Vertical Decompositions for Triangles in 3-Space*

Mark de Berg† Leonidas J. Guibas‡ Dan Halperin§

February 22, 1995

Abstract

We prove that, for any constant $\varepsilon > 0$, the complexity of the vertical decomposition of a set of n triangles in three-dimensional space is $O(n^{2+\varepsilon} + K)$, where K is the complexity of the arrangement of the triangles. For a single cell the complexity of the vertical decomposition is shown to be $O(n^{2+\varepsilon})$. These bounds are almost tight in the worst case.

We also give a deterministic output-sensitive algorithm for computing the vertical decomposition that runs in $O(n^2 \log n + V \log n)$ time, where V is the complexity of the decomposition. The algorithm is reasonably simple (in particular, it tries to perform as much of the computation in two-dimensional spaces as possible) and thus is a good candidate for efficient implementations.

The algorithm is extended to compute the vertical decomposition of arrangements of n algebraic surface patches of constant maximum degree in three-dimensional space in time $O(n\lambda_q(n) \log n + V \log n)$, where V is the combinatorial complexity of the vertical decomposition, $\lambda_q(n)$ is a near-linear function related to Davenport-Schinzel sequences, and q is a constant that depends on the degree of the surface patches and their boundaries. We also present an algorithm with improved running time for the case of triangles which is, however, more complicated than the first algorithm.

*Mark de Berg was supported by the Dutch Organization for Scientific Research (N.W.O.), and by ESPRIT Basic Research Action No. 7141 (project ALCOM II:*Algorithms and Complexity*). Leonidas Guibas was supported by NSF grant CCR-9215219, by a grant from the Stanford SIMA Consortium, by NSF/ARPA Grant IRI-9306544, and by grants from the Digital Equipment, Mitsubishi, and Toshiba Corporations. Dan Halperin was supported by a Rothschild Postdoctoral Fellowship, by a grant from the Stanford Integrated Manufacturing Association (SIMA), by NSF/ARPA Grant IRI-9306544, and by NSF Grant CCR-9215219.
A preliminary version of this paper appeared in *Proc. 10th ACM Symposium on Computational Geometry*, 1994, pp. 1-10.

†Department of Computer Science, Utrecht University, P.O.Box 80.089, 3508 TB Utrecht, the Netherlands.

‡Department of Computer Science, Stanford University, Stanford, CA 94305.

§Robotics Laboratory, Department of Computer Science, Stanford University, Stanford, CA 94305.

1 Introduction

The study of arrangements plays a fundamental role in geometric computing. An arrangement is the partition of a Euclidean space into cells, as induced by a collection of possibly highly inter-penetrating objects. A surprising number of seemingly unrelated geometric problems boil down to the study of certain cells in such an arrangement. A famous example is the motion planning problem in robotics. Here the underlying arrangement is the arrangement in configuration space of the constraint surfaces defined by the obstacles and the robot. Because of these numerous applications, much research has been devoted to bounding the combinatorial complexity of arrangements, and of certain important subsets of arrangements such as zones and single cells.

For most algorithmic uses, however, a raw arrangement is an unwieldy structure. The difficulty is that cells in an arrangement can have very complex topologies, so navigating around them is difficult. What we often want is a further refinement of the cells into pieces, such as simplices, that are each homeomorphic to a ball and have constant description complexity. Ideally, the number of cells after the refinement should be proportional to the overall complexity of the arrangement. For arrangements of hyperplanes the well-known bottom vertex triangulation [15] meets this criterion. For more general arrangements such refined decompositions are more difficult to find. For example, for algebraic hypersurfaces of constant maximum degree in d-dimensional space ($d \geq 3$) the best decomposition technique known so far results in $O(n^{2d-3}\beta(n))$ cells [10], where $\beta(n)$ is an extremely slowly growing function,[1] whereas the complexity of the arrangement itself is only $O(n^d)$.

In this paper we study decompositions for arrangements of triangles in three-dimensional space. The simplest way to decompose such an arrangement is to compute the bottom vertex triangulation of the arrangement of the planes containing the triangles. The resulting decomposition has size $\Theta(n^3)$, which is optimal in the worst case. In many applications, however, the actual complexity of the arrangement of triangles is much smaller. So the challenge is to obtain a decomposition whose size is sensitive to the complexity of the arrangement of the triangles.

Such a complexity-sensitive decomposition was given by Aronov and Sharir [4]: their *Slicing Theorem* states that one can decompose an arrangement of n triangles in 3-space into $O(n^2\alpha(n)+K)$ tetrahedra, where K is the complexity of the arrangement. This result is close to optimal: $\Omega(K)$ is clearly a lower bound on any decomposition, and Chazelle [7] shows that there are arrangements of complexity $O(n)$ such that any decomposition into convex cells has size $\Omega(n^2)$. (The triangles in Chazelle's example form the boundary of a simple polytope.) The Slicing Theorem obtains a decomposition by adding vertical walls for each of the triangle boundary edges, one after the other. The wall of an edge e is obtained by "flooding" the zone of

[1] To be precise: $\beta(n) = 2^{\alpha(n)^c}$, where c is a constant depending on the dimension and the degree of the surfaces. Here and throughout the paper, $\alpha(n)$ is the extremely slowly growing functional inverse of Ackermann's function.

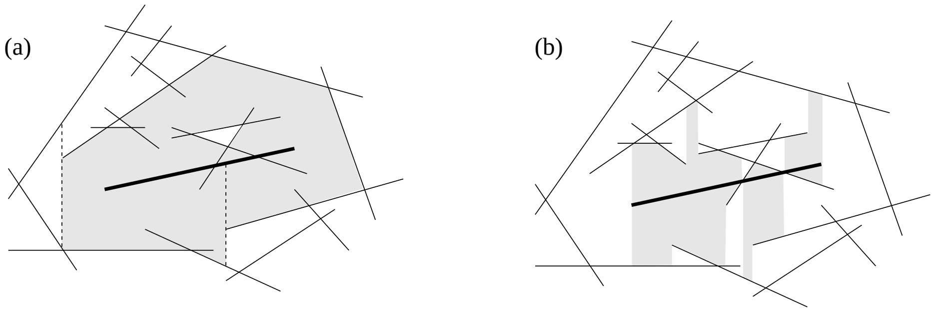

Figure 1: The vertical wall for the fat edge in the Slicing Theorem and in the vertical decomposition.

e in an arrangement on the vertical plane $H(e)$ containing e; this arrangement is defined by intersections of $H(e)$ with the triangles and with already added walls. See Figure 1(a); the dashed segments in this figure are previously added walls. After adding the walls one is left with convex cells that can easily be decomposed into tetrahedra. The Slicing Theorem decomposition has the unpleasant characteristic that it depends on the order in which triangle boundary edges are treated. Thus the tetrahedra in the decomposition are not defined "locally", and it is not canonical in the sense of Chazelle and Friedman [12]. This means that the decomposition is not very well suited for randomized incremental algorithms. It also makes it difficult to compute the decomposition efficiently.

A decomposition which does not have this problem—and one which we think is more simple to compute—is the following [14, 30, 31]. This decomposition is also obtained by erecting vertical walls. This time the wall for edge e simply consists of those points in $H(e)$ that can be connected to e with a vertical segment that does not cross any of the triangles in T. See Figure 1(b). This gives us a first decomposition $\mathcal{V}_1(T)$. Secondly, walls are erected from the intersection edges between pairs of triangles to produce a finer decomposition $\mathcal{V}_2(T)$. Observe that the wall erected from an edge is not obstructed by other walls, so the decomposition does not depend on the order in which the edges are treated. We call this decomposition the *vertical decomposition* for T and denote it by $\mathcal{V}(T) = \mathcal{V}_2(T)$. Note that the cells in $\mathcal{V}_2(T)$ need not be convex; in fact, they need not even be simply connected. But the decomposition can easily be refined into a convex subdivision $\mathcal{V}_3(T)$ where each cell has constant complexity, without increasing the asymptotic complexity of the subdivision—see Section 4.3 for details. We call the refined subdivision the *full vertical decomposition*[2] for T.

In this paper we prove bounds on the maximum combinatorial complexity of vertical decompositions. Our bounds are sensitive to the complexity of the arrangement of

[2]Mulmuley [31] calls $\mathcal{V}_3(T)$ the vertical decomposition, and he calls $\mathcal{V}_2(T)$ the cylindrical decomposition.

the triangles. More precisely, we show that, for any constant $\varepsilon > 0$, the complexity of the vertical decomposition of a set T of n triangles in 3-space is $O(n^{2+\varepsilon} + K)$, where K is the complexity of the arrangement $\mathcal{A}(T)$ induced by T. Our proof uses an interesting combination of efficient hierarchical cuttings [8, 29], the counting scheme used in hereditary segment trees [11], the Slicing Theorem [4], and random sampling [16, 26]. Our proof can be adapted to show that the vertical decomposition of a single cell in an arrangement of triangles has $O(n^{2+\varepsilon})$ complexity. We also give an example of a set T of triangles whose vertical decomposition has complexity $\Theta(n^2\alpha^2(n))$, whereas the complexity of $\mathcal{A}(T)$ is only $\Theta(n\alpha(n))$. This shows that our bound is close to optimal.

Secondly, we present a deterministic algorithm for constructing $\mathcal{V}_3(T)$ which runs in time $O(n^2 \log n + V \log n)$, where V is the complexity of $\mathcal{V}(T)$. The algorithm is reasonably simple (in particular, it tries to perform as much of the computation in two-dimensional spaces as possible) and thus is a good candidate for efficient implementations. The algorithm is also interesting as it is a three-dimensional version of a Bentley-Ottmann style sweep and may be adaptable to compute other partial or total information about the arrangement. Our approach is related to a space sweep algorithm that has recently been developed to compute a decomposition of certain arrangements for motion planning problems [24], and to the space sweep methods used to construct point location data structures for monotone subdivisions [22, 34].

We then extend the algorithm to compute the vertical decomposition of arrangements of n algebraic surface patches of constant maximum degree in three-dimensional space. The running time of the algorithm is $O(n\lambda_q(n) \log n + V \log n)$, where V is the combinatorial complexity of the vertical decomposition, $\lambda_q(n)$ is a near-linear function related to Davenport-Schinzel sequences, and q is a constant that depends on the degree of the surface patches and their boundaries.

For triangles, we also mention an alternative algorithm whose overhead term is subquadratic. This algorithm uses multi-level data structures (for ray shooting and similar problems) and so it is substantially more complicated. Its running time is $O(n^{4/5+\varepsilon}V^{4/5})$, which means that it is faster than the simple algorithm only when $V = O(n^{6/5-\delta})$ for some constant $\delta > 0$.

