Computing Depth Orders and Related Problems*

Pankaj K. Agarwal† Matthew J. Katz‡ Micha Sharir§

Abstract

Let $\mathcal{K}$ be a set of n non-intersecting objects in 3-space. A depth order of $\mathcal{K}$, if exists, is a linear order $<$ of the objects in $\mathcal{K}$ such that if $K, L \in \mathcal{K}$ and K lies vertically below L then $K < L$. We present a new technique for computing depth orders, and apply it to several special classes of objects. Our results include: (i) If $\mathcal{K}$ is a set of n triangles whose xy-projections are all 'fat', then a depth order for $\mathcal{K}$ can be computed in time $O(n \log^5 n)$. (ii) If $\mathcal{K}$ is a set of n convex and simply-shaped objects whose xy-projections are all 'fat' and their sizes are within a constant ratio from one another, then a depth order for $\mathcal{K}$ can be computed in time $O(n\lambda_s^{1/2}(n)\log^4 n)$, where s is the maximum number of intersections between the boundaries of the xy-projections of any pair of objects in $\mathcal{K}$, and $\lambda_s(n)$ is the maximum length of (n, s) Davenport-Schinzel sequences.

1 Introduction

We describe a general technique for solving problems of the following form: Let $\mathcal{C} = \{c_1, \ldots, c_n\}$ be a set of n objects in the plane, and let $\prec$ be a partial binary anti-symmetric relation over $\mathcal{C}$ such that, for any pair of objects $c_i, c_j \in \mathcal{C}$, if $c_i \cap c_j \neq \emptyset$ then either $c_i \prec c_j$ or $c_j \prec c_i$, and if $c_i \cap c_j = \emptyset$ then c_i and c_j are incomparable. Moreover, we assume that for any pair of intersecting objects in $\mathcal{C}$ it is possible to determine in constant time which of the two possibilities holds, and that the transitive closure of $\prec$ is acyclic, i.e., it is a partial order. We wish to compute a linear extension of this partial order. We refer to this problem as the *2-dimensional linear-extension problem.*

*Work on this paper by the first author has been supported by National Science Foundation Grant CCR–93–01259 and an NYI award. Work on this paper by the third author has been supported by NSF Grant CCR-91-22103, by a Max-Planck Research Award, and by grants from the U.S.-Israeli Binational Science Foundation, the Israel Science Fund administered by the Israeli Academy of Sciences, and the G.I.F., the German-Israeli Foundation for Scientific Research and Development.

†Department of Computer Science, Duke University

‡School of Mathematical Sciences, Tel Aviv University

§School of Mathematical Sciences, Tel Aviv University, and Courant Institute of Mathematical Sciences, New York University

The main motivation for studying this problem comes from the problem of computing a *depth order* in three dimensions. Specifically, we are given a set $\mathcal{K}$ of n non-intersecting, convex and simply-shaped objects in 3-space (by 'simple shape' we mean that each object can be described by a constant number of polynomial equalities and inequalities of constant maximum degree). For a pair of objects $K, L \in \mathcal{K}$, we say that K lies *below* L (and L lies *above* K) if there exists a vertical line λ that intersects both K and L and $\lambda \cap K$ lies fully below $\lambda \cap L$ (the convexity of K, L implies that this relation is anti-symmetric). We denote this relation by $K \prec L$. Assuming that the transitive closure of $\prec$ is an acyclic relation, any linear extension of it is called a *depth order* of $\mathcal{K}$. It is then clear that the set of the xy-projections of the objects in $\mathcal{K}$ is a proper input for the 2-dimensional linear-extension problem, as formulated above, where for any pair of projections K^*, L^* of two respective objects K, L of $\mathcal{K}$, we have $K^* \prec L^*$ if and only if $K \prec L$. Hence, the problem of computing depth orders in 3-space can be reduced to the 2-dimensional linear-extension problem. Computing depth orders is a preliminary step of many algorithms for *hidden surface removal* in computer graphics, and of several other algorithms; see [14] for more details.

We note that the 2-dimensional linear-extension problem arises in some other applications as well. For example, suppose that each $c_i \in \mathcal{C}$ designates some area of the plane to which we want to apply a certain process (e.g., spraying it with some substance, painting it, etc.), and that whenever two such areas overlap, there is some order in which the two respective processes must be scheduled. The problem is then to find a global scheduling order for the processes so that they are executed in the correct order for each point in the plane. In fact, the painter's algorithm for hidden surface removal (see e.g. [19]) is an instance of this more general problem.

The problem of computing depth orders has been studied by de Berg et al. [16]. They presented an algorithm for computing a depth order of a collection of n arbitrary non-intersecting triangles in 3-space; their algorithm runs in time $O(n^{4/3+\varepsilon})$, for any $\varepsilon > 0$. Their algorithm is conceptually fairly simple, but it uses rather involved range searching methods, and it does not extend to sets of more general objects. We note that the main challenge here is to obtain subquadratic solutions, because a quadratic algorithm is easy, even for more general objects: Simply compute, in a brute-force manner, all the k intersecting pairs of the xy-projections of the given objects. This gives us all the possible $\prec$-relationships, and we can then complete the resulting partial order into a linear order, using topological sorting. This method runs in $O(n^2)$ time. Thus the result of [16] is a significant progress over this naive technique in the case of triangles. (If k is much smaller than quadratic, we can also compute these k intersections (in the general case) in time $O((n+k)\log n)$, using a standard line-sweeping technique. Still, unless $k = o(n^{4/3+\varepsilon})$, this method is also inferior to the technique of [16], in the case of triangles.)

We present here a different approach to the problem (actually, to the 2-dimensional linear-extension problem), which is simpler than the approach of [16]. The algorithm applies to collections of objects that satisfy certain 'fatness' properties (so it does not apply to

arbitrary triangles). We call the objects in a collection $\mathcal{K}$ as above *α-fat* if the xy-projection of any object in $\mathcal{K}$ has the property that it is contained in some axis-parallel square S^+ and contains another axis-parallel square S^-, so that the ratio between the edge lengths of S^+ and S^- is at most α. (Note that the notion of fatness depends also on the 3-D orientations of the objects in $\mathcal{K}$ and not just on their shapes; for example, if $\mathcal{K}$ is a collection of disks in 3-space, then these disks are α-fat if and only if the angles between their normals and the z-axis are all smaller than some fixed angle $\delta = \delta(\alpha)$.) We say that the objects in $\mathcal{K}$ have the *α-ratio* property if the xy-projections K^*, L^* of any pair of objects in $\mathcal{K}$ have the property that K^* is contained in some axis-parallel square S^+, L^* contains another axis-parallel square S^-, and the ratio between the edge lengths of S^+ and S^- is at most α (and a symmetric condition holds when interchanging K^* and L^*).

Our main results are:

Theorem 1.1 *A depth order of a collection of n non-intersecting α-fat triangles in 3-space, for any $\alpha > 1$, can be computed (if it exists) in time $O(n \log^5 n)$, where the constant of proportionality depends on α (more precisely, it is bounded by $c\alpha^9 \log \alpha$, for some absolute constant c).*

Theorem 1.2 *Let $\mathcal{K}$ be a collection of n non-intersecting convex simply-shaped α-fat objects in 3-space with the α-ratio property. Then a depth order for $\mathcal{K}$ can be computed (if it exists), in an appropriate model of computation, in time $O(n\lambda_s^{1/2}(n) \log^4 n)$, where s is the maximum number of intersections between the boundaries of any pair of projections of objects in $\mathcal{K}$.*[1] *The constant of proportionality depends on α (more precisely, it is bounded by $c\alpha^2$, for some absolute constant c).*

Theorem 1.3 *The 2-dimensional linear-extension problem for a collection $\mathcal{C}$ of n disks can be solved in $O(n^{7/5+\varepsilon})$ time, for any $\varepsilon > 0$. Moreover, if $\mathcal{C}$ has the α-ratio property, this problem can be solved in $O(n \log^7 n)$ time, where the constant of proportionality is bounded by $c\alpha^2$, for some absolute constant c.*

