Local Polyhedra and Geometric Graphs*

Jeff Erickson†

University of Illinois at Urbana-Champaign
jeffe@cs.uiuc.edu
http://www.cs.uiuc.edu/~jeffe

Submitted to *Computational Geometry: Theory and Applications*: August 20, 2003

Abstract

We introduce a new realistic input model for geometric graphs and nonconvex polyhedra. A geometric graph G is *local* if (1) the longest edge at every vertex v is only a constant factor longer than the distance from v to its Euclidean nearest neighbor and (2) the the longest and shortest edges differ in length by at most a polynomial factor. A polyhedron is local if all its faces are simplices and its edges form a local geometric graph. We show that any boolean combination of two local polyhedra in $\mathbb{R}^d$, each with n vertices, can be computed in $O(n \log n)$ time using a standard hierarchy of axis-aligned bounding boxes. Using results of de Berg, we also show that any local polyhedron in $\mathbb{R}^d$ has a binary space partition tree of size $O(n \log^{d-2} n)$ and depth $O(\log n)$; these bounds are tight in the worst case when $d \leq 3$. Finally, we describe efficient algorithms for computing Minkowski sums of local polyhedra in two and three dimensions.

*A preliminary version of this paper was presented at the 19th Annual ACM Symposium on Computational Geometry [19]. See http://www.cs.uiuc.edu/~jeffe/pubs/local.html for the most recent version of this paper.

†Partially supported by a Sloan Foundation Fellowship, NSF CAREER award CCR-0093348, and NSF ITR grants DMR-0121695 and CCR-0219594. Portions of this work were done while the author was visiting Duke University.

1 Introduction

Nonconvex polyhedra are ubiquitous in computer graphics, solid modeling, computer aided design and manufacturing, robotics, and other geometric application areas. Unlike nonconvex polygons or convex objects in space, for which many problems can be solved easily, polyhedra are notoriously difficult to handle efficiently, at least in the worst case.

Collision detection is a textbook example of a problem that is relatively easy for polygons but hard for polyhedra. Although it is quite easy to detect whether two simple polygons intersect in $O(n \log n)$ time, the fastest algorithm for deciding whether two static nonconvex polyhedra intersect, due to Pellegrini, runs in time $O(n^{8/5+\varepsilon})$ [43]. For polyhedral terrains, the time bound can be improved to $O(n^{4/3+\varepsilon})$ [11]. Pellegrini's algorithm was generalized by Schömer and Thiel [46] to find the first collision between two translating polyhedra in time $O(n^{8/5+\varepsilon})$, or between two rotating polyhedra in time $O(n^{5/3+\varepsilon})$. To avoid directly checking all $\Omega(n^2)$ edge pairs, these algorithms employ complex multilevel range-searching data structures that would be difficult (if not impossible) to implement efficiently. Erickson [17] proved that the polyhedron intersection problem is at least as hard (in the the algebraic decision tree model of computation) as Hopcroft's problem: Given a set of points and lines in the plane, does any point lie on a line? The main idea of the reduction is to replace each point and line with an infinitesimally thin spike. In light of this reduction and Erickson's $\Omega(n^{4/3})$ lower bound for Hopcroft's problem [18], an algorithm that detects intersections in $o(n^{4/3})$ worst-case time appears unlikely.[1]

In practice, one of the most popular techniques for intersection detection uses a *hierarchy of bounding volumes*. For a given placement of two disjoint polyhedra, the algorithms refine their hierarchies only to the coarsest level at which the resulting bounding volumes are disjoint. Beginning with Guttmann's introduction of the R-tree in the early 1980s [24], several types of bounding volume hierarchies have been proposed and implemented [2, 22, 23, 26, 31, 33, 35]. Unfortunately, all of these methods—in fact, any related method that uses a hierarchy of convex bounding volumes—can be forced to spend $\Omega(n^2)$ time to determine whether two n-vertex polyhedra intersect. The worst-case example consists of two polyhedra whose edges approximate a twisted grid, similar to a construction of Chazelle [10, 41]. (See Section 4.)

Since worst-case efficient algorithms for detecting intersections seem unlikely, many authors have analyzed heuristics under the assumption that the input objects satisfy certain realistic constraints. For example, Suri and others have shown that for large collections of objects, a standard bounding box heuristic culls out most non-intersecting pairs, provided the objects are *fat* (at least on average) and all about the same size [53, 60]. Agarwal, Guibas, *et al.* [23, 1] and independently Lotan *et al.* [33] recently showed that in a certain hierarchy of bounding spheres for *well-behaved* necklaces of balls, only $O(n^{4/3})$ pairs of balls can intersect. Haverkort *et al.* [25] also recently showed that storing a set of boxes with *low slicing number* in a certain bounding box hierarchy allows box-intersection and approximate range queries to be answered in polylogarithmic time.

This paper introduces a new realistic input model for nonconvex polyhedra, called *locality*. A geometric graph is local if (1) the longest edge at every vertex v is only a constant factor longer than the distance from v to its Euclidean nearest neighbor and (2) the the longest and shortest edges differ in length by at most a polynomial factor. A simplicial polyhedron is local if its edges form a local geometric graph. Surprisingly, our model allows polyhedra that contain sharp spikes

[1] It should be emphasized, however, that Erickson's results [18, 17] are proved in incomparable models of computation and therefore do *not* imply an $\Omega(n^{4/3})$ lower bound for the polyhedron intersection problem. The strongest lower bound known for this problem is only $\Omega(n \log n)$, in the algebraic decision tree and algebraic computation tree models [52, 4].

and folds; however, it forbids many long edges to be packed closely together. See Section 2 for more formal definitions and basic properties.

We consider the complexity of three problems involving local polyhedra: detecting intersections, computing binary space partitions, and constructing Minkowski sums. Restricting the input to local polyhedra significantly improves the worst-case complexity of each problem. In Section 3, we prove that standard bounding volume hierarchy techniques can be used to detect whether two local polyhedra of any fixed dimension intersect in $O(n \log n)$ time. In fact, our algorithm can compute the intersection, union, or any other boolean combination of two local polyhedra in same time bound. In Section 5, we apply a result of de Berg [5] to show that any local polyhedron in $\mathbb{R}^d$ has a binary space partition tree of size $O(n \log^{d-1} n)$ and depth $O(\log n)$; we also show that these bounds are tight when $d \leq 3$. We develop upper and lower bounds on the complexity of Minkowski sums of local polyhedra in low dimensions in Sections 6 and 7.

2 Preliminaries

2.1 Definitions

A *geometric graph* $G = (V, E)$ is an undirected simple graph whose vertices V are distinct points in $\mathbb{R}^d$ and whose edges E are straight line segments. Planar straight-line graphs are examples of geometric graphs in the plane; however, the edges of a geometric graph may cross. The vertices and edges of any (convex or non-convex) polyhedron or piecewise linear complex also form a geometric graph. The *size* $n(G)$ of a geometric graph G is the number of vertices.

Let $N(v)$ denote the set of neighbors of a vertex v in a geometric graph G. We define the *local stretch* of a vertex, denoted $\sigma(v)$, to be the ratio between the length of the longest edge at v and the distance from that vertex v to its nearest Euclidean neighbor (which may not be a neighbor of v in the graph). The local stretch of a graph G, denoted $\sigma(G)$, is the maximum local stretch of its vertices. Similarly, we define the *global stretch* of G, denoted $\Sigma(G)$, as the ratio between the longest and shortest edge lengths in G.

$$\sigma(v) = \frac{\max\limits_{u \in N(v)} |uv|}{\min\limits_{u \in V \setminus \{v\}} |uv|} \qquad \sigma(G) = \max_{v \in V} \sigma(v) \qquad \Sigma(G) = \frac{\max\limits_{uv \in E} |uv|}{\min\limits_{uv \in E} |uv|}$$

Intuitively, $\sigma(G)$ and $\Sigma(G)$ respectively bound the local and global variation in the 'scale' of the graph. We easily observe that $\Sigma(G) \leq \sigma(G)^{n(G)}$. We will write $n = n(G)$, $\sigma = \sigma(G)$, and $\Sigma = \Sigma(G)$ whenever the graph G is clear from context.

We call a geometric graph *local* if its local stretch σ is less than some fixed constant and its global stretch Σ is less than some fixed *polynomial* in the number of vertices. Local graphs are a generalization of the *civilized* graphs considered by Teng [56], for which $\Sigma = O(1)$. The choice of the word "local" is meant to emphasize the much more important role of the local stretch; all of our complexity bounds are polynomial in σ but at most polylogarithmic in Σ (for any fixed dimension).

A *polytope* is the convex hull of a finite number of points. A *polyhedron* is the union of a finite number of polytopes, all of the same dimension. The boundary of any d-dimensional polyhedron P is a $(d-1)$-dimensional manifold, comprised of several connected $(d-1)$-dimensional polyhedra called the *facets* of P. A *face* of P is either P itself or a face of a facet of P; the latter are called *proper faces* of P. In particular, the empty set is the unique (-1)-dimensional face of every polyhedron. A polyhedron is *simplicial* if its facets (and thus its faces) are all simplices. A *boundary triangulation* of a polyhedron decomposes its facets into simplices that meet face to face, with no

additional vertices, Finally, we say that a polyhedron is *local* if it has a boundary triangulation whose edges form a local geometric graph.

Most of our bounds for simplicial polyhedra apply immediately to 'simplex soup': arbitrary collections of simplices in $\mathbb{R}^d$, possibly with shared faces. Thus, for example, we can replace the word 'polyhedron' with 'mesh' or 'piecewise linear complex' or 'immersed manifold' in almost all our results with no other changes. The only exceptions are the bounding volume hierarchy time bounds in Section 3.2, which require that the *spread* of the vertices is bounded by a polynomial, and the bounds for BSP trees in Section 5, which increase by a logarithmic factor if the simplices do not have disjoint interiors.

Finally, all the results in this paper hold under slightly weaker versions of locality. For example, we can allow a constant number of vertices to have non-constant local stretch, or allow polyhedra that can be partitioned into a constant number of local components. Our analysis also applies to small perturbations of local polyhedra, by identifying nearby pairs of vertices that are not graph neighbors. More significantly, let $\sigma_k(v)$ denote the ratio between the length of v's longest edge and the distance from v to its *kth nearest* Euclidean neighbor. Say that a geometric graph or polyhedron is *k-local* if $\Sigma = n^{O(1)}$ and $\sigma_k(v) = O(1)$ for all every vertex v (or all but a constant number). All our bounds for local polyhedra also apply to k-local polyhedra, up to a small polynomial factor in k; we omit the easy details.