We note that even the final refined trapezoidation $\mathcal{V}_3(T)$ is not simplicial, in the sense that a facet of a particular cell may have multiple cells bordering it on the other side. This raises a number of interesting questions when it comes to navigating across cell boundaries.

The rest of the paper is organized as follows. In Section 2 we introduce the basic assumptions and terminology that will be used throughout the paper. The combinatorial analysis of the complexity of the vertical decomposition is presented in Section 3. In Section 4 we present the output-sensitive algorithm to compute the vertical decomposition, together with variants of the algorithm. Some concluding remarks and open problems are given in Section 5.

2 Preliminaries

Let $T = \{t_1, \ldots, t_n\}$ be a collection of n (possibly intersecting) triangles in $\mathbb{R}^3$. Let $\mathcal{A}(T)$ denote the arrangement induced by T, namely, the subdivision of 3-space into cells of dimensions $0, 1, 2$ and 3, induced by the triangles in T. We make the same *general position* assumption as Aronov and Sharir [4]: no two edges of distinct triangles intersect, no vertex of a triangle is contained in another triangle, and so on. In particular we assume that no triangle is vertical, that is, no triangle is parallel to the z-axis. (In the sequel "vertical" will always mean parallel to the z-axis. When we are discussing arrangements on the xy-plane we will explicitly say "y-vertical" when we mean parallel to the y-axis.) By standard arguments [35] the combinatorial bound that we derive in Section 3 holds for degenerate arrangements as well. However, the algorithms described in Section 4 will have to undergo several technical adjustments (which we do not discuss in this paper) to accommodate for degenerate arrangements.

The *combinatorial complexity* (or *complexity* in short) of an arrangement $\mathcal{A}$ is defined to be the overall number of cells of various dimensions in $\mathcal{A}$; we denote the complexity of $\mathcal{A}$ by $|\mathcal{A}|$.

Central to the concept of vertical decompositions is the following notion of visibility: two points $p, q \in \mathbb{R}^3$ *(vertically) see each other with respect to* T if and only if the segment $\overline{pq}$ connecting them is vertical and the relative interior of $\overline{pq}$ does not intersect any triangle in T. Usually the set T is clear from the context and we just say that p and q see each other. This definition is extended to objects other than points as follows: two sets $P, Q \subset \mathbb{R}^3$ see each other if and only if there are points $p \in P$, $q \in Q$ that see each other.

We define two three-dimensional entities related to a three-dimensional curve γ. Let $H(\gamma)$ be the vertical surface that is the union of all the vertical lines which contain a point of γ. (Note that throughout the paper, whenever this definition is used, γ is a curve whose projection onto the xy plane is a simple curve, i.e., it is non-self-intersecting.) Let the *vertical wall extended from the three-dimensional curve* γ, denoted $W(\gamma, T)$, be defined as follows: $W(\gamma, T) := \{p \in \mathbb{R}^3 : p \text{ sees } \gamma\}$. In other words, $W(\gamma, T)$ is the union of all vertical segments of maximal length that have a point of γ as an endpoint and whose interior does not intersect any triangle in T. Note that some of these segments can be rays.

3 The Combinatorial Bounds

We first prove bounds on the size of the vertical decomposition of the full arrangement of a set of triangles in 3-space. The same proof technique is then used to derive a bound for the case of a single cell.

3.1 The Full Arrangement

Let $T = \{t_1, \ldots, t_n\}$ be a set of triangles in $\mathbb{R}^3$, as defined above. We investigate the maximum combinatorial complexity of $\mathcal{V}(T) = \mathcal{V}_2(T)$ as a function of n, the number of triangles in T, and K, the complexity of $\mathcal{A}(T)$. Note that the complexity of $\mathcal{V}(T)$ is at least K.

We denote by $E(T)$ the set of segments in 3-space that are either an edge of a triangle in T or the intersection of two triangles in T. We call the segments in $E(T)$ edges; when we discuss edges of triangles in T we will explicitly say triangle boundary edges, and when we discuss intersections between triangles we will say intersection edges. For an edge $e \in E(T)$ we define its vertical wall to be $W(e, T)$ (see Section 2), namely, $W(e, T) := \{p \in \mathbb{R}^3 : p \text{ sees } e\}$. Let $\mathcal{W}(T) := \{W(e, T) : e \in E(T)\}$ be the collection of all the vertical walls. The vertical decomposition for T is the subdivision induced by the set $T \cup \mathcal{W}(T)$.

We first consider the complexity of a single wall $W(e, T)$. Recall that $H(e)$ is the vertical surface containing e. By definition, the part of $W(e, T)$ above e is bounded by the lower envelope of the segments that are intersections of the other triangles in T with $H(e)$ and lie above e. See also Figure 1(b). Since the complexity of the lower envelope of n segments in the plane is $O(n\alpha(n))$ [25], the part of $W(e, T)$ above e has $O(n\alpha(n))$ complexity. A similar argument holds for the part of $W(e, T)$ below e. Hence, a single wall has complexity $O(n\alpha(n))$. Because there are $3n$ triangle boundary edges we can make the following observation.

Observation 3.1 *The total complexity of the walls $W(e, T)$ for all boundary edges e of the triangles in T is $O(n^2\alpha(n))$.*

The total number of edges in $E(T)$—and thus the total number of walls—is $O(n^2)$. It follows that the maximum complexity of $\mathcal{V}(T)$ is $O(n^3\alpha(n))$, as was noted by Mulmuley [31]. More precisely, it follows that $|\mathcal{V}(T)| = O((n + N)n\alpha(n)))$, where N is the number of pairwise intersections of triangles in the arrangement. However, it is not clear whether it is possible that all walls have $\Theta(n\alpha(n))$ complexity. Indeed, below we show that this cannot happen when K is large. But first we give an example showing that for small K most walls can have large complexity.

Theorem 3.1 *There exists a set T of n triangles in $\mathbb{R}^3$ with $|\mathcal{A}(T)| = \Theta(n\alpha(n))$ and $|\mathcal{V}(T)| = \Theta(n^2\alpha^2(n))$.*

Proof: Let S' be a set of $\lfloor n/2 \rfloor$ line segments in the yz-plane whose upper envelope has complexity $\Theta(n\alpha(n))$ [37], and such that S' lies completely below the plane $z = 0$. Extend each segment in the x-direction to obtain a set T' of infinitely long strips, that is, let $T' := \{[-\infty : \infty] \times s : s \in S'\}$. The upper envelope of T' contains $\Theta(n\alpha(n))$ lines that are parallel to the x-axis. (The construction can easily be modified to use bounded triangles instead of infinitely long strips.) In the same way we can construct a set T'' of $\lceil n/2 \rceil$ strips whose *lower* envelope contains $\Theta(n\alpha(n))$ lines that are parallel to the y-axis, and that lie completely above the plane $z = 0$. For the set $T = T' \cup T''$ we have $|\mathcal{A}(T)| = \Theta(n\alpha(n))$, and $|\mathcal{V}(T)| = \Theta(n^2\alpha^2(n))$. $\square$

Next, we will establish an upper bound on the complexity of the vertical decompositions of sets of triangles in 3-space. Bounding the complexity of vertical decompositions amounts to bounding the sum of the complexities of the walls in $\mathcal{W}(T)$. There are two types of walls: walls erected for triangle boundary edges and walls erected for intersection edges. The total complexity of the walls erected for the triangle boundary edges is $O(n^2\alpha(n))$ by Observation 3.1.

Now consider a wall erected from an intersection edge e. Let $S(e)$ be the set of segments that are the intersections of the other triangles with the vertical surface $H(e)$. As remarked before, the part of $W(e, T)$ above e is bounded by the lower envelope of (the parts of) the segments in $S(e)$ lying above e, and the part of $W(e, T)$ below e is bounded by the upper envelope of (the parts of) the segments in $S(e)$ lying below e. The complexity of $W(e, T)$ is therefore linear in the number of points of the following types:[3]

(1) intersections of a segment $l \in S(e)$ with e,

(2) endpoints of a segment $l \in S(e)$ that are vertically visible from e,

(3) intersections between two segments $l_1, l_2 \in S(e)$ that are vertically visible from e.

Figures 2 illustrates the three types. The first type of endpoint is the intersection of

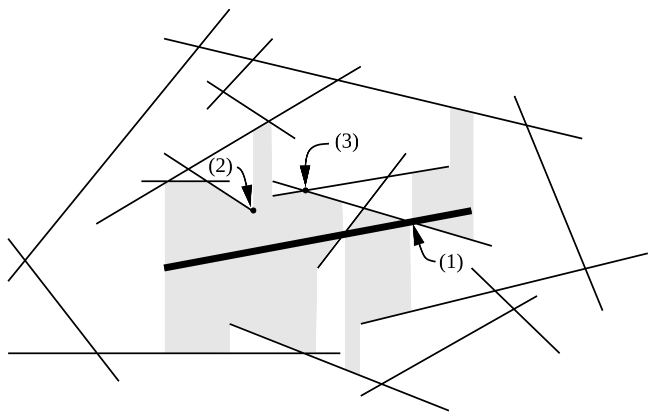

Figure 2: Three types of features on a vertical wall.

the two triangles that define e and the triangle that defines l. In other words, it is a vertex of $\mathcal{A}(T)$. We charge this feature in $W(e, T)$ to this vertex of $\mathcal{A}(T)$. This

[3]If the wall has constant complexity this may not be true, but the total complexity of all constant complexity walls is $O(n^2)$.

way every vertex of $\mathcal{A}(T)$ gets charged a constant number of times. Hence, the total number of such features over all walls in $\mathcal{W}(T)$ is $O(K)$.

Now consider the second type of endpoint. Note that the endpoint of l is the intersection of an edge e' of the triangle that defines l with $H(e)$. So there is a visibility between e and e', which implies that e defines a feature of $W(e', T)$. We charge the feature on $W(e, T)$ to the feature on $W(e', T)$. A feature gets charged at most once this way. Recall that the sum of the complexities of the walls erected from triangle boundary edges is $O(n^2\alpha(n))$. Hence, the total number of features of type 2 is also bounded by $O(n^2\alpha(n))$.

When the third type of endpoint occurs there is a visibility between two intersection edges, namely edge e and the intersection edge of the triangles that define l_1 and l_2. The remainder of this section is devoted to bounding the total number of such visibilities.

The bipartite case

We first study the following "bipartite" version of the problem. Let h be a fixed non-vertical plane, let T_1 be a set of n triangles lying completely below h, and let T_2 be a set of n triangles lying completely above h. We want to bound the number of pairs $e_1 \in E(T_1), e_2 \in E(T_2)$ that can see each other. Let $b(T_1, T_2)$ denote this number, and let $b(n)$ be the maximum value of $b(T_1, T_2)$ over all sets T_1 and T_2 of n triangles each, as defined above. The lower bound example in the proof of Theorem 3.1 shows that $b(n) = \Omega(n^2\alpha^2(n))$. We now establish upper bounds for $b(n)$. Recall that we only need to consider visibilities between intersection edges, as the number of remaining visibilities is $O(n^2\alpha(n))$.