Thus, for the case of fat triangles, we obtain an algorithm that is much faster and simpler than the algorithm of [16]. This result is especially significant since it enables us to perform hidden surface removal for a collection of fat triangles with unknown depth order, using the algorithm of [22], whose running time is close to linear. Prior to our result, one had to spend close to $O(n^{4/3})$ time on computing the depth order, before proceeding to apply the algorithm of [22], or, alternatively, one could apply the hidden surface removal algorithm of [15], which does not require a depth order to be given, but still runs in time close to $O(n^{4/3})$. The running time of our algorithm for general collections of objects, as in Theorem 1.2, which is close to $O(n^{3/2})$, is somewhat worse than the running time

[1] $\lambda_s(n)$ is the maximum length of (n, s) Davenport-Schinzel sequences, which is nearly linear in n for any fixed s [18, 28].

of the algorithm of [16], but it is nevertheless (significantly) subquadratic, and it applies to many cases where no previous subquadratic algorithm was known, such as the case of disks in 3-space (satisfying the conditions of Theorem 1.2). Theorem 1.3 is stated in its more general version (2-dimensional linear-extension rather than depth order), since a depth order for a collection of n non-intersecting horizontal disks in 3-space can be trivially computed in $O(n \log n)$ time, by simply sorting the corresponding horizontal planes by their z-coordinates. (Note that the parameter α in the three theorems above does not have to be a constant. For example, the algorithm of Theorem 1.1 is more efficient than the algorithm of [16] for any $\alpha = O(n^{1/27})$, and the algorithm of Theorem 1.2 remains subquadratic for any $\alpha = O(n^{1/4-\varepsilon})$, for $\varepsilon > 0$.)

We still do not know how to extend our technique to detect cycles in the 'above-below' relation $\prec$ (the algorithm of [16] can detect such cycles). We are currently studying this problem.

The paper is organized as follows. In Section 2 we describe the general technique for the 2-dimensional linear-extension problem. In Section 3, we specialize this technique to the case of fat triangles, to obtain Theorem 1.1, and, in Section 4, we specialize it to the case of general fat objects with the α-ratio property, to obtain Theorem 1.2. In Section 5 we consider the case of disks in the plane, and establish Theorem 1.3. We conclude in Section 6, with a discussion of our results and with some open problems.

2 Outline of the Method

We focus here on the 2-dimensional linear-extension problem. The algorithm consists of two stages. In the first stage we compute a collection of pairs of subsets of the given collection $\mathcal{C}$,

$$\mathcal{F} = \{(\mathcal{R}_1, \mathcal{B}_1), (\mathcal{R}_2, \mathcal{B}_2), \ldots, (\mathcal{R}_m, \mathcal{B}_m)\},$$

such that[2]

(i) For each intersecting pair of objects $c_1, c_2 \in \mathcal{C}$, there is an $1 \leq i \leq m$ such that either $c_1 \in \mathcal{R}_i$ and $c_2 \in \mathcal{B}_i$, or $c_2 \in \mathcal{R}_i$ and $c_1 \in \mathcal{B}_i$.

(ii) For each $i = 1, \ldots, m$, all the objects in $\mathcal{B}_i$ have a common point (we say that $\mathcal{B}_i$ has the *common point* property).

We will refer to such a family $\mathcal{F}$ of pairs of subsets as a *canonical decomposition* for $\mathcal{C}$. In the second stage we construct a directed graph G over $\mathcal{C}$ of size $O(\sum_{i=1}^{m}(|\mathcal{R}_i| + |\mathcal{B}_i|))$, such

[2]The idea of decomposing a set of objects $\mathcal{C}$ into a family $\mathcal{F}$ of pairs of subsets that satisfies property (i) has been used in several other geometric algorithms, see e.g. [1, 5, 8, 9, 11]. In most of these applications, however, one needs the property that each object in $\mathcal{R}_i$ intersects every object in $\mathcal{B}_i$, which is stronger than property (i).

that any permutation of $\mathcal{C}$ that is obtained by topologically sorting G is a linear extension of the underlying relation $\prec$. Initially, the edge set of G is empty. We process each pair $(\mathcal{R}, \mathcal{B})$ of the above collection separately, and obtain from it $O(|\mathcal{R}| + |\mathcal{B}|)$ edges of G, as follows. By property (ii), the set $\mathcal{B}$ is a chain of the relation $\prec$, and we can therefore sort $\mathcal{B}$ into a linear list, denoted as $b_1 \prec b_2 \prec \cdots \prec b_q$. We add the $q-1$ edges (b_i, b_{i+1}), for $i = 1, \ldots, q-1$, to the graph G. Next we 'merge' the elements of $\mathcal{R}$ into the list $\mathcal{B}$. That is, we want to compute, for each element r of $\mathcal{R}$, two pointers to its immediate predecessor and successor (if they exist) in the list $(b_1, \ldots, b_q)$; these are denoted, respectively, as $b_{pred(r)}$ and $b_{succ(r)}$. We then add, for each $r \in \mathcal{R}$, the (at most) two corresponding edges $(b_{pred(r)}, r)$ and $(r, b_{succ(r)})$ to G.

We claim that, after this step is repeated for all pairs $(\mathcal{R}, \mathcal{B})$, the relation $\prec$ is contained in the transitive closure of G; that is, for each pair $c \prec c'$ of objects of $\mathcal{C}$, there exists a directed path in G from c to c'. (Note that G is acyclic, by assumption.) Indeed, if c, c' is such a pair then $c \cap c' \neq \emptyset$, and therefore there exists a pair $(\mathcal{R}, \mathcal{B})$ in the collection computed in the first stage, such that one of the objects, say c, belongs to $\mathcal{B}$ and the other belongs to $\mathcal{R}$. In the second stage we find an object $c'' \in \mathcal{B}$ such that c'' is the immediate predecessor of c' in $\mathcal{B}$, and the edge (c'', c') is added to G (note that c'' exists, since c is a predecessor of c' in $\mathcal{B}$). If $c'' = c$ then the claim is obvious, and if $c'' \neq c$ then necessarily $c \prec c''$, so G contains a directed path from c to c'', which, concatenated with the edge (c'', c'), gives us the desired path from c to c'. The case where $c \in \mathcal{R}$ and $c' \in \mathcal{B}$ is completely symmetric.

Suppose we have an algorithm $\mathcal{A}$ that, given a subset $\mathcal{R}' \subseteq \mathcal{R}$ and a subset $\mathcal{B}' \subseteq \mathcal{B}$, determines for each object $r \in \mathcal{R}'$ whether r intersects any object of $\mathcal{B}'$ (which is the same as asking whether r intersects the region $\bigcup_{b \in \mathcal{B}'} b$), and, if so, it also computes a 'witness' object $b \in \mathcal{B}'$ that intersects r. Using $\mathcal{A}$ as a subroutine, we can perform the merge step for a pair of sets $(\mathcal{R}, \mathcal{B})$, with $\mathcal{B}$ already sorted, as follows. Let w be a point in $\bigcap \mathcal{B}$. Since the objects of $\mathcal{C}$ are assumed to be convex, the union of any subset of $\mathcal{B}$ is a star-shaped region with respect to w. Let T be a minimum-height binary tree over the objects $b_1, \ldots, b_q$, where these objects are stored at the leaves of T from left to right in increasing order. Denote by $\mathcal{B}_v$ the set of objects stored at the leaves of the subtree of T rooted at a node v. Associate with each internal node v of T the region $U_v = \bigcup_{b \in \mathcal{B}_v} b$. We process the nodes of T by levels, beginning at the root level. The output of the processing of the j-th level nodes is an assignment of a subset $\mathcal{R}_v \subseteq \mathcal{R}$ to each node v of the $(j+1)$-th level. The algorithm maintains the following invariant property: At the beginning of the current level (after the $\mathcal{R}$-subsets of the current level have been formed), each object r of $\mathcal{R}$ is present in at most four $\mathcal{R}$-subsets, $\mathcal{R}_{v_1}$, $\mathcal{R}_{v_2}$, $\mathcal{R}_{v_3}$, $\mathcal{R}_{v_4}$, of this level, so that the largest predecessor and the smallest successor of r in $\mathcal{B}$ (if they exist) are stored at one or two of these nodes. At the root level, we first assign $\mathcal{R}_{root} = \mathcal{R}$ to the root, and then apply the algorithm $\mathcal{A}$ to the two sets associated with the root, i.e., to $\mathcal{R}_{root}$ and $\mathcal{B}_{root}$. We then form the sets $\mathcal{R}_{left(root)}$ and $\mathcal{R}_{right(root)}$, as follows. For each object $r \in \mathcal{R}_{root} = \mathcal{R}$, if r was found not to intersect U_{root} we can ignore r, since it is $\prec$-incomparable with all objects of $\mathcal{B}$. If r was found to intersect U_{root} and the witness b that was found is greater than r (i.e., $r \prec b$), then, if b