2.2 Basic Properties

In a geometric graph with local stretch σ, any edge of length ℓ has two balls of radius ℓ/σ around its endpoints that contain no other vertices of the graph. Several basic properties of local geometric graphs and local polyhedra follow immediately from this simple observation by straightforward packing arguments.

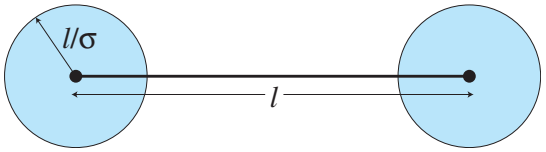

Figure 1. Any edge in a local geometric graph has large empty balls around its endpoints.

Lemma 2.1. *In any geometric graph G in $\mathbb{R}^d$, every vertex has degree at most $O(\sigma^d \log \sigma)$. This bound is tight in the worst case.*

Proof: Let v be an arbitrary vertex of G. Without loss of generality, suppose that the distance from v to its nearest neighbor is exactly 1. We define a sequence of open nested balls $B_0 \subset B_1 \subset B_2 \subset \cdots$, all centered at v, where each ball B_i has radius 2^i. Every neighbor of v lies inside the ball $B_{\lceil \lg \sigma \rceil}$. Let A_i denote the annulus $B_i \setminus B_{i-1}$.

Consider an edge uv of G where $u \in A_i$. The distance from u to v is at least 2^{i-1}, which implies that the ball of radius $2^{i-1}/\sigma$ centered at u does not contain any other vertex of G. The intersection of this empty ball and A_i has volume $\Omega(2^{id}/\sigma^d)$. On the other hand, the volume of A_i is $O(2^{id})$. A straightforward packing argument immediately implies that each annulus A_i contains at most $O(\sigma^d)$ neighbors of v.

For the matching lower bound, let v be an arbitrary point, pack as many points as possible into each annulus A_i, and let G be the graph connecting v to every other point. ■

Lemma 2.2. *Any n-vertex geometric graph G in $\mathbb{R}^d$ has at most $O(\sigma^d n)$ edges. This bound is tight in the worst case.*

Proof: Let u and v be the closest pair of vertices in G, and without loss of generality, assume that $|uv| = 1$. Let B be a ball of radius σ centered at v. Every neighbor of v in the graph lies inside B and is the center of an empty unit ball. A straightforward packing argument now immediately implies that v has $O(\sigma^d)$ neighbors. Let G' be the graph obtained from G by deleting v and all its incident edges. We easily observe that $\sigma(G') \le \sigma(G)$ and $\Sigma(G') \le \Sigma(G)$. By the inductive hypothesis, G' has $(n-1) \cdot O(\sigma^d)$ edges. The trivial base case is a one-vertex graph.

For the matching lower bound, consider the graph G^* whose vertices lie on the integer grid $\{1, 2, \ldots, m\}^d$, where $m = \lfloor n^{1/d} \rfloor$, with an edge between any two vertices whose Euclidean distance is at most σ. This graph clearly has $\Omega(m^d \sigma^d) = \Omega(n\sigma^d)$ edges. ■

Corollary 2.3. *Any n-vertex polyhedron in $\mathbb{R}^d$ has at most $O(\sigma^{d(d-1)} n)$ faces.*

Proof: Without loss of generality, we can assume that the polyhedron is simplicial, since triangulating any non-simplicial polyhedron only increases the number of faces. Let u and v be the closest pair of vertices in the polyhedron. The previous proof implies that v has at most $O(\sigma^d)$ neighbors in the polyhedron's edge graph. Every k-dimensional face that has v as a vertex is the convex hull of v and k of its graph neighbors. It follows that v is a vertex of at most $\binom{O(\sigma^d)}{k} = O(\sigma^{kd})$ k-dimensional faces, and thus $O(\sigma^{d(d-1)})$ faces altogether. By the inductive hypothesis, deleting v and all the faces that contain it leaves a polyhedron with at most $(n-1) \cdot O(\sigma^{d(d-1)})$ faces. ■

For general collections of simplices, this bound is tight in the worst case. The matching lower bound is attained by the simplicial complex whose vertices lie on an integer grid, where a subset of up to $d+1$ points form a simplex if and only if its diameter is at most σ. For polyhedra, however, the bound in Corollary 2.3 is quite conservative. In particular, Euler's formula implies that a three-dimensional polyhedron of arbitrary genus has fewer facets than edges, so Lemma 2.2 implies a tighter upper bound of $O(\sigma^3 n)$ when $d = 3$. Of course, if the polyhedron has genus zero, it has at most $3n - 6$ edges and $2n - 4$ facets, regardless of the value of σ.

Lemma 2.4. *Let G be a geometric graph in $\mathbb{R}^d$, all of whose edges have length at least 1. At most $O(\sigma^d \log \Sigma)$ edges of G intersect any unit-width hypercube.*

Proof: Let $\square$ be a hypercube of unit width, and let $E_\square$ be the set of edges in G that intersect $\square$. We define a sequence of nested hypercubes $\square_1 \subset \square_2 \subset \cdots$, all concentric with $\square$, where each hypercube $\square_i$ has width $2^{i+1} + 1$. We partition the edges in $E_\square$ into into disjoint *length classes* $E_1 \cup E_2 \cup \cdots$, where each set E_i contains all edges in $E_\square$ whose length is between 2^{i-1} and 2^i. Any edge in E_i has at least one endpoint in $\square_i$, and that endpoint is the center of an empty ball of radius at least $2^{i-1}/\sigma$. Thus, by a straightforward packing argument, $O(\sigma^d)$ edges in any length class E_i intersect $\square$. At most $\lg \Sigma$ of the length classes are nonempty. ■

Finally, let $G = (V, E)$ and $G' = (V', E')$ be geometric graphs (or possibly the same graph). We say that two edges $uv \in E$ and $u'v' \in E'$ are *α-close* if their Euclidean distance is less than α times the sum of their lengths:

$$\min_{x \in uv} \min_{x' \in u'v'} |xx'| < \alpha\big(|uv| + |u'v'|\big).$$

Two edges are *close* if they are 1-close.

Lemma 2.5. *Let G be a geometric graph in $\mathbb{R}^d$, all of whose edges have length at least 1. At most $O((1+\alpha)^d \sigma^{2d} \log\sigma \log\Sigma)$ edges of G are α-close to any line segment of length 1.*

Proof: Let s be a line segment of length 1. As in the previous lemma, we partition the edges of G into disjoint length classes $E_1 \cup E_2 \cup \cdots$, where each edge class E_i contains edges whose length is between 2^{i-1} and 2^i. If an edge $e \in E_i$ is α-close to s, then the distance between e and s is at most $\alpha + \alpha 2^i$. In particular, some endpoint of e is within distance $\alpha + \alpha 2^i + 2^{i-1}$ of segment s.

Let C_i denote the Minkowski sum of s with a ball of radius $\alpha + \alpha 2^i + 2^{i-1} = O((1+\alpha)2^i)$. Any edge $e \in E_i$ that is close to s must have at least one endpoint in C_i. We charge the close edge e to this endpoint. Each charged endpoint must lie at the center of a ball of radius $2^{i-1}/\sigma$ that contains no other vertex of G. Since the volume of C_i is $O((1+\alpha)^d 2^{id})$, a standard packing argument implies that at most $O((1+\alpha)^d \sigma^d)$ endpoints of edges in E_i are charged. Lemma 2.1 implies that each endpoint is charged $O(\sigma^d \log\sigma)$ times. Finally, at most $\lceil \lg \Sigma \rceil$ of the length classes are nonempty. ■

Corollary 2.6. *Two geometric graphs G and G' have at most $O(n(1+\alpha)^d \sigma^{2d} \log\sigma \log\Sigma)$ α-close edge pairs, where $\sigma = \max\{\sigma(G), \sigma(G')\}$, $\Sigma = \max\{\Sigma(G), \Sigma(G')\}$, and $n = n(G) + n(G')$.*

In the remainder of the paper, we will assume that σ is a fixed constant and Σ is a fixed polynomial in n, and we will omit explicit dependence on these parameters from our upper bounds.

2.3 Relationship to Other Input Models

Several different models of realistic or well-shaped geometric data have been proposed in the past [6, 7]. Perhaps the most well-known realistic input model is *fatness* [59]. An object X is fat if any ball centered inside X either contains X or has a constant fraction of its volume inside X. Thus, fat objects have no sharp spikes or folds. Local polyhedra, however, can have arbitrarily sharp features, and thus need not be fat; conversely, fat objects can have vertices with edges of arbitrarily different length, and thus need not be local. See Figure 2.

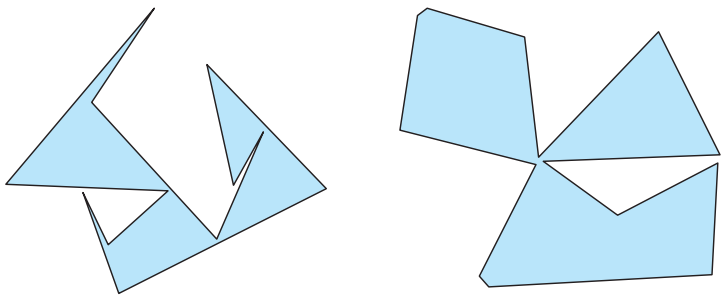

Figure 2. A local nonfat polygon, and a fat nonlocal polygon.

Another realistic input model, introduced by van der Stappen in the context of motion planning [51], is *low density*; see also [39, 48]. A set of objects have density λ if any ball of radius r intersects at most λ objects with diameter r or greater. Most bounds for low density scenes depend linearly on λ. Lemma 2.4 implies that a local polyhedron, viewed as a collection of facets, has density $O(\log n)$; thus, bounds for low-density environments apply to local environments with only a polylogarithmic penalty. On the other hand, low-density objects need not be local.

Two other realistic input models studied by de Berg *et al.* are *uncluttered* scenes and scenes with small *simple cover complexity* [7, 5]. Again, local polyhedra fit these models up to a logarithmic

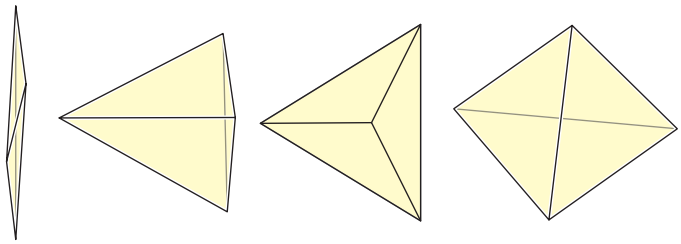

Figure 3. Local but badly-shaped tetrahedra. From left to right: spindle, wedge, cap, sliver.

factor, but neither uncluttered nor easily-covered polyhedra are necessarily local. We discuss the connection between locality and clutter in more detail in Section 5.

