 69] for heuristic solutions to other interpolation problems.

Bern et al. [23] recently considered the problem of finding the *minimum slope* interpolating surface for input points with elevations (an optimization criterion mentioned in [220]). The slope of an elevated triangle is the slope in the direction of steepest descent, and the slope of an elevated triangulation is the maximum slope of any of its elevated triangles. Bern et al. showed that this problem can be solved in time $O(n^3)$ using the edge-insertion paradigm, discussed in Section 2.2.2. The following lemma shows that the "weak anchor property" holds, thus establishing the applicability of edge insertion.

Lemma 11. *Assume abc is a maximum-slope triangle in elevated triangulation $\mathcal{T}$. Assume the line of steepest descent on abc passes through a (either ascending or descending from a). Assume elevated triangulation $\mathcal{T}'$ has smaller slope than $\mathcal{T}$. Then there is an edge of $\mathcal{T}'$ incident to a that crosses bc (in the projection onto the plane).*

Proof: Assume without loss of generality that the line of steepest descent ℓ descends from a to bc. If the lemma is false, then $\mathcal{T}'$ must contain an elevated triangle ade, with de intersecting both ab and ac in the projection onto the plane containing the input. (Edge de does not necessarily cross both ab and ac, so e could be identical to c.) A vertical plane V_ℓ through ℓ must cut de above ℓ, since $\mathcal{T}'$ has smaller slope than $\mathcal{T}$. So at least one of d and e, say d, must lie strictly above the plane containing abc. See Figure 29.

Now consider the elevated triangle adb (which is not necessarily a triangle of $\mathcal{T}$ or $\mathcal{T}'$). Because d lies above the plane containing abc, the slope of adb must be greater than the slope of abc. (Here notice that V_ℓ intersects the plane containing adb in a line with steeper slope than ℓ.) Let ℓ' be the line of steepest descent on adb. If ℓ' connects d with ab, we consider the triangles of $\mathcal{T}$ that intersect $V_{\ell'}$, a vertical

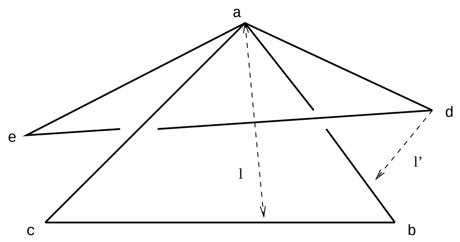


Figure 29. The weak anchor property holds for minmax slope.

plane through ℓ'. These triangles intersect $V_{\ell'}$ in a polygonal path from d down to ab; at least one edge of this path must have slope at least that of ℓ', a contradiction to the assumption that abc is a maximum-slope triangle in $\mathcal{T}$. Contradictions also follow in the other two cases: when ℓ' connects b with ad or a with bd. ∎

This lemma also holds for constrained triangulations, giving an $O(n^3)$ algorithm for finding a least-slope interpolating surface for polygonal inputs with holes and elevations. There are a host of open questions on optimal interpolation; we list three. Many interpolation problems, including the second one listed below, also make sense when Steiner points are allowed.

Open Problem 5. *For point set data with elevations, can a triangulation maximizing the minimum angle on an elevated triangle be computed in polynomial time?*

Open Problem 6. *For point set data with elevations, can a triangulation approximately minimizing the total surface area be computed in polynomial time? (An intriguing result [163] shows that for any such point set there exists a "flattening coefficient" $\epsilon > 0$, such that if all z-coordinates are multiplied by ϵ, Delaunay triangulation minimizes area.)*

Open Problem 7. *For point set data with elevations, can a triangulation with least-sharp sharpest dihedral angle be computed in polynomial time? (See [69].)*

2.5.2. Surfaces for three-dimensional models

Here the input is a "solid model" and the output is a triangulated surface. This step often precedes three-dimensional mesh generation, especially in advancing-front mesh generators.

Solid models take a number of rather varied forms, which complicates even the definition of problems. Constructive solid geometry (CSG) defines a polyhedron

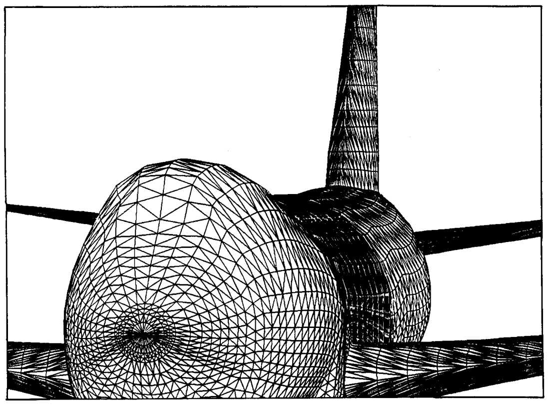


Figure 30. Surface triangulation of a Boeing 747-200 (Baker).

as the intersection and union of primitive polyhedra, such as half-spaces. (One may also allow sphere and cylinder primitives, in order to construct curved solids.) A polyhedron can also be more explicitly defined by a "boundary representation". Solids can be approximately defined by point sets, either all interior to the solid, or labeled "interior" and "exterior". Such inputs occur in scientific visualization, learning theory, and computer graphics. Inputs in medical and aerospace applications often take the form of regularly spaced planar cross-sections [11, 15, 31]. Finally, curved surfaces in design and graphics may be defined by spline patches [18], or by "implicit surfaces"—level sets of functions of three variables.

Triangulating the surface of a polyhedron reduces to triangulating a PSLG; Lindholm [136] uses an advancing-front generator within each face. Solids defined by point sets, however, present some fresh problems. One may want to represent such an input with an enclosing or interpolating polyhedron. The convex hull (the intersection of all half-spaces containing the point set) is an enclosing polyhedron, but it does not usually give a good representation of the "shape" of the point set. Roughly speaking, an α-shape generalizes the convex hull by replacing half-spaces with balls or complements of balls, with radius $1/\alpha$ [70, 74]. This generalization allows more faithful shape representation, but may have complexity $\Omega(n^2)$. Another approach to shape representation, about which little is known, is to find an enclosing or interpolating polyhedron (without Steiner points) that is optimal for some criterion. The following problem is an example.

Open Problem 8. *For points in three dimensions, can an approximate min-surface-area enclosing polyhedron be found in polynomial time?*

A number of authors (see [15, 102, 196] for surveys) have considered the

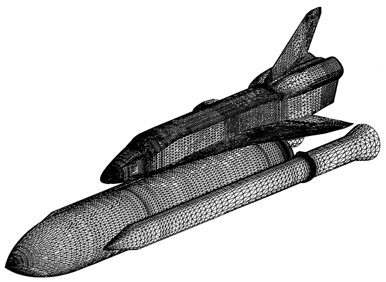


Figure 31. Surface triangulation of the U.S. Space Shuttle (Baker).

problem of computing a triangulated surface interpolating a number of parallel planar cross-sections. Boissonat [31] takes each adjacent pair of cross-sections and uses planar conforming Delaunay triangulations to help compute a three-dimensional tetrahedralization spanning the pair. Baker [11] uses a related heuristic to compute a surface triangulation of an aircraft. He places points interior to each polygonal cross-section, roughly one point for each of the polygon's vertices, and then computes the three-dimensional DT of all input and interior vertices. Each interior tetrahedron has at least two Steiner vertices, so that the union of the tetrahedra with three or four input vertices contains a surface triangulation of the aircraft. See Figures 30 and 31. For this algorithm to succeed, input vertices must be closely spaced relative to the thickness and separation of aircraft parts.

Barequet and Sharir [15] propose a different heuristic for this same problem. They take two adjacent cross-sections and project them onto the same parallel plane. Matching sections of polygons are "stitched" together with a back-and-forth triangulation. Also stitched together are short stretches on either side of each intersection, where a polygon from one cross-section crosses one from the other. At this point, remaining regions in the projection are all bounded by simple polygons; these are filled in by triangulations minimizing total lifted area (when cross-sections are separated again), using dynamic programming as in Section 2.2.3. This last step may use triangles contained in the cross-sectional planes; indeed this capability is necessary in order to form a surface that does not intersect itself [97].

A number of researchers [51, 84, 103, 191] have worked on meshing curved surfaces. Chew [51] has extended his two-dimensional mesh generator (Section 2.3.1) to curved surfaces by generalizing planar Delaunay triangulation. He defines the circumcircle of a triangle to be the intersection of the curved surface with a sphere that has its center on the surface and passes through the triangle's vertices. If the surface is not too curved relative to the size of the triangle, there is a unique such sphere. This definition has the advantage that circumcenters lie on the surface, facilitating the extension of the mesh generator. As before, the resulting mesh has all angles greater than $30°$, but now triangle sizes are controlled locally by deviation

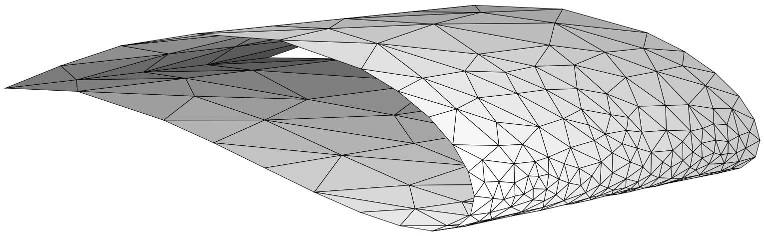


Figure 32. Surface triangulation of a wing (Chew).

from the curved surface. See Figure 32.

Edelsbrunner and Shah [74] recently gave a more formal development of Delaunay triangulation restricted to a subspace such as a curved surface. In their scheme, the restricted DT is the dual of the diagram obtained by intersecting the Voronoi diagram with the subspace. In general, their method produces a simplicial complex rather than a triangulation, meaning that lower-dimensional faces (vertices and edges) need not be contained in triangles. Finally, Bloomenthal [30] uses octrees to polygonize implicit surfaces; these surfaces can then be triangulated if desired. Octrees have been applied to polygonizing spline surfaces as well.

3. Three dimensional triangulations

Triangulation in three dimensions is called *tetrahedralization* (or sometimes *tetrahedrization*). A tetrahedralization is a partition of the input domain, point set or polyhedron, into a collection of tetrahedra, that meet only at shared faces (vertices, edges, or triangles). Tetrahedralization turns out to be significantly more complicated than triangulation.

As in two dimensions, n represents the number of vertices of the input domain, and we distinguish several different types of domains.