belongs to $\mathcal{B}_{left(root)}$, add r to $\mathcal{R}_{left(root)}$, otherwise (i.e., $b \in \mathcal{B}_{right(root)}$) add r to both sets. The case where the witness b that was found is smaller than r is treated symmetrically. The invariant maintained by the algorithm is easily seen to hold at the level of the children of the root. At a general level j, we apply the algorithm $\mathcal{A}$, for each node v of this level, to the sets $\mathcal{R}_v$, $\mathcal{B}_v$. We then inspect, for each object $r \in \mathcal{R}$, the (at most) four j-level nodes that store r, and identify (at most) two of these nodes (the rightmost of these nodes for which a predecessor witness of r was found, and the leftmost of these nodes for which a successor witness was found), whose children might still need to store r. We distribute r to these (at most four) children, based on a criterion similar to that used at the root. The validity of the invariant property easily follows by induction on j. At the leaf level, each object r of $\mathcal{R}$ is assigned to at most four leaves, and the desired objects $b_{pred(r)}$, $b_{succ(r)}$ (if they exist) are two of these leaves, and can thus be determined in constant time.

This concludes the outline of the algorithm. In the subsequent sections we will specialize this technique to a set of fat triangles (Section 3), and to a set of general fat objects with the α-ratio property (Section 4). We also discuss briefly, in Section 5, the special case of disks, which admits a more efficient solution than the general case.

3 Fat Triangles

In this section we apply the general technique outlined above to the case where the objects of the input collection $\mathcal{C}$ of the 2-dimensional linear extension problem are all *δ-fat triangles*, that is, all the angles of each triangle in $\mathcal{C}$ are at least δ, for some fixed constant δ. (For triangles, this is an equivalent definition of fatness.) Fat triangles have been investigated in [24], where it was shown that the combinatorial complexity of the union of m δ-fat triangles is $O(m \log\log m)$, where the constant of proportionality depends on δ (see also [7, 17, 29]). Moreover, such a union can be computed in time $O(m \log^2 m)$ by a deterministic algorithm [24], or in expected time $O(m 2^{\alpha(m)} \log m)$ by a randomized algorithm [26].

3.1 The First Stage

The first stage of the algorithm is implemented in this case as follows. We first fix a family $\mathcal{O}$ of $O(1/\delta)$ orientations, evenly spaced in $[0, 2\pi)$, so that the angle between any pair of successive orientations is $\leq \delta/3$. We then represent each triangle $\Delta \in \mathcal{C}$ as the union of three overlapping triangles, such that each subtriangle Δ' is 'semi-canonical', in the sense that it has two sides at orientations belonging to $\mathcal{O}$; we refer the reader to Figure 1 and to [24] for the easy details. This allows us to partition the collection of new triangles into $O(1/\delta^3)$ subfamilies, so that all triangles within any subfamily $\mathcal{C}'$ are 'almost-homothetic', that is, two sides of each triangle of $\mathcal{C}'$ are at two fixed orientations, and the orientation of the third side lies in some narrow range of orientations (of size, say, $\delta/3$).

Recall that in the first stage we need to compute a canonical decomposition for $\mathcal{C}$. We

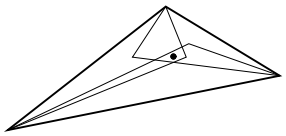

Figure 1: Representing a fat triangle Δ as the union of three 'semi-canonical' subtriangles (the middle point is the center of the inscribed circle of Δ)

do this by computing a canonical decomposition separately for each pair of (not necessarily distinct) subfamilies $\mathcal{C}'$ and $\mathcal{C}''$. That is, we compute a family $\mathcal{F} = \{(\mathcal{R}_1, \mathcal{B}_1), \ldots, (\mathcal{R}_m, \mathcal{B}_m)\}$ of pairs of subsets such that

(i) For each i, $\mathcal{R}_i \subseteq \mathcal{C}'$ and $\mathcal{B}_i \subseteq \mathcal{C}''$. Moreover, for each intersecting pair of triangles $(c_1, c_2) \in \mathcal{C}' \times \mathcal{C}''$, there is an i such that $c_1 \in \mathcal{R}_i$ and $c_2 \in \mathcal{B}_i$ (or vice versa, when $\mathcal{C}' = \mathcal{C}''$).

(ii) Each $\mathcal{B}_i$ has the common point property.

The union of these collections over all pairs of subfamilies (after replacing each semi-canonical triangle in $\mathcal{R}_i, \mathcal{B}_i$ by its original containing triangle) constitutes an appropriate collection for $\mathcal{C}$.

We now describe how to construct $\mathcal{F}$ for a fixed pair of subfamilies $\mathcal{C}'$ and $\mathcal{C}''$. First assign to each triangle $\Delta \in \mathcal{C}' \cup \mathcal{C}''$ the radius $\rho(\Delta)$ of the smallest circle containing Δ, and sort the triangles in $\mathcal{C}' \cup \mathcal{C}''$ in increasing order of these radii. Let $\mathcal{L}'$ (resp. $\mathcal{H}'$) denote the subset of triangles of $\mathcal{C}'$ that lie in the smaller (resp. larger) half of the sorted sequence. We define $\mathcal{L}'', \mathcal{H}'' \subseteq \mathcal{C}''$ in complete analogy. Our strategy is to compute a canonical decomposition for $\mathcal{L}', \mathcal{H}''$, in a manner that will be described in a moment, then compute a canonical decomposition for $\mathcal{L}'', \mathcal{H}'$, in the same manner, and finally handle recursively the pairs $(\mathcal{L}', \mathcal{L}'')$ and $(\mathcal{H}', \mathcal{H}'')$.

Next, we describe how to compute a canonical decomposition for $\mathcal{L}', \mathcal{H}''$. Associate with a semi-canonical triangle $\Delta = uvw$ two triangles, $\Delta^+ \supseteq \Delta$ and $\Delta^- \subseteq \Delta$, and a trapezoid Δ^*, as follows (see Figure 2 for an illustration). The triangle Δ^+ contains Δ and is obtained from Δ by replacing the side vw whose orientation is not fixed by another side. The new side has a vertex v in common with the former side, its orientation is in $\mathcal{O}$ and is greater than the orientation of the former side, and the angle between the former side and the new side is at most $\delta/3$. The triangle Δ^- is contained in Δ and is obtained by replacing the new side of Δ^+ by a parallel side through the other vertex w. Finally, $\Delta^* \stackrel{\text{def}}{=} \Delta^+ - \Delta^-$.

Lemma 3.1 *If $\Delta_1 \cap \Delta_2 \neq \emptyset$, where $\Delta_1 \in \mathcal{L}'$ and $\Delta_2 \in \mathcal{H}''$, then at least one of the following conditions must hold:*

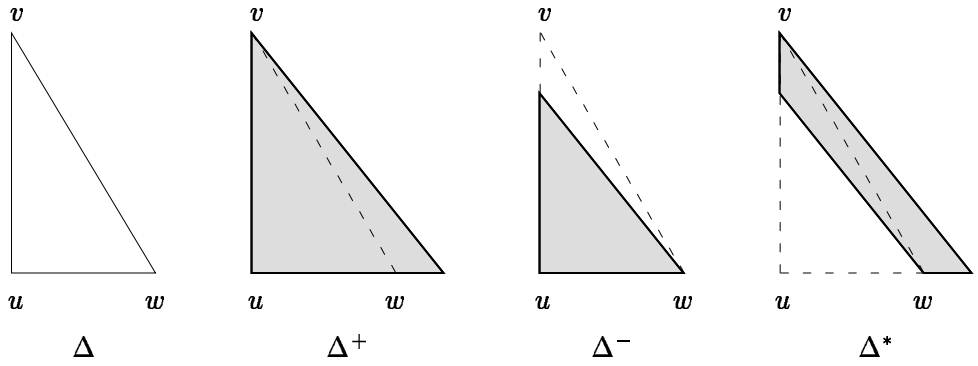

Figure 2: A triangle Δ, the corresponding triangles Δ^+, Δ^-, and the trapezoid Δ^*