We can also compare our model to quality metrics used for simplicial finite-element mesh generation. For example, Miller, Talmor, Teng, and others [32, 34, 49, 54, 56] define a triangulation to be *well-shaped* if the circumradius of each simplex is only a constant factor longer than the shortest edge of that simplex. Well-shaped triangulations have bounded local stretch σ, but they are not necessarily local, since the global stretch Σ could be exponential in the worst case. Conversely, local triangulations need not be well-shaped, even in two dimensions, since they can contain sharp angles; see Figure 3. Talmor [54] proved that a well-shaped triangulation with n vertices, in any fixed dimension, has only $O(n)$ simplices. Corollary 2.3 implies that this linear upper bound actually holds for any triangulation whose local stretch is bounded by a constant.

3 Intersecting Local Polyhedra

3.1 Combinatorial Bounds

Intuitively, one of the reasons that collision detection is difficult for arbitrary nonconvex polyhedra in $\mathbb{R}^3$ is that two polyhedra can intersect, or nearly intersect, in a quadratic number of different locations [10, 41]. For local polyhedra, this quadratic behavior is impossible, even in higher dimensions.

Lemma 3.1. *If two simplices $\triangle$ and $\triangle'$ intersect, then at least one edge of $\triangle$ is close to at least one edge of $\triangle'$.*

Proof: Let uv and $u'v'$ be the longest edges of $\triangle$ and $\triangle'$, respectively, and let x be an arbitrary point in the intersection $\triangle \cap \triangle'$. We easily observe that the distance from any point in $\triangle$ to uv is at most $|uv|$. In particular, the distance from uv to x is at most $|uv|$. Similarly, the distance from $u'v'$ to x is at most $|u'v'|$. Thus, by the triangle inequality, uv and $u'v'$ are close. ■

Corollary 3.2. *Between any two local polyhedra in $\mathbb{R}^d$, each with at most n vertices, there are $O(n \log n)$ pairs of intersecting faces. Thus, any boolean combination of two local polyhedra has complexity $O(n \log n)$.*

Proof: Without loss of generality, we can assume that both polyhedra are simplicial; triangulating the boundary of each polyhedron can only increase the number of intersecting face pairs. Corollary 2.6 implies that there are $O(n \log n)$ close edge pairs. We charge each intersecting pair of faces to some close pair of edges with one edge from each face. By Corollary 2.3, each edge belongs to $O(1)$ faces. It follows that each close edge pair is charged $O(1)$ times. ■

Our algorithmic results are based on the following stronger observation. A *bounding box* of a geometric object is a parallelepiped with orthogonal edges, such that each facet of the box touches the object. Unless specifically stated otherwise, we do *not* assume that bounding boxes are aligned with the coordinate axes.

Lemma 3.3. *If two simplices $\triangle$ and $\triangle'$ have bounding boxes that intersect, then at least one edge of $\triangle$ is close to at least one edge of $\triangle'$.*

Proof: Let $\square$ and $\square'$ denote the intersecting axis-aligned bounding boxes of $\triangle$ and $\triangle'$, respectively. Each facet of $\square$ contains at least one vertex of $\triangle$. Thus, $\triangle$ must have at least one edge e with vertices on the furthest pair of parallel facets of $\square$. This edge is close to every point in $\square$, that is, the distance from e to any point in $\square$ is at most the length of e. Similarly, $\triangle'$ has at least one edge e' that is close to every point in the bounding box $\square'$. The triangle inequality now implies that e and e' are close. ■

Corollary 3.4. *Between any two local, simplicial polyhedra in $\mathbb{R}^d$, each with at most n vertices, there are $O(n \log n)$ pairs of faces with intersecting bounding boxes.*

Proof: Essentially the same as Corollary 3.2. ■

Corollary 3.4 immediately suggests the following algorithm for detecting whether two local, simplicial polyhedra P and Q intersect. Let B_1 and B_2 be the set of *axis-aligned* bounding boxes of facets of P and Q, respectively. Since each polyhedron has $O(n)$ facets, we can clearly calculate B_1 and B_2 in $O(n)$ time. Using multidimensional range trees and segment trees [16, 50], we can find all pairs of intersecting boxes $(\square_1, \square_2) \in B_1 \times B_2$ in time $O(n \log^{d-1} n + k)$, where k is the number of intersecting pairs. Finally, for each pair of intersecting boxes, we can test in $O(1)$ time whether the corresponding pair of facets intersect. Corollary3.4 implies that $k = O(n \log n)$, so the overall running time of the algorithm is $O(n \log^{d-1} n)$. This algorithm can be made extremely practical, at least in low dimensions, by combining it with simple heuristics [61]. In fact, we can actually compute the intersection, or any other boolean combination, of two local polyhedra within the same time bound, by performing an additional constant amount of work for each pair of intersecting facets, plus a constant number of point-in-polyhedron tests to handle the special case where no pair of facets intersects.

Theorem 3.5. *Any boolean combination of two local simplicial polyhedra in $\mathbb{R}^d$, each with n vertices, can be computed in $O(n \log^d n)$ time.*

3.2 Graded Bounding Volume Hierarchies

We can obtain a faster and more general intersection algorithm by constructing a bounding volume hierarchy, called a *graded box-tree*, for each polyhedron. A graded box-tree is (as usual) a rooted tree with constant degree, where the root corresponds to the entire polyhedron, and the leaves correspond to individual facets. Each internal node v stores the axis-aligned bounding box of the facets (leaves) in its subtree. All the bounding volumes at the same level in a graded box-tree have approximately the same diameter. In the interest of simplifying the analysis, we will describe a bounding-volume hierarchy that can easily be improved in practice.

Let P be a local, simplicial polyhedron, or more generally, a local collection of $(d-1)$-simplices in $\mathbb{R}^d$, and let $\square(P)$ be a minimal axis-aligned bounding cube for P. To each internal node v in a graded box-tree, we associate an axis-aligned *control cube* $\boxdot_v$, an axis-aligned *bounding cube* $\square_v$, and a *facet set* P_v as follows.

- The control cubes are defined by a 2^d-tree of $\square(P)$—a quadtree in $\mathbb{R}^2$, an octtree in $\mathbb{R}^3$, and so forth. In particular, $\boxdot_{\text{root}} = \square(P)$. Nodes at each level of the graded box-tree have congruent, interior-disjoint control cubes. We emphasize that the control cubes are not the actual bounding volumes.

- The bounding cube $\square_v$ is the cube concentric with the corresponding control cube $\boxdot_v$, whose width is twice the width of $\boxdot_v$. We will use these cubes as bounding volumes in our intersection algorithm. (In practice, it would be more efficient to use bounding volumes that fit the corresponding facet sets tightly, but this is not required for our analysis.)

- Finally, the facet set P_v contains every facet f of P that is contained in the bounding cube $\square_v$ and whose centroid lies inside the control cube $\boxdot_v$. In particular, the facet set at the root of the tree is the entire polyhedron. At any level of our tree, the bounding cubes are all the same size, and at most 2^d of them contain any point. The facet sets are not explicitly stored in internal nodes of the tree, only their bounding cubes.

A facet $f \in P_v$ is called an *outlier* if it is not a member of P_w for any child w of v, or equivalently, if it is not contained in any child's bounding cube $\square_w$. Each outlier is attached to v as a new child, which becomes a leaf ℓ in our hierarchy. Because P is local, the proof of Lemma 2.4 implies that each internal node has $O(1)$ outlier children, in addition to its 2^d subcube children. We easily verify by induction that P_v is actually the set of facets that appear as leaves in the subtree rooted at v.

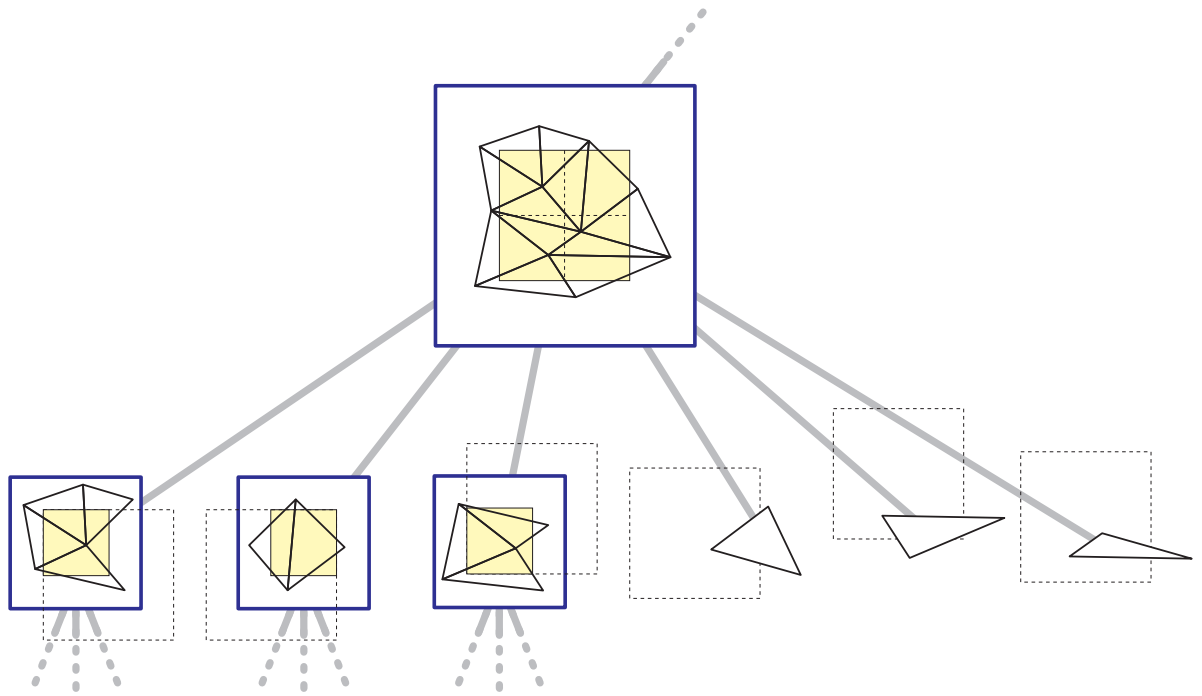

Figure 4. An internal node and its children in a graded box-tree. See the text for details.

A two-dimensional example of our construction is shown in Figure 4. The top of the figure shows a control square (shaded) and the bounding box for a set of eleven triangles. This node has three normal children (northwest, southwest, and southeast) and three outlier children. In each child, the control square (if any) is shaded and the parent's control square is dashed. Because none of the triangles are contained in the southeast bounding square, there is no southeast child.