- **Simple polyhedron.** A simple polyhedron is topologically equivalent to a sphere; it does not meet itself in a handle, or touch itself at a point or an edge. The boundary of such a polyhedron forms a connected planar graph. In triangulations without Steiner points, each tetrahedron's vertices must be vertices of the polyhedron.
- **Nonsimple polyhedron.** A nonsimple polyhedron may be multiply connected, topologically equivalent to a torus or a higher-genus surface. It may also have cavities, meaning that its boundary is not connected.
- **Point set.** As in two dimensions, a triangulation of a point set fills the convex hull. If Steiner points are allowed, then the boundary of the triangulation may be a larger convex polytope.

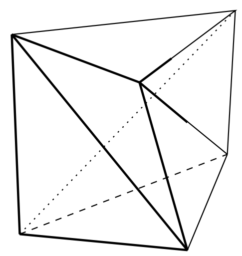


Figure 33. Schönhardt's untetrahedralizable polyhedron.

3.1. Tetrahedralization without Optimization

In this section, we concentrate on existence and construction of tetrahedralizations, without concern for optimality. Existence and construction are already interesting, since many two-dimensional triangulation properties break down in three dimensions.

The first surprise is that different triangulations of the very same input may contain different numbers of tetrahedra. For example, choose n points $v_i = (i, i^2, i^3)$ on the *moment curve.* It is not hard to show that their convex hull can be triangulated with the $\binom{n-2}{2}$ tetrahedra of the form $v_i v_{i+1} v_j v_{j+1}$. (In fact this is the Delaunay triangulation of these points.) A generalization of Euler's formula shows that any tetrahedralization of an n-vertex polyhedron has at most this many tetrahedra. If we choose the tetrahedralization carefully, however, we can achieve linear, rather than quadratic, complexity for this same input. In fact, any strictly convex polyhedron can be tetrahedralized with at most $2n - 7$ tetrahedra: choose a vertex v, triangulate each face of the polyhedron that is not adjacent to v, and then connect v to each triangle. This bound is within a factor of two of optimal, as any tetrahedralization of a simple polyhedron has at least $n - 3$ tetrahedra.

Edelsbrunner, Preparata, and West [77] show how to construct a linear-complexity tetrahedralization of point sets in general position. After tetrahedralizing the convex hull with only linear complexity as above, interior points are added one at a time. When a point is added, the tetrahedron containing it is replaced by four smaller tetrahedra. In contrast, linear complexity is not always possible for points in special position [4].

When we try to extend these results to nonconvex polyhedra, we meet a second surprise: not all polyhedra are tetrahedralizable [132]. The following counterexample is due to Schönhardt [193]. Start with a triangular prism, and twist one triangle relative to the other so that each rectangular face of the prism folds into two triangles with a reflex edge between them (Figure 33). Any set of four vertices must include a pair that face each other across such a reflex edge. So the polyhedron contains no tetrahedron, and tetrahedralization is impossible.

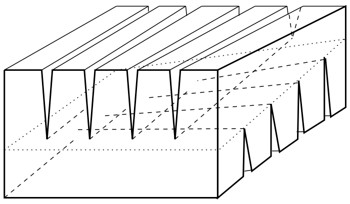


Figure 34. Chazelle's lower bound example.

Schönhardt's polyhedron can be tetrahedralized if we add one Steiner point. This leads to the question of how many Steiner points may be required for tetrahedralization. Chazelle [42] found a simple polyhedron in which $\Omega(n^2)$ Steiner points are needed even to partition the polyhedron into convex regions. Clearly, this is also a lower bound for tetrahedralization.

Chazelle's polyhedron (Figure 34) can be built from a cube by removing numerous thin wedges. Wedges parallel to the y-axis are removed from the top face of the cube, and wedges parallel to the x-axis are removed from the bottom face. The reflex edges at the tips of the wedges form two sets of lines, that almost meet at the center of the polyhedron, near the hyperbolic surface $z = xy$. Viewed from above, the lines partition this surface into $\Omega(n^2)$ small squares. The centers of the squares can be connected by lines that lie on the doubly-ruled hyperbolic surface, but by slightly turning the grid of centers we can find a set of $\Omega(n^2)$ points, no pair of which are mutually visible. Lines nearly parallel to the x-axis (respectively, y-axis) are blocked from above (below) by wedges. Hence no pair of points can lie in the same convex region, so there must be $\Omega(n^2)$ regions in any convex decomposition, and a fortiori $\Omega(n^2)$ tetrahedra in any tetrahedralization. As we now show, this bound is tight.

Theorem 13. *Any polyhedron can be triangulated with $O(n^2)$ Steiner points and $O(n^2)$ tetrahedra.*

Proof: Extend a vertical "wall" from each edge of the polyhedron boundary, up and down from that edge until it reaches some other part of the boundary. These walls divide the polygon into generalized cylinders. Triangulating the top and bottom faces of the cylinders partitions the polygon into $O(n^2)$ triangular prisms. Each vertical prism side is crossed at most once by a polyhedron edge, so the prisms are polyhedra with at most twelve vertices. Triangulate the faces of these polyhedra, making sure that tetrahedra from different prisms will meet face to face, and then triangulate each prism with at most 20 tetrahedra incident to a single interior Steiner point. (We need the Steiner point as the edges crossing the vertical faces make the prisms not strictly convex.) ∎

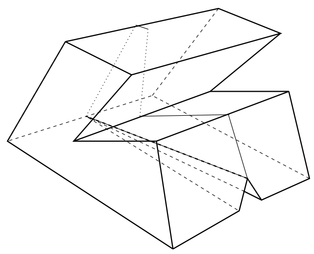


Figure 35. Extending a vertical wall from a reflex edge.

As we have seen, convex polyhedra can be triangulated with only $O(n)$ tetrahedra, while Chazelle's polyhedron requires $\Omega(n^2)$. This suggests the possibility that an appropriate measure of nonconvexity would interpolate these bounds and achieve $o(n^2)$ tetrahedra for "slightly nonconvex" polyhedra. One natural measure is r, the number of reflex edges. Chazelle and Palios [46] developed a triangulation algorithm sensitive to this measure.

Let $N(v)$ be the set of neighbors of vertex v, and define the *cap* of v to be the star-shaped polyhedron formed by removing the convex hull of $N(v)$ from the convex hull of $N(v) \cup \{v\}$. See Figure 36. Since the boundary of the polyhedron forms a planar graph, one can always find a vertex v with a cap with at most five other vertices. One would like to remove such a cap from the polyhedron, replacing it with at most three tetrahedra, and continue until the polyhedron is triangulated. This would be analogous to triangulating a polygon in the plane by removing a single ear triangle at a time. Since not all polyhedra are tetrahedralizable, this approach does not work—the difficulty is that the rest of the polyhedron might penetrate into the cap, so that it could not be removed without causing the polyhedron to intersect itself.

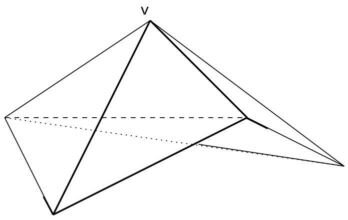


Figure 36. The cap of vertex v.

Lemma 12. *Let S be a set of vertices of a simple polyhedron with triangular faces, such that no vertex of S is adjacent to another vertex of S or to a reflex edge. Then at most $2r$ caps of vertices in S are penetrated by other portions of the polyhedron.*

Proof Sketch: Suppose the cap of vertex v is penetrated. The faces touching v are boundaries of the polyhedron, so the penetration must occur through the remaining faces of the cap. Moreover, since no edge can completely cross the cap without crossing a face adjacent to v, there must be a polyhedron vertex interior to the cap.

Project all interior vertices onto a line extending from v through the cap, and let w be the vertex with projection closest to v. Then w cannot be in any other cap penetrating the cap of v, and—it is not hard to show—w can be chosen as closest by at most two other caps not penetrating the cap of v. Line segment vw must be entirely contained in the polyhedron; otherwise it would cross a face, one vertex of which would be closer to v. And finally w must be an endpoint of at least three reflex edges; otherwise, one of the faces incident to w would have a vertex closer to v. So we can charge penetrated caps one-for-one to endpoints of reflex edges. ∎

Theorem 14 (Chazelle and Palios [46]). *Any simple polyhedron with n vertices and r reflex edges can be partitioned into $O(n+r^2)$ tetrahedra using $O(n+r^2)$ Steiner points.*

Proof: Start by triangulating the faces of the polyhedron. As in any planar graph, we can find $\Omega(n)$ nonadjacent vertices, all with degree at most six. By Lemma 12, unless n is $O(r)$, one of these vertices has a low-degree cap that is not penetrated. Remove this cap, leaving a smaller polyhedron. The cap itself can be split into $O(1)$ tetrahedra. After we remove all but $O(r)$ vertices, the remaining polyhedron can be triangulated by the vertical wall method. ∎

The resulting partition is not yet a triangulation, because the vertical walls subdivide the faces of the reduced polyhedron without matching the caps removed from those faces. If m wall edges occur on the faces of the cap of vertex v, then v's cap can be triangulated with $O(m)$ tetrahedra with apex v, and this subdivision does not propagate into other caps. The complexity of the triangulation may grow if an edge of the reduced polyhedron is shared by many removed caps. This complication can be handled by surrounding each such edge with a narrow prism-shaped polyhedron before doing the vertical wall construction. Now vertical walls subdivide the faces of the prism, rather than the edge itself; the prism can be triangulated with a single Steiner point. We omit the details of handling the tips of the prisms where they meet at vertices of the reduced polyhedron. The final product is a triangulation with $O(n+r^2)$ tetrahedra, that with careful implementation can be constructed in time $O((n+r^2)\log r)$ [46].

The algorithm just given does not use the assumption that the polyhedron is simple in any essential way, only in the analysis of the number of tetrahedra. Recently, Chazelle and Shouraboura [47] used the Gauss-Bonnet formula to show that the genus of a polyhedron is bounded by the number of reflex edges, thereby extending the same $O(n+r^2)$ bound to non-simple polyhedra.

Dey [59] and Hershberger and Snoeyink [107] analyze another algorithm—called *notch cutting* [42]—for convex partition and tetrahedralization of non-simple polyhedra. This algorithm incrementally bisects reflex dihedrals with planes, extending the plane in all directions away from the reflex edge until it first hits the polyhedron boundary. Hershberger and Snoeyink prove a tight worst-case bound of $O(nr + r^{7/3})$ tetrahedra.

