Discrete Comput Geom 14:429–444 (1995)

Discrete & Computational
Geometry

Bounds on the Size of Tetrahedralizations*

B. Chazelle[1] and N. Shouraboura[2]

[1] Department of Computer Science, Princeton University,
Princeton, NJ 08544, USA
chazelle@cs.princeton.edu

[2] Program in Applied and Computer Mathematics, Princeton University,
Princeton, NJ 08544, USA
nadia@cs.princeton.edu

Abstract. This paper gives lower and upper bounds on the complexity of triangulating the region between polyhedra. Particular attention is given to the case of convex polyhedra and terrains.

1. Introduction

We extend the work of Bern [5] on triangulating the region between three-dimensional polyhedra. Here is a summary of our results:

- We give a linear algorithm for triangulating the region between two convex polyhedra. No assumptions are made on the polyhedra: they can be disjoint, nested, or overlapping. The algorithm produces $O(n)$ tetrahedra, where n is the input size. This optimal bound improves on an $O(n \log n)$-size construction by Bern [5].
- We give a method for triangulating the region between a convex polyhedron and a disjoint polyhedral terrain (or a star-shaped polyhedron). The number of tetrahedra produced is $O(n \log n)$. We prove the rather surprising result that the bound is tight. More generally, we show that any convex decomposition of the space between a convex decomposition of the space between a convex polyhedron and a terrain requires $\Omega(n \log n)$ pieces in the worst case.

* The first author was supported in part by NSF Grant CCR-90-02352 and The Geometry Center, University of Minnesota, an STC funded by NSF, DOE, and Minnesota Technology, Inc. The second author was supported in part by NSF Grant PHY-90-21984.

- We prove that any polyhedron of genus g must have at least $g - 1$ reflex dihedral angles. To our surprise, this result appears to be new. Actually, we were not even able to find any proof in the literature that the number of reflex edges is $\Omega(g)$. Although our proof is quite simple it relies on the Gauss–Bonnet curvature formula and thus it is not completely self-contained. This result shows that the triangulation algorithm of Chazelle and Palios [8] works just the same for polyhedra of arbitrary genus: specifically, any polyhedron with n vertices and r reflex angles can be triangulated with $O(n + r^2)$ tetrahedra, regardless of its genus. The bound is tight in the worst case.

Tetrahedralization refers to the triangulation of three-dimensional polyhedra. The subject has been heavily researched because of its relevance to the finite-element method, mesh generation, topology of 3-manifolds, interpolation theory, tool design, etc. [1]–[5], [8], [9], [14], [15]. An odd assortment of results has emerged. For example, all polyhedra can be tetrahedralized, but nonconvex ones might require additional vertices, so-called Steiner points: the canonical example is the Schönhardt octahedron [17], which is made by connecting together two parallel, slightly twisted triangles. Checking whether Steiner points are needed or not has been shown to be NP-hard by Ruppert and Seidel [16]. Steiner points are somewhat of a nuisance in practice, because they make representations more complicated and cause round-off error problems. Unfortunately they can rarely be avoided. The number and the shape of the tetrahedra are parameters that are both more critical and malleable. The former is where we focus our attention in this paper.

It was shown in [8] that any n-vertex polyhedron can be triangulated using only $O(n^2)$ tetrahedra. This is known to be optimal in the worst case [6]. Of course, in many cases it is possible to do much better. For example, a convex polyhedron requires only $O(n)$ tetrahedra. Bern [5] provided motivation for looking at the region between two convex polyhedra. If the polyhedra are disjoint (side-by-side) it is quite easy to achieve an $O(n)$-size triangulation. The nested case is more difficult, and Bern proposed a construction consisting of $O(n \log n)$ tetrahedra. Our first result improves this to $O(n)$, which is optimal. Furthermore, the method allows the polyhedra to be intersecting. It is based on a dovetailed construction of the Dobkin–Kirkpatrick polyhedral hierarchy [10], somewhat in the style of [7] though much simpler. Unlike Bern's construction, our method uses Steiner points. Whether this can be avoided remains an open problem.

The polyhedron used in the quadratic lower bound proof of [6] is built by placing two polyhedral terrains,[1] one on top of the other. It is natural to ask whether the lower bound collapses if one terrain is convex. Our second result shows that $\Theta(n \log n)$ is the tight answer. The upper bound is obtained by carefully pruning the outer polyhedral hierarchy at critical places and tracing its interaction with the terrain by way of a fairly delicate accounting scheme.

The lower bound is achieved by creating an art gallery that requires on the order of $n \log n$ guards: the roof of the gallery is a hangar-like concave structure; the floor

[1] A polyhedral terrain is the graph of a bivariate piecewise-linear continuous function: any vertical line intersects it in exactly one point.

plan is a terrain parametrized by a permutation. We prove that the number of guards needed is precisely equal to the number of swaps required in *mergesorting* that permutation. Expectedly, we use the bit-reversal permutation to achieve the best possible lower bound. The essence of our proof technique is to be able to interpret a triangulation scheme as a permutation routing network, whereby each node corresponds to one or several distinct tetrahedra. This connection allows us to classify the floor plans that lead to easy triangulations by just looking at the complexity of their corresponding permutations.