Each facet is stored in exactly one leaf. If we remove nodes w whose facet sets P_w are empty and compress paths with no branches to single edges, we obtain a tree with constant degree, $O(n)$ leaves (by Corollary 2.3), and therefore $O(n)$ internal nodes. The diameter of any local polyhedron is at most $n\Sigma$ times the length of its shortest edge. Thus, the graded box-tree of any local polyhedron has has depth at most $\lceil \log_2(n\Sigma) \rceil = O(\log n)$. We can easily construct a graded box-tree for any local polyhedron in $O(n \log n)$ time.

Theorem 3.6. *Given graded box-trees for two local simplicial polyhedra P and Q, each with n vertices, we can determine whether P and Q intersect in $O(n \log n)$ time.*

Proof: We use the following standard recursive algorithm, which can be used with any bounding volume hierarchy. If the top-level bounding cubes $\square(P)$ and $\square(Q)$ are disjoint, we can halt immediately. Otherwise we replace one of the two bounding cubes, say $\square(P)$, with its $O(1)$ children, and recursively check for intersections between each of those children and Q. (In practice, it is usually more efficient to expand the larger of the two bounding volumes, but our analysis does not require this choice.) The recursion stops when both polyhedra are reduced to individual facets—leaves in their respective hierarchies—which we can test for intersection in $O(1)$ time.

The running time of this algorithm is clearly dominated by the number of recursive calls, each of which is caused by an intersection between two bounding cubes, or between a bounding cube and a facet. Each bounding cube is the parent of $O(1)$ leaves. Thus, to complete the proof, it suffices to show that between the two graded box-trees, there are only $O(n \log n)$ intersecting pairs of bounding cubes. (This is a conservative upper bound; not every pair of intersecting bounding cubes is tested by the algorithm.)

We claim that each bounding cube in one hierarchy intersects only $O(\log n)$ larger bounding cubes in the other hierarchy. Let v be a node in the graded box-tree of P, and let w be a node in the graded box-tree of Q, such that the bounding cubes $\square_v$ and $\square_w$ intersect. We charge this intersection to either v or w, whichever has the smaller bounding cube (breaking ties arbitrarily). Suppose v has the smaller bounding cube. Let B_w be the set of bounding cubes of nodes in Q's box-tree at the same level as w. All the cubes in B_w are congruent (and larger than $\square_v$), and at most $2^d = O(1)$ of them overlap at any point. It follows that $\square_v$ intersects at most a constant number of cubes in B_w. Thus, v is charged only $O(1)$ times for each level of Q's hierarchy, or $O(\log n)$ times overall. ■

As we noted earlier, it is easy to modify this algorithm to actually compute any boolean combination of the two polyhedra in the same asymptotic running time. Our algorithm allows for the boxes in the two trees to have different orientations or extremely different sizes. Thus, if P and Q undergo any type of rigid motion (or even scaling!), we can still test for intersection in $O(n \log n)$ time without recomputing their hierarchies.

Almost any type of bounding volume can be used in place of axis-aligned cubes in our hierarchy with no loss of efficiency; the only requirement is that the diameter of each bounding volume is at most a constant factor larger than the diameter of the set of objects it encloses. Similarly, the underlying 2^d-tree of control cubes can be replaced by any recursive decomposition into fat regions. Our definition for the facet sets F_v is also quite flexible. For example, we could redefine F_v to be the set of facets that are contained in the bounding cube $\square_v$ and intersect the control cube $\boxdot_v$. This redefinition allows facets to be stored in multiple leaves, but the overall increase in the size of the tree and the running time of the intersection is only a constant factor (exponential in d).

3.3 Related Problems

All the arguments in this section apply directly to local self-intersecting polyhedra, local piecewise-linear complexes, or more generally, any local 'simplex soup' whose edge graph is connected. For disconnected sets of simplices, however, our running-time analysis requires the *spread* of the vertices—the ratio between the largest and smallest pairwise distance [20]—to be bounded by a polynomial in n.

For example, given two local connected planar straight-line graphs, we can overlay them in $O(n \log n)$ time in two different ways. One method is to use the standard sweep-line algorithm, which runs in $O(n \log n + k)$ time, there k is the number of intersecting pairs of segments [8]; Corollary 3.2 implies that $k = O(n \log n)$ if each graph is local. Alternately, we can build a graded (semi-)R-tree for each planar graph and then merge them using the recursive algorithm described in the proof of Theorem 3.6; similar algorithms are described by Brinkoff *et al.* [9] and van Oosterom [38].

Many finite-element applications require multiple meshes, either to partition to the domain for parallel computation, to support the computation of different physical quantities over a single domain, or to track the evolution of a domain over time. In these applications, solutions must be transferred efficiently between overlapping meshes [29]. A key step in the solution transfer process is identifying pairs of overlapping elements. If the meshes are local, we can find all such pairs in near-linear time, using recursive bisection to define a graded bounding volume hierarchy. Similar methods can be use to overlay non-matching meshes of similar (or identical) surfaces [27, 28, 30]. In particular, this technique is efficient for the well-shaped tetrahedral meshes produced by Delaunay refinement algorithms [44, 49], even if they contain slivers.

4 The Harpsicordion

We now show that the results from the previous section are asymptotically optimal for polyhedra in $\mathbb{R}^3$ by constructing a pair of local polyhedra that intersect in $\Omega(n \log n)$ distinct points. Our lower bound construction also implies that our near-linear upper bounds do *not* hold under two obvious relaxations of our input model.

Our bad examples are variants of *Chazelle's polyhedron*, which was originally used to prove quadratic lower bounds for convex decomposition [10, 41]. Our version of Chazelle's construction consists of two polyhedra P and Q, each with total complexity $O(n)$. Each of these two polyhedra contains n edges on the saddle surface $z = xy$. Specifically, the 'vertical' saddle edges of P lie on the lines $z = iy$ for integers $1 \le i \le n$, and the 'horizontal' saddle edges of Q lie on the lines $z = xj$ for integers $1 \le j \le n$. Otherwise, P lies entirely above the saddle and Q lie entirely below. P and Q touch at $\Omega(n^2)$ distinct points of the form (i, j, ij). If we build bounding volume hierarchies for P and Q, the standard intersection algorithm must examine $\Omega(n^2)$ leaf pairs, no matter what shape the bounding volumes have.

A simple modification gives us a pair of *disjoint* polyhedra with similar worst-case behavior. Let $P^+ = P + (0, 0, \varepsilon)$ and $Q^- = Q - (0, 0, \varepsilon)$ be translations of P and Q away from the saddle, where $\varepsilon = O(1/n^2)$ is an arbitrarily small positive real number. Chazelle [10] proved that any convex decomposition of $\mathbb{R}^3 \setminus (P^+ \cup Q^-)$ has $\Omega(n^2)$ cells. The convex hull of any two saddle edges of P^+ intersects every saddle edge of Q^-. It easily follows that for any *convex* bounding volume hierarchies for P^+ and Q^-, the standard intersection algorithm requires $\Omega(n^2)$ time to prove that P^+ and Q^- are disjoint.

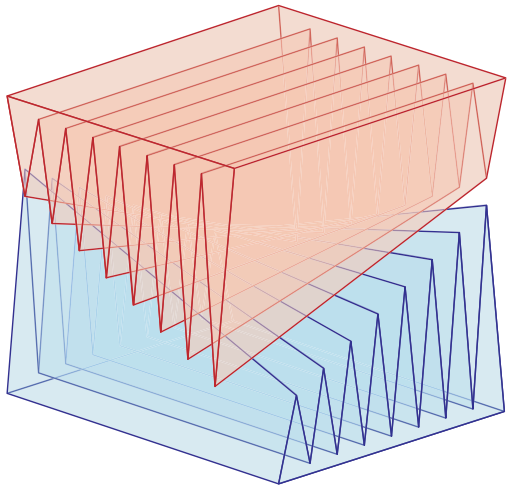

Figure 5. Chazelle's polyhedra.

Theorem 4.1. *For any sufficiently large n, σ, and Σ, there are two n-vertex polyhedra P and Q, where $\sigma(P) = \sigma(Q) = \sigma$ and $\Sigma(P) = \Sigma(Q) = \Sigma$, that intersect in $\Omega(n\sigma \log \Sigma) = \Omega(n \log n)$ distinct points.*

Proof: In Chazelle's original construction, the edges that meet along the saddle are all roughly the same length, but this is clearly not necessary. Instead, we use two parallel sets of line segments of exponentially decaying length, resembling the strings of a harpsichord; specifically, the ith segment in each set has length $(2+\sigma)^{-i}$. See Figure 6. These segments can be placed arbitrarily close together without increasing the local stretch above σ. If each set has m segments, the global stretch of each set is $((2+\sigma)/\sigma)^m$. Thus, for any desired σ and Σ, we can construct two local sets of m segments that meet in an $m \times m$ grid, where

$$m = \frac{\ln \Sigma}{\ln(1 + \frac{2}{\sigma})} > \frac{\sigma \ln \Sigma}{2}.$$

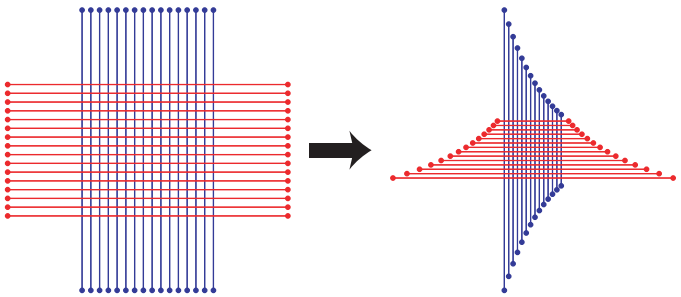

Figure 6. Replacing a regular grid with a harpsichord grid.

We now construct a local $O(m)$-vertex polyhedron with m edges on the saddle surface $z = xy$. Since our polyhedron resembles an accordion, with several saddle edges resembling the strings of a harpsichord, we call it a *harpsicordion*.

The harpsicordion is built by gluing together several copies of two local triangulated annuli $A^\flat$ and $A^\sharp$, shown in Figure 7. The annuli have the same convex outer boundary. The holes are similar triangles; the hole in $A^\flat$ is $(2+\sigma)/\sigma$ times as large as the hole in $A^\sharp$. Moreover, if we scale $A^\sharp$ by a factor of $(2+\sigma)/\sigma$ and align the two holes, the bottom edges of the annuli become collinear.