These results reveal that nearly convex polyhedra require few Steiner points. The question arises: can we find an efficient algorithm that uses the minimum number of Steiner points? Ruppert and Seidel [184] give a negative answer to this question. They show that testing whether Steiner points are necessary to triangulate a given polyhedron is NP-complete (see [92]), even for star-shaped polyhedra (which can trivially be triangulated with a single Steiner point). They similarly prove that, for any k, it is NP-hard to test whether k Steiner points suffice. The following open question asks for an approximation algorithm.

Open Problem 9. *Is there an efficient algorithm for triangulating any n-vertex polyhedron into $O(m)$ tetrahedra, where m is the minimum possible number?*

Certain very special polyhedra can be triangulated without Steiner points. Goodman and Pach [99] prove that the region *between* two convex polyhedra (the convex hull of the union, minus the polyhedra) can be tetrahedralized into $O(n^2)$ tetrahedra by lifting the polyhedra onto a folded plane in four dimensions. (This result generalizes to arbitrary dimension.) Bern [20] improves this bound to $O(n \log n)$ for the region between two nested convex polyhedra. He also shows that even for general polyhedra, only interior Steiner points are necessary. This result may have application to multiple domains (see also [156]); it follows from "fattening" each face into a roughly prism-shaped solid. Toussaint et al. [215] prove that the union of three convex polyhedra can be tetrahedralized without Steiner points.

Chazelle and Shouraboura [47] improve Bern's $O(n \log n)$ bound to linear by reintroducing Steiner points. They also prove that the region between a convex polyhedron and a terrain (a polyhedral surface intersected once by each vertical line) can be triangulated with $O(n \log n)$ tetrahedra, and—remarkably—this bound is tight. Their first result raises the following question.

Open Problem 10. *Can the region between c convex polyhedra (the convex hull of the union, minus the polyhedra) be triangulated with $O(n + c^2)$ tetrahedra?*

3.2. Optimal Tetrahedralization

In this section, we consider three-dimensional optimal triangulation without Steiner points. Since Steiner points are required simply to tetrahedralize nonconvex polyhedra, this section treats only point sets (and as a special case, convex polyhedra). Even for point sets very little is known.

Since a single input has tetrahedralizations of different complexity, a natural optimization question is the following. A more general open question asks for a minimum-complexity tetrahedralization of a point set.

Open Problem 11. *Is there a polynomial-time algorithm for triangulating an arbitrary convex polyhedron with the minimum number of tetrahedra?*

The Delaunay triangulation (DT) in $\mathbb{R}^3$ contains each tetrahedron with vertices from the input point set, whose circumsphere contains no other input points on its surface or in its interior. Assuming general position, no five points lie on a single sphere, so this defines a triangulation. The complexity of the DT may be as high as $\Omega(n^2)$, as shown by the moment curve example (Section 3.1). There does not seem to be a reasonable definition of constrained DT in three dimensions.

The *lifting* transformation defined in Section 2.2.1 generalizes to three (and higher) dimensions. We map an input point with Cartesian coordinates (x, y, z) to the point $(x, y, z, x^2 + y^2 + z^2)$. The image points all lie on a paraboloid in four dimensions; the projection of the lower convex hull back onto the xyz-hyperplane gives the DT. Coupled with an algorithm for computing four-dimensional convex hulls [197], this gives a worst-case optimal, quadratic-time algorithm to compute the DT.

There are also direct algorithms. Bowyer [33] and Watson [219, 82] gave incremental algorithms that are quite popular in practice. Watson's algorithm inserts points in sorted order by one coordinate, testing all old circumspheres that intersect the current sweep plane. Bowyer includes evidence that his algorithm runs in time $O(n^{4/3})$ for a random point set. Dwyer [68] gives a linear-expected-time algorithm for random points in the unit ball.

Joe [115] and Rajan [176] generalize the flip algorithm for DT construction. In three dimensions, flips involve sets of five points, forming a tetrahedral bipyramid. Such a figure can be tetrahedralized in two ways: either as a pair of tetrahedra separated by a face, or as three tetrahedra surrounding an interior diagonal. Thus flips trade two tetrahedra for three, or vice versa. See Figure 37. Starting from an arbitrary tetrahedralization, however, the flip algorithm can get stuck in a local optimum and fail to produce the DT [115].

Joe [117] showed that, if we start with the DT of some point set, and add a single point (dividing the tetrahedron containing it into four, or if the new point is outside the convex hull, adding tetrahedra connecting it to the triangles it can see), then flipping from the resulting triangulation never gets stuck. All tetrahedra involved in flips are neighbors of the new vertex, so in some sense this flipping procedure becomes two- rather than three-dimensional. This result gives another $O(n^2)$-time algorithm for computing the DT: add points one by one (say, in sorted order by x-coordinate) and, after each addition, flip until the DT is reached. Rajan [176] described a similar procedure for incrementally adding points and flipping

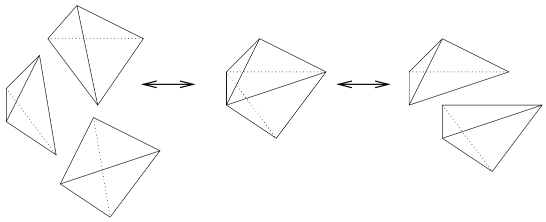


Figure 37. Two ways of tetrahedralizing five points.

tetrahedra to find the DT. His procedure flips tetrahedra in the order corresponding to the changes in the convex hull of the lifted points as the new point is moved vertically down onto the paraboloid. Thus, Rajan's algorithm generalizes to higher-dimensional DT construction. Edelsbrunner and Shah [73] extended Joe's algorithm to higher dimensions and to regular triangulations of weighted points.

Though the worst-case time bound for computing the three-dimensional DT must be $\Omega(n^2)$, an "output-sensitive" algorithm runs much more quickly on simple input instances. Techniques recently developed by Matoušek [148] for half-space range queries give an algorithm with running time $O(n^{4/3+\epsilon} + k \log n)$, where k is the complexity of the DT. See Fortune's survey [88] for more details on three-dimensional DT algorithms, including some important implementation issues.

Because the DT possesses so many optimality properties in two dimensions, geometers long suspected that it should optimize something in three dimensions. Recently, Rajan [176] discovered the first such result. (His result actually holds in all dimensions.) The *min-containment sphere* of a simplex t is the smallest sphere containing t. If t contains its circumcenter, then the min-containment sphere is identical to the circumsphere. Otherwise, the min-containment sphere circumscribes a lower-dimensional face of t. For example, in two dimensions, the min-containment sphere is either the circumcircle or the diameter circle of the longest edge. Rajan proved the following, which generalizes a result of D'Azevedo and Simpson for the planar case [55].

Theorem 15 (Rajan [176]). *The Delaunay triangulation is the triangulation that minimizes the maximum radius of a min-containment sphere.*

Proof Sketch: Lift the points to the paraboloid $(x, y, z, x^2+y^2+z^2)$. Any sphere in $\mathbb{R}^3$ corresponds to a hyperplane in four dimensions cutting this paraboloid. Let $\mathcal{T}$ be any triangulation, and for any tetrahedron t in $\mathcal{T}$, define $H(t)$ to be the half-space above the hyperplane corresponding to the circumsphere of t. If S is the min-containment sphere of t, the radius of S corresponds to the vertical distance between the lifted center of S and $H(t)$. Now form a polytope $P(\mathcal{T})$ as the intersection of all such halfspaces. (This projects to a power diagram—see Section 2.2.1—in the original space.) Then the largest min-containment sphere corresponds to the

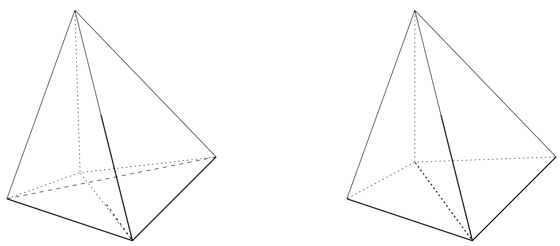


Figure 38. The DT (left) contains a very flat "sliver" tetrahedron.

largest distance between this polytope and the portion of the paraboloid to which the convex hull of the input can be lifted. The DT is the convex hull of the lifted points, so it is lower than any other possible polytope $P(\mathcal{T})$, and hence minimizes this distance. ∎

Dey (personal communication) observed that for cospherical points, all triangulations have the same maximum min-containment radius, so any completion of the Delaunay triangulation solves the degenerate case. Figure 38 gives a counterexample to the plausible conjecture that the DT also minimizes the radius of the largest circumsphere.

Recently, Schmitt and Spehner [192] proved a second optimality theorem that holds in arbitrary dimension. We describe the result for $\mathbb{R}^3$. Define the co-angle of a triangle in a tetrahedron t to be half the angle of the cone with apex at t's circumcenter and base equal to the triangle's circumcircle. Using a lifting argument as above, Schmitt and Spehner prove that each interior face in a DT has a sum of co-angles (one from each side) at most 180°, while any other triangulation has a face that exceeds 180°. This result is the three-dimensional analog of the fact that a reversed quadrilateral has unsplit angles summing to more than 180°.

3.3. Steiner Tetrahedralization

We have already touched on the subject of Steiner tetrahedralization in Section 3.1, because tetrahedralizing a nonconvex polyhedron may require Steiner points. We now discuss problems in which Steiner points are used to improve the quality of the solution.

3.3.1. Reducing Delaunay Triangulations

The Delaunay triangulation of a set of n points in $\mathbb{R}^3$ may have $\Omega(n^2)$ tetrahedra, though a "typical" point set has only $O(n)$ [68]. This raises the question of whether Steiner points can be used to reduce the complexity of the DT.

Chazelle et al. [44] answered this question affirmatively by showing that for any point set, there exists a set of $O(n^{1/2}\log^3 n)$ points, such that the Delaunay triangulation of the union of the two point sets has $O(n^{3/2}\log^3 n)$ tetrahedra. Their method repeatedly finds a point that lies inside a large number of Delaunay circumspheres; the addition of such a point removes all the corresponding tetrahedra and replaces them by $O(n)$ new tetrahedra.

The success of this method follows from a combinatorial lemma of independent interest that holds in arbitrary fixed dimension d. If there are m spheres, each passing through a pair of points, then some point of space (not necessarily one of the n input points) is interior to $\Omega(m^2/(n^2\log^{2d}(n^2/m)))$ spheres. Results stronger by log factors hold for diameter spheres and rectangular boxes.