1. *One of the two sides of Δ_2 of fixed orientations intersects a side of Δ_1^+.*
2. $\Delta_1^+ \cap \Delta_2^- \neq \emptyset$.
3. *A vertex of Δ_1^+ lies inside Δ_2^*.*

Proof. Since $\Delta_1 \cap \Delta_2 \neq \emptyset$, and since $\Delta_1^+ \supseteq \Delta_1$, we must also have $\Delta_1^+ \cap \Delta_2 \neq \emptyset$. If neither condition 1 nor condition 2 holds, then Δ_1^+ must intersect the triangle $\tilde{\Delta}_2 \stackrel{\text{def}}{=} \Delta_2 - \Delta_2^-$, but cannot intersect any of the two sides of $\tilde{\Delta}_2$ that are also sides of Δ_2^* (see Figure 3). Hence Δ_1^+ must have a vertex inside $\tilde{\Delta}_2$, thus also inside Δ_2^*. □

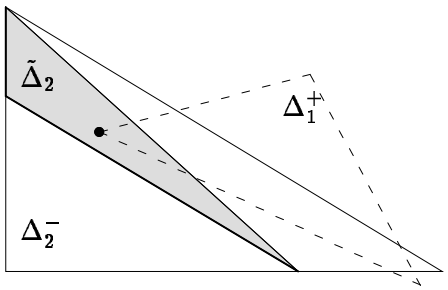

Figure 3: Illustration of the proof of Lemma 3.1

By the above lemma, a canonical decomposition for $\mathcal{L}', \mathcal{H}''$ can be obtained by reporting,

in compact form, all pairs $\Delta_1 \in \mathcal{L}', \Delta_2 \in \mathcal{H}''$ that satisfy one of the three conditions stated in the lemma.

Reporting intersections of type 1: Let us denote by $\mathcal{L}'^+$ the set of expanded triangles Δ_1^+, for $\Delta_1 \in \mathcal{L}'$. Fix one of the three possible orientations, θ_1, of the sides of expanded triangles in $\mathcal{L}'^+$ and one of the two fixed orientations, θ_2, of the sides of triangles in $\mathcal{H}''$. Let S_1 (resp. S_2) denote the collection of all sides at orientation θ_1 (resp. θ_2) of triangles in $\mathcal{L}'^+$ (resp. in $\mathcal{H}''$). Since the segments in S_1 are all parallel, and so are the segments of S_2, it is straightforward, using standard orthogonal range searching techniques (such as 2-dimensional *range trees* [25]), to report all intersecting pairs of segments in $S_1 \times S_2$, as a collection of complete bipartite graphs $\{\mathcal{R}_j \times \mathcal{B}_j\}$, so that $\sum_j |\mathcal{R}_j|, \sum_j |\mathcal{B}_j| = O(m \log^2 m)$, where m is the combined size of $\mathcal{L}'$ and $\mathcal{H}''$; the cost of this procedure is also $O(m \log^2 m)$. Repeating this step six times, for all possible pairs θ_1, θ_2, we obtain a collection of pairs $\{(\mathcal{R}_j, \mathcal{B}_j)\}$, so that:

(i) $\mathcal{R}_j \subseteq \mathcal{L}'$ and $\mathcal{B}_j \subseteq \mathcal{H}''$ for each j.

(ii) For each pair $(\mathcal{R}_j, \mathcal{B}_j)$, each pair of triangles $\Delta_1 \in \mathcal{R}_j$, $\Delta_2 \in \mathcal{B}_j$ is such that $\rho(\Delta_1) \leq \rho(\Delta_2)$.

(iii) For each pair $(\mathcal{R}_j, \mathcal{B}_j)$, each pair of triangles $\Delta_1 \in \mathcal{R}_j$, $\Delta_2 \in \mathcal{B}_j$ is such that Δ_1^+ and Δ_2 have nonempty intersection.

(iv) $\sum_j |\mathcal{R}_j|, \sum_j |\mathcal{B}_j| = O(m \log^2 m)$.

Note that property (ii) follows from property (i) and from the definition of the sets $\mathcal{L}'$ and $\mathcal{H}''$. Next we need to refine the pairs $(\mathcal{R}_j, \mathcal{B}_j)$, so as to enforce the common point property for one of the sets in each pair. Let $(\mathcal{R}, \mathcal{B})$ be one of the pairs produced above, and let Δ_0 be a fixed triangle in $\mathcal{R}$. Each triangle in $\mathcal{B}$ must contain, by property (iii), a point inside the expanded triangle Δ_0^+. Let z be the center of the inscribed circle of Δ_0^+, and let $\Delta_0^{++} = z + 2(\Delta_0^+ - z)$; that is, Δ_0^{++} is an expansion of Δ_0^+ about z by a factor of 2; see Figure 4.

Lemma 3.2 *For each triangle $\Delta \in \mathcal{B}$ we have $area(\Delta \cap \Delta_0^{++}) \geq \beta \cdot area(\Delta_0^{++})$, for some constant fraction $\beta \geq c\delta^3$, where c is an absolute constant.*

Proof. Suppose first that Δ has just one vertex inside Δ_0^{++} (as shown in Figure 4). The two edges of Δ incident to that vertex intersect Δ_0^{++} in two segments of length $\geq r$, where r is the radius of the inscribed circle of Δ_0^+. Since Δ is δ-fat, the area of $\Delta \cap \Delta_0^{++}$ is thus at least $\frac{1}{2} r^2 \sin\delta = \Omega(r^2\delta)$. On the other hand, since Δ_0^{++} is also δ-fat, its area is at most $O(r^2/\delta)$, as is easily verified. Hence $area(\Delta \cap \Delta_0^{++})/area(\Delta_0^{++}) = \Omega(\delta^2) = \Omega(\delta^3)$, as claimed.

Suppose next that Δ has another vertex inside Δ_0^{++}. It is easily verified that the length of each side of Δ is $\Omega(\rho(\Delta)\delta) = \Omega(\rho(\Delta_0)\delta) = \Omega(r\delta)$. In fact, a more refined argument

shows that the product of any two sides of Δ is $\Omega(\rho^2(\Delta)\delta) = \Omega(r^2\delta)$. We can thus apply the same arguments as above, to conclude that $area(\Delta \cap \Delta_0^{++}) = \Omega(r^2\delta^2)$, so that $area(\Delta \cap \Delta_0^{++})/area(\Delta_0^{++}) = \Omega(\delta^3)$. $\Box$

Lemma 3.2 enables us to use a simple construction, due to Fredman, that produces $l =$

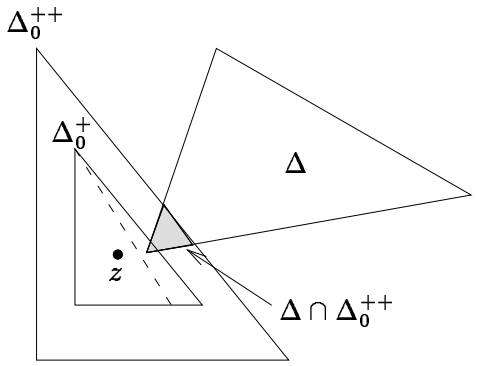

Figure 4: The area of $\Delta \cap \Delta_0^{++}$ must be 'large'

$O(\frac{1}{\delta^3}\log\frac{1}{\delta})$ points within Δ_0^{++} so that each triangle of $\mathcal{B}$ contains at least one of these points (the set produced by Fredman's construction 'stabs' any convex object that covers some constant fraction of a given convex region; an alternative construction is to draw l independent random points from the uniform distribution within Δ_0^{++}). Thus, by splitting $\mathcal{B}$ into l subfamilies $\mathcal{B}_1, \ldots, \mathcal{B}_l$, and by replacing the pair $(\mathcal{R}, \mathcal{B})$ by the l corresponding pairs $(\mathcal{R}, \mathcal{B}_1), \ldots, (\mathcal{R}, \mathcal{B}_l)$, we ensure the common point property for all the new sets $\mathcal{B}_j$, without affecting the other properties listed above.