2. Triangulating Between Two Convex Polytopes

Let P and Q be two convex polytopes,[2] with a total of n vertices. We wish to subdivide the region between the boundaries of P and Q into $O(n)$ tetrahedra. Naive methods such as merging the Dobkin–Kirkpatrick hierarchies of the two polytopes are doomed to fail. Instead, we define a process for "thickening" the boundaries of the polytopes in coordinated fashion, until they become so wide as to cover all of $\mathbb{R}^3$: during the growth process we maintain a triangulation of the union of the thickened boundaries. For this to work, the two boundaries have to grow at comparable speeds.

To flesh out this idea, we introduce the key notion of a *polyhedral layer*, which is simply the region between two nested convex polytopes. The outer and inner boundaries might share faces, so a layer is topologically equivalent to $S^2 \times [0, 1]$ pinched at various places. We define the *magnitude* of a layer to be the total number of faces (of all dimensions) among its inner and outer boundaries. Layers will always come along with a cell decomposition, so any standard data structure for three-dimensional cell complexes[3] will provide an adequate representation [11]. Specifically, we need lists of vertices, edges, facets, and cells, along with their incidence relations.

Thickening the Boundaries. A layer can grow inward or outward. We begin with the inward growth. Pick a maximal independent set of low-degree vertices on the inner boundary. For each such vertex v, form the convex hull of its adjacent vertices and keep only those faces nonadjacent to v: glue the resulting patch to the layer. The space between the patch and the layer is called a *pocket* (Fig. 1). It is a polyhedron of constant description size. Growing a convex polytope inward in this fashion produces the well-known Dobkin–Kirkpatrick hierarchy [10].

The outward growth [12], [13] is equivalent to the inward growth in dual space. We select a maximal independent set of nonadjacent, low-complexity facets on the

[2] A polytope is a bounded polyhedron: our restriction to the bounded case is meant only to simplify the explanation. It is routine to extend our results to unbounded polyhedra. Similarly, we assume that the polytopes are nondegenerate and well behaved. In particular, it is assumed that no vertex is completely surrounded by coplanar facets: if this were the case, we would remove the vertex and merge the incident facets together.

[3] Recall that a cell complex is a subdivision of a region into relatively open faces (i.e., vertices, edges, facets, cells) such that the closures of two adjacent faces intersect in the closure of a face.

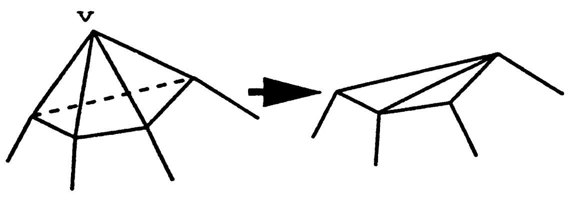

Fig. 1. Inward growth.

outer boundary of the layer. For each such facet f, we intersect the half-spaces bounded by f and its adjacent facets (on the relevant sides). This creates a pocket with at most a constant number of other facets, one of which is f (Fig. 2). There is one delicate point to address: the facets of the pocket are coplanar with the facets adjacent to f. On the one hand, we must merge those facets together, otherwise the complexity of the new outer boundary does not decrease (actually it strictly increases). On the other hand, if we merge the facets, we no longer have a cell complex. To resolve this dilemma, we enclose the new outer boundary B into a slightly enlarged copy of itself. This enlarged copy is triangulated in the same way as B except that coplanar faces are merged. We connect each edge of the enlarged copy to its corresponding edge in B by rectangular or trapezoidal facets (Fig. 3). This creates a layer of very thin *buffer* pockets that completely surrounds the outer boundary; such a pocket is a symbolic device and need not have actual volume.

A simple but important observation is that thickening both the inner and outer boundaries of the layer decreases its magnitude by a constant factor. Note that the magnitude of a layer may have nothing to do with its complexity as a cell complex: as a matter of fact, growing a layer increases the complexity of its cell complex but decreases its magnitude.

Coordinated Growth of Two Layers. We now look at the thickening of two layers, L and L', emanating from P and Q, respectively. We do not make any assumptions about the relative positioning of the layers. We assume that the union of the two layers is fully available as a cell complex (not necessarily simplicial); we call it the *union complex*. Note that it is also provides a cell decomposition of the boundary of $L \cup L'$. We do not assume, however, that it gives a cell decomposition of the four boundaries. For example, features of the boundaries strictly inside $L \cup L'$ might have been removed and have no representative features in the union complex.

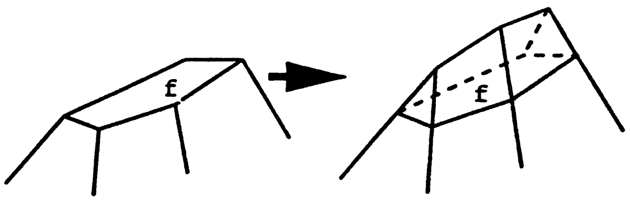

Fig. 2. Outward growth.