Bern, Eppstein, and Gilbert [25] showed how to use more Steiner points, and reduce the complexity of a Delaunay triangulation to $O(n)$. Their technique is completely different, and it works for any fixed dimension. The algorithm first computes a balanced octree (in general, a 2^d-ary tree) such that each point is alone in a cube surrounded by empty cubes the same size. The closest cube vertex to each input point is then replaced by that input point.

To reduce the size of the tree, the algorithm identifies long chains of cubes, say more than 2^d levels, in which each cube has only one nonempty child cube. Next it removes the middle-sized cubes of these chains, leaving small cubes floating inside large cubes; then the algorithm surrounds each small cube with a constant number of layers of cubes its own size. This guarantees that every d-sphere that contains input points from both inside and outside a small floating cube, also contains at least one vertex from these layers. Now every point is incident on $O(1)$ maximal empty spheres, so using the vertices of cubes as Steiner points gives a linear-size Delaunay triangulation. As just explained, the running time of this algorithm depends on the size of the initial tree, but long chains can be identified without actually computing them to give a time bound of $O(n\log n)$.

3.3.2. Nonobtuse Triangulation

There is more than one way to generalize nonobtuse triangulation to three dimensions. For a convergence bound on a certain numerical method, Vavasis [217] assumes that no dihedral angle is obtuse. Nonobtuse dihedrals, however, do not imply that the mesh is a Delaunay triangulation. For this implication we need a different generalization: each tetrahedron is *self-centered*, meaning that it contains its circumcenter [176]. Finally, in order to generalize perpendicular duals, we need a stronger condition than self-centered that we shall call *fully self-centered*: triangles and tetrahedra are all self-centered.

In a recent paper, Bern, Chew, Eppstein, and Mitchell [21] give an algorithm that solves all these generalizations for point sets in arbitrary dimension d. They show how to Steiner-triangulate n input points with $O(n^{\lceil d/2 \rceil})$ *path simplices*. A path simplex contains a path of d pairwise orthogonal edges; it is fully self-centered and all its dihedral angles (angles between $(d-1)$-dimensional faces) are nonobtuse.

The algorithm successively projects the point set onto two-dimensional planes, and solves $\lfloor d/2 \rfloor$ planar problems using a linear-complexity all-right-triangle algorithm for point sets [25, 29]. Back in $\mathbb{R}^d$, the products of right triangles form solids—right-triangular prisms in $\mathbb{R}^3$—that can be triangulated with path simplices.

Bern et al. [21] also give a lower bound result. To explain this result, we require some preliminaries. Any set of $k < d$ vertices—a $(k-1)$-simplex—in a d-dimensional simplex defines a $(d+1-k)$-dimensional angle. The angle can be measured by placing a small perpendicular sphere (that is, lying in a perpendicular $(d+1-k)$-flat) around the $(k-1)$-simplex, and determining what fraction of the sphere is subtended by the $(d+1-k)$-simplex opposite the $(k-1)$-simplex. For example, there are two kinds of angles in three dimensions: solid angles which are measured by subtended area on a sphere around a vertex, and dihedral angles which are measured by subtended arc length on a circle around an edge.

We can now define $2(d-1)$ different no-bad-angle problems. For each k, we can forbid either small or large $(d+1-k)$-angles. To forbid small angles, we require that all angles be bounded away from zero; to forbid large angles, we require that all angles be bounded away from half a sphere. No-small-angle and no-large-angle problems are each linearly ordered by difficulty. No small d-angles implies no bad angles (small or large) of any kind. No small $(d-1)$-angles implies no small $(d-2)$-angles, and so forth down to no small 2-angles (dihedrals). For large angles, the dimensions reverse, with no large dihedrals first, then no large 3-angles, on down to no large d-angles. A large solid (that, is d-) angle implies bad angles of all types. The hardest problem that can be solved with polynomial complexity is the no-large-dihedral problem, and the most stringent bound on dihedral angles is 90°.

Still open is the question of whether $O(n^{\lceil d/2 \rceil})$ is the best possible bound for the no-large-dihedral problem. And, getting back to more realistic mesh generation questions, no algorithms are known for nonobtuse tetrahedralization of polyhedra.

3.3.3. Bounded-Aspect-Ratio Mesh Generation

The most commonly used definition of the *aspect ratio* of a simplex is the ratio of the radii of the circumscribed sphere to the inscribed sphere [82]. Bounded aspect ratio is equivalent to no small d-angles. We now give descriptive names to the six different types of three-dimensional simplices, classified according to the types of bad angles allowed. (See [10, 60] for similar classifications.)

- **Round.** A round tetrahedron has no bad angles of any kind.
- **Needle.** A needle has one small solid angle, but no small or large dihedrals.

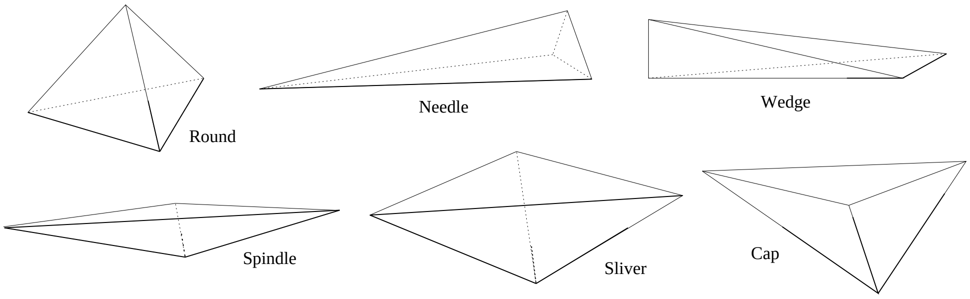


Figure 39. The six types of tetrahedra.

- **Wedge.** A wedge has small but not large dihedrals and no large angles of any kind. An example is a tetrahedron that is the convex hull of two far-apart, short perpendicular edges.

- **Spindle.** A spindle has small but not large solid angles, and large but not small dihedrals.

- **Sliver.** A sliver has small and large dihedrals, but no large solid angle. An example is a tetrahedron formed by four nearly coplanar points, fairly evenly spaced around a great circle of the circumsphere. Slivers are distinguished from the other bad tetrahedra by a low ratio of circumsphere radius to shortest edge length.

- **Cap.** A cap has a large—nearly flat—solid angle. The circumsphere's radius is hence much larger than the longest edge.

What is known on bounded-aspect-ratio triangulation in three dimensions? Bern et al. [25] use octrees (or in general 2^d-ary trees) to triangulate point set input, using only a constant times the minimum number of bounded-aspect-ratio tetrahedra. Dey, Bajaj, and Sugihara [60] generalize Chew's mesh generation algorithm to three dimensions. For point sets or convex polyhedra with point holes, this method avoids all bad tetrahedra except slivers.

Mitchell and Vavasis [157] have recently generalized the quadtree polygon mesh generation of Bern et al. [25] to three dimensions, avoiding all types of bad tetrahedra. The generalization is not straightforward, primarily because vertices of polyhedra may have arbitrary degree.

Their algorithm first computes a balanced octree that safely separates—that is, by some constant number of cubes—faces of the input polyhedron D. The octree is refined in three phases: first, vertices are separated from nonincident edges and faces; second, boxes away from vertices are split to separate edges from other edges and facets; and finally, boxes away from both vertices and edges are split to separate facets from other facets. Cubes are duplicated in Riemann sheets for faces close together in space, but far apart by geodesic distance.

Next the boxes around vertices are merged in order to approximately center each vertex in its box. The intersection of the boundary of D and the surface of a box must be triangulated with triangles of height a constant fraction of the maximum possible (as in Section 2.3.3). A complicated set of warping rules conforms the octree to the edges and facets of D. Finally, warped boxes are triangulated by adding Steiner points near their centers, with tetrahedra radiating from these points.

Let A be the minimum aspect ratio of a Steiner triangulation of domain D. (Here the aspect ratio of a triangulation is the maximum aspect ratio of its tetrahedra.) The algorithm just sketched gives a triangulation with aspect ratio at most cA (where c is a constant), that uses at most a constant factor times the minimum number of tetrahedra needed to achieve cA. The proof of this theorem also requires a new idea beyond [25]. Because no analog of the CDT is known in three dimensions, Mitchell and Vavasis must compare their tetrahedralization to an optimal tetrahedralization, showing that at each point in D the tetrahedron chosen by their algorithm is no more than a constant times smaller than the largest possible tetrahedron at that point.

Theorem 16 (Mitchell and Vavasis [157]). *There is an algorithm, based on octrees, that computes an approximate optimal-aspect-ratio tetrahedralization of an arbitrary polyhedral domain, using no more than a constant times the optimal number of tetrahedra.*

The theorem has special importance because the edge skeleton of a bounded-aspect-ratio tetrahedralization has a "separator" of complexity $O(n^{2/3})$ [153, 154]. (A separator is a set of vertices whose removal disconnects the graph into two pieces of roughly equal size.) Nested dissection then saves a factor of $O(n)$ in the asymptotic time to solve the linear equations that arise in the finite element method [139].

3.4. Heuristically Generated Three-Dimensional Meshes

Most of the techniques for generating and improving two-dimensional meshes can be generalized to three dimensions, though not without some difficulties. See [9, 95, 144] for surveys discussing both structured and unstructured meshes primarily for fluid flow problems. Overall, three-dimensional unstructured mesh generation is still in its early stages of development.

3.4.1. Mesh improvement

Laplacian smoothing (see Section 2.4.1) generalizes to three dimensions [41, 225], but the improvement it offers may not be as significant as in two dimensions.

There are local tranformations that trade two tetrahedra for three, and three tetrahedra for two, as discussed above (Section 3.2.1). These transformations represent the analog of the flip procedure in two dimensions, and may be used to improve a triangulation according to some criterion, such as the Delaunay empty circumsphere condition. In three dimensions, however, the flip procedure may get stuck in a local optimum that is not a global optimum [115]. Joe [117], Rajan [176], and Edelsbrunner and Shah [73] have shown that, for the Delaunay criterion, special starting triangulations always lead to a global optimum. Joe has also used the flip procedure to locally maximize the minimum solid angle [118]. Starting from the Delaunay triangulation, he significantly improved aspect ratios, while also slightly decreasing the number of tetrahedra.

