Output-Sensitive Construction of the Union of Triangles *

Esther Ezra and Micha Sharir
School of Computer Science
Tel Aviv University
{estere,michas}@post.tau.ac.il

May 9, 2004

Abstract

We present an efficient algorithm for the following problem: Given a collection $T = \{\Delta_1, \ldots, \Delta_n\}$ of n triangles in the plane, such that there exists a subset $S \subset T$ (unknown to us), of $\xi \ll n$ triangles, such that $\bigcup_{\Delta \in S} \Delta = \bigcup_{\Delta \in T} \Delta$, construct efficiently the union of the triangles in T. We show that this problem can be solved in randomized expected time $O(n^{4/3} \log n + n\xi \log^2 n)$, which is subquadratic for $\xi = o(n/\log^2 n)$. In our solution, we use a variant of the method of Brönnimann and Goodrich [10] for finding a set cover in a set system of finite VC-dimension. We present a detailed implementation of this variant, which makes it run within the asserted time bound. Our approach is fairly general, and we show that it can be extended to compute efficiently the union of simply shaped bodies of constant description complexity in $\mathbb{R}^d$, when the union is determined by a small subset of the bodies.

1 Introduction

Many problems in computational geometry involve the task of constructing the boundary of the union of n geometric objects in the plane or in higher dimensions. Problems of this kind include motion planning [22], where we wish to construct the forbidden portions of the configuration space; hidden surface removal for visibility problems in three dimensions [27]; finding the minimal Hausdorff distance between two sets of points (or of segments) in $\mathbb{R}^2$ [19]; applications in geographic information systems [13], and many others. In this paper, we focus mainly on the problem of constructing the union of n triangles in $\mathbb{R}^2$, but we also show that our algorithm can be extended to other geometric objects in the plane and in higher dimensions.

Computing the union by constructing the full arrangement of the n input triangles requires $\Theta(n^2)$ time in the worst case, which, in many instances, is wasteful, since the combinatorial

*Work on this paper has been supported by NSF Grants CCR-97-32101 and CCR-00-98246, by a grant from the U.S.-Israeli Binational Science Foundation, by a grant from the Israel Science Fund, Israeli Academy of Sciences, for a Center of Excellence in Geometric Computing at Tel Aviv University, and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University.

complexity of the union boundary might be considerably smaller. Nevertheless, an algorithm for this problem that runs in subquadratic time when the boundary of the union has subquadratic complexity[1] is unlikely to exist, since this problem belongs to the family of *3SUM-hard* problems [18], which are problems that are very likely to require $\Omega(n^2)$ time in the worst case; see below for more details.

However, subquadratic algorithms exist in several special cases, such as the case of *fat* triangles (namely, every angle of each triangle is at least some constant positive angle), or of triangles that arise in the union of Minkowski sums of a fixed convex polygon with a set of pairwise disjoint convex polygons (which is the problem one faces in translational motion planning of a *convex* polygon). In these cases, the union has only linear or near-linear complexity [20, 23, 24], and more efficient algorithms, based on either deterministic divide-and-conquer, or on randomized incremental construction, can be devised, and are presented in the above-cited papers.

If the input consists of general triangles, then the complexity of the union can be $\Theta(n^2)$ in the worst case. If it happens to be smaller, one can attempt to compute the union by employing the randomized incremental construction (RIC) of Agarwal and Har-Peled [1], whose analysis is based on Mulmuley's *theta series* [27]. Briefly, the algorithm inserts the triangles one at a time in a random order, and maintains the union incrementally, updating it after each insertion. As is well known (and discussed in [15]), the RIC algorithm has good performance, even when the size of the arrangement is quadratic, provided that the *depth* $d(v)$ (i.e., the number of input triangles containing v in their interior) of most of the vertices v in the arrangement induced by the n input triangles is large enough. We refer to such vertices as being *deep*. Otherwise, when most of the vertices in the arrangement are *shallow*, the RIC algorithm performs poorly. In this case, one can employ the *Disjoint Cover* (DC) algorithm, proposed in [15], which has good performance in practice. This algorithm also inserts the triangles one at a time, but it computes an insertion order that attempts to cover as many shallow vertices as possible in each insertion step. However, from a theoretical point of view (and in view of certain pathological examples, presented in [15]), the DC algorithm can produce $\Omega(n^2)$ vertices of the arrangement, even if the size of the output (i.e., the number of vertices on the boundary of the union) is only linear or constant, and it can be beaten by the RIC algorithm in such cases.

Output sensitivity. In this paper we present an efficient algorithm that computes the union in an "output-sensitive" manner. There are two obvious ways to define output sensitivity. The first is to measure the output size in terms of the size of the smallest subset $S \subset T$ that satisfies $\bigcup S = \bigcup T$, where $\bigcup S$ (resp., $\bigcup T$) denotes the union of the triangles in S (resp., in T). The second measure is in terms of the size of the smallest subset S' such that $\partial \bigcup T \subseteq \partial \bigcup S'$. See Figure 1 for an illustration of the two measures. Note that if the output size is ξ, according to either measure, the actual complexity of the union may be as large as $\Theta(\xi^2)$ (but not larger).

The second measure of output size is likely to be too weak. Indeed, consider the reduction, as presented in [18], of an instance of 3SUM (namely, the problem of determining whether there exist $a \in A$, $b \in B$, $c \in C$ satisfying $a+b+c=0$, for three given sets A, B, C of real numbers)

[1] This is one variant of *output sensitivity* that one may wish to attain. In this paper we use a different notion of output sensitivity, described later in the introduction.

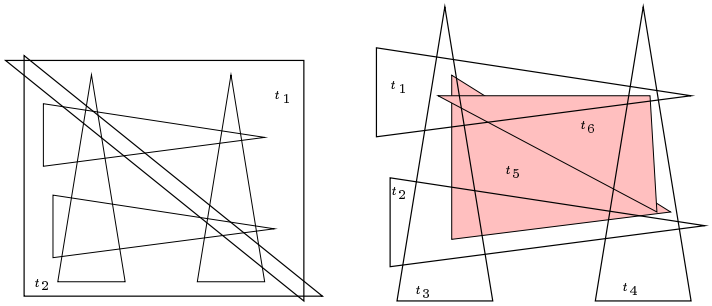

Figure 1: (a) An arrangement of six triangles, illustrating the first measure of output sensitivity. The triangles t_1 and t_2 cover the entire union, so the output size is 2. (b) Illustrating the second measure of output sensitivity. The union boundary is determined only by the triangles $t_1, \ldots, t_4$, even though the triangles t_5 and t_6 cover the hole created by $\bigcup_{i \leq 4} t_i$. The output size is 4 according to the second measure, and 6 according to the first one.

to an instance of the problem of determining whether the union of a given set of triangles fully covers the unit square. We can further reduce this latter problem to our problem, as follows. Let A denote an algorithm that efficiently computes the union of n triangles in the plane, in terms of the second measure, and let $T_A(n, \xi)$ denote its running time, expressed as a function of n and of the "output size" ξ. We assume that $T_A(n, \xi) = o(n^2)$ when $\xi = o(n)$. In order to determine efficiently whether the given triangles fully cover the unit square, we consider only the portions of the triangles that are contained in the unit square, and retriangulate them, if necessary. In addition, we add four thin and narrow triangles that cover the boundary of the unit square. We now run A on the newly constructed instance. Clearly, there are no holes in the union of the newly created triangles if and only if the original union contains the unit square. In this case, the boundary of the new union consists of only four triangles, and thus A will terminate in a predictable subquadratic time. We thus run A. If it terminates within the anticipated (subquadratic) time, we can determine, at no extra cost, whether the union covers the unit square. Otherwise, we stop A, and correctly report that the union of the original triangles does not cover the unit square. Hence an efficient output-sensitive solution, under the second measure, would have yielded a subquadratic solution to 3SUM, and is thus unlikely to exist.