Reporting intersections of type 2: We now find all pairs of intersecting triangles (Δ_1^+, Δ_2^-), for $\Delta_1 \in \mathcal{L}'$, $\Delta_2 \in \mathcal{H}''$. We begin by finding all such pairs for which a vertex of Δ_2^- lies inside Δ_1^+. Let $\mathcal{L}'^+ = \{\Delta^+ \mid \Delta \in \mathcal{L}'\}$. Since all the triangles in $\mathcal{L}'^+$ are homothets, we can assume, applying an appropriate affine transformation if necessary, that each triangle $\Delta^+ \in \mathcal{L}'^+$ is a right-angle triangle with a horizontal base, and that the hypotenuse of Δ^+ forms an angle of $3\pi/4$ from the base (in counterclockwise direction); see Figure 5. We construct a segment tree T on the bases of the triangles in $\mathcal{L}'^+$; see Preparata and Shamos [27] for details on segment trees. Each node v of T is associated with a vertical strip I_v. A triangle Δ^+ is stored at a node $v \in T$ if the x-projection of the base of Δ^+ contains the projection of I_v but does not contain that of $I_{p(v)}$, where $p(v)$ is the parent of v. For each such triangle Δ^+, we partition $\Delta^+ \cap I_v$ into a triangle and a parallelogram by drawing a segment, parallel to the hypotenuse of Δ^+, from the intersection point of the base of Δ^+ and the right boundary of I_v; see Figure 5. Let $\mathcal{L}_v'^+$ be the set of resulting triangles and P_v be the set of resulting parallelograms. We sort the triangles of $\mathcal{L}_v'^+$ by the

y-coordinates of their bases, and construct a minimum height binary tree τ_v whose leaves store $\mathcal{L}'^{+}_v$ in this sorted order. Note that this order is also equal to the increasing y-order of the hypotenuses of the triangles of $\mathcal{L}'^{+}_v$. We associate with each node ξ of τ_v the set of triangles stored in the leaves of the subtree rooted at ξ. Given a query point p, we can thus report, in $O(\log m)$ time, the triangles of $\mathcal{L}'^{+}_v$ that contain p, as a union of $O(\log m)$ disjoint subsets (where m is, as above, the combined size of $\mathcal{L}'$ and $\mathcal{H}''$). We next construct a minimum-height binary tree on the slanted edges of the parallelograms in P_v in increasing y-order, so that, for a query point p, we can report all parallelograms that contain p as a union of $O(\log m)$ disjoint subsets. Since the secondary structures at every node v require $O(|\mathcal{L}'^{+}_v| \log |\mathcal{L}'^{+}_v|)$ space, the size of the overall data structure is $O(m \log^2 m)$. The total preprocessing time is also $O(m \log^2 m)$.

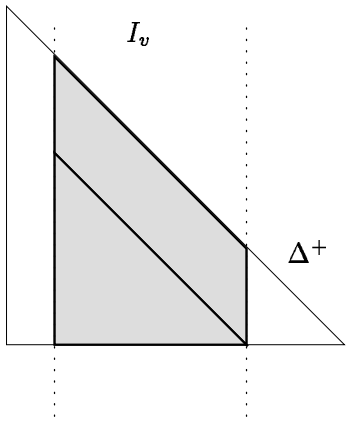

Figure 5: Partitioning $\Delta^+ \cap I_v$ into a triangle and a parallelogram

Let p be a point in the plane. We can report all triangles of $\mathcal{L}'^{+}$ that contain p as follows: We first find the nodes v of the segment tree T for which I_v contains p. For each such node v, we report the triangles of $\mathcal{L}'^{+}_v$ and the parallelograms of P_v that contain p. The total query time is $O(\log^2 m)$ and the output consists of $O(\log^2 m)$ disjoint subsets. Repeating this procedure for all vertices of Δ^-, for $\Delta \in \mathcal{H}''$, we can obtain a family of pairs $\{(\mathcal{R}_j, \mathcal{B}_j)\}$ that satisfy properties (i), (ii), and (iv) of the previous case, and the following property:

(iii') For each pair $(\mathcal{R}_j, \mathcal{B}_j)$, each pair of triangles $\Delta_1 \in \mathcal{R}_j, \Delta_2 \in \mathcal{B}_j$ is such that a vertex of Δ_2^- lies in Δ_1^+.

We enforce the common point property as in the previous type of intersection. Next, we find, in a symmetric manner, all pairs of triangles (Δ_1, Δ_2) for which a vertex of Δ_1^+ lies inside Δ_2^-, and, finally, using the technique for reporting intersections of type 1, we

compute all such pairs of triangles for which an edge of Δ_1^+ intersects an edge of Δ_2^-. The total size of the graphs, and the computation time, are both $O(m \log^2 m)$.

Reporting intersections of type 3:

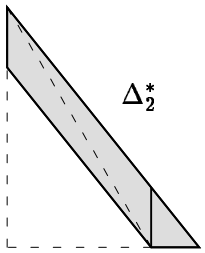

Figure 6: Partitioning Δ_2^* into a triangle and a parallelogram

We are left with the task of finding intersections between triangles Δ_1^+, for $\Delta_1 \in \mathcal{L}'$, and trapezoids Δ_2^*, for $\Delta_2 \in \mathcal{H}''$, where a vertex of Δ_1^+ lies inside Δ_2^*. Since Δ_2^* can be partitioned into a triangle and a parallelogram whose edges have fixed orientations, as shown in Figure 6, it is easily verified that all vertex-trapezoid containments can be reported as in the previous case. This produces a collection of pairs of sets of triangles $(\mathcal{R}_i, \mathcal{B}_i)$ that satisfies the above properties (where the overall size of these sets, and the time it takes to compute them, is $O(m \log^2 m)$).

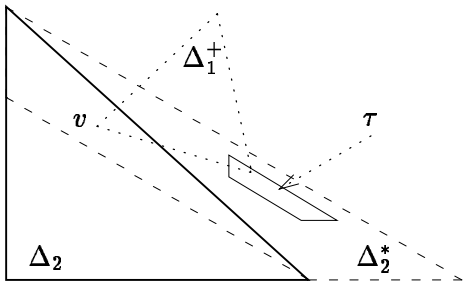

Figure 7: Reporting intersections of type 3: the case where Δ_2 can be discarded

Let $(\mathcal{R}, \mathcal{B})$ denote one of the canonical pairs of sets of triangles produced in this case, where $\mathcal{R}$ consists of 'small' triangles (in $\mathcal{L}'$) and $\mathcal{B}$ of 'large' triangles (in $\mathcal{H}''$). The orientations of the edges of all the trapezoids Δ^*, for $\Delta \in \mathcal{B}$, are fixed, so the intersection $\tau = \bigcap_{\Delta \in \mathcal{B}} \Delta^*$ is a nonempty trapezoid whose sides have the same three fixed orientations. We now test each of the original triangles $\Delta \in \mathcal{B}$ to determine whether Δ intersects τ.

Triangles that are disjoint from τ can be discarded from $\mathcal{B}$, because, if such a triangle Δ_2 intersects a triangle $\Delta_1 \in \mathcal{R}$, then either one of the other two conditions holds and thus the pair (Δ_1, Δ_2) will be reported by one of the previous procedures, or Δ_1^+ has another vertex v that lies in $\Delta_2 \cap \Delta_2^*$ (this is shown as in the proof of Lemma 3.1) and thus (Δ_1, Δ_2) appears in another pair $(\mathcal{R}', \mathcal{B}')$, obtained by the procedure for reporting type 3 intersections; see Figure 7. In the latter case, Δ_2 intersects $\tau' = \cap_{\Delta' \in \mathcal{B}'} (\Delta')^*$, because v lies in τ' and $v \in \Delta_2$. Hence, Δ_2 will not be discarded when processing $\mathcal{B}'$.

Let $\Delta \in \mathcal{B}$ be a triangle that intersects τ. Since two of the edges of Δ have the same slopes as the non-parallel edges of τ and since they do not intersect the interior of τ, it is easily verified that Δ contains one of the vertices of τ. Hence, by splitting $\mathcal{B}$ into at most four subcollections, we can guarantee that each subcollection has the common point property. (In fact, as is easy to verify, two subcollections suffice.)