Rivara refinement generalizes as follows: split a tetrahedron by adding a triangle with apex equal to the midpoint of the longest edge and base equal to the opposite edge. Recursively split all tetrahedra sharing the bisected edge. This algorithm performs quite well in practice [181], but the generalization of the minimum angle bound is currently open.

Open Problem 12. *Does repeated application of three-dimensional Rivara refinement keep the minimum solid angle bounded away from zero?*

3.4.2. Octrees

Yerry and Shephard generalized their quadtree algorithm to a three-dimensional algorithm using balanced octrees [225]. They kept the number of "patterns" for boundary cubes manageable by assuming that each cube was cut by at most three facets of the input polyhedron. In the final steps, patterns are warped to approximate the actual boundary of the input, and Laplacian smoothing is applied. Further progress on their octree algorithm is reported by Shephard et al. [203]. Perucchio et al. [169, 189] have also advanced the octree approach with a different way of handling boundary cubes.

Buratynski [35] uses rectangular octrees, that is, noncubical boxes, and a hierarchical set of warping rules. Boxes are first warped to input corners, then input edges, and finally input faces. The rules are somewhat simplified by the fact that the octree is initially refined so that input edges intersect boxes of only one size. (Hence, this method does not come with the theoretical size guarantee of Mitchell and Vavasis's method [157].) Empirically, Buratynski's method seems to give no tetrahedra with bad aspect ratio.

Field and Smith [85] and others [158, 166] suggest the use of a tetrahedral octree, built by recursively cutting a "bcc tetrahedron" (body-centered cubic) into eight copies of itself. Tetrahedra spawn fewer boundary patterns than hexahedra.

3.4.3. Polyhedron Decomposition

Cavendish, Field, and Frey [41] developed one of the first three-dimensional mesh generators. Their approach cuts the polyhedron into polygonal cross-sections, adds randomly chosen Steiner points with average spacing determined by a measure of local feature size [40], and then computes the Delaunay triangulation. (See [27, 152] for analyses of DTs of random points.) If Steiner points are sufficiently closely spaced on the boundaries of the polygonal cross-sections, the DT will be conforming. A final improvement step merges tetrahedra or moves Steiner points to remove slivers (very flat tetrahedra). Joe [118] is currently working on a generalization of his two-dimensional convex-decomposition generator [119].

3.4.4. Advancing Front

Löhner [144, 145], Baker [8, 9], Jameson et al. [111], Peraire et al. [168], and others have generated tetrahedral meshes for entire aircraft. Typically, something akin to the advancing front method places Steiner points in layers around the aircraft.

Jameson et al. [111] use a number of overlapping structured grids to place Steiner points, and then produce an unstructured tetrahedral grid with the Delaunay triangulation.

Baker's method [8] starts with a surface triangulation, as described in Section 2.5 above. Steiner points are then added exterior to the aircraft. Regular lattices of Steiner points surround the aircraft, with the density of Steiner points decreasing away from the surface. A shell of closer-in Steiner points is created by adding a few points along a normal to each surface point. A Delaunay triangulation is computed incrementally, but with violations allowed where the DT would pierce the surface. A final improvement step removes slivers. Empirically, the only remaining tetrahedra of poor aspect ratio are needles near junctures of aircraft parts, such as where the wing joins the fuselage.

4. Conclusions

We have described work in computational geometry motivated by finite element mesh generation. This material spans a spectrum from purely theoretical results (for example, Chazelle's linear-time triangulation algorithm), through a middle ground (our own work on Steiner triangulations), to practical heuristics devised by numerical analysts.

We believe that worthwhile research is spread throughout this spectrum. We have attempted to gather together these scattered results, and hope this compilation proves useful to both theorists and practitioners.

Acknowledgments

We would like to thank Tim Baker, Paul Chew, Mike Dillencourt, David Dobkin, Herbert Edelsbrunner, David Field, John Gilbert, David Goldberg, Barry Joe, Scott Mitchell, Lee Nackman, John Shaw, Shang-Hua Teng, Steve Vavasis, and Frances Yao for various helpful suggestions.

References

[1] A. Aggarwal, L.J. Guibas, J. Saxe, and P.W. Shor. A linear time algorithm for computing the Voronoi diagram of a convex polygon. *Disc. and Comp. Geometry* 4 (1989) 591–604.

[2] E. Anagnostou and D. Corneil. Polynomial-time instances of the minimum weight triangulation problem. *Comp. Geometry Theory and Applications* 3 (1992) 247–259.

[3] F. Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys* 23 (1991) 345–405.

[4] D. Avis and H. ElGindy. Triangulating point sets in space. *Disc. and Comp. Geometry* 2 (1987) 99–111.

[5] I. Babuška and A. Aziz. On the angle condition in the finite element method, *SIAM J. Numer. Analysis* 13 (1976) 214–227.

[6] I. Babuška and W.C. Rheinboldt. A-posteriori error estimates for the finite element method. *Int. J. Numer. Meth. Eng.* 12 (1978) 1597–1615.

[7] P.L. Baehmann, S.L. Wittchen, M.S. Shepard, K.R. Grice, and M.A. Yerry. Robust geometrically-based automatic two-dimensional generation, *Int. J. Numer. Meth. Eng.* 24 (1987) 1043–1078.

[8] T.J. Baker. Automatic mesh generation for complex three-dimensional regions using a constrained Delaunay triangulation. *Eng. with Computers* 5 (1989) 161–175.

[9] T.J. Baker. Developments and trends in three-dimensional mesh generation. *Appl. Numer. Math.* 5 (1989) 275–304.

[10] T.J. Baker. Element quality in tetrahedral meshes. *7th Int. Conf. on Finite Element Models in Flow Problems*, Huntsville, Alabama, 1989.

[11] T.J. Baker. Unstructured meshes and surface fidelity for complex shapes. In *Proc. 10th AIAA Comp. Fluid Dynamics Conf.*, Hawaii, 1991.

[12] B.S. Baker, E. Grosse, and C.S. Rafferty. Nonobtuse triangulation of polygons. *Disc. and Comp. Geometry* 3 (1988) 147–168.

[13] R.E. Bank. *PLTMG User's Guide.* SIAM, 1990.

[14] R.E. Bank, A.H. Sherman, and A. Weiser. Refinement algorithms and data structures for regular local mesh refinement. In R. Stepleman et al., *Scientific Computing*. IMACS/North-Holland, 1983, 3–17.

[15] G. Barequet and M. Sharir. Piecewise-linear interpolation between polygonal slices. In *Proc. 10th ACM Symp. Comp. Geometry* (1994) 93–102.

[16] R.E. Barnhill. Representation and approximation of surfaces. *Math. Software III*, J.R. Rice, ed., Academic Press, 1977, 69–120.

[17] R.E. Barnhill. Computer aided surface representation and design, *Surfaces in Computer Aided Geometric Design*, R. Barnhill and W. Boehm, eds., North-Holland, Amsterdam, 1983, 1–24.

[18] R.H. Bartels, J.C. Beatty, and B.A. Barsky. *An Introduction to Splines for Use in Computer Graphics and Modeling*, Morgan Kaufmann, 1987.

[19] T.J. Barth and D.C. Jespersen. The design and application of upwind schemes on unstructured meshes. In *Proc. AIAA 27th Aerospace Sciences Meeting*, Reno, 1989.

[20] M. Bern. Compatible tetrahedralizations. In *Proc. 9th ACM Symp. Comp. Geometry* (1993) 281–288.

[21] M. Bern, L.P. Chew, D. Eppstein, and J. Ruppert. Dihedral bounds for mesh generation in high dimensions. Manuscript, 1994.

[22] M. Bern, D. Dobkin, and D. Eppstein. Triangulating polygons without large angles. In *Proc. 8th ACM Symp. Comp. Geometry* (1992) 222–231.

[23] M. Bern, H. Edelsbrunner, D. Eppstein, S. Mitchell, and T.S. Tan. Edge-insertion for optimal triangulations. *Disc. and Comp. Geometry* 10 (1993) 47–65. Also in *Proc. Latin American Theoretical Informatics*, Springer-Verlag LNCS 583 (1992) 46–60.

[24] M. Bern and D. Eppstein. Polynomial-size nonobtuse triangulation of polygons. *Int. J. Comp. Geometry and Applications* 2 (1992) 241–255.

[25] M. Bern, D. Eppstein, and J.R. Gilbert. Provably good mesh generation. In *Proc. 31st IEEE Symp. Foundations of Computer Science* (1990) 231–241. To appear in *J. Comp. System Science*.

[26] M. Bern, D. Eppstein, and S.-H. Teng. Parallel construction of quadtrees and quality triangulations. In *Proc. 3rd Workshop on Algorithms and Data Structures*, Springer-Verlag LNCS 709 (1993) 188–199.

[27] M. Bern, D. Eppstein, and F. Yao. The expected extremes in a Delaunay triangulation. *Int. J. Comp. Geometry and Applications* 1 (1991) 79–92.

[28] M. Bern and J.R. Gilbert. Drawing the planar dual. *Inform. Process. Lett.* 43 (1992) 7–13.

[29] M. Bern, S. Mitchell, and J. Ruppert. Linear-size nonobtuse triangulation of polygons. In *Proc. 10th ACM Symp. Comp. Geometry* (1994) 221–230.

[30] J. Bloomenthal. Polygonization of implicit surfaces. *CAGD* 5 (1988) 341–355.

[31] J.D. Boissonat. Shape reconstruction from planar cross sections. *Comp. Vision, Graphics, and Image Processing* 44 (1988) 1–29.

[32] J.D. Boissonat, O.D. Faugeras, and E. Le Bras-Mehlman. Representing stereo data with the Delaunay triangulation. Tech. Rep. 788, INRIA, France, 1988.

[33] A. Bowyer. Computing Dirichlet tessellations. *Computer J.* 24 (1981) 162–166.

[34] K.Q. Brown. Voronoi diagrams from convex hulls. *Inform. Process. Lett.* 9 (1979) 223–228.

[35] E.K. Buratynski. A fully automatic three-dimensional mesh generator for complex geometries. *Int. J. Numer. Meth. Eng.* 30 (1990) 931–952.

[36] A. Bykat. Design of a recursive, shape controlling mesh generator. *Int. J. Numer. Meth. Eng.* 19 (1983) 1375–1390.

[37] G.F. Carey and J.T. Oden. *Finite Elements: Computational Aspects*. Prentice-Hall, 1984.