In contrast, the first measure does lend itself to an efficient output-sensitive solution, which is the main result of this paper.

Our results. Specifically, we present an efficient algorithm to construct the boundary of the union of a set $T = \{\Delta_1, \ldots, \Delta_n\}$ of n triangles in the plane, under the assumption that there exists a subset $S \subset T$ of $\xi \ll n$ triangles (unknown to us) such that $\bigcup S = \bigcup T$. We present an algorithm, whose running time is $O(n^{4/3} \log n + n\xi \log^2 n)$, which is subquadratic when $\xi = o(n/\log^2 n)$. Our approach is a randomized algorithm, based on the method of Brönnimann and Goodrich for finding a set cover or a hitting set in a set system of finite VC-dimension, as presented in [10] (see Section 2.1 for a brief review of this method). In our

case, the objects are the triangles of T, and each vertex v of the arrangement $\mathcal{A}(T)$ defines a set $T_v = \{\Delta \in T \mid v \in int(\Delta)\}$. A *hitting set* for this system is a set $S \subset T$ such that $\bigcup S = \bigcup T$, and thus a minimum-size hitting set is the object that we wish to compute. In general, the Brönnimann-Goodrich technique is not efficient enough for our purposes, but we use a variant of the algorithm which can be implemented efficiently. Specifically, we apply the algorithm of Brönnimann and Goodrich in an "approximate setting", fine-tuning it (using randomization) so that it constructs a subset T' of $O(\xi \log \xi)$ triangles of T, whose union covers the overwhelming majority of the vertices in the arrangement $\mathcal{A}(T)$. This allows us, with some care, to compute the portion of $\bigcup T$ that lies outside $\bigcup T'$ in an efficient explicit manner. We note that, when measuring the expected number of vertices generated by the algorithm, it suffices (and is appropriate) to consider only vertices at positive depth, since vertices at depth 0 are the vertices of the union, and they have to be constructed by any algorithm that computes the union. We call the latter quantity, namely the number of positive-depth vertices generated by the algorithm, the *residual cost* of the algorithm.

In Section 2.1 we briefly recall the algorithm of Brönnimann and Goodrich, and present our approximate version of it. Then we derive an upper bound on the expected residual cost of the algorithm in its approximate version. Section 3 describes a detailed implementation of our algorithm. In this implementation, we use generic and simple techniques, that can be easily extended to other geometric objects of constant description complexity[2] in the plane and in $\mathbb{R}^d$. These extensions are discussed in Section 4. We give concluding remarks and suggestions for further research in Section 5.

2 The Union Construction as a Set Cover Problem

2.1 An overview of the Brönnimann-Goodrich technique

A technique for finding a set cover of a set system of finite VC-dimension is described in detail by Brönnimann and Goodrich [10]; for the sake of completeness, we provide a brief overview of this approach, in the context of the union construction problem.

We denote by V the set of vertices of the arrangement $\mathcal{A}(T)$ at positive depth (considering only intersection points of the triangle boundaries and ignoring triangle vertices). Our set system is dual to the set system (V, T), and is defined as (T, V^*), where

$$V^* = \{T_v : v \in V\},$$

and where T_v consists of all the triangles $\Delta \in T$ that contain v in their interior. Since this set system is dual to (V, T), which has some finite VC-dimension d (see, e.g., [6]), it follows that the VC-dimension of (T, V^*) is also finite; as a matter of fact, it does not exceed 2^{d+1} [8]. As already mentioned, out goal is to find a *hitting set* for (T, V^*), that is, a subset $H \subseteq T$ that has a nonempty intersection with every set $T_v \in V^*$, $v \in V$.

[2] A set in $\mathbb{R}^d$ is said to have *constant description complexity* if it is a semi-algebraic set defined as a Boolean combination of a constant number of polynomial equalities and inequalities of constant maximum degree in a constant number of variables.

The algorithm of Brönnimann and Goodrich finds a hitting set, whose size is $O(h^* \log h^*)$, where h^* is the smallest size of any hitting set. Note that the reported hitting set is actually a *set cover* for the primal set system (V, T), where a set cover, in this case, is a collection $\mathcal{C} \subseteq T$ of triangles, whose union covers the entire set V. (For technical reasons, the method of Brönnimann and Goodrich computes a set cover via a hitting set of the dual set system, which is why we also work with the dual system; see [10] for further details.) Since, by definition, the size of the optimal cover is assumed to be ξ, it follows that the size of the set cover reported by the algorithm is at most $O(\xi \log \xi)$.

We first describe the algorithm of Brönnimann and Goodrich in its "ideal setting", where the entire set V is given, and then show how to modify this setting, so that it suffices to consider only a small subset of vertices.

The Brönnimann-Goodrich algorithm has two key subroutines: (i) A *net finder* $\mathcal{F}$ for (T, V^*), which is an algorithm that, given a parameter $r \geq 1$ and a weight distribution w on T, computes a $(1/r)$-*net* for the weighted system (T, V^*) [6]. A $(1/r)$-net is a subset $N \subseteq T$, which has a nonempty intersection with each set in V^* whose total weight is at least $1/r$ of the total weight of T. (ii) A *verifier* $\mathcal{V}$, that, given a subset $H \subseteq T$, either states (correctly) that H is a hitting set, or returns a nonempty "witness" set $T_v \in V^*$, for some v, such that $T_v \cap H = \emptyset$. In our context, $\mathcal{V}$ has simply to output a vertex $v \in V$ which is not contained in the interior of $\bigcup H$.

The Brönnimann-Goodrich algorithm then proceeds as follows. We guess the value of ξ (homing in on the right value using an exponential search). We assign weights to the triangles in T. Initially, all weights are 1. We then use the net finder $\mathcal{F}$ to construct a $(1/2\xi)$-net N for (T, V^*). If the verifier $\mathcal{V}$ outputs some set T_v that N does not hit, we double the weights of the triangles in T_v, and repeat the process with the new weights. As shown in [10], a hitting set is found after at most $4\xi \log(n/\xi)$ iterations.

The problem with this ideal setting is that it requires the construction of all the (positive-depth) vertices of $\mathcal{A}(T)$, which is much too much to ask for, since it can be too expensive (V can be quadratic in the worst case, while ξ can still be very small). Instead, we use a smaller randomly sampled subset $R \subseteq V$ of r elements, whose actual computation is presented in Section 3. We then feed the verifier $\mathcal{V}$ with R instead of the entire set V. We show that once the verifier $\mathcal{V}$ announces that the subset H, reported by the net finder $\mathcal{F}$, covers R, the actual number of vertices of V that remain uncovered is relatively small, with high probability. We then compute the uncovered vertices in an explicit manner, and thereby complete the construction of $\bigcup T$.

2.2 A subquadratic residual cost via sampling

We begin the analysis of our implementation of the Brönnimann-Goodrich technique with the following lemma, which provides a lower bound for the size of the sample R, which is sufficient to guarantee the property asserted at the end of the preceding subsection.

In what follows, we say that an event occurs with *overwhelming probability* (or w.o.p., for short), if the probability that it does not occur is at most $\frac{1}{n^c}$, for some constant $c \geq 1$.