To sum up, after performing the procedures described above, we end up with a collection of pairs $\{(\mathcal{R}, \mathcal{B})\}$ of sets of triangles, so that the union of the sets $\mathcal{R} \times \mathcal{B}$ contains all intersecting pairs of $\mathcal{L}' \times \mathcal{H}''$, the total size of all these sets is $O(m \log^3 m)$ (where the constant of proportionality is $O(\frac{1}{\delta^3} \log \frac{1}{\delta})$), and for each pair $(\mathcal{R}, \mathcal{B})$, the set $\mathcal{B}$ has the common point property. We repeat the same procedure for the pair $\mathcal{L}''$ and $\mathcal{H}'$. Taking into account the recursive handling of $(\mathcal{L}', \mathcal{L}'')$ and of $(\mathcal{H}', \mathcal{H}'')$, we obtain a canonical decomposition for $\mathcal{C}', \mathcal{C}''$ whose total size is $O((\frac{1}{\delta^3} \log \frac{1}{\delta})\, n \log^3 n)$. This completes the processing for the pair $\mathcal{C}', \mathcal{C}''$, and we repeat this procedure for the other $(O(1/\delta^6))$ pairs of subfamilies, to obtain an overall collection of pairs $(\mathcal{R}, \mathcal{B})$, whose total size is $O((\frac{1}{\delta^9} \log \frac{1}{\delta})\, n \log^3 n)$

3.2 The Second Stage

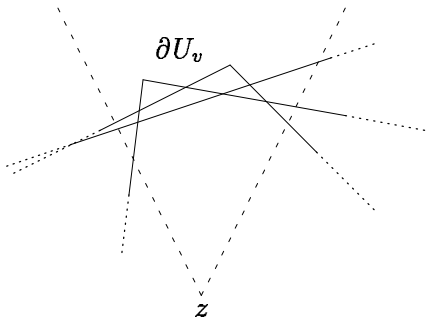

Figure 8: The proof that U_v has linear complexity

We now describe the implementation of the second stage of the algorithm. Let $(\mathcal{R}, \mathcal{B})$ be one of the pairs in the canonical decomposition for $\mathcal{C}$ produced in the first stage. Since $\mathcal{B}$ has the common point property, we can fully sort it, according to $\prec$, in $O(|\mathcal{B}| \log |\mathcal{B}|)$ time. Next we follow the scheme described in the preceding section. Each subproblem that arises

there is of the following type: We have two subsets $\mathcal{B}_v \subseteq \mathcal{B}$, $\mathcal{R}_v \subseteq \mathcal{R}$, and our goal is to find all triangles $\Delta \in \mathcal{R}_v$ that intersect the union U_v of the triangles in $\mathcal{B}_v$, and, for each such triangle Δ, to report a witness triangle of $\mathcal{B}_v$ that intersects Δ. Since $\mathcal{B}$ has the common point property, U_v is a star-shaped polygon, with respect to the common point z, whose complexity is linear in $n_v = |\mathcal{B}_v|$. To see this, note that the angle at which any side of a triangle in $\mathcal{B}_v$ is seen from z is at least δ. Hence, if we draw a collection of $O(1/\delta)$ rays from z, at evenly-spaced orientations, we can guarantee that each side of a triangle in $\mathcal{B}_v$ meets at least one of these rays. It follows that the intersection of ∂U_v with any wedge between two adjacent rays from z can be regarded as a portion of the boundary of a single face in an arrangement of n_v rays and lines (see Figure 8). As is well known [6], the complexity of such a face is $O(n_v)$. Hence the overall complexity of ∂U_v is $O(\frac{1}{\delta} n_v)$. U_v can be easily constructed in time $O(n_v \log n_v)$, as in [6, 20]. (More details concerning this construction can be inferred from the description given in the following section.)

We can thus preprocess ∂U_v, in $O(n_v)$ time, into a data structure of linear size, for logarithmic-time ray-shooting queries, as in [10, 12, 21]. Then, for each $\Delta \in \mathcal{R}_v$, we pick a vertex w of Δ, and test, in $O(\log n_v)$ time, whether $w \in U_v$; this can be done by a binary search with the orientation of zw through the list of orientations of the segments connecting z to the vertices of U_v. If this is the case, we report that Δ intersects U_v, and the above binary search also easily yields a triangle in $\mathcal{B}_v$ containing w (e.g., we can take the triangle whose boundary appears along ∂U_v in the direction zw from z). Otherwise, w lies outside U_v, and we perform three ray shooting queries along the sides of Δ (where the first two are with the sides adjacent to w), to determine whether Δ intersects U_v; if an intersection is found then a witness triangle can also be easily produced.

Hence the whole procedure at node v of the tree takes time $O((|\mathcal{R}_v| + |\mathcal{B}_v|) \log |\mathcal{B}_v|)$, and, summing this over all nodes of the tree, the total running time of the second stage is $O((|\mathcal{R}| + |\mathcal{B}|) \log^2 |\mathcal{B}|)$. Putting everything together, we have shown:

Theorem 3.3 *The 2-dimensional linear-extension problem for a collection of n δ-fat triangles can be solved in time $O(n \log^5 n)$, where the constant of proportionality is $O(\frac{1}{\delta^9} \log \frac{1}{\delta})$. Hence, within the same asymptotic time bound, we can also compute a depth order (if it exists) for a collection of n non-intersecting α-fat triangles in 3-space, for any $\alpha > 1$, where the constant of proportionality is $O(\alpha^9 \log \alpha)$.*

4 General Fat Objects

In this section we apply our technique to a collection $\mathcal{C} = \{c_1, \ldots, c_n\}$ of n convex, simply-shaped α-fat objects with the α-ratio property, as defined in the introduction. For each $c_i \in \mathcal{C}$, let s_i^+ be a smallest axis-parallel square containing c_i and let s_i^- be a largest axis-parallel square contained in c_i; denote the edge lengths of s_i^+, s_i^-, respectively, as ρ_i^+, ρ_i^-. By assumption, we have $\rho_i^+ \leq \alpha \rho_j^-$, for any pair of (not necessarily distinct) objects $c_i, c_j \in \mathcal{C}$.

Put $\mathcal{S}^+ = \{s_1^+, \ldots, s_n^+\}$. Clearly, for any pair of objects $c_i, c_j \in \mathcal{C}$, $c_i \cap c_j \neq \emptyset$ implies that $s_i^+ \cap s_j^+ \neq \emptyset$. Moreover, two axis-parallel squares intersect if and only if a vertex of one of them lies inside the other. This observation allows us, using orthogonal range searching techniques as in the preceding section, to obtain all pairs of intersecting squares in $\mathcal{S}^+$ as a collection of complete bipartite graphs $\{\mathcal{S}_i' \times \mathcal{S}_i''\}$, where the total size of their vertex sets is $O(n \log^2 n)$, and where one set of each such pair has the common point property (all squares in that set contain a vertex of some square in the second set of the pair). This representation can be constructed in time $O(n \log^2 n)$.

Now replace each such pair $(\mathcal{S}_i', \mathcal{S}_i'')$ by the corresponding pair $(\mathcal{R}_i, \mathcal{B}_i)$ of subsets of $\mathcal{C}$. Property (i) of Section 2 clearly holds for the new collection, but property (ii) (the common point property) may fail to hold. This property can be enforced as follows. Let p be a point that lies in, say $\bigcap \mathcal{S}_i''$. Then all the squares in $\mathcal{S}_i''$, and thus also all the objects in $\mathcal{B}_i$, are contained in an axis-parallel square s centered at p whose edge length is $2\rho_{max}$ (where $\rho_{max} = \max_j \rho_j^+$). Partition s into $l = O(\alpha^2)$ small squares of edge length less than $\rho_{min}/2$, where $\rho_{min} = \min_j \rho_j^-$. This induces a partitioning of $\mathcal{B}_i$ into l subsets $\mathcal{B}_{i,1}, \ldots, \mathcal{B}_{i,l}$, so that for each of these subsets $\mathcal{B}_{i,k}$, all sets in $\mathcal{B}_{i,k}$ fully contain one of the smaller squares. Hence each $\mathcal{B}_{i,k}$ clearly has the common point property. We replace the pair $(\mathcal{R}_i, \mathcal{B}_i)$ by the l pairs $(\mathcal{R}_i, \mathcal{B}_{i,1}), \ldots, (R_i, \mathcal{B}_{i,l})$. Note that this increases the overall size of the vertex sets of our bipartite graphs by only a factor of $O(\alpha^2)$.