We say that a planar curve γ is *convex* if and only if γ is contained in the boundary of its convex hull. In other words, any line intersects γ at most twice. The following simple lemma is crucial in our upper bound proof.

Lemma 3.1 *Let γ be a convex curve in the plane h. Then the number of visibilities between segments in $E(T_i)$ and γ is $O(n2^{\alpha(n)})$, for $i = 1, 2$.*

Proof: Recall that $H(\gamma)$ is the vertical surface containing γ. Define $t^* = t \cap H(\gamma)$ and $T_1^* = \{t^* : t \in T_1\}$. Each t_i^* consists of at most three connected components. The curve γ sees a segment $e \in E(T_1)$ if and only if $e \cap H(\gamma)$ is a vertex of the upper envelope of T_1^*. An intersection of two curves t_i^* and t_j^* corresponds to an intersection of $H(\gamma)$ and $t_i \cap t_j$. Since γ is a convex curve, there are at most two such intersections. Hence, the complexity of the upper envelope of T_1^* is at most $\lambda_4(n) = O(n2^{\alpha(n)})$ [3]. A similar argument holds for $E(T_2)$. $\square$

Using this lemma we can prove an almost tight upper bound on $b(n)$. A basic ingredient in the proof are efficient hierarchical cuttings [8, 29], which we define next. Let H be a set of n hyperplanes in $\mathbb{R}^d$. A $(1/r)$-cutting for H is a subdivision of $\mathbb{R}^d$

into disjoint simplices such that the interior of each simplex is intersected by at most n/r hyperplanes in H. The size of a cutting is the number of simplices it consists of. We say that a cutting Ξ' C-refines a cutting Ξ if every simplex of Ξ' is completely contained in a single simplex of Ξ and every simplex of Ξ contains at most C simplices of Ξ'. Now let C, ρ be constants and r a parameter with $1 \leq r \leq n$. A sequence $\boldsymbol{\Xi} = \Xi_0, \Xi_1, \ldots, \Xi_k$ of cuttings is called an *efficient hierarchical* $(1/r)$*-cutting (for* H*)* if Ξ_0 is the single "simplex" $\mathbb{R}^d$, every Ξ_i $(1 \leq i \leq k)$ is a $(1/\rho^i)$-cutting of size $O(\rho^{id})$ (where the constant in $O(\rho^{id})$ does not depend on i) that C-refines Ξ_{i-1}, and $\rho^{k-1} < r \leq \rho^k$. Notice that the last condition implies that $k = \Theta(\log r)$. We call the simplex in Ξ_{i-1} that contains a certain simplex $s \in \Xi_i$ the parent of s, denoted by parent(s). Chazelle [8] has shown (see also Matoušek [29]) that for any given H and r there exists an efficient hierarchical cutting (for certain constants C, ρ).

We are now ready to prove an upper bound on $b(n)$.

Lemma 3.2 $b(n) = O(n^2 2^{\alpha(n)} \log n)$.

Proof: Let T_1, T_2, and h be as defined above. Project the triangles of T_1 and T_2 vertically onto the plane h. Construct an efficient hierarchical $(1/6n)$-cutting for the $6n$ lines containing the projected triangle edges. The interior of each simplex in the final cutting is crossed by at most one edge; we add one more level to the hierarchy to get a cutting where the simplices have no edges crossing their interior. Thus we get a hierarchy $\boldsymbol{\Xi} = \Xi_0, \Xi_1, \ldots, \Xi_k$ such that Ξ_0 is the single "simplex" $\mathbb{R}^d$, every Ξ_i $(1 \leq i \leq k)$ is a $(1/\rho^i)$-cutting of size $O(\rho^{id})$ that C-refines Ξ_{i-1}, $k = O(\log n)$, and the simplices of Ξ_k have no edges crossing their interior. For a simplex $s \in \Xi_i$, let $T_1^{\subset}(s) \subset T_1$ be the set of triangles in T_1 whose projection fully contains s but whose projection does not contain parent(s), and let $T_1^{\times}(s)$ be the set of triangles whose projection intersects the interior of s but does not contain it; see Figure 3. Observe that $T_1^{\times}(s)$ is empty for the simplices of Ξ_k. Let $T_1(s) := T_1^{\subset}(s) \cup T_1^{\times}(s)$. Define $T_2^{\subset}(s)$, $T_2^{\times}(s)$, and $T_2(s)$ analogously. Finally, let $T^{\subset}(s) := T_1^{\subset}(s) \cup T_2^{\subset}(s)$, $T^{\times}(s) := T_1^{\times}(s) \cup T_2^{\times}(s)$, and $T(s) := T_1(s) \cup T_2(s)$. In the remainder we only work with "clipped" triangles, that is, for each triangle $t \in T(s)$ we only consider the part that projects onto s.

Consider a visibility between intersection edges $e_1 \in E(T_1)$ and $e_2 \in E(T_2)$. Let $t(e_1)$ and $t'(e_1)$ be the two triangles that define e_1, that is, $e_1 = t(e_1) \cap t'(e_1)$. Similarly, let $e_2 = t(e_2) \cap t'(e_2)$. We denote the projection of e_i onto h by $\overline{e_i}$. The basic observation behind our proof is the fact that there must be a simplex s in some cutting Ξ_i such that the intersection point $p := \overline{e_1} \cap \overline{e_2}$ lies in s, the triangles $t(e_1), t'(e_1), t(e_2), t'(e_2)$ are in $T(s)$, and at least one of these triangles is in $T^{\subset}(s)$.

To see why this observation is true, follow the path of p down the hierarchical cutting and consider what happens to the triangles $t(e_1)$, $t'(e_1)$, $t(e_2)$, and $t'(e_2)$. Let s_i denote the simplex in the cutting Ξ_i that contains p. (If p is on the boundary between some simplices of Ξ_i then any one of these simplices can be chosen. We may assume that p lies in the interior of the projections of the four triangles. Hence, all

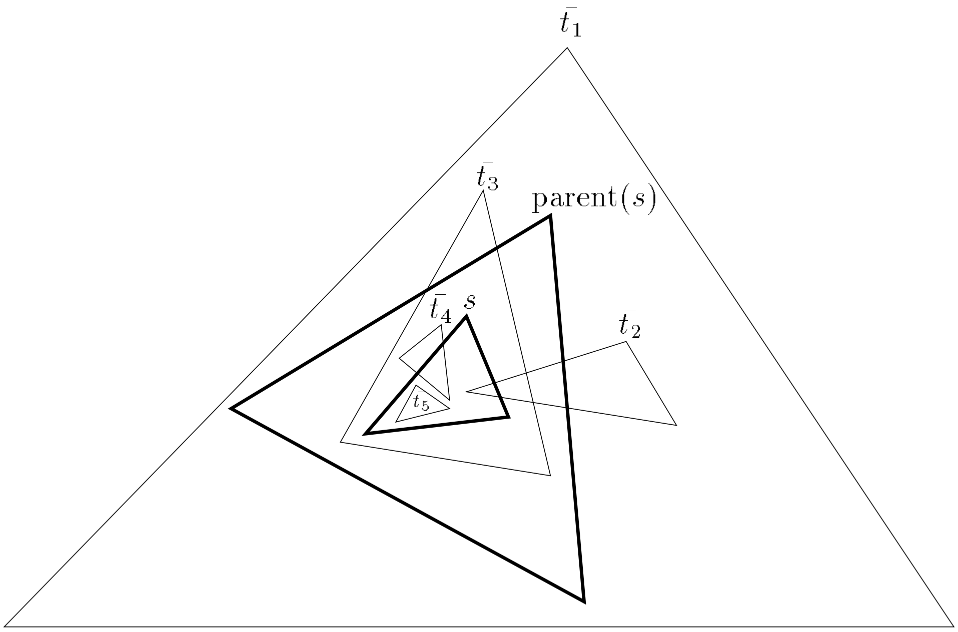

Figure 3: Triangle $\bar{t}_i$ is the projection of triangle $t_i \in T_1$. For the example depicted in the figure, $T_1^{\subset}(s) = \{t_3\}$, and $T_1^{\times}(s) = \{t_2, t_4, t_5\}$.

four triangles intersect the interior of the chosen simplex, which is sufficient for the proof.) Each of the four triangles is contained in $T^{\times}(s_0)$, because $s_0 = h$. This means that each of them is either in $T^{\times}(s_1)$ or in $T^{\subset}(s_1)$. If a triangle is in $T^{\times}(s_1)$, then it must be in either $T^{\times}(s_2)$ or $T^{\subset}(s_2)$. Continuing this argument, we see that for each of the triangles there is an i with $0 < i \leq k$ such that the triangle is in $T^{\times}(s_j)$ for $j = 0, \ldots, i-1$, and in $T^{\subset}(s_i)$. Hence, if i^* is the smallest i for which any one of the four triangles is in $T^{\subset}(s_i)$, then s_{i^*} is the simplex that we are looking for. Because $T^{\times}(s_k)$ is empty, the index i^* must exist.

The observation implies that we only have to count for each simplex $s \in \mathbf{\Xi}$ the number of visibilities where all involved triangles are in $T(s)$ and at least one of them is in $T^{\subset}(s)$. (A similar observation is often made when one uses hereditary segment trees [11]: only long-long and long-short intersections need to be considered, not short-short intersections.)

So let's count the number of such visibilities for a given simplex s. Let $n_s := |T(s)|$. Consider some triangle $t \in T_1(s)$. We shall count the number of visibilities involving intersections of t and triangles $t' \in T_1^{\subset}(s)$. To this end we consider the intersection of t with the upper envelope of $T_1^{\subset}(s)$; an intersection between t and a part of some $t' \in T_1^{\subset}(s)$ that is not on the upper envelope can never be visible from above. The upper envelope of $T_1^{\subset}(s)$ is a convex polyhedral surface (that is, it is on the boundary of a convex polyhedron), and the projection of the intersection of t with the upper envelope of $T_1^{\subset}(s)$ is contained in a convex curve γ. Lemma 3.1 now tells us that the total number of visibilities between γ and some edge in $E(T_2(s))$ is $O(n_s 2^{\alpha(n_s)})$. In other words, the number of visibilities involving intersections of t and some triangle $t' \in T_1^{\subset}(s)$ is $O(n_s 2^{\alpha(n_s)})$. A similar argument holds of course for the triangles in $T_2(s)$, so the total number of visibilities at s that involve at least one triangle in $T^{\subset}(s)$ is $O(n_s^2 2^{\alpha(n_s)})$.