Lemma 2.1 *Let $T = \{\Delta_1, \ldots, \Delta_n\}$ be a given collection of n triangles in the plane, let V denote the set of vertices of the arrangement $\mathcal{A}(T)$ at positive depth, let κ denote the size of V, and suppose that there are only ξ triangles of T whose union is equal to $\bigcup T$. Let $S \subseteq T$ denote a subset of triangles, and let $R \subseteq V$ be a random sample of $r = \Omega(t \log n)$ positive-depth vertices sampled after S has been fixed, for some parameter $t \geq 1$ and with a sufficiently large constant of proportionality. If S covers all but $r_{_S} < r$ vertices of R, then, w.o.p., the actual number $\kappa_{_S}$ of vertices of V that are not covered by the elements of S satisfies*

$$\kappa_{_S} \leq \max\left\{\frac{\kappa}{t}, \beta\frac{\kappa}{r}r_{_S}\right\}, \tag{1}$$

for some absolute constant $\beta > 1$.

Proof: For simplicity of exposition, we present the analysis under the model where R is obtained by drawing each point of V independently with probability $p = \frac{r}{\kappa}$. Nevertheless, the assertion of the lemma also holds for other models of sampling R, in particular, for the model we use in the actual implementation of the algorithm; see Section 3 and Appendix A for details. Since each point in $V \setminus \bigcup S$ is chosen independently with probability $\frac{r}{\kappa}$, the expected number of vertices of R that are not covered by S is $\frac{r}{\kappa}\kappa_{_S}$.

It suffices to consider the case $\kappa_{_S} > \frac{\kappa}{t}$, for otherwise (1) clearly holds.

Since R is sampled *after* S has been fixed, the number $r_{_S}$ of vertices of R that are not covered by $\bigcup S$ is a random variable, which can be expressed as the sum of $\kappa_{_S}$ mutually independent indicator variables, $X_1, \ldots, X_{\kappa_{_S}}$, each satisfying

$$Pr[X_i = 1] = p; \;\; Pr[X_i = 0] = 1 - p, \;\; \text{for } i = 1, \ldots, \kappa_{_S}.$$

Fix a parameter $r_0 > 0$, and consider the event

$$A_S: \quad r_{_S} - \frac{r}{\kappa}\kappa_{_S} < -r_0.$$

Using a large deviation bound given in [6, Theorem A.13], it follows that

$$Pr[A_S] < e^{-\frac{{r_0}^2}{2\frac{r}{\kappa}\kappa_{_S}}}. \tag{2}$$

Putting $r_0 = \sqrt{2c_0\frac{r}{\kappa}\kappa_{_S}\log n}$, for some constant $c_0 \geq 1$, (2) implies that the probability that the event A_S does not occur is at most $\frac{1}{n^{c_0}}$. Hence, w.o.p.,

$$r_{_S} - \frac{r}{\kappa}\kappa_{_S} \geq -\sqrt{2c_0\frac{r}{\kappa}\kappa_{_S}\log n},$$

or

$$r_{_S} \geq \sqrt{\frac{r}{\kappa}\kappa_{_S}}\left[\sqrt{\frac{r}{\kappa}\kappa_{_S}} - \sqrt{2c_0\log n}\right].$$

Since we have assumed that $\kappa_{_S} > \frac{\kappa}{t}$, and that $r = \Omega(t\log n)$, with a sufficiently large constant of proportionality, it follows that, w.o.p.,

$$\sqrt{\frac{r}{\kappa}\kappa_{_S}} - \sqrt{2c_0\log n} > \alpha\sqrt{\frac{r}{\kappa}\kappa_{_S}}, \tag{3}$$

for some absolute constant $0 < \alpha < 1$, which implies that

$$\kappa_S \leq \frac{\kappa}{\alpha r} r_S,$$

and thus the lemma follows. □

Remarks: 1) Note that Lemma 2.1, as well as its variant discussed in the Appendix, deal with abstract sets, and do not exploit any special property of vertices in arrangements of triangles. We will therefore be able to use the lemma, more or less verbatim, in the extensions presented in Section 4.
2) We re-emphasize that Lemma 2.1 relies on the assumption that R is sampled *after* S has been chosen (in our implementation, this choice will also be random). In particular, for the lemma to be applicable at each iteration of the Brönnimann-Goodrich algorithm, R should be redrawn from scratch before applying the verifier $\mathcal{V}$. (See Section 3 for further details.)

Lemma 2.1 implies that if the triangles in S cover all the elements of R (and thus $r_S = 0$), then, w.o.p., $\kappa_S \leq \frac{\kappa}{t}$ (in fact, it is sufficient that $r_S = O\left(\frac{r}{t}\right)$). We thus construct the union of the input triangles in two steps, where in the first we find a set H of $O(\xi \log \xi)$ triangles that covers all but at most $\frac{\kappa}{t}$ vertices of V, and compute the union $\bigcup H$, and in the second we handle efficiently all the remaining vertices of V that H does not cover; see below for details. It thus follows that the overall expected number of positive depth vertices generated by the algorithm is $O(\xi^2 \log^2 \xi)$ (which is the number of vertices of the arrangement of the triangles in H) in the first part, and at most $\frac{\kappa}{t}$ in the second part.

In summary, we have shown

Theorem 2.2 *Let $T = \{\Delta_1, \ldots, \Delta_n\}$ be a given collection of n triangles in the plane, and assume that there exists a subset $H \subset T$ of $\xi \ll n$ triangles (unknown to us) such that $\bigcup H = \bigcup T$. Let V, κ and t be as in Lemma 2.1. Then one can implement the Brönnimann-Goodrich algorithm, so that its residual cost is $O(\xi^2 \log^2 \xi + \frac{\kappa}{t})$, w.o.p. In particular, for $t = \max\left\{\frac{\kappa}{\xi^2}, 1\right\}$, the residual cost is $O(\xi^2 \log^2 \xi)$.*

Discussion. Clearly, if our only concern is to have the algorithm generate as few positive-depth vertices as possible, we should choose t as large as possible, thereby making R larger, and the set of vertices of V not covered by H smaller. For example, as noted, if we choose $t = \max\left\{\frac{\kappa}{\xi^2}, 1\right\}$, then the residual cost of the algorithm is at most $O(\xi^2 \log^2 \xi)$, w.o.p. Since there are only ξ triangles that define the union, the combinatorial complexity of the boundary of the union is only $O(\xi^2)$. This implies that, for the above choice of t, the overall number of vertices that the algorithm generates is $O(\xi^2 \log^2 \xi)$, which is subquadratic for $\xi = o(n/\log n)$. However, if we are concerned with the actual running time, large values of t will slow down the algorithm, because sampling the sets R will be more expensive. Hence, in the actual implementation of the algorithm, presented in Section 3 below, we will choose a smaller value for t, in order to optimize the bound on the actual running time of the algorithm. This will also affect the bound on the residual cost.

We also note that the bound $O(\xi^2 \log^2 \xi)$ on the complexity of the union of the triangles

computed in the first part of the algorithm may be too pessimistic in practice. If the complexity of the union $\bigcup H$ turns out to be smaller, the residual cost will be smaller too.

3 Implementation of the Algorithm

The actual cost of the algorithm depends on the cost of several support routines (in addition to the cost of the actual generation of positive-depth vertices), such as (i) constructing the random samples R; (ii) finding a $(1/2\xi)$-net for the set system (T, V^*); (iii) implementing the verifier $\mathcal{V}$, which, in our case, is an algorithm that efficiently decides whether a given subset S of triangles covers another given subset R of positive-depth vertices; and (iv) the actual construction of the union of the input triangles, after an approximate hitting set has been found. We present here an implementation that uses generic and simple techniques, and yields a subquadratic output-sensitive algorithm for constructing the union.