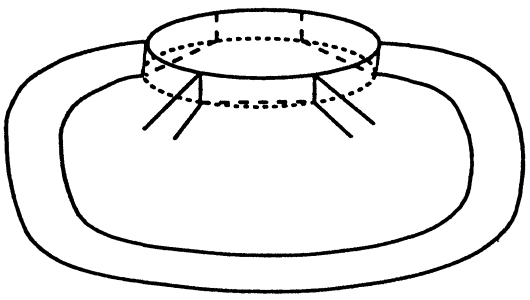

Fig. 3. Construction of buffer pockets. The outer boundary, an enlarged copy, and part of the triangulation between them are drawn.

We assume that all four boundaries of the layers are available separately, i.e., with no regard to the fact that some might intersect: this is the same as having a representation of four distinct convex polytopes. However, we also provide pointers from the facets of the four boundaries to their corresponding facets of the union complex: for this to make sense, we require that any boundary facet that does not lie completely inside $L \cup L'$ should have its "outside" part appear as the union of at most a constant number of facets of the union complex. (Just one is not enough, because such a facet might be homeomorphic to an annulus, so it needs to be cut in two in order to keep all the facets topologically equivalent to disks.)

The reason we need this correspondence will now become clear. Suppose that we wish to grow, say, the layer L inward. Our first task is to determine the intersection of the new pockets of L with the current union complex. If a pocket has a facet in (old) ∂L that intersects L', then routine navigation through the cell complex starting at that intersection does the job. If not, then things are a little more risky, as the pocket might penetrate L' at some random location. We use the optimal algorithm of [7] to intersect the boundaries of L and L' pairwise. With the correspondence mentioned above, this gives us access to all the union complex boundary facets that intersect the pockets, in time linear in the complexity of the boundaries. (There are technical details which we omit.)

What do we do with the new pockets? A pocket that does not intersect either boundary of L' falls in one of two categories (Fig. 4): If it lies entirely inside L', it is ignored, i.e., it has no effect on the union complex. Otherwise, it is simply added to the union complex. The more interesting case arises if a pocket partly intersects L' (Fig. 5). Then we triangulate the complement of L' within the pocket and add these

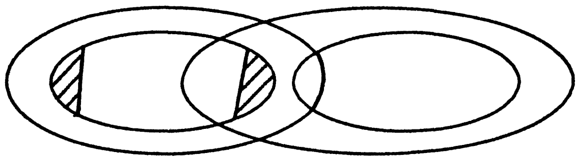

Fig. 4. The shaded region on the left is a pocket of L which lies entirely outside L'. The shaded region in the center is a pocket of L which lies entirely in L'.

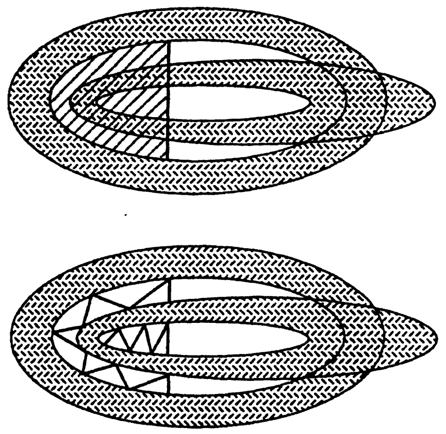

Fig. 5. The shaded region in the top figure is a pocket of L which partly intersects L'. We triangulate the complement of L' in the pocket (the shaded region in the bottom figure).

faces to the current union complex. In Fig. 5 this complement consists of two full-dimensional components; in general, it can have up to a constant number of them (because the pocket has $O(1)$ complexity). We should also observe that the union complex is only modified by growing: once it contains a face it keeps it forever. Also, as we mentioned before, we should not expect the boundary of the new layer L to be represented in the union complex in its totality.

Triangulating the complement of L' in the pocket is quite easy because the boundary of L' is convex and the pocket has constant size. Outward growth is handled similarly. The only difference is that a pocket might now have a large size (remember those thin buffer pockets). It is an easy exercise, however, to carry out the triangulation in linear time.

The Triangulation Algorithm. Returning now to our original problem, we grow the layer of P first inward and then outward. Then we switch and do the same with respect to Q. We iterate this process back and forth until the boundaries become of constant size. At that point, a simple finishing touch adds a few unbounded tetrahedra to create a full-fledged union complex that subdivides all of $\mathbb{R}^3$. Buffer pockets that have not yet been triangulated can be handled now, which produces the desired simplicial complex. The total space and time complexity is easily seen to follow a geometric series summing up to $O(n)$.

Theorem 2.1. *It is possible to triangulate the region between the boundaries of two convex polytopes in linear time, using a linear number of faces.*

Remark. The dovetailing mechanism can be replaced by variants such as always thickening the bigger layer. However, it would be a big mistake to grow one layer all the way, and then work on the other. Similarly, simply merging the two hierarchies fails miserably. In both cases, on the order of $n \log n$ tetrahedra might end up being produced. As should be noted, a key idea to avoid this blowup has been to avoid refining the interior of the layers as they grow.