[38] G.F. Carey, M. Sharma, and K.C. Wang. A class of data structures for 2-d and 3-d adaptive mesh refinement. *Int. J. Numer. Meth. Eng.* 26 (1988) 2607–2622.

[39] J.E. Castillo, ed. *Mathematical Aspects of Numerical Grid Generation. SIAM*, 1991.

[40] J.C. Cavendish. Automatic triangulation of arbitrary planar domains for the finite element method. *Int. J. Numer. Meth. Eng.* 8 (1974) 679–696.

[41] J.C. Cavendish, D.A. Field, and W.H. Frey. An approach to automatic three-dimensional finite element mesh generation. *Int. J. Numer. Meth. Eng.* 21 (1985) 329–347.

[42] B. Chazelle. Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm. *SIAM J. Comput.* 13 (1984) 488–507.

[43] B. Chazelle. Triangulating a simple polygon in linear time. *Disc. and Comp. Geometry* 6 (1991), 485–524.

[44] B. Chazelle, H. Edelsbrunner, L.J. Guibas, J.E. Hershberger, R. Seidel, and M. Sharir. Selecting multiply covered points and reducing the size of Delaunay triangulations. In *Proc. 6th ACM Symp. Comp. Geometry* (1990) 116–127.

[45] B. Chazelle and J. Incerpi. Triangulation and shape complexity. *ACM Trans. on Graphics* 3 (1984) 135–152.

[46] B. Chazelle and L. Palios. Triangulating a nonconvex polytope. *Disc. and Comp. Geometry* 5 (1990), 505–526.

[47] B. Chazelle and N. Shouraboura. Bounds on the size of tetrahedralizations. In *Proc. 10th ACM Symp. Comp. Geometry* (1994) 231–239.

[48] L.P. Chew. There are planar graphs almost as good as the complete graph. *J. Comp. System Science* 39 (1989) 205–219.

[49] L.P. Chew. Constrained Delaunay triangulations. *Algorithmica* 4 (1989) 97–108.

[50] L.P. Chew. Guaranteed-quality triangular meshes. Tech. Rep. TR-89-983, Comp. Science Dept., Cornell University, 1989.

[51] L.P. Chew. Guaranteed-quality mesh generation for curved surfaces. In *Proc. 9th ACM Symp. Comp. Geometry* (1993) 274–280.

[52] N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. *SIAM J. Computing* 14 (1985) 210–223.

[53] K. Clarkson. Approximation algorithms for planar traveling salesman tours and minimum-length triangulations. In *Proc. 2nd ACM-SIAM Symp. Disc. Algorithms* (1991) 17–23.

[54] A.K. Cline and R.J. Renka. A constrained two-dimensional triangulation and the solution of closest node problems in the presence of barriers. *SIAM J. Numer. Analysis* 27 (1990) 1305–1321.

[55] E.F. D'Azevedo and R.B. Simpson. On optimal interpolation triangle incidences. *SIAM J. Sci. Stat. Comput.* 10 (1989) 1063–1075.

[56] L. De Floriani, B. Falcidieno, and C. Pienovi. Delaunay-based representation of surfaces defined over arbitrarily shaped domains. *Computer Vision, Graphics, and Image Processing* 32 (1985) 127–140.

[57] L. De Floriani, B. Falcidieno, C. Pienovi, and G. Nagy. On sorting triangles in a Delaunay tessellation. *Algorithmica* 6 (1991) 522–532.

[58] B. Delaunay. Sur la sphère vide. *Izvestia Akademia Nauk SSSR, VII Seria, Otdelenie Matematicheskii i Estestvennyka Nauk* 7 (1934) 793–800.

[59] T. Dey. Triangulation and CSG representation of polyhedra with arbitrary genus. In *Proc. 7th ACM Symp. Comp. Geometry* (1991) 364–372.

[60] T. Dey, C. Bajaj, and K. Sugihara. On good triangulations in three dimensions. *Int. J. Comp. Geometry and Applications* 2 (1992) 75–95.

[61] T. Dey, M.B. Dillencourt, and S.K. Ghosh. Triangulating with high connectivity. Tech. Report 94-24, Dept. of Inform. and Comp. Science, UC-Irvine, 1994. To appear in *6th Canad. Conf. Comp. Geometry*.

[62] M.T. Dickerson, R.L.S. Drysdale, S.A. McElfresh, E. Welzl. Fast greedy triangulation algorithms. In *Proc. 10th ACM Symp. Comp. Geometry* (1994) 211–220.

[63] M.B. Dillencourt and W.D. Smith. A simple method for resolving degeneracies in Delaunay triangulations. In *Proc. 20th Int. Colloq.— Automata, Languages, and Programming*. (1993) Springer-Verlag LNCS 700, 177–188.

[64] H. Djidjev and A. Lingas. On computing the Voronoi diagram for restricted planar figures. In *Proc. 2nd Worksh. Algorithms and Data Structures*. Springer-Verlag LNCS 519 (1991) 54–64.

[65] D. Dobkin, S. Friedman, and K. Supowit. Delaunay graphs are almost as good as complete graphs. *Disc. and Comp. Geometry* 5 (1990) 399–407.

[66] D. Dobkin, S. Levy, W. Thurston, and A. Wilks. Contour tracing by piecewise linear approximations. *ACM Trans. on Graphics* 9 (1990) 389–423.

[67] R.D. Düppe and H.J. Gottschalk. Automatische Interpolation von Isolinien bei willkürlichen Stützpunkten. *Allgemeine Vermessungsnachrichten* 77 (1970) 423–426.

[68] R.A. Dwyer. Higher-dimensional Voronoi diagrams in linear expected time. *Disc. and Comp. Geometry* 6 (1991) 343–367.

[69] N. Dyn, D. Levin, and S. Rippa. Data dependent triangulations for piecewise linear interpolation. *IMA J. Numer. Analysis* 10 (1990) 137–154.

[70] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, 1987.

[71] H. Edelsbrunner. An acyclicity theorem for cell complexes in d dimensions. *Combinatorica* 18 (1990) 251–260.

[72] H. Edelsbrunner and E.P. Mücke. Simulation of simplicity, a technique to cope with degenerate cases in geometric computations. *ACM Trans. Graphics* 9 (1990) 66–104.

[73] H. Edelsbrunner and N.R. Shah. Incremental topological flipping works for regular triangulations. In *Proc. 8th ACM Symp. Comp. Geometry* (1992) 43–52.

[74] H. Edelsbrunner and N.R. Shah. Triangulating topological spaces. In *Proc. 10th ACM Symp. Comp. Geometry* (1994) 285–292.

[75] H. Edelsbrunner and T.S. Tan. A quadratic time algorithm for the minmax length triangulation. In *Proc. 32nd IEEE Symp. Foundations of Comp. Science* (1991) 414–423.

[76] H. Edelsbrunner and T.S. Tan. An upper bound for conforming Delaunay triangulations. *Disc. and Comp. Geometry* 10 (1993) 197–213.

[77] H. Edelsbrunner, F.P. Preparata, and D.B. West. Tetrahedrizing point sets in three dimensions. *J. Symbolic Comp.* 10 (1990) 335-347.

[78] H. Edelsbrunner, T.S. Tan, and R. Waupotitsch. A polynomial time algorithm for the minmax angle triangulation. *SIAM J. Sci. Stat. Comp.* 13 (1992) 994–1008.

[79] H. ElGindy and D. Avis. A linear algorithm for computing the visibility polygon from a point. *J. Algorithms* 2 (1981) 186–197.

[80] D. Eppstein. The farthest point Delaunay triangulation minimizes angles. *Comp. Geometry Theory and Applications* 1 (1992) 143–148.

[81] D. Eppstein. Approximating the minimum weight triangulation. *Disc. and Comp. Geometry* 11 (1994) 163–191.

[82] D.A. Field. Implementing Watson's algorithm in three dimensions. In *Proc. 2nd ACM Symp. Comp. Geometry* (1986) 246–259.

[83] D.A. Field. Laplacian smoothing and Delaunay triangulations. *Comm. in Applied Numer. Analysis* 4 (1988) 709–712.

[84] D.A. Field. Delaunay criteria for triangulating surfaces. In *Curves and Surfaces in Comp. Vision and Graphics III. Proc. SPIE* 1830 (1992) 237–246.

[85] D.A. Field and W.D. Smith. Graded tetrahedral finite element meshes. *Int. J. Numer. Meth. Eng.* 31 (1991) 413–425.

[86] R.A. Finkel and J.L. Bentley. Quad trees: a data structure for retrieval on composite keys. *Acta Inform.* 4 (1974) 1–9.

[87] S. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica* 2 (1987) 153–174.

[88] S. Fortune. Voronoi diagrams and Delaunay triangulations. In *Computing in Euclidean Geometry*, F.K. Hwang and D.Z. Du, eds., World Scientific, 1992.

[89] W.H. Frey. Selective refinement: a new strategy for automatic node placement in graded triangular meshes. *Int. J. Numer. Meth. Eng.* 24 (1987) 2183–2200.

[90] W.H. Frey and D.A. Field. Mesh relaxation: a new technique for improving triangulations. *Int. J. Numer. Meth. Eng.* 31 (1991) 1121–1133.

[91] I. Fried. Condition of finite element matrices generated from nonuniform meshes. *AIAA J.* 10 (1972) 219–221.

[92] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[93] M.R. Garey, D.S. Johnson, F.P. Preparata, and R.E. Tarjan. Triangulating a simple polygon. *Inform. Process. Lett.* 7 (1978) 175–179.

[94] P.L. George, F. Hecht, and E. Saltel. Constraint of the boundary and automatic mesh generation. In *Proc. 2nd Int. Conf. on Numer. Grid Generation in Comp. Fluid Mechanics* (1988).

[95] P.L. George, F. Hecht, and E. Saltel. Fully automatic mesh generator for 3D domains of any shape. *Impact of Com. in Sci. and Eng.* 2 (1990) 187–218.

[96] P.D. Gilbert. New results in planar triangulations. Report R–850, Univ. Illinois Coordinated Science Lab (1979).

[97] C. Gitlin, J. O'Rourke, and V. Subramanian. On reconstructing polyhedra from parallel slices. Tech. Report 025, Dept. of Computer Science, Smith College, 1993.

[98] C. Gold, T. Charters, and J. Ramsden. Automated contour mapping using triangular element data structures and an interpolant over each irregular triangular domain, In *Proc. Siggraph*, 1977, 170–175.