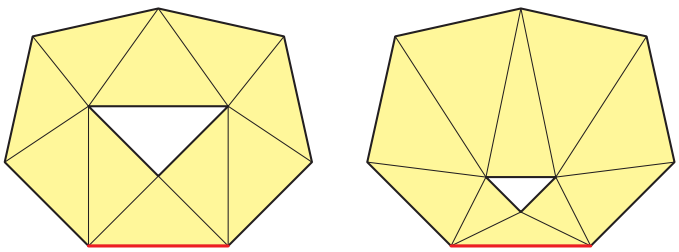

Figure 7. The template annuli $A^\flat$ and $A^\sharp$.

The harpsicordion consist of a sequence of m *folds*. Each fold is built by gluing a copy of $A^\sharp$ to a copy of $A^\flat$ along their common outer boundary, and translating the holes slightly away from the plane through this outer boundary. Successive folds, which differ in size by a factor of $(2+\sigma)/\sigma$, are glued together along a common hole boundary. We fill the holes in the first and last annuli with triangles to close the polyhedron. Finally, we slightly tilt the bottom edges of the folds to lie on the saddle surface. If $A^\sharp$ and $A^\flat$ are constructed carefully, the resulting polyhedron has local stretch σ and global stretch Σ, no matter how thin we make the folds.

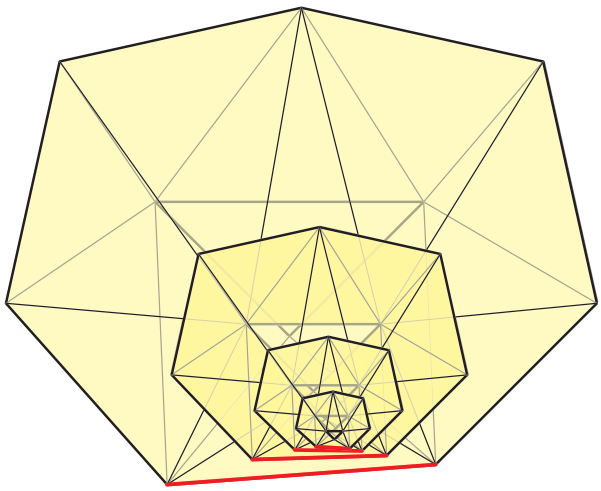

Figure 8. A harpsicordion with four folds.

Finally, polyhedra P and Q each consist of $O(n/m)$ harpsicordia, positioned so that each harpsicordion in P meets one harpsicordion in Q in m^2 distinct points in some saddle. The total number of intersection points is $nm = \Omega(n\sigma \log \Sigma)$. We can make P and Q connected by adding prisms between pairs of harpsicordia. ■

Similar collections of harpsicordia can be used to prove the following lower bounds.

Theorem 4.2. *There are two disjoint, local, simplicial, n-vertex polyhedra in $\mathbb{R}^3$, such that for any hierarchy of convex bounding volumes, the standard intersection algorithm requires $\Omega(n \log n)$ time.*

Theorem 4.3. *There is a local, simplicial, n-vertex polyhedron P in $\mathbb{R}^3$ such that any convex decomposition of $\mathbb{R}^3 \setminus P$ has $\Omega(n \log n)$ cells.*

Our lower bound construction implies that both the local stretch and the global stretch must be bounded in order to obtain our near-linear upper bounds. Specifically, we obtain pairs of n-vertex

polyhedra with $\Omega(n^2)$ intersection points either by setting $\sigma = \Omega(n)$ and $\Sigma = 2$, or by setting $\Sigma = 2^{\Omega(n)}$ and $\sigma = 4$. In fact, for the case $\sigma = \Omega(n)$, we can simplify our lower bound construction by using a single annulus to construct a non-local accordion, as shown in Figure 9.

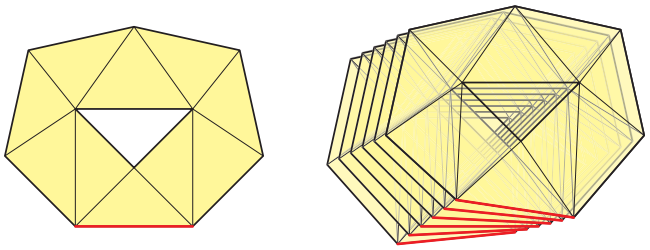

Figure 9. A non-local accordion and its template annulus.

5 Binary Space Partitions

A *binary space partition tree*, or *BSP*, is a binary tree where every internal node v has an associated *cutting hyperplane* h_v in $\mathbb{R}^d$. We can recursively associate an open convex polyhedral *cell* Δ_v with every node v in a BSP as follows. The cell associated with the root is $\mathbb{R}^d$. If u and w are the children of some internal node v, then $\Delta_u = \Delta_v \cap h_v^+$ and $\Delta_w = \Delta_v \cap h_v^-$, where h_v^+ and h_v^- are the open halfspaces bounded by the cutting hyperplane h_v. The (closures of the) leaf cells of a BSP form a convex decomposition of $\mathbb{R}^d$. We say that a BSP $\mathcal{B}$ *respects* a polyhedron P—or less formally, that $\mathcal{B}$ is a BSP for P—if no facet of P intersects the interior of any leaf cell of $\mathcal{B}$. The *size* of a BSP is the number of cuts, or equivalently, one less than the number of leaves.

Fuchs *et al.* [21] introduced BSP trees, following earlier work by Schumacker *et al.* [47], as a tool for computing depth orders for rendering. Since their introduction, BSP trees have been used for many other applications in computer graphics, including shadow generation [12, 13], solid modeling [37, 57], geometric data repair [36], and visibility culling for interactive walkthroughs [55].

As in the case for intersection detection, the worst-case complexity bounds for BSPs of polyhedra in $\mathbb{R}^3$ are quite pessimistic. Chazelle's polyhedron, described in the previous section, gives an $\Omega(n^2)$ lower bound in general [41]. For orthogonal polyhedra in $\mathbb{R}^3$, a construction of Thurston gives a lower bound of $\Omega(n^{3/2})$ [42]. In both cases, matching upper bounds were first proved by Paterson and Yao [41, 42]. In fact, Paterson and Yao's techniques imply that any set of n interior-disjoint simplices in $\mathbb{R}^d$ has a BSP of size $O(n^{d-1})$.

De Berg [5] defined the *clutter factor* κ of a set of objects to be the largest number of objects that intersect a hypercube that does not contain a vertex of the axis-aligned bounding box of any object. A set of objects is *uncluttered* if its clutter factor is smaller than some fixed constant. de Berg also proved that any uncluttered set of n objects has a BSP of size $O(n)$. For scenes with non-constant clutter factor κ, de Berg's construction, combined with the earlier results of Paterson and Yao [41], yields a BSP of size $O(\kappa^{d-1} n)$. By adapting and slightly improving de Berg's results, we show that any local polyhedron has a BSP of near-linear size.

Lemma 5.1. *The facets of any local, simplicial, n-vertex polyhedron in $\mathbb{R}^d$ have clutter factor $O(\log n)$.*

Proof: Let $\triangle$ be a simplex and let $\square$ be a hypercube of width w, such that $\triangle$ and $\square$ intersect, but no vertex of the bounding box of $\triangle$ lies inside $\square$. The longest edge of $\triangle$ must have length greater

than w. Following the proof of Lemma 3.3, we conclude that this edge must be close to some edge of $\square$.

Let P be a local simplicial polyhedron, such that no facet of P has a bounding box vertex inside $\square$. We charge each facet that intersects $\square$ to its longest edge, which is close to some edge of $\square$ by the previous argument. Lemma 2.5 implies that at most $O(\log n)$ edges of P are close to any edge of $\square$, and by Lemma 2.1, each edge belongs to $O(1)$ facets. Finally, $\square$ has $d2^{d-1} = O(1)$ edges. ■

Lemma 5.2. *Let P be a set of n interior-disjoint simplices in $\mathbb{R}^d$ with clutter factor κ. We can construct a BSP of size $O(n\kappa^{d-2})$ for P in time $O(n\kappa^{d-2} + n\log n)$.*

Proof: De Berg [5] describes a two-level BSP of linear complexity for any uncluttered collection of n objects. The first level is an orthogonal BSP of size $O(n)$ that covers the vertices of the bounding boxes of the objects; this bound does not depend at all on the clutter factor. Moreover, each leaf cell in this orthogonal BSP can be covered by $O(1)$ hypercubes that contain no bounding box vertices. This orthogonal BSP can be constructed in $O(n\log n)$ time. We modify this construction slightly, building an orthogonal BSP of size $O(n/\kappa)$, where every leaf cell contains at most κ bounding box vertices.[2]

We build the orthogonal BSP recursively as follows. Let $\square$ be an axis-aligned bounding *hypercube* for P, and let V be the set of bounding box vertices that lie inside $\square$. (Initially, V contains all $2^d n$ bounding box vertices.) If $|V| \le \kappa$, we return the trivial BSP. Otherwise, we split $\square$ into 2^d smaller hypercubes, each with half the width of $\square$. If no subcube contains $|V| - \kappa$ points in V, we cut along the d bisecting planes to construct d complete levels of the BSP, and then continue recursively in each subcube. Otherwise, let $\square'$ be the subcube containing at least $|V| - \kappa$ points in V, let v be the only vertex of $\square$ that is also a vertex of $\square'$, and let $\square''$ be the smallest cube that contains $|V| - \kappa$ points in V and has v as a vertex. We cut along the facets of $\square''$ that are not also facets of $\square$, forming the first d levels of a kd-tree, and continue recursively inside $\square''$. The entire orthogonal BSP can be constructed in time $O(n\log n)$.

There are clearly at most 2^d leaves of depth less than d. If ℓ is a leaf with depth at least d, the dth direct ancestor of ℓ (for example, the grandparent of ℓ if $d = 2$) has a cell containing more than κ bounding box vertices. Thus, the BSP has at most $2^d n/\kappa$ such leaf-ancestors, and therefore at most $4^d n/\kappa + 2^d = O(n/\kappa)$ leaves altogether.

By construction, each leaf cell in our orthogonal BSP contains at most κ bounding box vertices, and each leaf cell can be covered by at most $2^{d-1} = O(1)$ hypercubes. It follows that each leaf cell intersects at most $(2^{d-1} + 1)\kappa = O(\kappa)$ simplices in P. Using an auxiliary data structure, de Berg [5] describes an algorithm to determine the k leaf cells that intersect any constant-complexity query range in $O(k\log n)$ time. Using this query structure, we can determine the simplices in P that intersect each leaf cell in time $O(n\log n)$.

To construct the second level of our BSP, we apply the algorithm of Paterson and Yao [41] to the set of $O(\kappa)$ simplices that intersect any leaf of the orthogonal BSP. Each second-level BSP has size $O(\kappa^{d-1})$ and can be constructed in time $O(\kappa^{d-1})$. (Paterson and Yao claim a running time of $O(k^{d+1})$, where k is the number of input objects, but this can be reduced to $O(k^{d-1})$ using standard randomized techniques [14].) Since there are $O(n/\kappa)$ second-level BSPs, the overall size of the resulting BSP is $O((n/\kappa)\kappa^{d-1}) = O(n\kappa^{d-2})$. ■

The following theorem is now almost immediate.