We next consider the implementation of the second stage. As described in Section 2, we need to solve efficiently the following subproblem: We are given two sets, $\mathcal{R}$, $\mathcal{B}$, of objects of $\mathcal{C}$, so that $\mathcal{B}$ has the common point property. For each $r \in \mathcal{R}$, we want to determine whether it intersects any object in $\mathcal{B}$, and, if so, find a witness $b \in \mathcal{B}$ that intersects r. Put $p = |\mathcal{R}|$, $q = |\mathcal{B}|$.

We form the union U of all objects in $\mathcal{B}$. Since all these objects contain a common point, z, and are convex, we can represent each $b \in \mathcal{B}$ as a function $f_b(\theta)$, in polar coordinates about z, where $f_b(\theta)$ is the distance from z to the point of intersection of ∂b with the ray emerging from z in direction θ. Then the boundary of the union U is the graph, in polar coordinates, of the upper envelope $f_U(\theta) = \max_{b \in \mathcal{B}} f_b(\theta)$ of these functions. If s is the maximum number of intersections between the boundaries of any pair of objects in $\mathcal{B}$, then the number of connected portions of the boundaries of the objects of $\mathcal{B}$ that constitute the boundary of U is $O(\lambda_s(q))$, where $\lambda_s(n)$ is the maximum length of (n, s) Davenport-Schinzel sequences, and, under an appropriate model of computation, f_U (and thus U) can be computed in time $O(\lambda_s(q) \log q)$ [18]. (Note that here we did not make use of the fact that the given objects are fat; we only needed the common point property and the assumption that the objects are simply shaped.)

We next partition $\mathcal{R}$ into $\left\lceil p/\sqrt{\lambda_s(q)} \right\rceil$ subsets, each of size at most $\sqrt{\lambda_s(q)}$. For each subset $\mathcal{R}^{(j)}$, we perform a line-sweeping procedure on the collection of all arcs constituting ∂U and all arcs constituting the boundaries ∂r of the objects $r \in \mathcal{R}^{(j)}$. The goal of this procedure is to detect intersections between ∂U and the boundaries ∂r, and also to

detect containments of boundaries ∂r within U. When the sweep encounters an intersection between two boundaries ∂r, $\partial r'$, it processes this intersection in a standard manner (there are only $O(\lambda_s(q))$ such intersections). However, when the sweep reaches an intersection between an arc of ∂U and some ∂r, or when it reaches the leftmost point of some ∂r and finds that it lies inside the union U, the algorithm first reports that r intersects the union, and finds a witness object of $\mathcal{B}$ that r intersects (as in the case of fat triangles described in Section 3), but then it removes ∂r from the collection of arcs being processed. This guarantees that the total number of events processed by the sweep is only $O(\lambda_s(q))$, and the whole procedure thus takes $O(\lambda_s(q)\log q)$ time. We repeat this procedure for all sets $\mathcal{R}^{(j)}$, and thus solve this subproblem of the second stage in time $O((p/\sqrt{\lambda_s(q)}+1)\lambda_s(q)\log q)$. (Similar ideas were used in [3].)

Recall that in the second stage of the algorithm we need to solve the above subproblem for each pair of subsets $(\mathcal{R}_v, \mathcal{B}_v)$ at each node v of the tree constructed for each of the pairs $(\mathcal{R}_i, \mathcal{B}_i)$ generated in the first stage. Summing the cost of the above procedure over all these subproblems, we conclude that the overall cost of stage 2 is

$$\sum_{(\mathcal{R}_i,\mathcal{B}_i)} O((|\mathcal{R}_i|/\sqrt{\lambda_s(|\mathcal{B}_i|)}+1)\lambda_s(|\mathcal{B}_i|)\log^2 n) = O(\alpha^2 n\lambda_s^{1/2}(n)\log^4 n)\ .$$

We have thus shown:

Theorem 4.1 *The 2-dimensional linear-extension problem for a collection $\mathcal{C}$ of n convex, simply-shaped, α-fat objects with the α-ratio property can be solved, in an appropriate model of computation, in time $O(n\lambda_s^{1/2}(n)\log^4 n)$, where s is the maximum number of intersections between the boundaries of any pair of objects in $\mathcal{C}$. Hence, within the same time bound, we can also compute a depth order (if it exists) for a collection of n non-intersecting convex objects in 3-space, whose xy-projections have the above properties. The constant of proportionality is bounded by $c\alpha^2$, for some absolute constant c.*

Remark 1. While the cost of the implementation of the first stage in the case of such objects is rather low, i.e., close to linear, the cost of the implementation of the second stage is relatively high, i.e., close to $n^{3/2}$. More precisely, the costly part of the second stage is the procedure for 'merging' the sets $\mathcal{R}$ and $\mathcal{B}$, for the pairs $(\mathcal{R},\mathcal{B})$ obtained in the first stage. In some special cases, however, it is possible to perform this merge much more efficiently. One such interesting case is that of disks with the α-ratio property, which we consider in the next section.

Remark 2. Consider the following related problem. Given a collection of n non-intersecting convex simply-shaped α-fat objects in 3-space, preprocess it for *vertical ray shooting* queries, that is, preprocess it so that, for a given query point, the object lying immediately under it (if exists) can be found efficiently. Using techniques similar to those used above, we can solve the vertical ray shooting problem with nearly linear preprocessing time and storage and $O(\text{polylog}(n))$ query time. Thus this problem seems to be easier than the problem of computing a depth order. This result will be presented in a forthcoming companion paper.

5 Disks

In this section we apply our technique to a collection $\mathcal{C} = \{c_1, \ldots, c_n\}$ of n disks in the plane. The implications of the results of this section to the problem of computing the depth order are not very interesting, since the depth order of a collection of non-intersecting horizontal disks in 3-space can be computed easily by sorting the planes containing the disks by their z-coordinates. Nevertheless, the results are of interest in the more general context of the 2-dimensional linear-extension problem that we are studying. We first sort the disks in $\mathcal{C}$ according to their radii. Let $\mathcal{L}$ (resp. $\mathcal{H}$) denote the subset of disks of $\mathcal{C}$ that appear in the smaller (resp. larger) half of the sorted sequence. Our strategy is to interact $\mathcal{L}$ with $\mathcal{H}$, in a manner that will be described below, and then handle recursively the sets $\mathcal{L}$ and $\mathcal{H}$. For each $c_i \in \mathcal{L}$, let s_i^+ be the axis-parallel square circumscribing c_i; denote by $\mathcal{S}_\mathcal{L}^+$ the set of these squares. Clearly, for any pair of disks $c_1 \in \mathcal{L}$ and $c_2 \in \mathcal{H}$, $c_1 \cap c_2 \neq \emptyset$ implies that either (i) the boundary of c_2 intersects the boundary of s_1^+, or (ii) c_2 contains s_1^+. In other words, $c_1 \cap c_2 \neq \emptyset$ implies that there exists a side of s_1^+ that intersects c_2.

We now compute a compact representation of the set of all side-disk intersections, for a side of a square in $\mathcal{S}_\mathcal{L}^+$ and a disk in $\mathcal{H}$. We first compute such a representation for the intersections involving vertical sides, and then compute, in a similar manner, a representation for the intersections involving horizontal sides. Let $\mathcal{Y}_\mathcal{L}^+$ denote the set of vertical sides of squares in $\mathcal{S}_\mathcal{L}^+$. According to an observation of van Kreveld et al. [30], a side $v \in \mathcal{Y}_\mathcal{L}^+$ intersects a disk $c \in \mathcal{H}$ iff either (i) the center of c lies inside the horizontal slab defined by the horizontal lines passing through the endpoints of v, and the line containing v intersects c, or (ii) one of the endpoints of v lies inside c. Since all the sides in $\mathcal{Y}_\mathcal{L}^+$ are vertical, we can report all intersections of the first type using standard orthogonal range-searching techniques in close to linear time. The costly part is reporting the intersections of the second type. This can be done using more sophisticated range searching techniques, such as the techniques of Agarwal and Matoušek [2], Matoušek [23] and of Chazelle et al. [13], which take time $O(n^{7/5+\varepsilon})$, for any $\varepsilon > 0$; this is also a bound on the total size of the sets in the resulting representation. Taking into account the recursive handling of $\mathcal{L}$ and of $\mathcal{H}$, we obtain an overall collection of pairs of sets $\{(\mathcal{R}_j, \mathcal{B}_j)\}$ that covers all intersecting pairs of disks in $\mathcal{C} \times \mathcal{C}$, where the total size of all these sets is still $O(n^{7/5+\varepsilon})$, for any $\varepsilon > 0$.