In the following description, we denote by h the size of the set H computed in the first stage of the algorithm.

Sampling R

The task at hand is to construct, at each iteration of the algorithm, a random sample of (an expected number of) $r = ct \log n$ positive-depth vertices of $\mathcal{A}(T)$, for appropriate values of the parameter t and the constant c. (As already mentioned, and will be discussed below, we have to draw a new subset R in each iteration of the algorithm, in order to eliminate any dependence between the present subset of triangles reported by the net finder $\mathcal{F}$ and the (current) sample R.)

We sample R using the following simple-minded approach. Suppose that we have a guess for the values of ξ and κ (see below for details concerning these guesses). Let κ^* denote the number of vertices on the boundary of $\bigcup T$. If $\kappa = O(\kappa^*)$ then the entire arrangement has only $O(\kappa^*) = O(\xi^2)$ vertices, and can thus be constructed in time $O(n \log n + \xi^2)$, using any of the standard techniques [27]. We may thus assume that $\kappa \gg \kappa^*$. We also may assume that $\kappa \gg \max\{\xi^2, n^{4/3}\}$. Otherwise, we construct the entire arrangement in time $O((n + \xi^2 + n^{4/3}) \log n) = O((\xi^2 + n^{4/3}) \log n)$.

We now perform $\frac{c' r \binom{n}{2}}{\kappa}$ sampling steps, where in each step we choose, uniformly and independently, a pair of edges of distinct triangles in T, for an appropriate constant $c' > 1$. Clearly, a real vertex of the arrangement $\mathcal{A}(T)$ is chosen in a single step with probability $\frac{\kappa+\kappa^*}{9\binom{n}{2}}$, and thus the expectation of the number r' of pairs of edges that actually intersect is

$$\frac{\kappa + \kappa^*}{9\binom{n}{2}} \cdot \frac{c' r \binom{n}{2}}{\kappa} = \Theta(r).$$

Using the same deviation bound, as shown in Lemma 2.1, it can be shown that, w.o.p., the

actual number of such pairs satisfies

$$r' \geq \mathbf{E}(r') - \sqrt{\gamma \frac{\kappa}{\binom{n}{2}} \frac{r\binom{n}{2}}{\kappa} \log n} = \mathbf{E}(r') - \sqrt{\gamma r \log n},$$

for some constant $\gamma \geq 1$. Since $\sqrt{\gamma r \log n} \ll r$ (by the choice of r and γ), there is a constant $0 < \alpha < 1$, which can be made arbitrarily small (for a proper choice of γ) such that, w.o.p.,

$$r' \geq (1-\alpha)\mathbf{E}(r') = \Theta(r),$$

for a sufficiently large constant of proportionality, that depends on c' and γ.

Not all sampled vertices have positive depth. However, since $\kappa \gg \kappa^*$, the overwhelming majority of the sampled vertices will have positive depth. By choosing c' to be sufficiently large, at least r of these vertices will have positive depth, w.o.p.

Implementing a net finder $\mathcal{F}$ and a verifier $\mathcal{V}$

As already described in the preceding section, we assign weights to the elements of T (initially, each triangle gets the weight 1), and use a net finder $\mathcal{F}$ to construct a $(1/2\xi)$-net for the weighted dual system (T, V^*). We then apply the verifier $\mathcal{V}$, in order to decide whether H covers (the newly resampled subset) R. If it does, the first part of the algorithm terminates, and we proceed to the actual construction of the union; otherwise, $\mathcal{V}$ returns a particular witness subset $T_v \in V^*$, for some $v \in R$, such that $T_v \cap H = \emptyset$. We then double the weights of the triangles in T_v, construct a new $(1/2\xi)$-net and a new sample R, and repeat this process until we find a subset of triangles that *fully covers* R. The analysis in [10] can be modified to show that the number of iterations that this algorithm performs is $O(\xi \log(n/\xi))$. Indeed, as long as there exists some vertex of the new sample R that is not covered by the set H constructed by $\mathcal{F}$, we keep on doubling the weights of the triangles covering this vertex, and according to the analysis of the algorithm [10], the overall number of such iterations does not exceed $4\xi \log(n/\xi)$. On the other hand, if R is fully covered by H, we stop this process (and may perform a smaller number of iterations), and start the actual construction of the union.

We start with the description of the net finder $\mathcal{F}$. We use a simple method, reviewed briefly in [10] and presented by Matoušek [25], for reducing the weighted case to the unweighted one. In this method, we scale all weights of the triangles in T, such that the sum $w(T)$ of the weights of all the elements of T satisfies $w(T) = n$. We then take $\lfloor w(\Delta) + 1 \rfloor$ copies of each element $\Delta \in T$ (where $w(\Delta)$ is the scaled weight of Δ). Note that the multiset T', that we have constructed, contains all the elements of T and has at most $2n$ elements. It is shown in [25] that an ε-net for (the unweighted set) T' is also an ε-net for the weighted set T. Finding a $(1/2\xi)$-net for T' can be done by drawing $O(\xi \log \xi)$ random elements of T'. As shown, e.g., in [6], an appropriate choice of the constant of proportionality ensures that such a random sample is a $(1/2\xi)$-net, with overwhelming probability. Clearly, creating the multiset T' takes $O(n)$ time, and drawing $O(\xi \log \xi)$ random elements of T' takes an additional $O(\xi \log \xi)$ time. Thus the overall running time of the net finder is $O(n)$, for total time of $O(n\xi \log(n/\xi))$ over all iterations of the algorithm. (Note that if the random sample is not a $(1/2\xi)$-net (which may

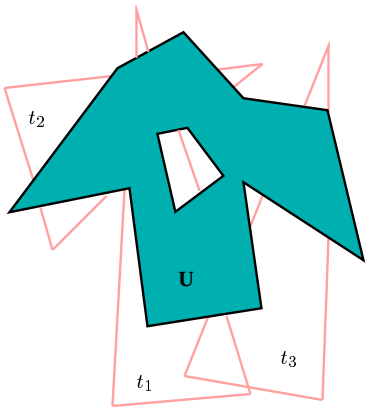

Figure 2: The second stage of the actual construction of the union. U denotes the union of the h triangles in the hitting set H, and t_1, t_2 and t_3 denote the remaining triangles to be inserted into the union. Only the portions of t_1, t_2 and t_3 that lie outside U are relevant.

happen with an overwhelmingly small probability), the number of iterations of the algorithm may exceed $4\xi \log(n/\xi)$, and, in this case, we may stop the whole process and restart it from scratch. Nevertheless, the fact that the process fails with an overwhelmingly small probability ensures that the number of such trials is not larger than some constant factor.)

In the implementation of the verifier $\mathcal{V}$, we use brute force, and iterate over all the vertices of R and the triangles of H in $O(r\xi \log \xi)$ time, to determine whether there exists a vertex in R that is not covered by the triangles of H. We denote the set of all such vertices of R by R_H. Suppose R_H is not empty (otherwise, the first part of the algorithm terminates). We sample a random vertex v from R_H, and obtain, by brute force, the set T_v of all triangles in T that contain v in their interior (clearly, $T_v \cap H = \emptyset$), and double their weights. However, since R may in general also contain zero-depth vertices, T_v will be empty for such vertices. In this case we continue sampling vertices out of R_H, and stop when we find a positive-depth vertex. Since the overwhelming majority of the vertices in R have positive depth, we will obtain such a vertex, w.o.p., after at most $O(\log n)$ samples, as is easily verified. Hence, w.o.p., the total cost of this substep is $O(n \log n)$. Since we repeat this procedure for $O(\xi \log(n/\xi))$ steps, the overall cost of this stage is

$$O(\xi \log(n/\xi)(r\xi \log \xi + n \log n)) = O(r\xi^2 \log \xi \log(n/\xi) + n\xi \log(n/\xi) \log n),$$

and this bounds the overall running time, for both the net finder $\mathcal{F}$ and the verifier $\mathcal{V}$, over all iterations of the first part of the algorithm.