[2]This modification was suggested by Mark de Berg (personal communication).

Theorem 5.3. *Any local, simplicial, n-vertex polyhedron P in $\mathbb{R}^d$ has a BSP of size $O(n \log^{d-1} n)$ and depth $O(\log n)$, which can be constructed in $O(n \log^{d-1} n)$ time.*

Proof: We construct the two-level BSP described in the proof of the previous lemma. The upper bounds on the size and construction time follow immediately from Lemmas 5.1 and 5.2. It remains only to show that the resulting BSP has logarithmic depth. Without loss of generality, assume that the shortest edge of P has length 1. The diameter of P, and thus the width of the initial control cube, is at most $n\Sigma$, where $\Sigma = \Sigma(P) = n^{O(1)}$ is the global stretch of P.

Consider an internal node v with depth $d \log(\Sigma n)$ in the first-stage orthogonal BSP. The corresponding cell $\square_v$ is a hypercube of width at most 1. If $\square_v$ contains a vertex of the bounding box of any facet $\triangle$ of P, then (following the proof of Lemma 3.3) at least one edge of $\triangle$ is close to at least one edge of $\square_v$. Lemma 2.4 implies that each edge of $\square_v$ is close to $O(\log n)$ edges of P, and and Lemma 2.1 implies that each edge of P lies on $O(1)$ facets. Thus, $\square_v$ contains $O(\log n)$ bounding box vertices. Our construction guarantees that v is the ancestor of $O(1)$ leaves in the orthogonal BSP. We conclude that depth of the orthogonal BSP is at most $d \log(\Sigma n) + O(1) = O(\log n)$.

Let k be the number of facets intersecting some leaf cell in the orthogonal BSP; Paterson and Yao's algorithm [41] constructs a BSP of depth at most k for these k facets. Since our construction guarantees that $k = O(\log n)$, every second-level BSP also has depth $O(\log n)$. ■

Theorem 4.3 implies that the size bound is tight in the worst case when $d \leq 3$. All of these results apply directly to any local collection of *interior-disjoint* simplices. If we allow self-intersections, however, the worst-case complexity increases by a single logarithmic factor.

Lemma 5.4. *Any set of n $(d-1)$-simplices in $\mathbb{R}^d$ with clutter factor κ has a BSP of size $O(n\kappa^{d-1})$, which can be constructed in time $O(n \log n + n\kappa^{d-1})$.*

Proof: Again we start with by constructing an orthogonal BSP of size $O(n/\kappa)$. We can then trivially construct a BSP of size $O(\kappa^d)$ for the $O(\kappa)$ simplices that intersect each leaf cell, in $O(\kappa^d)$ time, by incrementally constructing the arrangement of hyperplanes through those simplices. ■

Corollary 5.5. *Any local set of n $(d-1)$-simplices in $\mathbb{R}^d$ has a BSP of size $O(n \log^{d-1} n)$, which can be constructed in $O(n \log^{d-1} n)$ time.*

The following theorem shows that this bound is tight in the worst case.

Theorem 5.6. *For any d and any sufficiently large n, σ, and Σ, there is a set X of n $(d-1)$-dimensional simplices in $\mathbb{R}^d$, with $\sigma(X) = \sigma$ and $\Sigma(X) = \Sigma$, such that any BSP for X has $\Omega(\sigma^{d-1} n \log^{d-1} \Sigma) = \Omega(n \log^{d-1} n)$ cells.*

Proof: Generalizing the planar harpsichord grid used in the proof of Theorem 4.3, we can construct d sets, each containing $m = (\sigma \ln \Sigma)/2$ parallel $(d-1)$-simplices, that intersect in a regular cubical grid. The complement of the union of these simplices has $\Omega(\sigma^d \log^d \Sigma)$ connected components. Collecting n/md well-separated copies of these sets, we obtain a set of n simplices whose complement has $\Omega(\sigma^{d-1} n \log^{d-1} \Sigma)$ connected components. Any BSP has at least one leaf cell in each component. ■

6 Larger Combinations

Unions and intersections of local polyhedra are not necessarily local. Thus, if we want to efficiently construct a boolean combination of more than two local polyhedra, we cannot combine the objects in pairs; we must combine everything at once.

Theorem 6.1. *Any boolean combination of r local, simplicial polyhedra in $\mathbb{R}^3$, each with n vertices, has complexity $O(r^3 n \log^2 n)$.*

Proof: Let $P_1, P_2, \ldots, P_r$ be local polyhedra. We will show that the *arrangement* of these polyhedra has total complexity $O(r^3 n \log^2 n)$; any boolean combination of the polyhedra consists of a subset of the faces of the arrangement, possibly with some faces merged together. We can analyze the complexity of this arrangement using an argument similar to Paterson and Yao's analysis of their three-dimensional binary space partition trees [41].

First we count the vertices of the arrangement. Each arrangement vertex is either a vertex of a polyhedron P_i, the intersection of the edge of some polyhedron P_i with a facet of another polyhedron P_j, or the mutual intersection of three facets of three different polyhedra. There are clearly rn vertices of the first type, and Corollary 3.2 implies there are $O(r^2 n \log n)$ vertices of the second type. We charge each triple intersection point to the triangle whose longest edge is shortest among the three intersectors. By Lemma 3.1, the longest edge of the charged triangle is close to the longest edges of the other two triangles.

Consider three polyhedra P_i, P_j, P_k. Each polyhedron has $O(n)$ edges, and each edge lies on $O(1)$ facets. Each edge of P_i is close to $O(\log n)$ longer edges in P_j and $O(\log n)$ longer edges in P_j. Thus, each edge of P_i is charged $O(\log^2 n)$ times by triple intersections with P_j and P_k. Since there are $O(r^2)$ choices for j and k, each edge of P_i is charged $O(r^2 \log^2 n)$ times, so P_i is charged $O(r^2 n \log^2 n)$ times altogether. We conclude that the total number of triple intersections, and thus the total number of vertices, is $O(r^3 n \log^2 n)$.

The edges of the arrangement can be grouped into collinear *super-edges*, where each super-edge is either an edge of some polyhedron P_i, or the intersection of two facets of different polyhedra. Corollary 3.2 implies that there are $O(k^2 n \log n)$ super-edges. To count the actual arrangement edges, we charge each edge to one of its endpoints. Each triple intersection point is charged at most six times; all the remaining charges go to endpoints of super-edges. Thus, the total number of edges is $O(r^3 n \log^2 n)$.

Finally, each facet of each polyhedron P_i is decomposed into several arrangement facets, which we will call *fragments*, by the other polyhedra. Let F be a facet of P_i. Euler's formula implies that the number of fragments of F is less than $2v_F - 4$, where v_F is the number of arrangement vertices on F. If F charges two of its fragments to every vertex on F except its original vertices from P_i, only 2 fragments are uncharged. Except for polyhedra vertices, each vertex in the arrangement lies on exactly three polyhedron facets and thus is charged at most six times. Therefore, the total number of fragments is twice the number of facets plus six times the number of vertices, which by our earlier analysis is $O(r^3 n \log^2 n)$. ■

Theorem 6.2. *We can compute any boolean function of any r local, simplicial polyhedra in $\mathbb{R}^3$, each with n vertices, in time $O(r^3 n \log^2 n)$.*

7 Minkowski Sums of Local Polygons and Polyhedra

Finally, we consider the complexity of Minkowski sums of local polyhedra in two and three dimensions. In the worst case, the Minkowski sum of two n-gons has complexity $\Theta(n^4)$, and the Minkowski sum of two polyhedra in $\mathbb{R}^3$, each with n faces, has complexity $\Theta(n^6)$.

Theorem 7.1. *The Minkowski sum of any two local n-gons in the plane has complexity $O(n^3 \log n)$ and $\Omega(n^3)$ in the worst case.*

Proof: Let P and Q be two local n-gons with vertices labeled $p_1, \ldots, p_n$ and $q_1, \ldots, q_n$, respectively. The Minkowski sum $P + Q$ is the union of cells in the arrangement of the $r = 2n$ polygons $p_i + Q$ and $P + q_j$. Simplifying the proof of Theorem 6.1 to the two-dimensional case, we can prove that the arrangement of r local n-gons has complexity $O(r^2 n \log n)$.

For the lower bound, we construct two local $O(n)$-gons P and Q, with $\sigma \approx 2$ and $\Sigma = \Theta(n)$, whose Minkowski sum has complexity $\Omega(n^3)$. Each polygon consists of a comb with n widely-spaced extremely thin spikes, each of length $O(n)$, and a vertical zigzag of $2n$ edges, each of length 1. To maintain locality, a series of $O(\log n)$ edges interpolates between the large and small features of each polygon. Figure 10(a) shows the comb polygon, plus a simplified geometric graph with the same salient features.

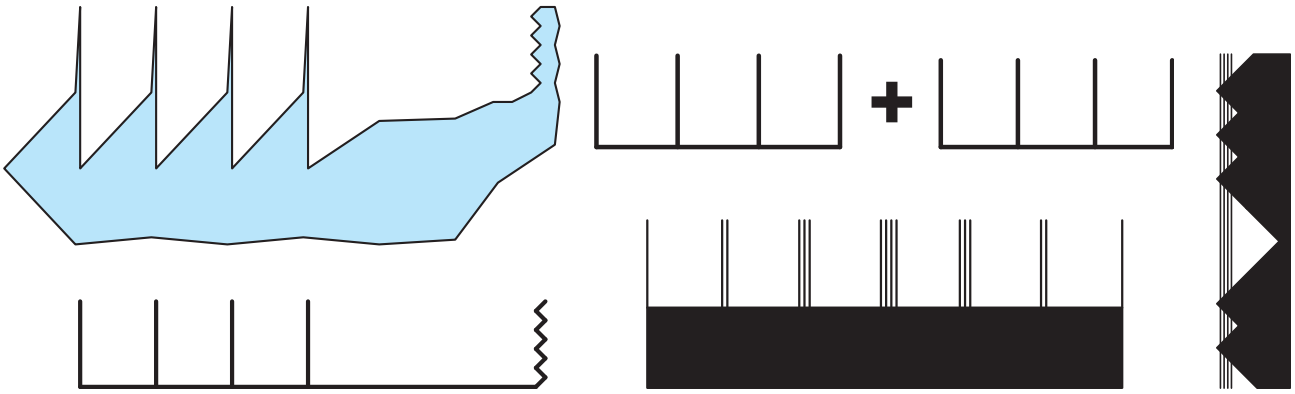

Figure 10. (a) One comb polygon and its salient features. (b) The Minkowski sum of two combs has several bundles of spikes. (c) The Minkowski sum of two zigzags cutting through a bundle.