[99] J.E. Goodman and J. Pach. Cell decomposition of polytopes by bending. *Israel J. of Math.* 64 (1988) 129–138.

[100] J. Gregory. Error bounds for linear interpolation on triangles, *The Mathematics of Finite Elements and Application II*, J. R. Whiteman, ed., Academic Press, London, 1975, 163–170.

[101] L.J. Guibas, D.E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica* 7 (1992) 381–413. Also in *Proc. 17th Int. Colloq.— Automata, Languages, and Programming*. Springer-Verlag LNCS 443 (1990) 414–431.

[102] H. Hagen, M. Müller, and G.M. Nielson. *Focus on Scientific Visualization*, Springer-Verlag, 1993.

[103] B. Hamann. A data reduction scheme for triangulated surface. *CAGD* 11 (1994) 197–214.

[104] D. Hansford. The neutral case for the min-max triangulation. *CAGD* 7 (1990) 431–438.

[105] L.R. Hermann. Laplacian-isoparametric grid generation scheme. *J. of the Eng. Mechanics Div. of the American Soc. of Civil Engineers* 102 (October 1976) 749–756.

[106] J. Hershberger. Finding the visibility graph of a polygon in time proportional to its size. *Algorithmica* 4 (1989) 141–155.

[107] J. Hershberger and J. Snoeyink. Convex polygons made from few lines and convex decompositions of polyhedra. In *Proc. 3rd Scand. Workshop on Algorithm Theory*, Springer-Verlag LNCS 621 (1992) 376–387.

[108] S. Hertel and K. Mehlhorn. Fast triangulation of simple polygons. *4th Conf. Foundations of Computation Theory* . Springer-Verlag LNCS 158 (1983) 207–218.

[109] K. Ho-Le. Finite element mesh generation methods: a review and classification. *Computer-Aided Design* 20 (1988) 27–38.

[110] T.C. Hu and A.C. Tucker. Optimal computer search trees and variable length alphabetic codes. *SIAM J. Applied Math.* 21 (1971) 514–532.

[111] A. Jameson, T.J. Baker, and N.P. Weatherill. Calculation of inviscid transonic flow over a complete aircraft. In *Proc. AIAA 24th Aerospace Sciences Meeting*, Reno, 1986.

[112] P. Jamet. Estimations d'erreur pour des elements finis droits presque degeneres. Tech. Report CRM-447, Centre d'Etudes de Limeil.

[113] K. Jansen. One strike against the min-max degree triangulation problem. *Comp. Geometry Theory and Applications* 3 (1993) 107–120.

[114] B. Joe. Delaunay triangular meshes in convex polygons. *SIAM J. Sci. Stat. Comput.* 7 (1986) 514–539.

[115] B. Joe. Three-dimensional triangulations from local transformations. *SIAM J. Sci. Stat. Comput.* 10 (1989) 718–741.

[116] B. Joe. On the correctness of a linear-time visibility polygon algorithm. *Intern. J. Computer Math.* 32 (1990) 155-172.

[117] B. Joe. Construction of three-dimensional Delaunay triangulations using local transformations. *CAGD* 8 (1991) 123–142.

[118] B. Joe. Delaunay versus max-min solid angle triangulations for three-dimensional mesh generation. *Int. J. Numer. Meth. Eng.* 31 (1991) 987–997.

[119] B. Joe and R.B. Simpson. Triangular meshes for regions of complicated shape. *Int. J. Numer. Meth. Eng.* 23 (1986) 751–778.

[120] B. Joe and R.B. Simpson. Corrections to Lee's visibility polygon algorithm. *BIT* 27 (1987) 458–473.

[121] M.T. Jones and P.E. Plassmann. Parallel algorithms for adaptive mesh refinement. Tech. Report MCS-P421-0394, Mathematics and Computer Science Division, Argonne National Laboratory, 1994.

[122] J.M. Keil and C.A. Gutwin. The Delaunay triangulation closely approximates the complete Euclidean graph. *1st Worksh. on Algorithms and Data Structures*. Springer-Verlag LNCS 382 (1989) 47–56.

[123] J.M. Keil and C.A. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Disc. and Comp. Geometry* 7 (1992) 13–28.

[124] D.G. Kirkpatrick. A note on Delaunay and optimal triangulations. *Inform. Process. Lett.* 10 (1980) 127–128.

[125] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science* 220 (1983) 671–680.

[126] R. Klein and A. Lingas. A linear-time randomized algorithm for the bounded Voronoi diagram of a simple polygon. In *Proc. 9th ACM Symp. on Comp. Geometry* (1993) 124–132.

[127] G.T. Klincsek. Minimal triangulations of polygonal domains. *Ann. Disc. Math.* 9 (1980) 121–123.

[128] C.L. Lawson. Software for C^1 surface interpolation. in J. Rice, ed., *Mathematical Software III*, Academic Press (1977) 161–194.

[129] D.T. Lee. Proximity and reachability in the plane. Tech. Rep. R–831, Coordinated Science Laboratory, Univ. Illinois, 1978.

[130] D.T. Lee. Visibility of a simple polygon. *Comput. Vision, Graphics, and Image Proc.* 22 (1983) 207–221.

[131] D.T. Lee and A. Lin. Generalized Delaunay triangulation for planar graphs. *Disc. and Comp. Geometry* 1 (1986) 201–217.

[132] N.J. Lennes. Theorems on the simple finite polygon and polyhedron. *Am. J. Math.* 33 (1911) 37–62.

[133] C. Levcopoulos. An $\Omega(\sqrt{n})$ lower bound for non-optimality of the greedy triangulation. *Inform. Process. Lett.* 25 (1987) 247–251.

[134] C. Levcopoulos and A. Lingas. On approximation behavior of the greedy triangulation for convex polygons. *Algorithmica* 2 (1987) 175–193.

[135] C. Levcopoulos and A. Lingas. Fast algorithms for greedy triangulation. *BIT* 32 (1992) 280–296. Also in *Proc. 2nd Scand. Worksh. Algorithm Theory*, Springer-Verlag LNCS 447 (1990) 238–250.

[136] D.A. Lindholm. Automatic triangular mesh generation on surfaces of polyhedra. *IEEE Trans. Magnetics* MAG-19 (1983) 2539–2542.

[137] A. Lingas. Advances in minimum weight triangulation. Ph.D. thesis, Linköping Univ., 1983.

[138] A. Lingas. Voronoi diagrams with barriers and their applications. *Inform. Process. Lett.* 32 (1989) 191–198.

[139] R.J. Lipton, D.J. Rose, and R.E. Tarjan. Generalized nested dissection. *SIAM J. Numer. Analysis* 16 (1979) 346–358.

[140] E.L. Lloyd. On triangulations of a set of points in the plane. In *Proc. 18th IEEE Symp. Found. Comp. Sci.* (1977) 228–240.

[141] S.H. Lo. A new mesh generation scheme for arbitrary planar domains. *Int. J. Numer. Meth. Eng.* 21 (1985) 1403–1426.

[142] S.H. Lo. Delaunay triangulation of nonconvex planar domains. *Int. J. Numer. Meth. Eng.* 28 (1989) 2695–2707.

[143] S.H. Lo. Automatic mesh generation and adaptation by using contours. *Int. J. Numer. Meth. Eng.* 31 (1991) 689–707.

[144] R. Löhner. Finite elements in CFD: what lies ahead. *Int. J. Numer. Meth. Eng.* 24 (1987) 1741–1756.

[145] R. Löhner. Generation of three-dimensional unstructured grids by the advancing-front method. In *Proc. AIAA 26th Aerospace Sciences Meeting*, Reno, 1988.

[146] G.K. Manacher and A.L. Zobrist. Neither the greedy nor the Delaunay triangulation approximates the optimum. *Inform. Process. Lett.* 9 (1979) 31–34.

[147] O. Marcotte and S. Suri. Fast matching algorithms for points on a polygon. *SIAM J. Comput.* 20 (1991) 405–422.

[148] J. Matoušek. Reporting points in halfspaces. In *Proc. 32nd IEEE Symp. Foundations of Comp. Science* (1991) 207–215.

[149] A. Maus. Delaunay triangulation and the convex hull of n points in expected linear time. *BIT* 24 (1984) 151–163.

[150] D.J. Mavriplis. Unstructured and adaptive mesh generation for high Reynolds number viscous flows. ICASE Report 91-25, NASA Langley Research Center, 1991.

[151] E. Melissaratos and D. Souvaine. Coping with inconsistencies: a new approach to produce quality triangulations of polygonal domains with holes for the the finite element method. In *Proc. 8th ACM Symp. Comp. Geom.* (1992) 202–211.

[152] R.E. Miles. On the homogeneous planar Poisson point process. *Mathematical Biosciences* 6 (1970) 85–127.

[153] G.L. Miller and W. Thurston. Separators in two and three dimensions. In *Proc. 22nd ACM Symp. Theory of Computing* (1990) 300–309.

[154] G.L. Miller, S.-H. Teng, and S.A. Vavasis. A unified geometric approach to graph separators. In *Proc. 32nd IEEE Symp. on Foundations of Comp. Science* (1991) 538–547.

[155] S.A. Mitchell. Refining a triangulation of a planar straight-line graph to eliminate large angles. In *Proc. 34th IEEE Symp. on Foundations of Comp. Science* (1993) 583–591.

[156] S.A. Mitchell. Finding a covering triangulation whose maximum angle is provably small. In *Proc. 17th Computer Science Conference*, Australian Comp. Science Comm. 16 (1994) 55–64.

[157] S.A. Mitchell and S. Vavasis. Quality mesh generation in three dimensions. In *Proc. 8th ACM Symp. Comp. Geom.* (1992) 212-221.

[158] D. Moore and J. Warren. Adaptive mesh generation I: packing space. Tech. Report TR 90-106, Dept. of Computer Science, Rice University, 1990.

[159] D.M. Mount and A. Saalfeld. Globally-equiangular triangulations of co-circular points in $O(n \log n)$ time. In *Proc. 4th ACM Symp. Comp. Geom.* (1988) 143–152.

[160] J.-D. Müller. Proven angular bounds and stretched triangulations with the frontal Delaunay method. In *Proc. 11th AIAA Comp. Fluid Dyanmics*, Orlando, 1993.

[161] S. Müller, K. Kells, and W. Fichtner. Automatic rectangle-based adaptvie mesh generation without obtuse angles. *IEEE Trans. Computer-Aided Design* 11 (1992) 855–863.