3. Triangulating Between a Polytope and a Terrain

Let P and Q be respectively a convex polytope and a polyhedral terrain. We could also choose an arbitrary star-shaped polyhedron as Q, since it can be made into a terrain by a suitable projective transformation. For example, if a polyhedron Q is star-shaped with respect to the origin, then Q is transformed into a polyhedral terrain by a projective transformation sending the origin to the point at infinity. We assume that P and Q do not intersect, or if they do, that the input size n accounts for all the intersection features as well.

Theorem 3.1. *It is possible to triangulate the region between two disjoint polytopes (one convex, the other star-shaped), using $O(n \log n)$ tetrahedra, where n is the size of the input polyhedra. The size of the triangulation is asymptotically optimal in the worst case.*

3.1. The Upper Bound

A natural idea would be to compute the outer Dobkin–Kirkpatrick hierarchy of P and merge the terrain Q within it. Such a naive approach runs into problems, however, because terrain facets may intersect too many features of the hierarchy. We get around this problem by pruning the resulting cell complex appropriately. Recall from the previous section that the outer hierarchy of P subdivides $\mathbb{R}^3$ into constant-complexity pockets and arbitrarily thin buffer pockets. Suppose that two facets f, g of the terrain intersect the same pocket and:

1. Both facets intersect the same set of pocket edges.
2. No edge of either facet intersects the pocket.
3. The portion D of the pocket between the two facets does not intersect the terrain (except at f, g).

A polytope D that satisfies all these conditions is called a *drum* (Fig. 6): it consists of two facets (which are pieces of the terrain facets f, g that cut all across the pocket) and at most a constant number of lateral facets (pieces of pocket facets);

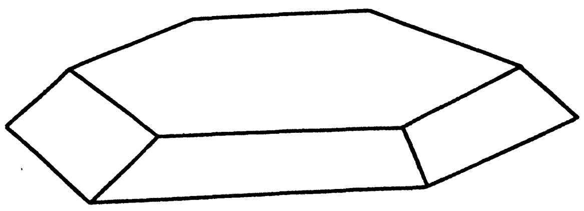

Fig. 6. A drum.

the pair (f, g) constitutes the *type* of the drum. We say that two drums are adjacent if they share a common lateral facet. Our pruning strategy involves removing all the lateral facets between adjacent drums (along with the incident edges and vertices, unless they are needed by neighboring nondrum cells). Note that because two adjacent drums are always of the same type, pruning never brings in contact drums of different type. The merged drums are called *superdrums*. Two obvious questions are: How do we turn the resulting subdivision into a triangulation? Why is the triangulation of size $O(n \log n)$?

Completing the Triangulation. We examine each pocket separately. If the pocket is free of any terrain features, then triangulating it is routine (as usual, care must be taken that the triangulations of a facet should agree on both sides, but this is easy to enforce; see [5] for a thorough treatment of a more general and difficult version of that issue). If the pocket does not give rise to any drum, then no pruning takes place within and we must triangulate a portion of the terrain inside a single pocket. If the pocket is not of the buffer type, then it is of constant size. So we can simply erect vertical walls from the edges and triangulate the resulting cylinders (which is a two-dimensional problem), while also adding enough internal Steiner points to make the facet triangulations compatible (see [1] and [8] for a similar idea). If the pocket is a buffer, then we can further assume that it contains no terrain vertex, so there is not even any need to erect walls.

The only difficult case is when a pocket contains one or several drums. The portion of the pocket outside its drums can be handled as before. The question is how to triangulate a superdrum R. Such a polyhedron resembles a thick polygon (Fig. 7). It consists of two facets F, G (arbitrarily complex simple polygons within f and g) connected together by lateral facets. As in a drum, the edges of the lateral facets provide a bijection between the vertices of the polygons F and G. Note that a given pair (f, g) can give rise to more than one polyhedron R.

Naive methods for triangulating R seem to run into all sorts of snags, as care must be taken to keep the size small. Recall that the pockets contributing the drums of R are arranged in a hierarchical fashion. The cross section of the outer hierarchy of P with the plane supporting F is a nonempty collection of $O(\log n)$ nested convex polygons, C_0, C_1, etc. The $2d$-pockets between them are convex polygons consisting of a base made of some number of collinear segments and a constant-size polygonal arc (Fig. 8). The nondotted $2d$-pockets in the figure indicate the drums. The nondotted region, which is precisely the facet F of R, is surrounded by a set of

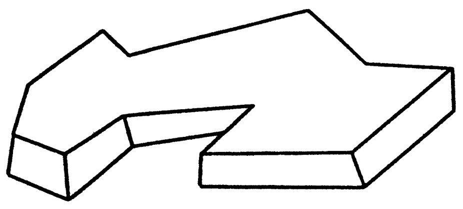
Fig. 7. A superdrum.