The distance between the spikes in P is very slightly larger than in Q, so that the Minkowski sum of the two cones has n^2 spikes, grouped into $2n - 1$ bundles; see Figure 10(b). Similarly, the Minkowski sum of the two zigzags has n^2 teeth. The zigzags of P and Q are positioned so that their Minkowski sum cuts through the middle bundle of n spikes. Each of the n^2 teeth cuts all the way through this bundle, intersecting all n spikes; see Figure 10(c). Thus, $P + Q$ has $\Omega(n^3)$ vertices. ■

Theorem 7.2. *The Minkowski sum of any two local, simplicial polyhedra in $\mathbb{R}^3$, each with n vertices, has complexity $O(n^4 \log^2 n)$ and $\Omega(n^4)$ in the worst case.*

Proof: Let P and Q be two local n-vertex polyhedra. The Minkowski sum $P + Q$ is the union of cells in the arrangement of the $r = 2n$ polyhedra $p_i + Q$ and $P + q_j$. The proof of Theorem 6.1 implies that this arrangement has complexity $O(r^3 n \log^2 n) = O(n^4 \log^2 n)$.

The lower bound construction is a generalization of the two-dimensional case. Each polyhedron approximates a piecewise linear complex consisting of two sets of square 'shelves', one set parallel

to the xz plane and one set parallel to the yz plane, along with a vertical 'staircase' with edges parallel to the plane $x = y$. The Minkowski sum of the two sets of shelves contains a tight $n \times n$ grid of planes, all parallel to the z axis. The Minkowski sum of the two staircases cuts through this grid n^2 times to create $\Omega(n^4)$ vertices. We can construct a local polyhedron with the same salient features as this complex by gluing together several annuli, similarly to the harpsicordion construction in Section 4. See Figure 11. We omit the straightforward but tedious details. ■

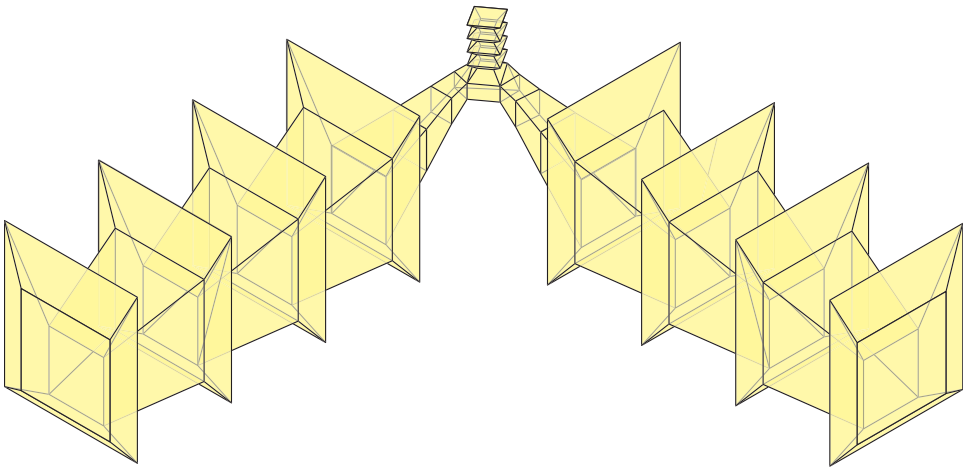

Figure 11. The shelves-and-staircases polyhedron; compare with Figure 10(a).

8 Discussion and Open Problems

We have introduced a new realistic input model, called *locality*, for nonconvex simplicial polyhedra and other sets of simplices. Unlike many other realistic input models, our model permits objects with arbitrarily sharp features.

Unlike most previously studied models, locality is not a function of the *shape* of geometric objects, but rather a function of their *representation*. Any polyhedron can be "localized" by carefully decomposing each face with a graded mesh. For example, the $n \times n \times 1$ rectangular box has a local boundary mesh with $O(\log n)$ vertices; see Figure 12. On the other hand, our results imply that any local boundary mesh of Chazelle's polyhedron, or of a regular convex n-gonal cylinder with constant height and radius, must have $\Omega(n^2/\log n)$ vertices.

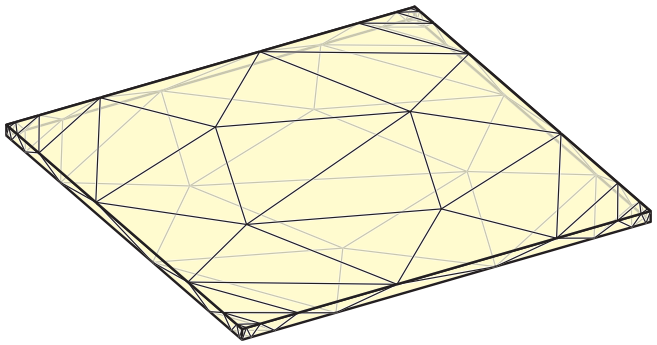

Figure 12. A local triangulation of an $n \times n \times 1$ 'pizza box' with $O(\log n)$ facets.

In order to gauge the realism of our 'realistic' input model, we measured the local and global stretches of several geometric models. While the global stretch Σ was less than the number of vertices for every model we tested, the local stretch σ was not as well-behaved.

Not surprisingly, for most of the CAD models we tested, the local stretch σ was extremely large, usually because the model contained nearly-coincident vertices from different components of the model, or several long skinny rectangles modeling a portion of a cylinder or cone. Our analysis can be modified to work for models with nearly-coincident vertices, but for objects with large cylindrical or conical regions, this bad behavior is simply unavoidable without introducing a large number of additional vertices.

We also tested standard models that were automatically reconstructed from scattered surface points [15, 58]. The maximum local stretch σ was often extremely large, especially for models like the happy Buddha and the dragon that were reconstructed from several two-dimensional range images. Even for these models, however, the kth-order stretch σ_k decayed quickly as a function of k; even for the worst model we tried, the 10th-order stretch was a small constant. See Table 1.

model	n	Σ	σ	σ_2	σ_3	σ_4	σ_6	σ_8	σ_{10}
teapot	1177	59.00	17.675	12.325	8.215	6.519	5.775	4.916	4.008
bunny	35947	27.59	302.626	8.818	4.720	4.096	3.400	2.718	2.045
armadillo	172974	2683.81	1705.08	1160.72	921.857	3.240	2.475	2.075	1.726
hand	327323	129.31	122.274	100.082	48.427	7.358	2.098	1.680	1.500
dragon	437645	315093.	80172.9	35132.0	991.039	294.512	68.046	22.458	21.611
buddha	543652	261019.	42065.3	26466.3	4105.940	608.669	14.324	10.277	8.960
blade	882954	214.27	210.559	197.254	80.889	27.025	3.931	2.936	2.219

Table 1. Global and local stretch of some large geometric models

We also examined the distribution of local stretch over the vertices of each model. Rather surprisingly, this distribution often had an extremely heavy tail, suggesting that the *average* local stretch might be a more realistic measure than the maximum. (Although most realistic input models restrict the worst-case behavior of some parameter, a few results are known for sets of objects that are fat on average [40, 60].) Higher-order local stretch, on the other hand, was more tightly concentrated around the mean. First- and tenth-order local stretch distributions for the dragon model are displayed in Figure 13; the other models had similar (but tighter) distributions.

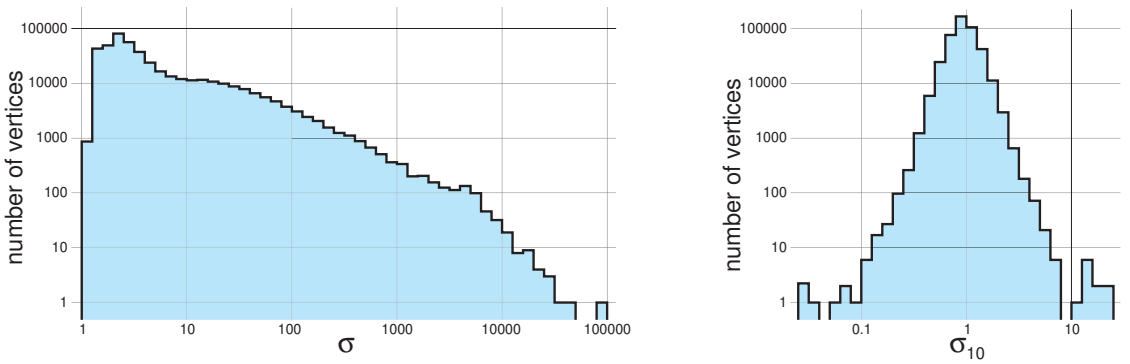

Figure 13. Distributions of σ and σ_{10} for the dragon model.

We derived nearly-matching upper and lower bounds for the Minkowski sum of two local polyhedra in two or three dimensions. The closeness of these bounds is somewhat misleading, however, since they have very different dependencies on the global stretch Σ; the two-dimensional bounds

are more accurately written as $O(n^3 \log \Sigma)$ and $\Omega(n^2\Sigma)$. How complex is the Minkowski sum of two *civilized* polyhedra, where $\Sigma = O(1)$? Our input model allows vertices to be arbitrarily close to higher-dimensional facets. Can we obtain better bounds by replacing the nearest neighbor distance in the definition of σ with, say, the local feature size at each vertex?

Intersection, convex decomposition, and Minkowski sum are only three of many problems involving nonconvex polyhedra that are that are difficult in the worst case, but may be easier for local or other 'realistic' polyhedra. For example, can local polyhedra be triangulated using only a near-linear number of simplices? How hard is constructing the triangulation of a local polyhedron with the minimum number of Steiner points [45] or tetrahedra [3]? How complex is the medial axis of a local polyhedron in the worst case?

Acknowledgments. Thanks to Pankaj Agarwal, Mark de Berg, Leo Guibas, Sariel Har-Peled, and Mark van Kreveld for helpful discussions, questions, and observations. I am especially grateful to an anonymous SoCG reviewer for pointing out de Berg's BSP results [5], Mark de Berg for outlining the improvement in the results of Section 5, and Sariel Har-Peled for writing code to measure σ and Σ in real world models. Thanks also to the graphics groups at Stanford [15] and Georgia Tech [58] for making their models publicly available.

References

[1] P. K. Agarwal, L. Guibas, A. Nguyen, D. Russel, and L. Zhang. Collision detection for deforming necklaces. To appear in *Comput. Geom. Theory Appl.*, 2003.