To obtain the total number of visibilities we have to sum over all simplices s in the hierarchical cutting. Each triangle $t \in T(s)$ has an edge intersecting parent(s). Hence, for a simplex $s \in \Xi_i$ we have $n_s \leq n/\rho^{i-1}$. Thus the total number of visibilities can be bounded as follows:

$$\begin{aligned}
\textstyle\sum_s O(n_s^2 2^{\alpha(n_s)}) &= 2^{\alpha(n)} \textstyle\sum_{0\leq i\leq k} \sum_{s\in\Xi_i} O(n_s^2) \\
&= 2^{\alpha(n)} \textstyle\sum_{0\leq i\leq k} O(\rho^{2i}) \cdot O((n/\rho^{i-1})^2) \\
&= n^2 2^{\alpha(n)} \textstyle\sum_{0\leq i\leq k} O(1) \\
&= O(n^2 2^{\alpha(n)} \log n) \ .
\end{aligned}$$

$\square$

Remark. The method of the proof of Lemma 3.2 can also be applied in other situations. As an example, consider the upper envelope of a set T of n $(d-1)$-simplices in $\mathbb{R}^d$. The maximum complexity of this envelope is $\Theta(n^{d-1}\alpha(n))$ [21]. Our proof technique does not give this optimal result, but an almost tight upper bound of $O(n^{d-1}\log n)$. The following proof, however, is very simple:

The complexity of the upper envelope of T is determined by the number of visible d-wise intersections, that is, intersections between d of the simplices in T. To count the number of visible d-wise intersections we use the above proof method. Project the simplices onto the hyperplane $x_d = 0$, and compute in this hyperplane an efficient hierarchical cutting for the set of $(d-2)$-hyperplanes containing the facets of the projected simplices. We use the term *box* for the simplices of the cutting, to avoid confusion with (the projections of) the simplices in T.

For a box s in the hierarchical cutting, let $T^{\subset}(s)$ be the set of simplices in T whose projections fully contain s but not parent(s), let $T^{\times}(s)$ be the set of simplices whose projections intersect the interior of s but do not contain it, and let $T(s) := T^{\times}(s) \cup T^{\subset}(s)$. We must count the number of visible d-wise intersections that involve k simplices from $T^{\times}(s)$, where $0 \leq k < d$; the intersections involving d simplices from $T^{\times}(s)$ are counted at lower levels in the hierarchy. There are $\Theta(n_s^k)$ k-tuples of simplices in $T^{\times}(s)$, where $n_s := |T(s)|$. Each k-tuple defines a k-dimensional simplex σ. Now σ may participate in several d-wise intersections. But each *visible* intersection must be an intersection with the convex polyhedron $\mathcal{P}(T^{\subset}(s))$ that is the intersection of the positive half-spaces bounded by the hyperplanes through the simplices in $T^{\subset}(s)$. In other words, the visible d-wise intersections in which σ participates are vertices of $\sigma \cap \mathcal{P}(T^{\subset}(s))$, which is a convex polytope in $(d-k)$-space. Hence, the number of such vertices is $O(n_s^{\lfloor (d-k)/2 \rfloor})$. So the total number of vertices on the upper envelope that we have to count at s is

$$\sum_{0\le k<d} O(n_s^k n_s^{\lfloor (d-k)/2\rfloor}) = O(n_s^{d-1}).$$

Summing over all boxes of the hierarchical cutting, we see that the total complexity of the upper envelope is $O(n^{d-1}\log n)$.

The general case

Before we can prove a bound on the complexity of vertical decompositions that is sensitive to the complexity of the arrangement, we need to prove the following worst-case bound.

Lemma 3.3 *The complexity of the vertical decomposition of a set of n triangles in $\mathbb{R}^3$ is $O(n^3)$.*

Proof: As observed before, it suffices to count the number of visibilities between intersection edges. Let $\boldsymbol{\Xi} = \Xi_0, \ldots, \Xi_k$ be an efficient hierarchical $(1/n)$-cutting for the set $H(T)$ of planes containing the triangles in T. For a simplex $s \in \Xi_i$, let $T(s) \subset T$ be the set of triangles intersecting the interior of s.

We wish to bound the number of vertical visibilities between two intersection edges inside the single "simplex" $s \in \Xi_0$. Consider the vertical line segment that connects two intersection edges that can see each other. There are two possibilities: this segment intersects a (non-vertical) facet of some simplex $s' \in \Xi_1$, or it is completely contained in some simplex. From Lemma 3.2 it follows that the total number of visibilities crossing a single facet is $O(n^2 2^{\alpha(n)}\log n)$. (This is true even though there can be triangles penetrating the facet: such triangles can be partitioned into a triangle and a quadrilateral (which can be further decomposed into two triangles) that each lie entirely on one side of the facet.) To obtain the total number of such visibilities we simply sum over all facets of simplices in Ξ_1. We are left with the visibilities of the second type, where the connecting segment is contained in simplex $s' \in \Xi_1$. In this case all four triangles involved are elements of the set $T(s')$, so we can count them recursively.

Thus we can count the total number of visibilities by summing over all simplices s in the hierarchical cutting the values $O(n_s^2 2^{\alpha(n_s)}\log n_s)$, where $n_s := |T(s)|$. To simplify the calculations below (and without influencing the final result) we first replace $O(n_s^2 2^{\alpha(n_s)}\log n_s)$ by the crude upper bound of $O(n_s^{2.5})$.

$$\begin{aligned}
\textstyle\sum_s O(n_s^2 2^{\alpha(n_s)}\log n_s) &= \textstyle\sum_s O(n_s^{2.5}) \\
&= \textstyle\sum_{0\le i\le k}\sum_{s\in\Xi_i} O(n_s^{2.5}) \\
&= \textstyle\sum_{0\le i\le k} O(\rho^{3i})\cdot O((n/\rho^i)^{2.5}) \\
&= n^{2.5}\textstyle\sum_{0\le i\le k} O(\rho^{i/2}) \\
&= n^{2.5} O(\rho^{k/2}) = O(n^3)\ .
\end{aligned}$$

□

We are almost ready to prove the main result of this section. The only remaining ingredient that we need is the following.

Lemma 3.4 *Let T be a set of n triangles in $\mathbb{R}^3$ with $|\mathcal{A}(T)| = K$, and let r be a parameter with $1 \leq r \leq n$. There exists an $O(\log r/r)$-cutting of size $O(r^2\alpha(r) + Kr^3/n^3)$ for T.*

Proof: Let $R \subset T$ be a random sample of size r. By the Slicing Theorem [4] we can triangulate $\mathcal{A}(R)$ into $O(r^2\alpha(r) + |\mathcal{A}(R)|)$ simplices. From ε-net theory [26] it follows that with high probability each simplex in the triangulation will be intersected by $O(n \log r/r)$ triangles in T. In other words, the triangulation will be an $O(\log r/r)$-cutting for T with high probability. The expected value of $|\mathcal{A}(R)|$ is $O(r^2 + A)$, where A is the expected number of triple intersections between triangles in R. Since the probability of a triple intersection in $\mathcal{A}(T)$ showing up in $\mathcal{A}(R)$ is $r(r-1)(r-2)/n(n-1)(n-2)$, the expected value of A is $Kr(r-1)(r-2)/n(n-1)(n-2)$. We have proved that the triangulation of a random sample $R \subset T$ of size r is an $O(\log r/r)$-cutting for T with high probability, and that its expected size is $O(r^2\alpha(r) + Kr^3/n^3)$. Hence, a sample with these properties must exist. □

Finally we can prove the main result of this section.

Theorem 3.2 *Let T be a set of n triangles in $\mathbb{R}^3$ with $|\mathcal{A}(T)| = K$. Then, for any constant $\varepsilon > 0$, the complexity of the vertical decomposition for T is $O(n^{2+\varepsilon} + K)$, where the constant of proportionality depends on ε.*

Proof: Define $f(n,m)$ to be the maximum number of visibilities between edges in $E(T)$ over all sets T of n triangles where m is the number of triple intersections of the triangles in T.

Let $\varepsilon > 0$ be given. Fix a large enough constant $r = r(\varepsilon)$. (In fact, r does not only depend on ε but also on the constants c_2 and c_3 defined below.) We shall prove inductively that there exists a constant $c = c(\varepsilon)$ such that $f(n,m) \leq cn^{2+\varepsilon} + crm$, which proves the theorem. (We choose c large enough so that this statement is true for small n.) The following two cases can arise:

- case (i): $m > n^3/r$

  By Lemma 3.3 we know that there exists a constant c_1 such that $f(n,m) \leq c_1 n^3$. It readily follows that $f(n,m) \leq crm$, for a constant $c > c_1$.

- case (ii): $m \leq n^3/r$

  We proceed in about the same way as in the proof of Lemma 3.3. The difference is that we now use the "complexity-sensitive" cuttings of Lemma 3.4 instead of Chazelle's efficient hierarchical cuttings. Because $m \leq n^3/r$, Lemma 3.4 implies that there exists a $(c_2 \log r/r)$-cutting Ξ for T of size $c_3 r^2\alpha(r)$, for some constants c_2, c_3. Let $T(s) \subset T$ denote the subset of triangles that intersect the interior of a simplex $s \in \Xi$. As usual we work with clipped triangles: for a triangle $t \in T(s)$ we work with the part $t \cap s$. This part is a convex polygon with at most six vertices, so we can triangulate it into at most four triangles. So n_s, the number of triangles in $T(s)$, is at most $4c_2 n \log r/r$.

As before, there are two types of visibilities between intersection edges of the triangles in T: visibilities such that there is a set $T(s)$ that contains all four triangles involved, and visibilities where there is no such set $T(s)$. The first type of visibilities is counted recursively. For visibilities of the second type we again observe that the segment connecting the two intersection edges must cross a facet of some simplex $s \in \Xi$. By Lemma 3.2 the number of such visibilities for a fixed facet is at most $c_4 n^2 2^{\alpha(n)} \log n$, for some constant c_4.