Next we need to refine the pairs $(\mathcal{R}_j, \mathcal{B}_j)$ produced in the first stage of the algorithm so as to enforce the common point property. For this, we adapt the technique used in the case of fat triangles. Let $(\mathcal{R}, \mathcal{B})$ be one of the pairs produced above, and let c_0 be a fixed disk in $\mathcal{R}$. Each disk in $\mathcal{B}$ must contain a point of the square s_0^+, whose side length is $2r_0$, where r_0 is the radius of c_0. Let z be the center of s_0^+, and let s_0^{++} be an axis-parallel square of side length $4r_0$ centered at z. It is easy to verify that for each disk $c \in \mathcal{B}$, we have $area(c \cap s_0^{++}) \geq \beta \cdot area(s_0^{++})$, for some constant fraction β (this follows from the fact that c_0 is smaller than the disks in $\mathcal{B}$). Therefore, there exist a constant number of points within s_0^{++} so that each disk of $\mathcal{B}$ contains at least one of these points. We can thus split $\mathcal{B}$ into a constant number of subfamilies $\mathcal{B}_1, \ldots, \mathcal{B}_l$, each having the common point property, and

replace the pair $(\mathcal{R}, \mathcal{B})$ by the l corresponding pairs $(\mathcal{R}, \mathcal{B}_1), \ldots, (\mathcal{R}, \mathcal{B}_l)$, without affecting the other properties.

The second stage of the algorithm is implemented in a way that is very similar to the one described for the case of fat triangles. The only difference is that here we need to perform circle shooting into a collection of circular arcs that form the boundary of a star-shaped region, rather than ray shooting. This can be done in polylogarithmic time, using the results of [4]. Putting everything together, we obtain an algorithm for the 2-dimensional linear-extension problem for a collection of n disks, whose running time is $O(n^{7/5+\varepsilon})$.

Finally, note that if the collection $\mathcal{C}$ of disks has the α-ratio property then we can do much better. In this case we can implement the first stage as in the case of general fat objects (with the α-ratio property), and the second stage as in the case of general disks, as just obtained. This approach yields an algorithm for this special case whose running time is $O(n \log^7 n)$.

Theorem 5.1 *The 2-dimensional linear-extension problem for a collection $\mathcal{C}$ of n disks can be solved in $O(n^{7/5+\varepsilon})$ time, for any $\varepsilon > 0$. Moreover, if $\mathcal{C}$ has the α-ratio property, this problem can be solved in $O(n \log^7 n)$ time, where the constant of proportionality is bounded by $c\alpha^2$, for some constant c.*

6 Conclusion

In this paper we have presented a new technique for computing depth orders for several useful classes of objects in 3-space. In the case of fat triangles we obtain a considerably improved (and simpler) solution than the previous algorithm of [16], and for the case of general fat objects our solution is the first known subquadratic algorithm.

The paper raises, however, many open problems. First, can our solution for the case of general fat objects be improved to run in close to $O(n^{4/3})$ time? Second, can we dispense with the α-ratio requirement, and obtain an efficient solution for the case of arbitrary fat objects (as we did in the case of triangles)? Finally, can we fine-tune our technique so that it can also detect cycles in the depth order?

References

[1] P.K. Agarwal, H. Edelsbrunner, O. Schwarzkopf and E. Welzl, Euclidean minimum spanning trees and bichromatic closest pairs, *Discrete Comput. Geom.* 6 (1991), 407–422.

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

[3] P.K. Agarwal and M. Sharir, Red-blue intersection detection algorithms, with applications to motion planning and collision detection, *SIAM J. Computing* 19 (1990), 297–321.

[4] P.K. Agarwal and M. Sharir, Circle shooting in a simple polygon, *J. Algorithms* 14 (1993), 68–87.

[5] P.K. Agarwal, M. Sharir and S. Toledo, Applications of parametric searching in geometric optimization, *J. Algorithms* 17 (1994), in press.

[6] P. Alevizos, J.D. Boissonnat and F.P. Preparata, An optimal algorithm for the boundary of a cell in a union of rays, *Algorithmica* 5 (1990), 573–590.

[7] H. Alt, R. Fleischer, M. Kaufmann, K. Mehlhorn, S. Näher, S. Schirra and C. Uhrig, Approximate motion planning and the complexity of the boundary of the union of simple geometric figures, *Algorithmica* 8 (1992), 391–406.

[8] P. Callahan and S. Kosaraju, A decomposition of multi-dimensional point-sets with applications to k-nearest-neighbors and n-body potential fields, *Proc. 24th ACM Symp. Theory of Computing*, 1992, 546–556.

[9] P. Callahan and S. Kosaraju, Faster algorithms for some geometric graph problems in higher dimensions, *Proc. 4th ACM-SIAM Symp. Discrete Algorithms*, 1993, 291–300.

[10] B. Chazelle, H. Edelsbrunner, M. Grigni, L. Guibas, J. Hershberger, M. Sharir and J. Snoeyink, Ray shooting in polygons using geodesic triangulations, *Algorithmica* 12 (1994), 54–68.

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

[12] B. Chazelle and L. Guibas, Visibility and intersection problems in plane geometry, *Discrete Comput. Geom.* 4 (1989), 551–581.

[13] B. Chazelle, M. Sharir and E. Welzl, Quasi-optimal upper bounds for simplex range searching and new zone theorems, *Algorithmica* 8 (1992), 407–429.

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

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

[16] M. de Berg, M. Overmars and O. Schwarzkopf, Computing and verifying depth orders, *SIAM J. Computing* 23 (1994), 437–446.

[17] A. Efrat, G. Rote and M. Sharir, On the union of fat wedges and separating a collection of segments by a line, *Comp. Geom. Theory and Appls* 3 (1993), 277–288.

[18] S. Hart and M. Sharir, Nonlinearity of Davenport–Schinzel sequences and of generalized path compression schemes, *Combinatorica* 6 (2) (1986), 151–177.

[19] D. Hearn and M.P. Baker, *Computer Graphics*, Prentice-Hall International, 1986.

[20] J. Hershberger, Finding the upper envelope of n line segments in $O(n \log n)$ time, *Inf. Proc. Letters* 33 (1989), 169–174.

[21] J. Hershberger and S. Suri, A pedestrian approach to ray shooting: Shoot a ray, take a walk, *Proc. 4th ACM-SIAM Symp. Discrete Algorithms*, 1993, 54–63.

[22] M.J. Katz, M.H. Overmars, M. Sharir, Efficient hidden surface removal for objects with small union size, *Comp. Geom. Theory and Appls* 2 (1992), 223–234.

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

[24] J. Matoušek, J. Pach, M. Sharir, S. Sifrony and E. Welzl, Fat triangles determine linearly many holes, *SIAM J. Computing* 23 (1994), 154–169.

[25] K. Mehlhorn, *Data Structures and Algorithms 3: Multidimensional Searching and Computational Geometry*, Springer-Verlag, Berlin, 1984.

[26] N. Miller and M. Sharir, Efficient randomized algorithms for constructing the union of fat triangles and of pseudodiscs, manuscript, 1991.

[27] F. Preparata and M. Shamos, *Computational Geometry – an introduction*, Springer-Verlag, New York, 1985.

[28] M. Sharir and P.K. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications*, Cambridge University Press, to appear.

[29] M. van Kreveld, On fat partitioning, fat covering, and the union size of polygons, *Proc. 3rd Workshop on Algorithms and Data Structures*, 1993, 452–463.

[30] M. van Kreveld, M. Overmars and P.K. Agarwal, Intersection queries in sets of disks, *Bit* 32 (1992), 268–279.