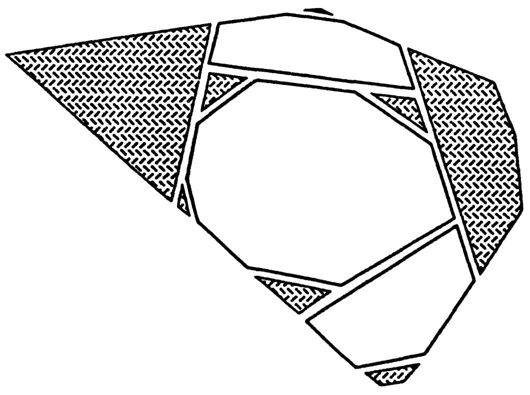

Fig. 8. A top view of the vicinity of a superdrum R. The union of the nondotted regions is the facet F of R, and the dotted regions are the blocking pockets.

blocking $2d$-pockets (the dotted polygons). Note that these blocking pockets might not necessarily be all adjacent to R.

We describe a method for subdividing F and explain how it can be carried over to R. Let B_i be the set of vertices of C_i incident to blocking $2d$-pockets. Note that the B_i are not necessarily disjoint. By analogy, the points of B_i are called *blocking*. We define D_i to be the convex hull of all the points in $B_0 \cup \cdots \cup B_i$. An easy way to visualize D_i is to think of the blocking points of B_j as nails in a board and C_i as a tight rubber band which is to be snapped: the new position of C_i is precisely D_i.

Recall that we have an identical structure on the facet G, so we can similarly define a polygon D_i' from C_i'. Note, however, that although C_i and C_i' are in bijection, the same need not be true of D_i and D_i'. We now match each D_i and D_i' by taking the convex hull H_i of their union. Since the sequences $D_0, D_1, \ldots$ and $D_0', D_1', \ldots$ are nested, the same is true of the sequence of convex hulls H_0, H_1, etc. Let $K_i = (H_i \cap R) \cup H_{i-1}$, so that $R = \bigcup_{i=1}^{n}(K_i \setminus H_{i-1})$. Note that $K_i \setminus H_{i-1}$ is the region between two nested convex polytopes: it is free of any blocking material, so we can apply Theorem 2.1 to triangulate it.

The Complexity of the Triangulation. An edge of the terrain can appear in only $O(\log n)$ pockets. It follows immediately that the number of tetrahedra produced outside of the drums is $O(n \log n)$. For the same reason the combined complexity of all the merged drums is at most $O(n \log n)$. The only question remaining is how many tetrahedra are produced during the triangulation of the superdrum R. The final triangulation step adds only a constant factor to the size, so it suffices to estimate the total complexity of the H_0, H_1, etc. The size of H_i is proportional to the total number of edges in D_i and D_i'. However, notice that the edges in all the D_i's form a planar graph, therefore it suffices to count the number of vertices, which is $|B_0| + |B_1| + \cdots$. Unfortunately, this estimate is too crude. We must take into consideration the fact that not all blocking points contribute to the complexity. In particular, it is clear that only points on blocking $2d$-pockets that are *adjacent* to R can become vertices of the $(R \cap H_i)$'s. Furthermore, if more than two vertices lie on

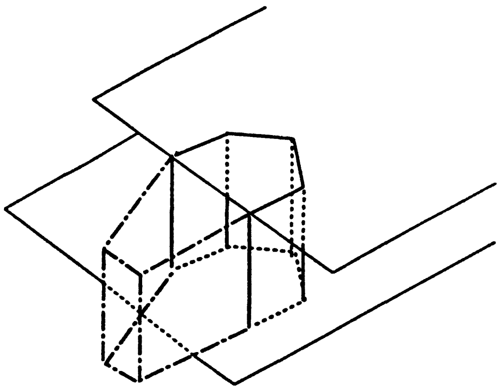

Fig. 9. A blocking pocket.

the base of a blocking $2d$-pocket, only the two endpoints can play any role. It thus appears that the contribution of superdrums to the overall size of the triangulation is at most proportional to the total number of blocking $2d$-pockets adjacent to superdrums. Such a $2d$-pocket is blocking for one of two reasons:

1. It is intersected by an edge of f or g (Fig. 9).
2. It is not intersected by any edge of f or g but it is adjacent to what would be a drum D, if it were not for a terrain edge that intersects D (Fig. 10).

Obviously, there are at most $O(n \log n)$ $2d$-pockets of the first kind. The same is true of the second kind of pockets, but to see why is a little more subtle. The crux is that no single terrain edge can create more than two $2d$-pockets of type 2 within the same three-dimensional pocket G. The reason for this is that D cuts all across G and therefore no terrain edge can intersect both D and $G \setminus D$. We conclude that the triangulation is of size $O(n \log n)$.

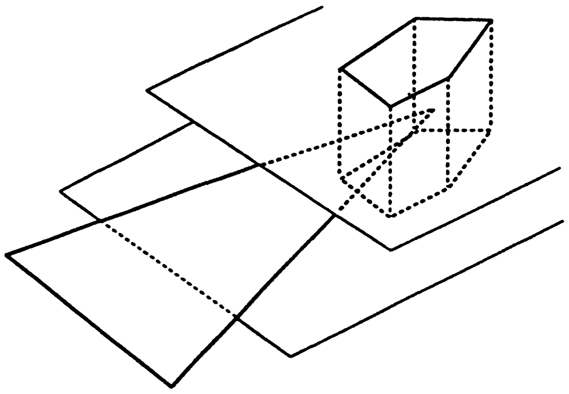

Fig. 10. Another blocking pocket.