Define m_s to be the number of triple intersections in $\mathcal{A}(T(s))$. Notice that $\sum_{s \in \Xi} m_s \leq m$. There are $4c_3 r^2 \alpha(r)$ facets of simplices in Ξ. Hence, for n and r large enough, we can bound $f(n,m)$ as follows:

$$
\begin{aligned}
f(n,m) &\leq 4c_3 r^2 \alpha(r) c_4 n^2 2^{\alpha(n)} \log n \ + \ \textstyle\sum_{s \in \Xi} f(n_s, m_s) \\
&\leq 4c_3 r^2 \alpha(r) c_4 n^2 2^{\alpha(n)} \log n \ + \ \textstyle\sum_{s \in \Xi} [c n_s^{2+\varepsilon} + crm_s] \\
&\leq 4c_3 r^2 \alpha(r) c_4 n^2 2^{\alpha(n)} \log n \ + \ \textstyle\sum_{s \in \Xi} c(4c_2 n \log r/r)^{2+\varepsilon} \ + \ crm \\
&\leq cn^{2+\varepsilon}\left[\tfrac{4c_3 r^2 \alpha(r) c_4 2^{\alpha(n)} \log n}{n^\varepsilon} + \tfrac{c_3 \alpha(r)(4c_2 \log r)^{2+\varepsilon}}{r^\varepsilon}\right] \ + \ crm \\
&\leq cn^{2+\varepsilon} + crm.
\end{aligned}
$$

□

3.2 A Single Cell

It turns out that both the lower bound proof and the upper bound proof that we gave for the full arrangement can easily be adapted to the case of a single cell. The lower bound construction in the proof of Theorem 3.1 in fact resulted in a single cell whose vertical decomposition has complexity $\Omega(n^2\alpha^2(n))$. To prove an upper bound we note that a single cell in an arrangement of n triangles can be decomposed into $O(n^2 \log n)$ simplices; this follows from the Slicing Theorem and a bound on the complexity of a single cell [5]. We designate a single three-dimensional cell in the arrangement by a point π in its interior. Using ε-net theory (as in the proof of Lemma 3.4) we see that there is a sample $R \subset T$ of size r such that each of the $O(r^2 \log r)$ simplices in the decomposition of the single cell defined by R and containing the designating point π is intersected by $O(n \log r/r)$ triangles in T. We can now follow the proof of Theorem 3.2 almost verbatim. The only difference is that in the inductive analysis in the proof we now have to recurse only on the $O(r^2 \log r)$ simplices of the single cell in $\mathcal{A}(R)$. In each subproblem we still deal with a single cell, so we get for $g(n)$, the maximum number of vertical visibilities between intersection edges in a single cell defined by n triangles, the following recurrence:

$$g(n) \leq c_1 r^2 \log r \cdot n^2 2^{\alpha}(n) \log n + c_2 r^2 \log r \cdot g(c_3 n \log r/r),$$

where c_1, c_2, and c_3 are suitable constants. This leads to the following result.

Theorem 3.3 *The maximum combinatorial complexity of the vertical decomposition of a single cell in an arrangement of n triangles in 3-space is $\Omega(n^2\alpha^2(n))$ and, for any constant $\varepsilon > 0$, it is $O(n^{2+\varepsilon})$, where the constant of proportionality depends on ε.*

Recently de Berg et al. [18] described a simple randomized incremental algorithm to compute a single cell in an arrangement of triangles. Their algorithm uses vertical decompositions, and its running time is $O(g(n) \log n)$, where $g(n)$ is the maximum complexity of the vertical decomposition of a single cell. From our results it follows that, for any $\varepsilon > 0$, their algorithm runs in time $O(n^{2+\varepsilon})$.

4 The Algorithm

In this section we present a deterministic algorithm for constructing the vertical decomposition $\mathcal{V}_3(T)$ of the arrangement $\mathcal{A}(T)$ induced by a set of triangles T. Our algorithm constructs the decomposition in stages corresponding to the partial decompositions described in the Introduction.

Let $T = \{t_1, \ldots, t_n\}$ be a collection of triangles as defined in Section 2. To simplify the description of our algorithm, we assume that the entire collection T of triangles is contained inside a *bounding simplex*, whose faces are special triangles in T. They are special in the sense that they violate the general position assumption. Also, unlike all the other triangles in T, we are interested in only one side of each of these four triangles (the side that faces the interior of the simplex).

As mentioned in the introduction, we will consider a decomposition carried out in three steps: first we extend a vertical wall from every boundary edge of any triangle in T, thus we obtain $\mathcal{V}_1(T)$; in addition, we extend a vertical wall from every intersection edge of two triangles in T, and we get $\mathcal{V}_2(T)$; and finally we refine the subcells of $\mathcal{V}_2(T)$ into constant size subcells, to produce $\mathcal{V}_3(T)$. In the first two steps of the algorithm we shall compute vertices and edges of the decompositions, and only at the final step we will create a representation that puts everything together: vertices, edges, faces and three-dimensional cells.

The data structure that we obtain after carrying out the entire algorithm provides a comprehensive and convenient representation of the arrangement: each subcell in this representation has constant description complexity—it is bounded by up to six quadrilateral walls, it is homeomorphic to a ball, and the structure provides connectivity information between adjacent cells across vertical walls (see the end of Subsection 4.3 for more details on the connectivity issue). A significant advantage of the one-step decomposition $\mathcal{V}_1(T)$ is that in return for a relatively low overhead it already captures the three-dimensional structure of the arrangement and it makes the next steps of the algorithm fairly simple.

4.1 Computing the Features of $\mathcal{V}_1(T)$

As before, we denote the complexity of the arrangement $\mathcal{A}(T)$ by K. We start with computing the one-step decomposition $\mathcal{V}_1(T)$ of the arrangement. In order to compute the features of $\mathcal{V}_1(T)$, we need to compute the vertical wall $W(e,T)$ for each boundary edge e of a triangle in T (in addition to the features of $\mathcal{A}(T)$). In Observation 3.1 we have already seen that the complexity of all these walls is $O(n^2\alpha(n))$.

Lemma 4.1 *The set of walls $W(e,T)$ for all boundary edges e of the triangles in T can be computed in $O(n^2 \log n)$ time.*

Proof: Recall that Observation 3.1 was based on the fact that each wall $W(e,T)$ is bounded from above (below) by the lower (upper) envelope of the intersections of other triangles with the vertical flat surface $H(e)$; see Figure 1(b). This fact also implies that we can compute a single wall in $O(n \log n)$ time [28]. Since there are $3n$ walls to compute, the time bound follows. □

We define a pair of two-dimensional arrangements of segments for each triangle t_i. We consider each triangle to be two-sided, and let t_i^- denote the side of t_i facing downward and t_i^+ be the side of t_i facing upward. We use the notation t_i^* to refer to either side of t_i. For the bounding simplex, we need only one side of each triangle—the side facing the internal part of the simplex. The arrangement on t_i^+ is defined by a set $\Gamma(t_i^+)$ consisting of two types of segments: intersections of t_i with other triangles of T and boundary pieces of walls $W(e,T)$ (for boundary edges e of triangles in T) that lie on t_i^+. Segments in $\Gamma(t_i^+)$ of the second type, in other words, are each the contribution of t_i to the lower envelope that bounds some wall $W(e,T)$ from below. Similarly, the arrangement on t_i^- is defined by the set of segments $\Gamma(t_i^-)$ consisting of intersections of t_i with other triangles of T, and boundary pieces of walls $W(e,T)$ (again, for boundary edges e of triangles in T) that lie on t_i^-. To each segment in $\Gamma(t_i^*)$ we attach a label denoting its origin: either the label of the other triangle intersecting t_i or the edge e on whose envelope it appears. Let $n_i^* := |\Gamma(t_i^*)|$. We use the abbreviation $\mathcal{A}_i^*$ to denote the two-dimensional arrangement $\mathcal{A}(\Gamma(t_i^*))$.

The next step of the algorithm is to compute, for each side of each triangle $t_i \in T$, the arrangement induced by $\Gamma(t_i^*)$. To this end we use a standard plane sweep algorithm that detects all the intersection points between segments[4] [33]. (At this

[4]From this point on, when discussing two-dimensional arrangements, we will distinguish between a segment and an edge. A *segment* is a maximal portion of a line that appears in the arrangement, possibly intersected by other segments. An *edge* is a maximal portion of a segment not intersected by any other segment.

point one may use the optimal algorithm by Chazelle and Edelsbrunner [9]. However, due to other steps of the algorithm, this will not make a difference in the overall running time of the algorithm. Also, the sweep paradigm is more appropriate for the extension that we present below to handling arrangements of surface patches.

Lemma 4.2 *After having computed all walls of triangle boundary edges, all the two-dimensional arrangements $\mathcal{A}_i^*$ can be computed in time $O(|\mathcal{V}_1(T)| \log n)$.*

Proof: In order to compute all the two-dimensional arrangements $\mathcal{A}_i^*$ we need to know all the segments resulting from pairwise intersection of triangles, and all the segments in $\Gamma(t_i^*)$ that are contributing to wall boundaries. This information is immediately available from computing all the walls of triangle boundary edges. For the latter type of segments this is clear. For the former type we note that every endpoint of an intersection segment must be the intersection of a triangle boundary edge with a triangle; hence, it is a feature of a wall boundary. Next, given the collection of n_i^* segments defining the arrangement $\mathcal{A}_i^*$, the algorithm for computing the arrangement runs in time $O((n_i^* + k_i^*) \log n)$, where k_i^* is the number of intersections of segments in the arrangement. Clearly, the sum of $O(n_i^* + k_i^*)$ over all the arrangements on the triangles is bounded by $O(|\mathcal{V}_1(T)|)$. □

What we actually need for the next steps of the algorithm is the collection of vertices of all these arrangements, ordered by increasing x order, together with additional local information for each vertex, including the edge(s) that it bounds. The ordering plus the additional information can also be computed in $O(|\mathcal{V}_1(T)| \log n)$ time.

After the first step of the decomposition, the entire three-dimensional arrangement is connected, that is, there are no "floating" parts. In particular, the two-dimensional boundary of each three-dimensional cell is connected. The shape of a three-dimensional cell in $\mathcal{V}_1(T)$ may still be rather convoluted and in particular it need not even be xy-monotone, as is illustrated in Figure 4. Therefore, instead of handling each three-dimensional cell at a time we carry out a space sweep over the entire decomposition $\mathcal{V}_1(T)$. A similar approach has recently been used to obtain a decomposition of certain arrangements related to a motion planning problem [24]; see also [22, 34] for dynamic maintenance of a monotone subdivision in a space-sweep.

Let P_{x_1} denote the plane $x = x_1$. Let $\mathcal{A}_{x_1} := \mathcal{A}(P_{x_1} \cap \mathcal{V}_1(T))$ be the two-dimensional arrangement of segments induced on the plane P_{x_1} by intersecting it with[5] $\mathcal{V}_1(T)$. We use $\mathcal{A}_x$ to denote this arrangement for an arbitrary x-value. We need the following property of $\mathcal{V}_1(T)$.

Lemma 4.3 *Every face in $\mathcal{A}_x$ is convex.*

[5] With a slight abuse of notation, we use $\mathcal{V}_1(T)$ to denote the subdivision of 3-space after the one-step decomposition, and also to denote the collection of triangles and walls that induce this subdivision.