The actual construction of the union

The implementation of the actual construction of the union proceeds through two stages. We first construct the union of the triangles in the set H, and then compute the portion of $\mathcal{A}(T)$ outside this union. As argued earlier, this portion contains, w.o.p., at most $\frac{\kappa}{t}$ positive-depth vertices of $\mathcal{A}(T)$.

We first construct the union of the h triangles of H in $O(h^2) = O(\xi^2 \log^2 \xi)$ time (using, e.g., randomized incremental construction [27]). Next, we efficiently find the intersections of the boundary of each of the remaining triangles Δ with the boundary of $\bigcup H$, in order to collect all the portions of $\partial\Delta$ lying outside $\bigcup H$. We denote the set of all such portions, over all the remaining triangles, by $\mathcal{C}$. (See Figure 2 for an illustration.)

In order to find those portions efficiently, we use the algorithm of Bentley and Ottmann [9] for reporting all k intersections in a set of n simply shaped Jordan arcs in $O(n \log n + k \log n)$ time. We partition the set of the remaining triangles into $\lceil \frac{n}{\xi \log \xi} \rceil$ subsets, each containing $O(\xi \log \xi)$ triangles. We denote the collection of all these subsets by $\mathcal{S} = \left\{ S_1, \ldots, S_{\lceil \frac{n}{\xi \log \xi} \rceil} \right\}$. Next, we compute, for every subset $S \in \mathcal{S}$, the arrangement $\mathcal{A}(S)$ induced by the triangles in S, and then run the Bentley-Ottmann algorithm on the combined collection of the edges of $\mathcal{A}(S)$ and the $O(h^2)$ edges of $\bigcup H$. Since the edges of $\mathcal{A}(S)$ are pairwise openly disjoint, and so are the edges of $\bigcup H$, the algorithm will only report intersections between the boundary of $\bigcup H$ and the remaining triangles. Since the overall number of such intersections, over all subsets in $\mathcal{S}$, is at most $\frac{\kappa}{t}$, the overall cost of reporting all intersections is

$$O\left(\left(\frac{n}{\xi \log \xi} \cdot \xi^2 \log^2 \xi\right) \log n + \frac{\kappa}{t} \log n\right) = O(n\xi \log \xi \log n + \frac{\kappa}{t} \log n).$$

Next, we trim the edges of the remaining triangles to their portions outside $\bigcup H$, and then construct the entire union using another line sweeping procedure on these exterior edge portions and the boundary edges of $\bigcup H$ [9]. Since there are at most $\frac{\kappa}{t}$ positive-depth vertices that are constructed during this process, the algorithm takes $O\left(\left(n + \xi^2 \log^2 \xi + \frac{\kappa}{t}\right) \log n\right)$ time.

This completes the detailed description of our algorithm, which is summarized in the following procedure, for which ξ is an input parameter. Since ξ is not known a priori, we run this procedure with the values $\xi = 1, 2, 4, \ldots, 2^i, \ldots$ (where $i \ll \log n$), thereby guaranteeing a constant approximation of the actual value of ξ. The choice of r (that is, of the parameter t) in this procedure will be specified later.

Procedure ConstructUnion(T, ξ)

1. Construct $\bigcup T$ by a line sweeping procedure on the triangles in T. Stop the procedure as
. soon as it constructs more than $\max\{\xi^2, n^{4/3}\}$ vertices. If it terminates **goto** 16.
2. Initialize all weights of the triangles in T to 1.
3. **repeat**
4. $H \leftarrow$ $(1/2\xi)$-net of size $O(\xi \log \xi)$ for the weighted system (T, V^*).
5. Construct a new random sample R of r vertices out of the vertices of $\mathcal{A}(T)$.
6. Apply the verifier $\mathcal{V}$ to H and R.
7. **if** H covers R **goto** 11.
8. **else**
9. Double the weights of all the triangles in the subset T_v reported by $\mathcal{V}$.
10. **endrepeat**
11. Construct the union of the triangles in H.
12. Partition T into subsets $S_1, \ldots, S_{\lceil \frac{n}{\xi \log \xi} \rceil}$ of size $O(\xi \log \xi)$ each.
13. For each S_i, compute $\mathcal{A}(S_i)$ and find all intersections between its edges and $\partial \bigcup H$,
. using a line-sweeping procedure.

14. Trim the edges of the remaining triangles to their portions outside $\bigcup H$. Denote the set of the resulting segments by $\mathcal{C}$.
15. Construct $\bigcup T$ by a line sweeping procedure on $\mathcal{C}$ and the boundary edges of $\bigcup H$.
16. **end**

We substitute $r = ct \log n$, for some absolute constant c, and for the parameter t that we still need to fix. Since h the size of H is $O(\xi \log \xi)$, and since the algorithm terminates after $O(\xi \log (n/\xi))$ iterations, the overall cost of the algorithm is

$$\min \left\{ \begin{array}{l} O((n+\kappa)\log n), \\ O\left(\frac{n^2}{\kappa} r\xi \log (n/\xi) + n\xi \log (n/\xi) \log n + hr\xi \log (n/\xi) + nh \log n + \frac{\kappa}{t} \log n + h^2 \log n\right) \end{array} \right\} =$$

$$\min \left\{ \begin{array}{l} O((n+\kappa)\log n), \\ O\left(\frac{n^2}{\kappa} t\xi \log n \log (n/\xi) + n\xi(\log (n/\xi) + \log \xi) \log n + \xi^2 t \log \xi \log n \log (n/\xi) + \frac{\kappa}{t} \log n\right) \end{array} \right\}.$$

Choosing

$$t = \max \left\{ \frac{\sqrt{\kappa}}{\xi \log n}, 1 \right\},$$

the running time bound becomes

$$\min \left\{ O((n+\kappa)\log n), O\left(\frac{n^2}{\sqrt{\kappa}} \log (n/\xi) + \xi\sqrt{\kappa} \log^2 n + n\xi(\log (n/\xi) + \log \xi) \log n\right) \right\}.$$

Since $\kappa = O(n^2)$ and $\xi \leq n$, this is upper bounded by

$$\min \left\{ O((n+\kappa)\log n), O\left(\frac{n^2}{\sqrt{\kappa}} \log n + n\xi \log^2 n\right) \right\}.$$

The two terms involving κ are equal when $\kappa = n^{4/3}$. Hence the running time is always bounded by $O(n^{4/3} \log n + n\xi \log^2 n)$.

In summary, we have shown:

Theorem 3.1 *Let T be a set of n triangles in the plane whose union is equal to the union of an unknown subset of $\xi \ll n$ triangles. Then the union can be constructed in randomized expected time $O\left(n^{4/3} \log n + n\xi \log^2 n\right)$, which is subquadratic for any $\xi = o\left(\frac{n}{\log^2 n}\right)$.*

4 Extensions

In this section we show how to extend our algorithm to compute the union of other planar shapes, as well as unions of simply shaped bodies in three and higher dimensions.