[2] G. Barequet, B. Chazelle, L. Guibas, J. Mitchell, and A. Tal. BOXTREE: A hierarchical representation for surfaces in 3D. *Comput. Graph. Forum* 15(3):C387–C396, C484, 1996. Proc. Eurographics'96.

[3] A. Below, J. A. de Loera, and J. Richter-Gebert. Finding minimal triangulations of convex 3-polytopes is NP-hard. *Proc. 11th Annu. ACM-SIAM Sympos. Discrete Algorithms*, 65–66, 2000.

[4] M. Ben-Or. Lower bounds for algebraic computation trees. *Proc. 15th Annu. ACM Sympos. Theory Comput.*, 80–86, 1983.

[5] M. de Berg. Linear size binary space partitions for uncluttered scenes. *Algorithmica* 28:353–366, 2000.

[6] M. de Berg, M. Katz, M. Overmars, A. F. van der Stappen, and J. Vleugels. Models and motion planning. *Proc. 6th Scand. Workshop Algorithm Theory*, 83–94, 1998. Lecture Notes Comput. Sci. 1432, Springer-Verlag.

[7] M. de Berg, M. J. Katz, A. F. van der Stappen, and J. Vleugels. Realistic input models for geometric algorithms. *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, 294–303, 1997.

[8] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[9] T. Brinkhoff, H.-P. Kriegel, and B. Seeger. Efficient processing of spatial joins using R-trees. *Proc. ACM SIGMOD Conf. on Management of Data*, 237–246, 1993.

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

[11] B. Chazelle, H. Edelsbrunner, L. J. Guibas, and M. Sharir. Algorithms for bichromatic line segment problems and polyhedral terrains. *Algorithmica* 11:116–132, 1994.

[12] N. Chin and S. Feiner. Near real-time shadow generation using BSP trees. *Proc. SIGGRAPH '89*, 99–106, 1989.

[13] N. Chin and S. Feiner. Fast object-precision shadow generation for areal light sources using BSP trees. *Comput. Graph.* 25:21–30, 1992. Proc. 1992 Sympos. Interactive 3D Graphics.

[14] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.* 4:387–421, 1989.

[15] B. Curless. The Stanford 3D Scanning Repository, June 2003. ⟨http://graphics.stanford.edu/data/3Dscanrep/⟩.

[16] H. Edelsbrunner. A new approach to rectangle intersections, Part I. *Internat. J. Comput. Math.* 13:209–219, 1983.

[17] J. Erickson. On the relative complexities of some geometric problems. *Proc. 7th Canad. Conf. Comput. Geom.*, 85–90, 1995. ⟨http://www.uiuc.edu/~jeffe/pubs/relative.html⟩.

[18] J. Erickson. New lower bounds for Hopcroft's problem. *Discrete Comput. Geom.* 16:389–418, 1996.

[19] J. Erickson. Local polyhedra and geometric graphs. *Proc. 19th Annu. ACM Sympos. Comput. Geom.*, 171–180, 2003.

[20] J. Erickson. Nice point sets can have nasty Delaunay triangulations. *Discrete Comput. Geom.* 30(1):109–132, 2003.

[21] H. Fuchs, Z. M. Kedem, and B. Naylor. On visible surface generation by a priori tree structures. *Comput. Graph.* 14(3):124–133, 1980. Proc. SIGGRAPH '80.

[22] S. Gottschalk, M. C. Lin, and D. Manocha. OBB-tree: A hierarchical structure for rapid interference detection. *Proc. SIGGRAPH '96*, 171–180, 1996.

[23] L. Guibas, A. Nguyen, D. Russel, and L. Zhang. Collision detection for deforming necklaces. *Proc. 18th Annu. ACM Sympos. Comput. Geom.*, 33–42, 2002.

[24] A. Guttman. R-trees: A dynamic index structure for spatial searching. *Proc. ACM SIGMOD Conf. Principles Database Systems*, 47–57, 1984.

[25] H. H. Haverkort, M. de Berg, and J. Gudmundsson. Box-trees for collision checking in industrial applications. *Proc. 18th Annu. ACM Sympos. Comput. Geom.*, 53–62, 2002.

[26] P. M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Trans. Graph.* 15(3):179–210, 1996.

[27] X. Jiao, H. Edelsbrunner, and M. T. Heath. Mesh association: Formulation and algorithms. *Proc. 8th International Meshing Roundtable*, 75–82, 1999. ⟨http://www.andrew.cmu.edu/user/sowen/abstracts/Ji681.html⟩.

[28] X. Jiao and M. T. Heath. Efficient and robust algorithms for overlaying surface meshes. *Proc. 10th International Meshing Roundtable*, 281–292, 2001. ⟨http://www.andrew.cmu.edu/user/sowen/abstracts/Ji834.html⟩.

[29] X. Jiao and M. T. Heath. Common-refinement based data transfer between nonmatching meshes in multiphysics simulations. Preprint, April 2003. ⟨http://www.cse.uiuc.edu/~jiao/papers/datatransfer.pdf⟩.

[30] X. Jiao and M. T. Heath. Overlaying surface meshes, part I: Algorithms. Preprint, February 2003. ⟨http://www.cse.uiuc.edu/~jiao/papers/overlay_p1.pdf⟩.

[31] J. Klosowski, M. Held, J. S. B. Mitchell, K. Zikan, and H. Sowizral. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Trans. Visualizat. Comput. Graph.* 4(1):21–36, 1998.

[32] X.-Y. Li and S.-H. Teng. Generating well-shaped Delaunay meshes in 3D. *Proc. 12th Annu. ACM-SIAM Sympos. Discrete Algorithms*, 28–37, 2001.

[33] I. Lotan, F. Schwarzer, D. Halperin, and J.-C. Latombe. Efficient maintenance and self-collision testing for kinematic chains. *Proc. 18th Annu. ACM Sympos. Comput. Geom.*, 43–52, 2002.

[34] G. L. Miller, D. Talmor, S.-H. Teng, and N. Walkington. A Delaunay based numerical method for three dimensions: generation, formulation, and partition. *Proc. 27th Annu. ACM Sympos. Theory Comput.*, 683–692, 1995.

[35] B. Mirtich and J. Canny. Impulse-based dynamic simulation. *The Algorithmic Foundations of Robotics*, 1995. A. K. Peters.

[36] T. M. Murali and T. A. Funkhouser. Consistent solid and boundary representations from arbitrary polygonal data. *Proc. 1997 Sympos. Interactive 3D Graphics*, 1997.

[37] B. Naylor, J. A. Amanatides, and W. Thibault. Merging BSP trees yields polyhedral set operations. *Comput. Graph.* 24(4):115–124, 1990. Proc. SIGGRAPH '90.

[38] P. van Oosterom. An R-tree based map-overlay algorithm. *Proc. EGIS '94*, 318–327, 1994.

[39] M. H. Overmars and A. F. van der Stappen. Range searching and point location among fat objects. *J. Algorithms* 21:629–656, 1996.

[40] J. Pach and G. Tardos. On the boundary complexity of the union of fat triangles. *Proc. 41th Annu. IEEE Sympos. Found. Comput. Sci.*, 423–431, 2000.

[41] M. S. Paterson and F. F. Yao. Efficient binary space partitions for hidden-surface removal and solid modeling. *Discrete Comput. Geom.* 5:485–503, 1990.

[42] M. S. Paterson and F. F. Yao. Optimal binary space partitions for orthogonal objects. *J. Algorithms* 13:99–113, 1992.

[43] M. Pellegrini. Ray shooting on triangles in 3-space. *Algorithmica* 9:471–494, 1993.

[44] J. Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms* 18:548–585, 1995.

[45] J. Ruppert and R. Seidel. On the difficulty of triangulating three-dimensional non-convex polyhedra. *Discrete Comput. Geom.* 7:227–253, 1992.

[46] E. Schömer and C. Thiel. Efficient collision detection for moving polyhedra. *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, 51–60, 1995.

[47] R. A. Schumacker, R. Brand, M. Gilliland, and W. Sharp. Study for applying computer-generated images to visual simulation. Tech. Rep. AFHRL–TR–69–14, U.S. Air Force Human Resources Laboratory, 1969.

[48] O. Schwarzkopf and J. Vleugels. Range searching in low-density environments. *Inform. Process. Lett.* 60:121–127, 1996.

[49] J. R. Shewchuk. Tetrahedral mesh generation by Delaunay refinement. *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, 86–95, 1998.

[50] H.-W. Six and D. Wood. Counting and reporting intersections of D-ranges. *IEEE Trans. Comput.* C-31:181–187, 1982.

[51] A. F. van der Stappen. *Motion Planning amidst Fat Obstacles.* Ph.D. dissertation, Dept. Comput. Sci., Utrecht Univ., Utrecht, Netherlands, 1994.

[52] J. M. Steele and A. C. Yao. Lower bounds for algebraic decision trees. *J. Algorithms* 3:1–8, 1982.

[53] S. Suri, P. M. Hubbard, and J. F. Hughes. Collision detection in aspect and scale bounded polyhedra. *Proc. 9th ACM-SIAM Sympos. Discrete Algorithms*, 127–136, 1998.

[54] D. Talmor. *Well-Spaced Points and Numerical Methods.* Ph.D. thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, August 1997. Technical report CMU-CS-97-164. ⟨http://reports-archive.adm.cs.cmu.edu/anon/1997/abstracts/97-164.html⟩.

[55] S. J. Teller and C. H. Séquin. Visibility preprocessing for interactive walkthroughs. *Comput. Graph.* 25(4):61–69, 1991. Proc. SIGGRAPH '91.

[56] S.-H. Teng. *Points, Spheres, and Separators: A Unified Geometric Approach to Graph Partitioning.* Ph.D. thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1992. Technical report CMU-CS-91-184.

[57] W. C. Thibault and B. F. Naylor. Set operations on polyhedra using binary space partitioning trees. *Comput. Graph.* 21(4):153–162, 1987. Proc. SIGGRAPH '87.

[58] G. Turk and B. Mullins. Large Geometric Models Archive, 2003. ⟨http://www.cc.gatech.edu/projects/large_models/⟩.

[59] J. Vleugels. *On Fatness and Fitness — Realistic Input Models for Geometric Algorithms.* Ph.D. thesis, Dept. Comput. Sci., Univ. Utrecht, Utrecht, The Netherlands, 1997.

[60] Y. Zhou and S. Suri. Analysis of a bounding box heuristic for object intersection. *Proc. 10th Annu. ACM-SIAM Sympos. Discrete Algorithms*, 830–839, 1999.

[61] A. Zomorodian and H. Edelsbrunner. Fast software for box intersection. *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, 129–138, 2000.