[162] K. Mulmuley. *Computational Geometry: An Introduction through Randomized Algorithms.* Prentice-Hall, Englewood Cliffs, 1994.

[163] O.R. Musin. Delaunay triangulation and optimality. *ARO Worksh. Comp. Geometry*, Raleigh, North Carolina, 1993.

[164] L.R. Nackman and V. Srinivasan. Point placement for Delaunay triangulation of polygonal domains. In *Proc. 3rd Canadian Conf. Comp. Geometry* (1991) 37-40.

[165] S. Olariu, S. Toida, and M. Zubair. On a conjecture by Plaisted and Hong. *J. Alg.* 9 (1988) 597–598.

[166] M.E.G. Ong. Uniform refinement of a tetrahedron. To appear in *SIAM J. Sci. Stat. Comp.*

[167] C.H. Papadimitriou, A.A. Schäffer, and M. Yannakakis. Simple local search problems that are hard to solve. *SIAM J. Comput.* 20 (1991) 56–87.

[168] J. Peraire, J. Peiro, L. Formaggia, K. Morgan, and O.C. Zienkiewicz. Finite element Euler computations in three dimensions. In *Proc. AIAA 26th Aerospace Sciences Meeting*, Reno, 1988.

[169] R. Perucchio, M. Saxena, and A. Kela. Automatic mesh generation from solid models based on recursive spatial decomposition. *Int. J. Numer. Meth. Eng.* 28 (1989) 2469–2502.

[170] D.A. Plaisted and J. Hong. A heuristic triangulation algorithm. *J. Algorithms* 8 (1987) 405–437.

[171] M.-A. Posenau. Approaches to high aspect ratio triangulations. In *Proc. 5th Canadian Conf. Comp. Geometry* (1993) 30–35.

[172] M.-A. Posenau and D.M. Mount. Delaunay triangulation and computational fluid dynamics meshes. In *Proc. 4th Canadian Conf. Comp. Geometry* (1993) 316–321.

[173] F.P. Preparata and S.J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Commun. Assoc. Comput. Mach.* 20 (1977) 87–93.

[174] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction*, Springer-Verlag, 1985.

[175] J.F. Randolph. *Calculus and Analytic Geometry.* Wadsworth, 1961, 373–374.

[176] V.T. Rajan. Optimality of the Delaunay triangulation in R^d. In *Proc. 7th ACM Symp. Comp. Geometry* (1991) 357–363.

[177] S. Rippa. Minimal roughness property of the Delaunay triangulation. *CAGD* 7 (1990) 489–497.

[178] S. Rippa and B. Schiff. Minimum energy triangulations for elliptic problems. *Comp. Meth. in Applied Mech. and Eng.* 84 (1990) 257–274.

[179] M.-C. Rivara. Algorithms for refining triangular grids suitable for adaptive and multigrid techniques. *Int. J. Numer. Meth. Eng.* 20 (1984) 745–756.

[180] M.-C. Rivara. A discussion on the triangulation refinement problem. In *Proc. 5th Canadian Conf. Comp. Geometry* (1993) 42–47.

[181] M.-C. Rivara and C. Levin. A 3-d refinement algorithm suitable for adaptive and multi-grid techniques. *Comm. in Appl. Numer. Meth.* 8 (1992) 281–290.

[182] I.G. Rosenberg and F. Stenger. A lower bound on the angles of triangles constructed by bisecting the longest side. *Math. Comp.* 29 (1975) 390–395.

[183] J. Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. In *Proc. 4th ACM-SIAM Symp. on Disc. Algorithms* (1993) 83–92.

[184] J. Ruppert and R. Seidel. On the difficulty of tetrahedralizing 3-dimensional non-convex polyhedra. *Disc. and Comp. Geometry* 7 (1992) 227–253.

[185] A. Saalfeld. Delaunay edge refinements. In *Proc. 3rd Canadian Conf. Comp. Geometry* (1991) 33-36.

[186] S. Salzberg, A. Delcher, D. Heath, and S. Kasif. Learning with a helpful teacher. In *Proc. 12th Int. Joint Conf. Artificial Intelligence* (1991).

[187] H. Samet. The quadtree and related hierarchical data structures. *Computing Surveys* 16 (1984) 188-260.

[188] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1990.

[189] N. Sapidis and R. Perucchio. Advanced techniques for automatic finite element meshing from solid models. *Computer-Aided Design* 21 (1989) 248–253.

[190] N. Sapidis and R. Perucchio. Delaunay triangulation of arbitrarily shaped planar domains. *CAGD* 8 (1991) 421–438.

[191] L.L. Scarlatos and T. Pavlidis. Optimizing triangulations by curvature equalization. In *Proc. Visualization '92*, IEEE Comput. Soc. Press, 333–339.

[192] D. Schmitt and J.-C. Spehner. On equiangularity of Delaunay diagrams in every dimension. In *Proc. 5th Canadian Conf. Comp. Geometry* (1993) 346–351.

[193] E. Schönhardt. Über die Zerlegung von Dreieckspolyedern in Tetraeder. *Math. Annalen* 98 (1928) 309–312.

[194] W.J. Schroeder and M.S. Shephard. Geometry-based fully automatic mesh generation and the Delaunay triangulation. *Int. J. Numer. Meth. Eng.* 26 (1988) 2503–2515.

[195] L.L. Schumaker. Triangulation methods. In *Topics in Multivariate Approximation*, C.K. Chui, L.L. Schumaker, and F.I. Utreras, eds., Academic Press, 1987, 219–232.

[196] L.L. Schumaker. Reconstructing 3D objects from cross-sections. In *Computation of Curves and Surfaces*, W. Dahmen, M. Gasca, and C.A. Micchelli, eds., Kluwer Academic Publishers, 1989, 275–309.

[197] R. Seidel. A convex hull algorithm optimal for point sets in even dimensions. Report 81-14, Dept. of Computer Science, U. British Columbia, 1981.

[198] R. Seidel. Constrained Delaunay triangulations and Voronoi diagrams with obstacles. In H.S. Poingratz and W. Schinnerl, eds., *1978-1988 Ten Years IIG* (1988) 178–191.

[199] R. Seidel. Backwards analysis of randomized geometric algorithms. ICSI Tech. Report TR-92-014, 1992.

[200] M.I. Shamos and D. Hoey. Closest-point problems. In *Proc. 16th IEEE Symp. Foundations of Comp. Science* (1975) 151–162.

[201] J.G. Shaw. Xerox Webster Research Center, Webster, New York.

[202] M.S. Shephard. Approaches to the automatic generation and control of finite element meshes. *Appl. Mech. Rev.* 41 (1988) 169–185.

[203] M.S. Shephard, F. Guerinoni, J.E. Flaherty, R.A. Ludwig, and P.L. Baehmann. Finite octree mesh generation for automated adaptive three-dimensional flow analysis. In *Proc. of 2nd Int. Conf. on Numer. Grid Generation in Computational Fluid Mechanics* (1988) 709–718.

[204] R. Sibson. Locally equiangular triangulations. *Computer J.* 21 (1978) 243–245.

[205] D. Sleator, R.E. Tarjan, and W. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *J. Amer. Math. Soc.* 1 (1988) 647–682.

[206] W.D. Smith. *Studies in Discrete and Computational Geometry.* PhD thesis, Princeton University, 1988.

[207] W.D. Smith. Implementing the Plaisted-Hong min-length plane triangulation heuristic. Manuscript cited by [53], 1989.

[208] V. Srinivasan, L.R. Nackman, J-M. Tang, and S.N. Meshkat. Automatic mesh generation using the symmetric axis tranformation of polygonal domains. Tech. Rept. RC 16132, Comp. Science, IBM Research Division, Yorktown Heights, NY, 1990.

[209] G. Strang and G.J. Fix. *An Analysis of the Finite Element Method.* Prentice-Hall, 1973.

[210] T.-S. Tan. An optimal bound for conforming quality triangulations. In *Proc. 10th ACM Symp. Comp. Geometry* (1994) 240–249.

[211] R.E. Tarjan and C.J. Van Wyk. An $O(n \log \log n)$-time algorithm for triangulating a simple polygon. *SIAM J. Comput.* 17 (1988) 143–178.

[212] W.C. Thacker. A brief review of techniques for generating irregular computational grids. *Int. J. Numer. Meth. Eng.* 15 (1980) 1335–1341.

[213] J.F. Thompson, ed. *Numerical Grid Generation.* North-Holland, 1982.

[214] J.F. Thompson, Z.U.A. Warsi, C.W. Mastin. *Numerical Grid Generation: Foundations and Applications*. North-Holland, 1985.

[215] G.T. Toussaint, C. Verbrugge, C. Wang, and B. Zhu. Tetrahedralization of simple and non-simple polyhedra. In *Proc. 5th Canadian Conf. Comp. Geometry* (1993) 24–29.

[216] P.J.M. van Laarhoven and E.H.L. Aarts. *Simulated Annealing: Theory and Practice*. Kluwer Academic Publishers, Dordrecht, the Netherlands, 1987.

[217] S.A. Vavasis. Stable finite elements for problems with wild coefficients. Tech. Report TR93-1364, Dept. of Comp. Science, Cornell University, 1993.

[218] C. Wang and L. Schubert. An optimal algorithm for constructing the Delaunay triangulation of a set of line segments. In *Proc. 3rd ACM Symp. Comp. Geometry* (1987) 223–232.

[219] D.F. Watson. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *Computer J.* 24 (1981) 167–171.

[220] D.F. Watson and G.M. Philip. Systematic triangulations. *Computer Vision, Graphics, and Image Processing* 26 (1984) 217–223.

[221] R.D. Williams. Adaptive parallel meshes with complex geometry. Tech. Report CRPC-91-2, Center for Research on Parallel Computation, California Inst. of Technology, 1991.

[222] A.M. Winslow. An irregular triangle mesh generator. Report UCXRL-7880, National Technical Information Service, Springfield, VA, 1964.

[223] F.F. Yao. Speed-up in dynamic programming. *SIAM J. Algebraic and Disc. Methods* 3 (1982) 532–540.

[224] M.A. Yerry and M.S. Shephard. A modified quadtree approach to finite element mesh generation. *IEEE Computer Graphics and Applications* 3 (January/February 1983) 39–46.

[225] M.A. Yerry and M.S. Shephard. Automatic three-dimensional mesh generation by the modified-octree technique. *Int. J. Numer. Meth. Eng.* 20 (1984) 1965–1990.