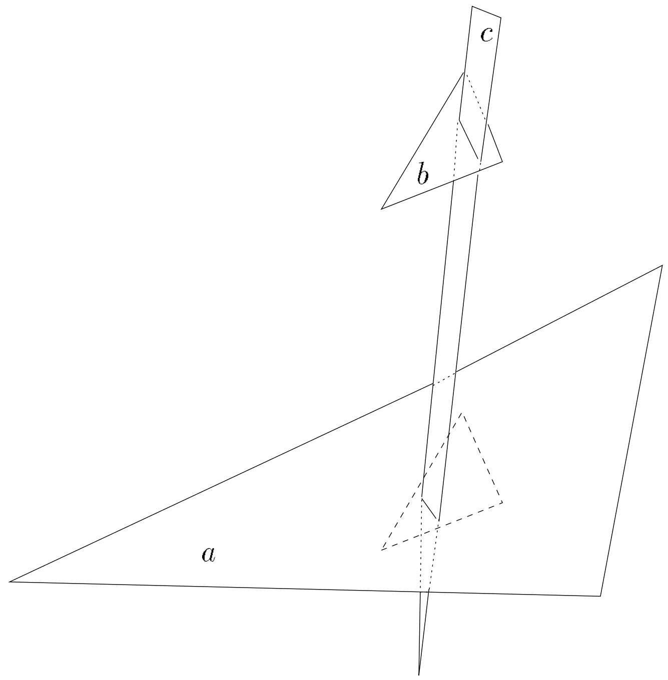

Figure 4: The cylindrical cell of $\mathcal{V}_1(T)$ above a and below b is not xy-monotone

Proof: Let S_x be the set of segments that are the intersection of P_x with $\mathcal{V}_1(T)$. S_x contains two types of segments: intersections of P_x with triangles in T, and intersections with vertical walls erected for the triangle boundary edges. The endpoint of a segment of the first type is the intersection of P_x with a triangle boundary edge. Hence, there must be a segment of the second type incident to this vertex; this is the vertical segment of maximal length through the endpoint that does not intersect the interior of any segment in S_x. Thus any vertex of $\mathcal{A}_x$ is either an intersection of two segments, or a T-junction where one segment endpoint lies on another segment. It follows that every face of $\mathcal{A}_x$ must be convex. $\square$

4.2 Computing the Features of $\mathcal{V}_2(T)$

The major difficulty in computing the features of $\mathcal{V}_2(T)$ is to efficiently detect the vertical visibilities of pairs of intersection edges, one on the ceiling of a cell of $\mathcal{V}_1(T)$ and the other on the floor of that cell. To this end we perform a space sweep with a plane P_x parallel to the yz-plane over $\mathcal{V}_1(T)$, from $x = -\infty$ to $x = +\infty$. Throughout the sweep we maintain dynamic data structures that describe $\mathcal{A}_x$—below we will indicate which data structures we need. Roughly, our goal is to subdivide each face of any arrangement $\mathcal{A}_i^+$ with the vertical projection of all the edges of ceiling faces that are visible when looking vertically upward from that face. Similarly, we wish

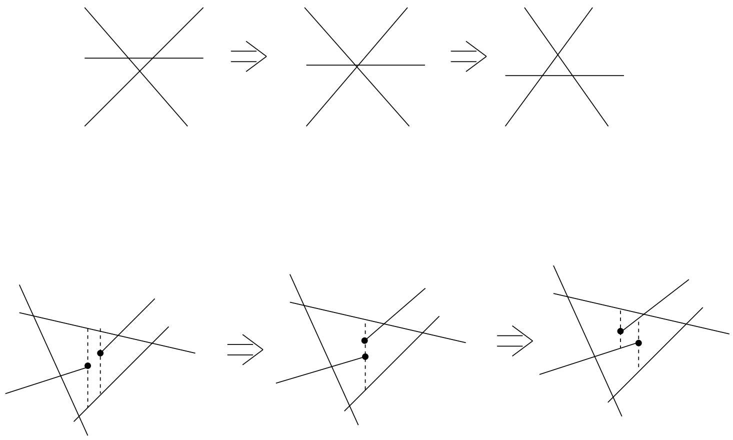

Figure 5: Examples of changes to $\mathcal{A}_x$ when sweeping over a vertex of $\mathcal{V}_1(T)$.

to subdivide each face of any arrangement $\mathcal{A}_i^-$ with the vertical projection of all the edges of floor faces that are visible when looking vertically downward from that face.

The arrangement $\mathcal{A}_x$ changes continuously as we sweep the plane P_x. At a finite number of "events", however, the combinatorial structure of $\mathcal{A}_x$ changes; these events are exactly the vertices of $\mathcal{V}_1(T)$. At such an event the following changes to $\mathcal{A}_x$ can occur: (i) a vertex of $\mathcal{A}_x$ may (dis)appear; (ii) an edge of $\mathcal{A}_x$ may (dis)appear; and (iii) a face of $\mathcal{A}_x$ may (dis)appear. In fact, several such changes occur simultaneously at each event. Two examples are given in Figure 5. By the general position assumption, each event involves only a constant number of changes to a constant number of faces in $\mathcal{A}_x$.

Since our goal is to detect vertical visibilities between two (intersection) edges in $\mathcal{V}_1(T)$, we wish to detect when there is a vertical segment connecting two intersection edges that does not intersect any triangle in T. Consider the moment when the sweep plane P_x contains such a vertical segment connecting intersection edges e_1 and e_2. At that moment the intersection of e_1 and e_2 with P_x must define two vertices of $\mathcal{A}_x$ that are on the boundary of the same face and lie on a common vertical line. Thus we define a new type of event in our space sweep, called a *vertical* event, which occurs when two vertices of the same face become aligned along a vertical line.

We maintain all the events in a priority queue $\mathcal{Q}$, ordered by increasing x-coordinate. The operations we will perform on $\mathcal{Q}$ are *insert* an event, *delete* an event, and *fetch* the next event, that is, fetch the event with minimum x-value (we delete each event after it has been fetched and handled). We also need to perform membership check of an event in the queue, to avoid inserting the same event several times. Such a queue can be implemented so that the time for each operation is $O(\log m)$, where m is the maximum number of events held simultaneously in the queue [33]. To each event that we insert into the queue, we attach the local geometric and combinatorial information relevant to that event. Observe that we can insert all the events where the structure of $\mathcal{A}_x$ changes—the vertices of $\mathcal{V}_1(T)$—into $\mathcal{Q}$ before we start the sweep. The vertical events, however, have to be computed on the fly. Next we describe the data structures we need to represent $\mathcal{A}_x$ in order to be able to compute the vertical events.

Let $f = f(x)$ be a face of $\mathcal{A}_x$. Recall that f is convex. We split the vertices on the boundary of f into two y-monotone chains: the lower chain $L(f)$, and the upper chain $U(f)$. The vertices in each chain are ordered by increasing y-coordinate, and we implement each chain as a balanced binary tree. See Figure 6. Whenever a vertex on the boundary of f disappears or newly appears, we update the relevant chain by a deletion or an insertion, respectively. When an existing face disappears, we "free" its attached chains. When a face newly appears, we allocate a pair of new (trivial) structures for its lower and upper chains. When a face f is split into several faces (for instance when an endpoint of a segment penetrates the face—inducing three new faces) we create new pairs of chains for these faces. These are easily constructed by splitting the chains of f and adding a small number of new vertices. Similarly, when two faces merge into one, we create the two chains of the new face by *join* operations on the chains of the merged faces plus possibly a few deletions of vertices. All these operations (insertion, deletion, split and join) can be carried out in $O(\log n)$ time each, using, e.g., red-black trees [23], [36, Chapter 4].

To detect vertical visibilities between vertices on the boundary of f we proceed as follows. Whenever a new vertex v is created on the lower chain of f (the operations for a new vertex on the upper chain are symmetric), we look for its "neighbors" in the upper chain of f, that is, we look for the two vertices of $U(f)$ whose projection onto the y-axis lie nearest to the projection of v onto the y-axis (see Figure 6 for an illustration). Let u_1 and u_2 be the two neighbors of v on the upper chain. Each vertex w on the boundary of f is the cross-section of P_x with an edge $e(w)$ of some $\mathcal{A}_i^*$. Since we have attached to each vertex some additional information, we can compare $e(v)$ with each of $e(u_1)$ and $e(u_2)$ to see if their projections onto the xy-plane intersect. If they intersect, we have detected a potential vertical visibility between two intersection edges; we add this event to $\mathcal{Q}$, with the x-coordinate of the intersection point. We say *potential* because when we will come to handle this event, we may find that this "vertical visibility" is obscured by another triangle (or triangles) in a manner that we were unable to predict when the event was inserted into the queue. Therefore, we distinguish between two types of events: *actual* events, which are either the events corresponding to features of $\mathcal{V}_1(T)$ or additional events that indeed correspond to a vertical visibility, and *false* events, which are potential vertical visibilities that are found out to be obscured when handled.

When we handle an actual event q that corresponds to vertical visibility of a pair of vertices u and v on the upper and lower chains respectively, each of u and v now have new neighbors on the opposite chain, so we check these new neighbors for additional potential events as above.

The next lemma proves the correctness of our approach.

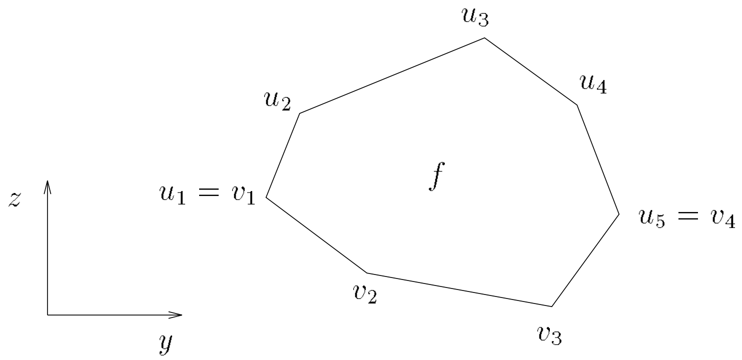

Figure 6: A face f and its two chains: the upper chain $U(f) = \{u_1, u_2, u_3, u_4, u_5\}$ and the lower chain $L(f) = \{v_1, v_2, v_3, v_4\}$. The *neighbors* of v_2 on the upper chain are the vertices u_2 and u_3.

Lemma 4.4 *The set of events that is computed during the sweep contains the set of visibilities between pairs of intersection edges. The false events can be efficiently distinguished from the actual events. Each false event can be charged to an actual event, and no actual event gets charged more than a constant number of times this way.*

Proof: First we claim that no vertical visibility is missed by our algorithm. The reason is that before a pair of edges of $\mathcal{V}_1(T)$ become vertically visible, their corresponding cross-sections must become neighbors (in the sense defined above) in $L(f)$ and $U(f)$ for some face f of $\mathcal{A}_x$. This is similar to a basic argument in the Bentley-Ottmann algorithm for detecting intersections of line segments [6], [33, Section 7.2].