The naive algorithm for constructing the triangulation involves intersecting each facet of the terrain with the entire outer hierarchy, which certainly takes no more than $O(n^2)$ time. Within that running time, it is routine to carry out the entire pruning. It is possible to improve the running time slightly, by using fairly complicated batching techniques based on multidimensional searching. It is not worthwhile pursuing that line of attack. Whether a simple, quasi-linear algorithm exists remains an open question.

3.2. *The Lower Bound*

We construct a convex polytope and a terrain disjoint from it, such that any convex decomposition of the space between them requires on the order of $n \log n$ pieces, where n is the combined size of the input.

Building the Polyhedra. The polytope is a cylinder with half of a regular $2m$-gon as a base, where $m = 2^p$. It consists of:

(i) a horizontal square of unit area facing up,
(ii) m *slabs* facing down,
(iii) two congruent vertical facets whose normals are parallel to the y-axis (Fig. 11).

To define the terrain it is helpful to (conceptually) enclose the cylinder inside a tall box, with the square of the cylinder coinciding with the top of the box. The terrain consists of the plane supporting the bottom of the box, plus m blade-like protrusions abutting the slabs of the cylinder (Fig. 11 shows only one of them). Each protrusion is associated with a slab of the cylinder: it is essentially a trapezoid (with small but positive thickness) parallel to the x-axis; its top side is parallel to a slab of the cylinder and comes extremely close to it without touching it. The trapezoids are in one-to-one correspondence with the m slabs of the cylinder. We assume that each

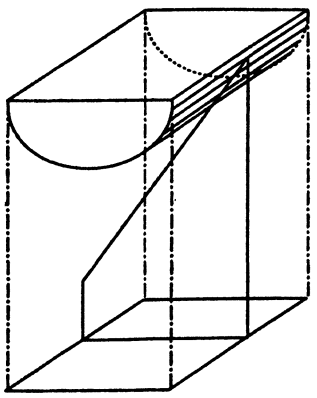
Fig. 11. A convex polytope facing a terrain.

trapezoid extends all across the vertical box. This is why it is necessary to make the box tall enough: a simple calculation shows that the height of the box should be on the order of m. Clearly, the total description size of the input scene is $n = O(m)$.

Our construction is complete once we specify:

(i) The permutation that defines the correspondence between slabs and trapezoids.
(ii) The placement of the trapezoids along the y-direction.

Because the box is tall enough, any permutation on m elements is feasible. We choose the bit-reversal permutation $\pi = (i_0, \dots, i_{m-1})$, where i_k is 1 plus the number obtained by writing k in binary over p bits and reversing the bits. For example, if $p = 3$, we obtain

$$\pi = (1, 5, 3, 7, 2, 6, 4, 8).$$

Placing Guards. The slabs can be thought of as the leaves of a complete binary tree of depth p. An internal node v is thus naturally associated with a sequence of consecutive slabs $s_i, \dots, s_j$. Let $W(v)$ be the wedge formed by two planes tangent to the cylinder along the segments $s_{i-1} \cap s_i$ and $s_j \cap s_{j+1}$. We choose the planes so that their normals bisect the angles between their respective slab pairs. If s_i or s_j is an extreme slab, then we take the vertical plane supporting the relevant wall of the box. Figure 12 shows such a wedge (with some liberty in the drawing to make it clearer). Given a node v (not a leaf, not a parent of a leaf, not the root), consider the bounding line L_v of the wedge $W(v)$. A simple geometric observation is that since the trapezoids are extremely close to the slabs, the line L_v intersects (strictly) the trapezoids in correspondence with the slabs $s_i, \dots, s_j$, and only those. Furthermore, because of the bit-reversal permutation, as we move along the line we encounter trapezoids from the left and right children of v in alternation. The intersection with the trapezoids are such tiny segments that we might as well as think

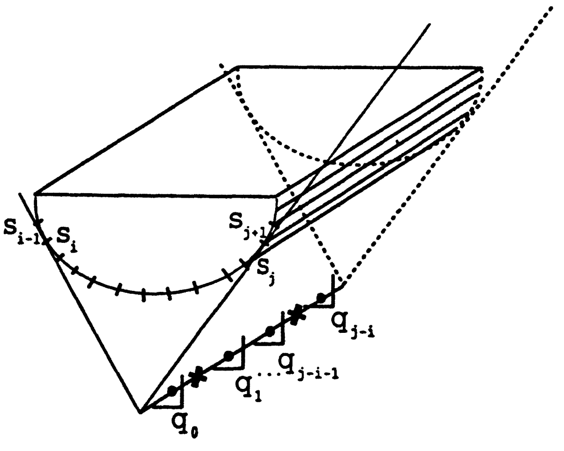

Fig. 12. Placing guards.

of them as points $q_0, \dots, q_{j-i}$ (given in ascending y-order). The midpoint of a segment of the form $q_l q_{l+1}$, for $0 \leq l < j - i$, is called a *guard* (the crosses in Fig. 12). Obviously, there is a total of about $m \log m$ guards overall. The lower bound will follow directly from the fact that by judicious placement of the trapezoids in the y-direction we can ensure that no two guards can see each other.