The analysis of the algorithm of [10] holds for any range space of finite VC dimension. Consider an input set S of bodies in $\mathbb{R}^d$, and let V denote the set of positive-depth vertices of $\mathcal{A}(S)$. It is well known that the range space (S, V^*) has finite VC dimension if the objects have constant description complexity. This can be shown, for instance, by the linearization

technique (see, e.g., [26]). In this case, the number of vertices that the objects in the set H, reported by the net finder $\mathcal{F}$, can generate, among themselves, is $O(\xi^d \log^d \xi)$. In addition, Lemma 2.1 continues to hold in this case, since it does not make any assumptions on the input shapes. It thus follows that Theorem 2.2 can be easily extended to bodies in $\mathbb{R}^d$ of constant description complexity, and that the residual cost of the algorithm, in this case, is $O(\xi^d \log^d \xi + \frac{\kappa}{t})$, w.o.p.

The actual implementation of the various stages of the algorithm can also be easily extended to bodies in $\mathbb{R}^d$ of constant description complexity. We begin with the planar case, and then discuss in Section 4.1 the extension to higher dimensions.

In the case of simply shaped planar regions, we apply similar subroutines, that run within the same time bounds as stated in Section 3. In the sampling procedure, each pair of region boundaries intersect in a constant number of points, and we collect all these intersections to form R. Since our system has finite VC-dimension, we can construct a $(1/2\xi)$-net for this system in much the same way as in Section 3. In addition, the verifier $\mathcal{V}$ can still detect whether a given vertex v is contained in the interior of another given region in $O(1)$ time, and thus these two subroutines will run within the same asymptotic time bounds as in the case of triangles. (In fact, these properties hold for bodies of constant description complexity in higher dimensions as well, and thus the net finder $\mathcal{F}$ and the verifier $\mathcal{V}$ will run within the same asymptotic time bounds in these cases too). In the actual construction of the union, we use the algorithm of Bentley and Ottmann [9], which can be applied for any set of Jordan arcs of constant description complexity, with the same asymptotic time bound, as stated in Section 3.

We can thus easily derive the following theorem:

Theorem 4.1 *Let S be a set of n planar regions of constant description complexity, whose union is equal to the union of an unknown subset of $\xi \ll n$ regions. Then the union can be constructed in randomized expected time $O\left(n^{4/3} \log n + n\xi \log^2 n\right)$, which is subquadratic for any $\xi = o\left(\frac{n}{\log^2 n}\right)$.*

4.1 The union of simply shaped bodies in $\mathbb{R}^d$

We begin with the extension of our algorithm to the case of bodies of constant description complexity in three dimensions, and then describe the generalization to higher dimensions.

In three dimensions, we may assume in the sampling procedure that $\kappa \gg \max\{\xi^3, n^2\}$. Otherwise, we construct the union in time $O((n^2 + \xi^3) \log n)$, as follows. We fix a body $B \in S$ and intersect its boundary F with each object $B' \in S \setminus \{B\}$. We obtain a collection of $n - 1$ Jordan regions of constant description complexity on F. The complement of their union is the portion of F that appears on $\partial \bigcup S$. Computing this complement can be done in time $O(n \log n + \kappa_B \log n)$, where κ_B is the number of vertices of $\mathcal{A}(S)$ that lie on F, using an appropriate variant of the line-sweeping algorithm of Bentley and Ottmann [9]. Repeating this procedure for each boundary F, the total cost is $O((n^2 + \kappa) \log n) = O((n^2 + \xi^3) \log n)$, as claimed.

The main part of the algorithm then proceeds in much the same way as before. For example,

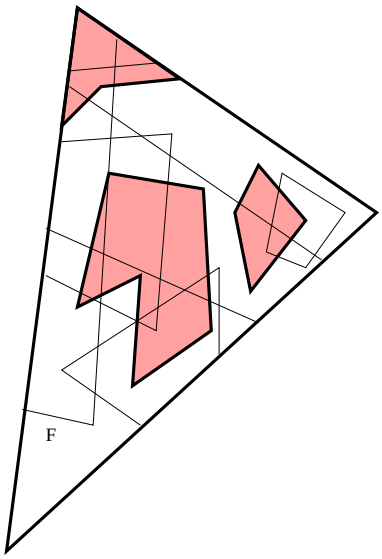

Figure 3: The case where the input bodies are simplices in three dimensions. The facet F belongs to one of the first h simplices. The thick lines are the boundaries of $\bigcup H$ on F. The thin lines are the intersections of the $n-h$ remaining simplex boundaries with F. The intersections appearing in the shaded regions lie in the interior of the union of the n simplices, and need not be computed explicitly.

when we construct a sample R of vertices, we perform, in analogy with the two-dimensional procedure, $\frac{c'r\binom{n}{3}}{\kappa}$ sampling steps, for an appropriate constant $c' > 1$, where in each step we choose, uniformly and independently, a triple of distinct input bodies in S, and collect all resulting boundary intersections to form R. A similar analysis to that described in Section 3 shows that, with an appropriate choice of the constant c', at least r of the chosen triples generate real vertices that have positive depth, w.o.p.

As noted above, the net finder $\mathcal{F}$ and the verifier $\mathcal{V}$ can be implemented in a similar manner to that described in Section 3, and run within the same asymptotic time bounds (and this holds in higher dimensions as well). It follows that, choosing $t = \max\left\{\frac{\sqrt{\kappa}}{\xi \log n}, 1\right\}$, the first part of the algorithm computes a subset H of S of size $h = O(\xi \log \xi)$, in time $O(r\xi^2 \log \xi \log(n/\xi) + n\xi \log(n/\xi) \log n)$, such that at most $\frac{\kappa}{t}$ positive-depth vertices of $\mathcal{A}(S)$ lie outside the (interior of the) union $\bigcup H$.

After constructing $\bigcup H$, we need to compute all the intersections between the remaining bodies and the boundary of $\bigcup H$. This is done as follows. For each body $B \in S$ (particularly, B may belong to H), we take its boundary F, and compute the set of its exposed portions that lie outside $\bigcup H \setminus \{B\}$. This is done by constructing the intersections $B'_F = B' \bigcap F$ for each $B' \in H \setminus \{B\}$, and then compute the complement of their union within F. Since the regions B'_F are bounded by curves of constant description complexity, their arrangement has $O(h^2)$ complexity, and it can be constructed in $O(h^2 \log n)$ time. We denote by E_F the set of edges of the arrangement that appear on the boundary of the union of the regions B'_F. Clearly $|E_F| = O(h^2)$. We then intersect F with all the remaining $n-h$ input bodies, obtaining a

set of curves $\mathcal{S}_F$ bounding the intersection regions. Our goal is to find the portions of the curves in $\mathcal{S}_F$ that are not contained in the interior of $\bigcup H$; see Figure 3 for an illustration. We first report the intersections between the curves in $\mathcal{S}_F$ and E_F in $O(nh \log n + I_F \log n)$ time, where I_F is the number of such intersections, in a similar manner to that described in the two-dimensional case. Since the overall number of these intersections, over all facets F, is less than $\frac{\kappa}{t}$, the overall time needed to report all these intersections, over all these facets, is

$$O(n^2 h \log n + \frac{\kappa}{t} \log n).$$

We now trim, on each boundary F, the edges of the cross sections of the remaining input bodies, to their portions within the exposed bodies on F, and continue in a similar manner to that described in the two-dimensional case; that is, we run a line sweeping procedure on these portions and the curves in E_F. The running time of this procedure, over all boundaries F, is $O\left(\left(n^2 + nh^2 + \frac{\kappa}{t}\right) \log n\right)$.