Once we handle a vertical visibility event it is easy to determine whether it is actual or false by checking whether the lower and upper chains involved in this event belong to the same face. The event is actual if and only if both chains belong to the same face. This could be done, for example, by maintaining cross pointers between the roots of the trees describing the lower and upper chains of a face.

We charge each false event q' to the actual event q that has "spawned" it. Every vertical event is added to the queue $\mathcal{Q}$ only at actual events; a false event does not create new events. Since no event creates more than a constant number of additional events, no actual event will be charged more than a constant number of times for false events. $\square$

As in the previous stage, we are not yet interested in actually representing the decomposition $\mathcal{V}_2(T)$, but only in collecting all the features (vertices and edges) of this decomposition for the third and final step. So we augment each two-dimensional arrangement $\mathcal{A}_i^*$ with information on all the edges that are vertically visible from $\mathcal{A}_i^*$. More precisely, we add to $\mathcal{A}_i^+$ all the edges that are visible when looking vertically upward from t_i^+. Similarly, we add to $\mathcal{A}_i^-$ all the edges that are visible when looking vertically downward from t_i^-. We also record on these arrangements the intersection of these edges with the original edges of $\mathcal{A}_i^*$. These intersection points correspond exactly to the events computed during the space sweep.

Lemma 4.5 *The space sweep algorithm described above for computing all the features of $\mathcal{V}_2(T)$ runs in time $O(V \log n)$, where $V = |\mathcal{V}_2(T)|$.*

Proof: We have seen that every event can be handled with a constant number of operations on some chains $U(f)$ and $L(f)$, and on $\mathcal{Q}$. Thus every event takes $O(\log n)$ time. It remains to observe that every event is either a feature of $\mathcal{V}_2(T)$ or, as shown in Lemma 4.4, it can be charged to a feature of $\mathcal{V}_2(T)$ such that no feature of $\mathcal{V}_2(T)$ gets charged more than a constant number of times. □

Let $\mathcal{B}_i^*$ denote the arrangement $\mathcal{A}_i^*$ augmented with all the features of $\mathcal{V}_2(T)$ that appear on t_i^*. Every face of every $\mathcal{B}_i^*$ represents a vertical cylindrical cell that has the same xy-projection as that face, and that has a unique triangle bounding it on the top and a unique triangle bounding it on the bottom (one of which is t_i). A face of $\mathcal{B}_i^*$ may still be rather complex: it need not be simply connected, and it may have a large number of edges on its boundary.

4.3 Computing the Full Decomposition $\mathcal{V}_3(T)$

The final step of our algorithm is to refine each $\mathcal{B}_i^*$ further to obtain the full vertical decomposition, in the following standard manner. Consider the projection of $\mathcal{B}_i^*$ onto the xy-plane. We extend a y-vertical segment from each vertex of the projected arrangement upward and downward until it reaches another segment, or the boundary of the projection of t_i. In other words, we compute a trapezoidal decomposition of the arrangement. This can be carried out by a plane sweep of each of the arrangements $\mathcal{B}_i^*$. This will take $O(V \log n)$ time, where $V = |\mathcal{V}_2(T)|$.

The added segments are projected back to t_i^* to obtain a refinement of $\mathcal{B}_i^*$ into trapezoids. Finally, we extend each of these newly added segments in the z-direction into vertical walls inside the respective three-dimensional cells to obtain $\mathcal{V}_3(T)$. Thus the full vertical decomposition consists of vertical prisms, which are bounded by a pair a trapezoids (or triangles) that are connected by vertical walls.

We represent $\mathcal{V}_3(T)$ by a graph $\mathcal{G}$. The nodes in $\mathcal{G}$ correspond to the cells (vertical prisms) of the decomposition. With each node we store an explicit description of the prism it represents. There is an edge between two prisms if they share a vertical wall. Thus we get a complete "network" that allows us to navigate from one point in a cell of the arrangement $\mathcal{A}(T)$ to any other point in that cell. We cannot, however, go from one cell in $\mathcal{A}(T)$ to an adjacent cell, because $\mathcal{G}$ only stores connections through vertical walls, not through triangles. We will come back to this issue shortly.

We have shown how to compute a trapezoidal decomposition of the arrangements $\mathcal{B}_i^*$ on the triangles. Constructing the graph $\mathcal{G}$ is now an easy task. Every prism in $\mathcal{V}_3(T)$ is bounded from below by a trapezoid in one of the (decomposed) arrangements $\mathcal{B}_i^+$. For each of these trapezoids we create a node in $\mathcal{G}$. To be able to store an explicit description of the corresponding prism at each node we must also know the triangle in T that bounds the prism from above. We could compute those by sorting all the trapezoids on all arrangements $\mathcal{B}_i^-$ suitably, but this is not necessary: with a little extra bookkeeping in the previous steps of the algorithm we can make sure that we know for every vertex of some $\mathcal{B}_i^+$ the triangle that lies directly above it. The connections between the prisms, which define the arcs in $\mathcal{G}$, are also easily derived from the information we have computed. Hence, constructing the graph $\mathcal{G}$ can be done without extra (asymptotic) overhead.

We conclude that $\mathcal{V}_2(T)$ can be transformed into $\mathcal{V}_3(T)$ in $O(V \log n)$ time. The previous steps of the algorithm, computing $\mathcal{V}_1(T)$ and transforming $\mathcal{V}_1(T)$ to $\mathcal{V}_2(T)$, take $O(n^2 \log n + V \log n)$ time in total by Lemmas 4.1, 4.2, and 4.5. The main algorithmic result of the paper is thus as follows.

Theorem 4.1 *Given a collection T of n triangles in general position in three-dimensional space, the time needed to compute the full vertical decomposition of the arrangement $\mathcal{A}(T)$ is $O(n^2 \log n + V \log n)$, where V is the combinatorial complexity of the vertical decomposition.*

The connectivity information of cells that are adjacent through a vertical wall is supplied by our structure. There are applications where this connectivity information is all one needs. Such is the case, for example, if the triangles are the so-called constraint surfaces of a motion planning problem. In this situation, if we start navigating the arrangement from a point inside a free cell (representing a placement of the robot where it does not collide with any of the obstacles) then the constraint surfaces bound forbidden regions and may not be crossed; the path that we trace is only allowed to cross vertical walls.

The connectivity structure can be easily enhanced to allow for crossing from one cell to an adjacent cell provided both cells have faces on the same side of the same triangle such that these faces share a common edge.

There is, however, one type of connectivity that is not readily available from the computations carried out by our algorithm: An application may require to move from a point on the ceiling of a cell to the floor of the cell lying right above that point. There are various ways to augment our data structures to support such ceiling/floor crossings.

One idea is to merge each pair of subdivisions $\mathcal{B}_i^+$ and $\mathcal{B}_i^-$. But now the subdivisions on the floor and ceiling of a cell are no longer the same. Hence, the subdivisions must be propagated further. To avoid blowing up the complexity of our structure, the subdivisions must be properly coarsened during the propagation. So what we need is two-dimensional version of fractional cascading [13]. Currently we do not know how to achieve this—we leave this is an open problem for further research.

Another possible solution is to augment each arrangement $\mathcal{B}_i^*$ with a point location structure [33]. This solution will not increase the preprocessing time of our structure asymptotically, but will cost $O(\log n)$ time per ceiling/floor crossing.

A slight modification of our structure enables a certain "compromise" solution, with no increase in the asymptotic complexity of the structure or in the preprocessing time. Namely, we augment each vertex on the arrangement $\mathcal{B}_i^-$ with a pointer to the face lying above it in $\mathcal{B}_i^+$, and vice versa. That is, we do not enable a direct crossing from every point in a face of a ceiling to a point on the floor right above it, but we supply such a crossing from any vertex on the boundary of the face (and symmetrically for vertices on the boundary of each floor face, a direct crossing to the face on the ceiling right below it). We omit the simple technical details of how to add these pointers.

4.4 An Algorithm with Subquadratic Overhead

The overhead in the algorithm described above—the time we always need, irrespective of the complexity of the decomposition—is $O(n^2 \log n)$. A question that comes to mind is whether a quadratic overhead is really necessary. The answer is no. We have devised an algorithm that achieves a subquadratic overhead time. This algorithm is, however, substantially more complicated than the algorithm described above, and the savings are small. Therefore, we summarize the performance of the alternative algorithm in the following theorem, and refer the reader to a Technical Report [19] where a detailed description of the algorithm is given.

Theorem 4.2 *Let T be a collection of n triangles in general position in three-dimensional space. For any $\varepsilon > 0$, the full vertical decomposition of the arrangement $\mathcal{A}(T)$ can be computed in time $O(\min(n^{4/5+\varepsilon}V^{4/5}, n^2 \log n) + V \log n)$, where V is the combinatorial complexity of the vertical decomposition.*

4.5 Extension to Arrangements of Surface Patches

In this subsection, we extend the algorithm described in Subsections 4.1 through 4.3 (not the improved-overhead algorithm) to the case of arrangements of surface patches. The definition of vertical decompositions for arrangements of surfaces are similar to those for triangles; see [10],[14]. We adhere, as much as possible, to the notation set in the previous subsections.

Let $T = \{t_1, \ldots, t_n\}$ be a given collection of n surface patches in 3-space that satisfy the following conditions:

(i) Each t_i is monotone in the xy-direction (that is, every vertical line intersects t_i in at most one point). Moreover, each t_i is a portion of an algebraic surface of constant maximum degree.

(ii) The vertical projection of t_i onto the xy-plane is a planar region bounded by a constant number of algebraic arcs of constant maximum degree.

(iii) The surface patches in T are in *general position*; one way of defining this is to require that the coefficients of the polynomials defining the surfaces and their boundaries are algebraically independent over the rationals (i.e., no multivariate polynomial with rational coefficients vanishes when substituting into it some of the given coefficients), thereby excluding all kinds of 'degenerate' configurations; see [35] for more details.

We need to assume here a model of computation where each elementary operation on the surface patches required by the algorithm is performed in constant time. We can assume the model used for algorithms in real algebraic geometry [27], where each algebraic operation involving a constant number of polynomials of constant maximum degree can be performed exactly, using rational arithmetic, in constant time.

The key factors that allow for an easy extension of the algorithm presented earlier (in Subsections 4.1 through 4.3) to arrangements of surface patches are the following:

(i) Since each surface t_i is xy-monotone, we can still employ a central tool in our algorithm, namely, a planar line sweep algorithm, when handling a two-dimensional arrangement $\mathcal{A}_i^*$ or $\mathcal{B}_i^*$. We simply project the arrangement to the xy-plane.

(ii) The significant property required for the spatial sweep carried out to compute the features of $\mathcal{V}_2(T)$, namely that every face in the cross-sectional arrangement $\mathcal{A}_x$ is y-monotone, still holds (see below for more details).