Instead of assigning each trapezoid an explicit y-coordinate, which would complicate the proof, we define their placement in a recursive manner. We process the tree of trapezoids bottom-up. For a node to be processed means that the entire block of trapezoids associated with it has been placed, up to translation (i.e., its degrees of freedom have been reduced to one). Since we proceed bottom-up, to process v simply means deciding the relative positioning of the blocks associated with the two children of v. To simplify matters further, we make the placements of all the blocks at a given level in the tree identical up to translation. Thus, it suffices to specify the placement of the nodes on the leftmost path of the tree, $v_0, \dots, v_p$.

Let $\varepsilon_k = (c/m)^k$, for some small enough constant $c > 0$. For any $1 \leq k \leq p$, we place the block for the right child of v_k ahead of the block for the left child of v_k by precisely ε_k (Fig. 13). Observe that it is always easy (for c small enough) to fit the entire block of trapezoids within the box. The lower bound follows from the next lemma.

Lemma 3.2. *No two guards can see each other.*

Proof. If the two guards p, q correspond to nodes v, w that are not in any ancestral relationship, then the convex cylindeı itself obstructs their mutual visibility. If $v = w$, then the trapezoids passing through the points q_0, q_1, etc., hide the guards from one another. So, the only case remaining is if, say, v is an ancestor of w. Let j and k $(j < k)$ be the heights of w and v, respectively. If we project trapezoids and guards onto the xz-plane, we find that not only the projection of p lies in that of every trapezoid associated with v, but actually the projected point is at least at a distance proportional to d/m away from any edge of the trapezoid's projection (Fig. 14), where D is the distance from the bounding line L_v to either of the two lines of contact between the wedge $W(v)$ and the cylinder.

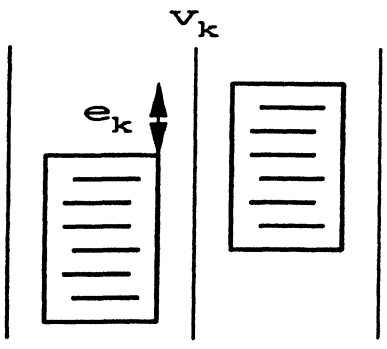

Fig. 13. Offset blocks.

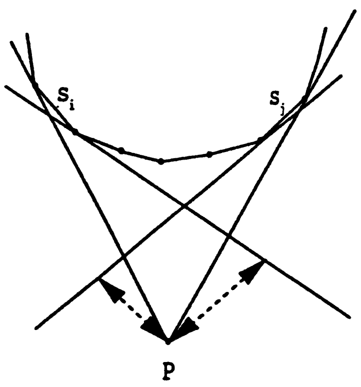

Fig. 14. Projected guards.

Thus, two disks of radius $\Omega(D/m)$ exist which any infinite ray of visibility from p must necessarily avoid: these disks are centered on and normal to the line L_v, and they lie on each side of p at a distance $\varepsilon_k/2$ from it (Fig. 15).

Let p' be the point of L_w nearest to p (Fig. 16). Because of the two obstructing disks, p cannot see any point of L_w away from p' by more than $O(|pp'|\varepsilon_k m/D)$, which is $O(c|pp'|\varepsilon_j/D)$. However, $|pp'|$ is at most D, so by setting c small enough, we find that no point on L_w away from p' by more than $\varepsilon_j/3$ can see p. Nevertheless, q lies at least $(\varepsilon_j - \varepsilon_k)/2$ away from p', which gives a contradiction. □

4. Genus and Reflex Angles

Intuitively, it seems clear that a polyhedron with large genus should have many reflex dihedral angles (we call the corresponding edges *reflex*). Recall that a polyhedron of genus g is a 3-manifold with piecewise-linear boundary that is

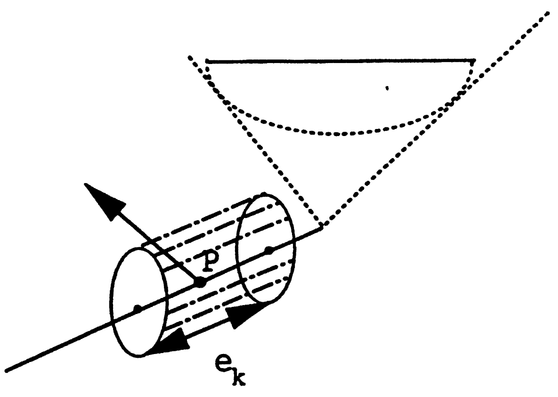

Fig. 15. Avoiding disks.