The overall running time of the algorithm, in this case, is thus

$$\min \left\{ \begin{array}{l} O((n^2+\kappa)\log n), \\ O\left(\frac{n^3}{\kappa} r\xi \log(n/\xi) + n\xi \log(n/\xi) \log n + hr\xi \log(n/\xi) + n^2 h \log n + \frac{\kappa}{t} \log n\right) \end{array} \right\} =$$

$$\min \left\{ \begin{array}{l} O((n^2+\kappa)\log n), \\ O\left(\frac{n^3}{\kappa} t\xi \log n \log(n/\xi) + \xi^2 t \log \xi \log n \log(n/\xi) + n^2 \xi \log \xi \log n + \frac{\kappa}{t} \log n\right) \end{array} \right\}.$$

Choosing, as above,

$$t = \max\left\{\frac{\sqrt{\kappa}}{\xi \log n}, 1\right\},$$

the running time bound becomes

$$\min\left\{O((n^2+\kappa)\log n), O\left(\frac{n^3}{\sqrt{\kappa}} \log(n/\xi) + \xi\sqrt{\kappa} \log^2 n + n^2 \xi \log \xi \log n\right)\right\}.$$

Since $\kappa = O(n^3)$ and $\xi \leq n$, this is upper bounded by

$$\min\left\{O((n^2+\kappa)\log n), O\left(\frac{n^3}{\sqrt{\kappa}} \log n + n^2 \xi \log^2 n\right)\right\}.$$

The two terms involving κ are equal when $\kappa = n^2$. Hence the running time is always bounded by

$$O(n^2 \log n + n^2 \xi \log^2 n) = O(n^2 \xi \log^2 n),$$

which is subcubic for $\xi = o\left(\frac{n}{\log^2 n}\right)$.

Let $\mathcal{B}$ be a set of n bodies of constant description complexity in $\mathbb{R}^d$, and let $\mathcal{S} \subset \mathcal{B}$ be the (unknown) subset of ξ bodies whose union is equal to $\bigcup \mathcal{B}$. We compute the union by recursing on the dimension. That is, we fix a body $B \in \mathcal{B}$, take its boundary F, and intersect it with each body $B' \in \mathcal{B} \setminus \{B\}$. We then compute the union of these intersection bodies, and construct

its component within F. The union of all these components, over all boundaries F, yields the boundary of $\partial\mathcal{B}$. Note that if $B \in \mathcal{B} \setminus \mathcal{S}$ then the union of the intersection bodies along ∂B covers the entire boundary of B. In fact, the union of the intersections with the bodies of $\mathcal{S}$ already covers the boundary. Similarly, if $B \in \mathcal{S}$ then the union of the intersection bodies along ∂B is equal to the union of the intersections with the bodies of $\mathcal{S}$. In either case, with an appropriate parametrization of the boundaries, we obtain n $(d-1)$-dimensional instances of the union construction problem, each with output size $\leq \xi$, according to our measure. We thus compute these $(d-1)$-dimensional unions recursively, and stop the recursion when $d = 3$. This leads to an overall algorithm that runs in randomized expected time $O(n^{d-1}\xi \log^2 n)$. That is, we have:

Theorem 4.2 *Let S be a set of n bodies of constant description complexity in $\mathbb{R}^d$, whose union is equal to the union of an unknown subset of $\xi \ll n$ bodies. Then the union can be constructed in randomized expected time $O(n^{d-1}\xi \log^2 n)$, which is asymptotically smaller than n^d for any $\xi = o\left(\frac{n}{\log^2 n}\right)$.*

5 Concluding Remarks

We have presented an output-sensitive algorithm for the problem of constructing efficiently the union of n triangles in the plane, whose running time is expressed in terms of the smallest size ξ of an unknown subset of the triangles whose union is equal to the union of the entire set. We have used a variant of the technique of Brönnimann and Goodrich [10] for finding a set cover in a set system of finite VC-dimension. We have also presented a detailed and fairly generic implementation of this method, showing that the above problem can be solved in randomized expected time $O(n^{4/3} \log n + n\xi \log^2 n)$, which is subquadratic for $\xi = o\left(\frac{n}{\log^2 n}\right)$. The algorithm does not have to know the value of ξ in advance. Instead, it runs an exponential search on ξ, which approximates well the correct value of ξ, up to a constant factor.

We showed that our approach can be easily extended to simply shaped bodies of constant description complexity in $\mathbb{R}^d$, for $d \geq 2$, where the union is determined by ξ bodies. In the planar case, the running time remains $O(n^{4/3} \log n + n\xi \log^2 n)$. In $d \geq 3$, the union can be constructed in randomized expected time $O(n^{d-1}\xi \log^2 n)$, which is asymptotically smaller than n^d for $\xi = o\left(\frac{n}{\log^2 n}\right)$. For $d > 3$, we computed the union recursively on d, by constructing the union along each object boundary separately. However, this recursion had to stop at $d = 3$. Indeed, for $d = 3$, applying the two-dimensional algorithm on the boundary of each input body, yields an overall $O\left(n^{7/3} \log n + n^2\xi \log^2 n\right)$ expected running time, which is worse than the bound that we have obtained when $\xi = o\left(\frac{n^{1/3}}{\log n}\right)$.

A direction for further research is to determine whether there exist simpler efficient approaches to the union construction problem studied in this paper. We note that the standard randomized incremental construction (RIC) of [27] may fail in this case. In fact, the standard bad example for the RIC, consisting of n triangles that form $\Theta(n^2)$ shallow vertices that are all covered by one large triangle (or, more generally, sparsely covered by $\xi \ll n$ triangles), shows that the RIC may fail to construct the union in an output-sensitive manner.

Another direction for further research is to extend our approach to instances involving unions in three dimensions where the worst-case complexity of the union is only quadratic or near-quadratic (see [4, 7, 28] for known instances of this kind). Our approach runs in *subcubic* time, when ξ is small, but does not improve upon standard, output-insensitive techniques when the union complexity is near-quadratic. The simplest instance of such a problem would be: Given a collection of n balls in $\mathbb{R}^3$, whose union is equal to the union of some $\xi \ll n$ of the balls, can the union be constructed in subquadratic time?

Finally, we note that in an earlier version of the algorithm [17], we used a different approach, based on a careful implementation of the DC algorithm of [15]. The previous approach is more complicated, yields a somewhat less efficient solution, which is subquadratic only for a smaller range of the values of the parameter ξ, and is more difficult to extend to other geometric shapes and to higher dimensions. Our new approach, based on the technique of Brönnimann and Goodrich, is simpler, more generic, improves our previous result, and extends to other shapes and to higher dimensions.

Acknowledgments. The authors wish to thank Ken Clarkson and Sariel Har-Peled for useful discussions on this problem. In particular, Sariel's insistence that we use the Brönnimann-Goodrich technique (instead of the DC algorithm in the earlier version [17]) has finally led to the improved algorithm presented in this paper.

References

[1] P. K. Agarwal and S. Har-Peled. Two randomized incremental algorithms for planar arrangements, with a twist. Manuscript, 2001.

[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. Pellegrini, and M. Sharir. Counting circular arc intersections. *SIAM J. Comput.*, 22(4):778–793, 1993.

[4] P. K. Agarwal and M. Sharir. Pipes, cigars, and kreplach: The union of minkowski sums in three dimensions. *Discrete Comput. Geom.*, 24:645–657, 2000.

[5] P. K. Agarwal, M. Sharir, and S. Toledo. Applications of parametric searching in geometric optimization. *J. Algorithms*, 17:292–318, 1994.

[6] N. Alon and J. H. Spencer. *The Probabilistic Method.* 2*nd* Edition, Wiley-Interscience, New York, USA, 2000.