We need to make a few simple adaptations in order for the algorithm to work in the case of surface patches almost as it is described for arrangements of triangles. One point to note is that we should consider any singular point (resp. curve) on the patch t_i as a vertex (resp. a not necessarily connected curve) on the corresponding arrangements $\mathcal{A}_i^+$ and $\mathcal{A}_i^-$. Also for every curve in any arrangement $\mathcal{A}_i^*$, we introduce a vertex for every point of y-vertical tangency of that curve (more precisely, we add a vertex on the curve γ at every point whose projection onto the xy-plane is a y-vertical tangency point of the projection of γ onto that plane). The reason for this adjustment is that when we perform our space sweep with a plane parallel to the yz-plane, those added vertices are locally the first (or last) point of that curve that the sweep plane intersects.

A *wall* in the current situation is defined for every maximal connected portion γ of the boundary of a surface patch in T, that belongs to a single algebraic curve. We intersect $H(\gamma)$ with all the other surfaces in T, and consider the lower and upper envelopes of the resulting curves relative to the curve γ. The overall number of curves on $H(\gamma)$ is clearly $O(n)$ and by standard arguments the complexity of the envelopes is $O(\lambda_q(n))$ for some constant q that depends on the algebraic degree of the surface patches and their boundaries. The time to compute each envelope is $O(\lambda_q(n) \log n)$ which can be done either by a straightforward divide and conquer, or, for a small saving in the constant q above, by the algorithm of Hershberger mentioned earlier [28]. Computing all the $O(n)$ walls, therefore, will take $O(n\lambda_q(n)\log n)$ for some constant q.

We also need to extend Lemma 4.3 to this more general setting.

Lemma 4.6 *Every face in $\mathcal{A}_x$ is y-monotone, that is, every z-vertical line intersects any face of $\mathcal{A}_x$ in at most one connected component.*

Proof: Intersecting the plane P_x with the patches in T leads to an arrangement of y-monotone curves on that plane. Let S denote this set of curves. The additional walls added at the first step, when intersected with P_x, give maximal vertical segments through the endpoints of the curves in S and not intersecting other curves of S. Consider a face f of $\mathcal{A}_x$. Let y^- be the y coordinate of the point with smallest y value on the boundary of the face f. This y value is attained either by a meeting point of two curves in S, or by a vertical extension through an endpoint of a curve in S (assuming general position). In both cases there is a point $z^+(y)$ that lies on a curve of S such that locally, and slightly to the right of y^- bounds f from above, and similarly a point $z^-(y)$ that bounds the face f from below there. Similarly, let y^+ be the y coordinate of the point with maximum y value on the boundary of the face f; which, by the same assumption, is attained either by a point or by a vertical extension.

Next, suppose we start advancing the points $z^+(y)$ and $z^-(y)$ from $y = y^-$ and to the right simultaneously (so that they are z-vertically aligned at all times) and staying on the boundary of f. Namely if we reach an intersection of a curve with another we switch to the other curve that now bounds f. We stop the process at the earliest (in y) of the following events: (i) we have reached $y = y^+$; or (ii) $z^+(y)$ and $z^-(y)$ stop seeing one another. Since we proceed only as long as $z^+(y)$ and $z^-(y)$ are visible to one another, we can maintain the requirement that they are z-vertically aligned.

If we stop because of (i), this means that throughout the way $z^+(y)$ and $z^-(y)$ are visible to one another, and therefore the face is y-monotone. If we stop because $z^+(y)$ and $z^-(y)$ stop seeing one another, then it must be the case that the intrusion is due to the left endpoint of a curve of S, which induces a vertical segment there. But then necessarily the union of segments $\overline{z^+(y)z^-(y)}$ for y ranging from y^- through the point where we stopped constitute the face f and therefore we must have also reached $y = y^+$, and f is evidently y-monotone in this case. $\square$

In summary, we have the following result

Theorem 4.3 *Given a collection T of n xy-monotone algebraic surface patches of constant maximum degree, bounded by algebraic curves of constant maximum degree as well, and in general position in three-dimensional space, the time needed to compute the full vertical decomposition of the arrangement $\mathcal{A}(T)$ is $O(n\lambda_q(n)\log n +$*

$V \log n$), where V is the combinatorial complexity of the vertical decomposition, and q is a constant that depends on the degree of the surface patches and their boundaries.

5 Conclusions and Open Problems

We have proved bounds on the maximum combinatorial complexity of the vertical decomposition of sets of triangles in 3-space that are sensitive to the complexity of the arrangement of triangles. We have also given a simple deterministic output-sensitive algorithm for computing the vertical decomposition, and we have extended the algorithm to handle surface patches.

The paper raises several open problems:

- The generalization of the combinatorial result to the case of surface patches, namely, to obtain a bound on the vertical decomposition of an arrangement of surface patches that is sensitive to the size of the arrangement; this seems to be a major open problem in the study of general arrangements.
- Tightening the small gap that remains between the upper and lower bounds for vertical decompositions. We believe that the n^{ε}-factor in the upper bound is only an artifact of our proof technique and that the lower bound is closer to the truth than the upper bound.
- Towards the end of Subsection 4.3, we have discussed the issue of navigating around the decomposed arrangement. An interesting problem that arises in this respect is to devise an efficient decomposition (i.e., a decomposition with a small number of cells) with a constant number of neighbors per cell.
- Is it possible to adapt the algorithm of Subsections 4.1 through 4.3 to efficiently compute the vertical decomposition of a single cell in a three-dimensional arrangement?

References

[1] P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. *SIAM J. Comput.*, 22(4):794–806, 1993.

[2] P. K. Agarwal and J. Matoušek. On range searching with semialgebraic sets. *Discrete Comput. Geom.*, 11:393–418, 1994.

[3] P. K. Agarwal, M. Sharir, and P. Shor. Sharp upper and lower bounds on the length of general Davenport-Schinzel sequences. *J. Combin. Theory Ser. A*, 52:228–274, 1989.

[4] B. Aronov and M. Sharir. Triangles in space or building (and analyzing) castles in the air. *Combinatorica*, 10(2):137–173, 1990.

[5] B. Aronov and M. Sharir. Castles in the air revisited. *Discrete Comput. Geom.*, 12:119–150, 1994.

[6] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28:643–647, 1979.

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

[8] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9:145–158, 1993.

[9] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *J. ACM*, 39:1–54, 1992.

[10] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. A singly-exponential stratification scheme for real semi-algebraic varieties and its applications. In *Proc. 16th Internat. Colloq. Automata Lang. Program.*, volume 372 of *Lecture Notes in Computer Science*, pages 179–192. Springer-Verlag, 1989.

[11] B. Chazelle, H. Edelsbrunner, L. J. Guibas, and M. Sharir. Lines in space: combinatorics, algorithms, and applications. In *Proc. 21st Annu. ACM Sympos. Theory Comput.*, pages 382–393, 1989.

[12] B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10(3):229–249, 1990.

[13] B. Chazelle and L. J. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1:133–162, 1986.

[14] K. Clarkson, H. Edelsbrunner, L. Guibas, M. Sharir, and E. Welzl. Combinatorial complexity bounds for arrangements of curves and spheres. *Discrete Comput. Geom.*, 5:99–160, 1990.

[15] K. L. Clarkson. A randomized algorithm for closest-point queries. *SIAM J. Comput.*, 17:830–847, 1988.

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

[17] M. de Berg. *Ray Shooting, Depth Orders and Hidden Surface Removal*, volume 703 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1993.

[18] M. de Berg, K. Dobrindt, and O. Schwarzkopf. On lazy randomized incremental construction. In *Proc. 26th Annu. ACM Sympos. Theory Comput.*, pages 105–114, 1994.

[19] M. de Berg, L. J. Guibas, and D. Halperin. Vertical decompostions for triangles in 3-space. Technical Report UU-CS-1994-29, Utrecht University, 1994.

[20] M. de Berg, D. Halperin, M. Overmars, J. Snoeyink, and M. van Kreveld. Efficient ray shooting and hidden surface removal. *Algorithmica*, 12:30–53, 1994.

[21] H. Edelsbrunner. The upper envelope of piecewise linear functions: tight complexity bounds in higher dimensions. *Discrete Comput. Geom.*, 4:337–343, 1989.

[22] M. Goodrich and R. Tamassia. Dynamic trees and dynamic point location. In *Proc. 23rd Annu. ACM Sympos. Theory Comput.*, pages 523–533, 1991.

[23] L. J. Guibas and R. Sedgewick. A dichromatic framework for balanced trees. In *Proc. 19th Annu. IEEE Sympos. Found. Comput. Sci.*, Lecture Notes in Computer Science, pages 8–21, 1978.

[24] D. Halperin and M. Sharir. Near-quadratic bounds for the motion planning problem for a polygon in a polygonal environment. In *Proc. 34th Annu. Sympos. on Foundations of Computer Science*, pages 382–391, 1993.

[25] S. Hart and M. Sharir. Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes. *Combinatorica*, 6:151–177, 1986.

[26] D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete Comput. Geom.*, 2:127–151, 1987.

[27] J. Heintz, T. Recio, and M.-F. Roy. Algorithms in real algebraic geometry and applications to computational geometry. In J.E. Goodman, R. Pollack, and W. Steiger, editors, *Discrete and Computational Geometry: Papers from the DIMACS Special Year*, volume 6 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 137–163. Amer. Math. Soc., 1991.

[28] J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Inform. Process. Lett.*, 33:169–174, 1989.

[29] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete Comput. Geom.*, 10:157–182, 1993.

[30] K. Mulmuley. Hidden surface removal with respect to a moving point. In *Proc. 23rd Annu. ACM Sympos. Theory Comput.*, pages 512–522, 1991.

[31] K. Mulmuley. Randomized multidimensional search trees: further results in dynamic sampling. In *Proc. 32nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 216–227, 1991.

[32] M. Overmars and M. Sharir. Output-sensitive hidden surface removal. In *Proc. 30th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 598–603, 1989.

[33] F. P. Preparata and M. I. Shamos. *Computational Geometry: an Introduction.* Springer-Verlag, New York, NY, 1985.

[34] F. P. Preparata and R. Tamassia. Efficient point location in a convex spatial cell-complex. *SIAM J. Comput.*, 21:267–280, 1992.

[35] M. Sharir. Almost tight upper bounds for lower envelopes in higher dimensions. *Discrete Comput. Geom.*, 12:327–345, 1994.

[36] R. E. Tarjan. *Data Structures and Network Algorithms.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.

[37] A. Wiernik and M. Sharir. Planar realizations of nonlinear Davenport-Schinzel sequences by segments. *Discrete Comput. Geom.*, 3:15–47, 1988.