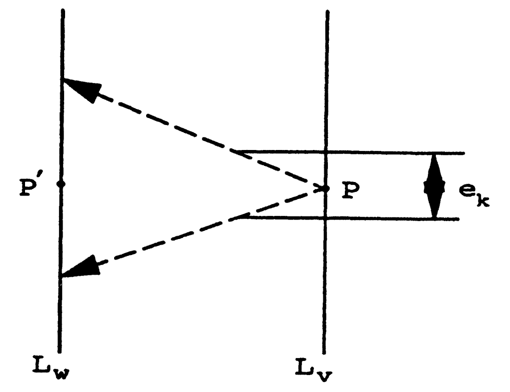

Fig. 16. Completing the proof.

homeomorphic to a torus with g holes. If r is the number of reflex edges, we show that

$$g \leq r + 1. \tag{1}$$

The curvature k_v at the vertex v is defined to be

$$k_v = \frac{1}{2\pi}\left(2\pi - \sum_i \theta_i\right),$$

where θ_i is the angle at v of a triangle incident to v. The Gauss–Bonnet formula relates the total curvature to the genus:

$$\sum_v k_v = 2 - 2g.$$

Inequality (1) follows easily from the next lemma.

Lemma 4.1. *The number of reflex edges incident to a vertex v is at least $-k_v$.*

Proof. Project the facets incident to v onto a unit sphere centered at v. This produces a simple, closed curve made of geodesic pieces. In other words, it is a "polygon" on the sphere made of arcs of great circles. A reflex edge gives rise to a reflex angle on the polygon and vice versa. It is easy to see that

$$L \leq 2\pi(R + 1), \tag{2}$$

where L is the length of the curve and R is its number of reflex angles. Indeed, if R is zero, then the inequality simply states the classical fact that the curvature at a locally convex point is nonnegative. If R is nonzero, then we can draw arcs of great circles from each reflex point along the bisector of its reflex angle, and thus decompose the unit sphere into at most $R + 1$ convex regions (i.e., regions where any two points are visible along an arc of great circle connecting them). The case $R = 0$ can now be used $R + 1$ times to establish (2). The lemma follows immediately from the fact that

$$L = \sum_i \theta_i = 2\pi(1 - k_v). \qquad \square$$

Chazelle and Palios [8] have given an algorithm for triangulating a genus-0 polyhedron with n vertices and r reflex angles, using $O(n + r^2)$ tetrahedra. The algorithm works for arbitrary genus g, but the bound becomes $O(n + (r + g)^2)$ tetrahedra. The inequality $g \leq r + 1$ implies that the true bound is still $O(n + r^2)$. In other words, the complexity bound for genus 0 applies to polyhedra of arbitrary genus as well.

References

1. Aronov, B., Sharir, M. Triangles in space or building (and analyzing) castles in the air. *Combinatorica* **10** (1990), 137–173.
2. Aronov, B., Sharir, M. Castles in the air revisited, *Proceedings of the 8th Annual ACM Symposium on Computational Geometry,* 1992, pp. 146–156.
3. Aronov, B., Sharir, M. The union of convex polyhedra in three dimensions, *Proceedings of the 34th Annual IEEE Symposium on the Foundations of Computer Science,* 1993, pp. 518–527.
4. Bajaj, C. L., Dey, T. K. Convex Decompositions of Polyhedra and Robustness, *SIAM Journal on Computing* **21** (1992), 339–364.
5. Bern, M. Compatible tetrahedralizations, *Proceedings of the 9th Annual ACM Symposium on Computational Geometry,* 1993, pp. 281–288.
6. Chazelle, B. Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm, *SIAM Journal on Computing* **13** (1984), 488–507.
7. Chazelle, B. An optimal algorithm for intersecting three-dimensional convex polyhedra, *SIAM Journal on Computing* **21** (1992), 671–696.
8. Chazelle, B., Palios, L. Triangulating a nonconvex polytope, *Discrete & Computational Geometry* **5** (1990), 505–526.
9. Dey, T. K. Triangulation and CSG representation of polyhedra with arbitrary genus, *Proceedings of the 7th Annual ACM Symposium on Computational Geometry,* 1991, pp. 364–372.
10. Dobkin, D. P., Kirkpatrick, D. G. Fast detection of polyhedral intersection, *Theoretical Computer Science* **27** (1983), 241–253.
11. Dobkin, D. P., Laszlo, M. J. Primitives for the manipulation of three-dimensional subdivisions, *Algorithmica* **4** (1989), 3–32.
12. Martin, A. A simple primal algorithm for intersecting 3-polyhedra in linear time, Tech. Report 91-16, University of British Columbia, Vancouver, July 1991.
13. Mehlhorn, K. *Data Structures and Algorithms, 3: Multidimensional Searching and Computational Geometry,* Springer-Verlag, Heidelberg, 1984.
14. Mulmuley, K. *Computational Geometry: An Introduction Through Randomized Algorithms,* Prentice-Hall, Englewood Cliffs, NJ, 1994.
15. Pellegrini, M. Building convex space partitions induced by pairwise interior-disjoint simplices, ICSI TR-93-039, Berkeley, CA, Aug. 1993.
16. Ruppert, J., Seidel, R. On the difficulty of triangulating three-dimensional nonconvex polyhedra, *Discrete & Computational Geometry* **7** (1992), 227–253.
17. Schönhardt, E. Über die Zerlegung von Dreieckspolyedern in Tetraeder, *Mathematische Annalen* **98** (1928), 309–312.

Received May 1994, and in revised form January 1995.