[7] B. Aronov, A. Efrat, V. Koltun, and M. Sharir. On the union of κ-round objects in three and four dimensions. In *Proc. 20th Annu. ACM Sympos. Comput. Geom.*, ACM Press, 2004, to appear.

[8] P. Assouad. Density and dimensions. *Ann. Inst. Fourier (Grenoble)*, 33:233–282, 1983.

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

[10] H. Brönnimann and M. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete Comput. Geom.*, 14:463–479, 1995.

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

[12] M. de Berg, L. J. Guibas, and D. Halperin. Vertical decompositions for triangles in 3-space. *Discrete Comput. Geom.*, 15:35–61, 1996.

[13] M. de Berg, M. Katz, A. F. van der Stappen, and J. Vleugels. Realistic input models for geometric algorithms. *Algorithmica*, 34:81–97, 2002.

[14] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications.* 2*nd* Edition, Springer-Verlag, Berlin, 2000.

[15] E. Ezra, D. Halperin, and M. Sharir. Speeding up the incremental construction of the union of geometric objects in practice. *Comput. Geom. Theory Appl.*, 27:63–85, 2004.

[16] E. Ezra and M. Sharir. Counting and representing intersections among triangles in three dimensions. In *Proc. 20th Annu. ACM Sympos. Comput. Geom.*, ACM Press, 2004, to appear.

[17] E. Ezra and M. Sharir. Output-sensitive construction of the union of triangles. In *Proc. 15th Annu. ACM-SIAM Sympos. Discr. Alg. (SODA'04)*, pages 413–422. SIAM, 2004.

[18] A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom. Theory Appl.*, 5:165–185, 1995.

[19] D. P. Huttenlocher, K. Kedem, and M. Sharir. The upper envelope of Voronoi surfaces and its applications. *Discrete Comput. Geom.*, 9:267–291, 1993.

[20] K. Kedem, R. Livne, J. Pach, and M. Sharir. On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete Comput. Geom.*, 1:59–71, 1986.

[21] V. Koltun. Almost tight upper bounds for vertical decompositions in four dimensions. In *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Science (FOCS'01)*, pages 56–65. 2001. Also in SIAM comput., to appear.

[22] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, 1991.

[23] J. Matoušek, N. Miller, J. Pach, M. Sharir, S. Sifrony, and E. Welzl. Fat triangles determine linearly many holes. In *Proc. 32nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 49–58, 1991.

[24] J. Matoušek, J. Pach, M. Sharir, S. Sifrony, and E. Welzl. Fat triangles determine linearly many holes. *SIAM J. Comput.*, 23(1):154–169, 1994.

[25] J. Matoušek. Cutting hyperplane arrangements. *Discrete Comput. Geom.*, 6:385–406, 1991.

[26] J. Matoušek. *Lectures on Discrete Geometry.* Springer-Verlag, Berlin, 2002.

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

[28] J. Pach, I. Safruti, and M. Sharir. The union of congruent cubes in three dimensions. *Discrete Comput. Geom.*, 30:133–160, 2003.

[29] B. Tagansky. A new technique for analyzing substructures in arrangements of piecewise linear surfaces. *Discrete Comput. Geom.*, 16(4):455–479, 1996.

A The Actual Model for Sampling R

As described in Section 3, we draw the elements of R by randomly making $\frac{c'r\binom{n}{2}}{\kappa}$ independent selections of a vertex out of V^+, for some constant $c' \geq 1$, where in each trial, each vertex (or more precisely, each pair of triangles) is chosen with probability $\frac{1}{\binom{n}{2}}$ (thus the same vertex may be sampled more than once). The probability p that a vertex $v \in V^+$ is chosen (at least once) is equal to

$$p = 1 - \left(1 - \frac{1}{\binom{n}{2}}\right)^{c'r\frac{\binom{n}{2}}{\kappa}}. \tag{4}$$

It is easily checked that p is smaller than $c'\frac{r}{\kappa}$. Moreover, one can also easily show that

$$p > c'\frac{r}{\kappa} - \frac{(c'r)^2}{\kappa^2}. \tag{5}$$

In this model, the variables $X_1, \ldots, X_{\kappa_S}$ (as were defined in Lemma 2.1) are no longer independent. Examining the proof of the deviation bound given in [6, Theorem A.13], we note that the only place where it uses the assumption that these variables are independent, is in the derivation of the equality

eti says: I have checked that (more than once).

$$\mathbf{E}\left[e^{\sum_{i=1}^{\kappa_S}\lambda X_i}\right] = \Pi_{i=1}^{\kappa_S}\mathbf{E}\left[e^{\lambda X_i}\right],$$

for any λ. Moreover, the analysis in [6] only uses the value $\lambda = \frac{r_0}{p\kappa_S}$, where r_0 is defined as in Lemma 2.1. An inspection of the derivation of these bounds in [6] shows that they continue to hold when

$$\mathbf{E}\left[e^{\sum_{i=1}^{\kappa_S}\lambda X_i}\right] \leq \Pi_{i=1}^{\kappa_S}\mathbf{E}\left[e^{\lambda X_i}\right].$$

Furthermore, Lemma 2.1 continues to hold when the weaker inequality

$$\mathbf{E}\left[e^{\sum_{i=1}^{\kappa_S}\lambda X_i}\right] \leq \gamma\Pi_{i=1}^{\kappa_S}\mathbf{E}\left[e^{\lambda X_i}\right] \tag{6}$$

holds, for some positive constant γ. This has the effect of multiplying the probability that (1) fails by γ, which implies that (1) still holds, w.o.p. Hence, it suffices to show that (6) holds for the above value of λ.

In our model,

$$\Pi_{i=1}^{\kappa_S}\mathbf{E}\left[e^{\lambda X_i}\right] = \left(e^{\lambda}p + (1-p)\right)^{\kappa_S} = \left(1 + p(e^{\lambda}-1)\right)^{\kappa_S}. \tag{7}$$

and

$$\mathbf{E}\left[e^{\sum_{i=1}^{\kappa_S}\lambda X_i}\right] = \sum_{m=0}^{r^*} Pr\left[r_{_S} = m\right] e^{\lambda m}, \tag{8}$$

where $r^* = \min\left\{\frac{c'r\binom{n}{2}}{\kappa}, \kappa_{_S}\right\}$. (Note that $Pr\left[r_{_S} = m\right] = 0$, for any $m > r^*$.)

In each of the $\frac{c'r\binom{n}{2}}{\kappa}$ drawing trials, the probability that we have selected a vertex v, and that it is not covered by S, is $q = \frac{\kappa}{\binom{n}{2}} \cdot \frac{\kappa_S}{\kappa} = \frac{\kappa_S}{\binom{n}{2}}$. Since these trials are independent, we have

$$Pr\left[r_{_S} = m\right] = \binom{r^*}{m} q^m [1-q]^{r^*-m}.$$

Hence (8) becomes

$$\sum_{m=0}^{r^*}\binom{r^*}{m} q^m [1-q]^{r^*-m} e^{\lambda m} =$$

$$\left(e^{\lambda}q + 1 - q\right)^{r^*}.$$

In other words, putting $e^{\lambda} - 1 = \lambda_0$, we need to show that

$$(1+\lambda_0 q)^r \leq \gamma\,(1+\lambda_0 p)^{\kappa_S},$$

for some constant $\gamma > 0$. We will show that

$$(1+\lambda_0 q)^{r^*} \leq (1+\lambda_0 p)^{2c'r}\,(1+\lambda_0 p)^{\kappa_S},$$

which implies the preceding inequality because $(1+\lambda_0 p)^{2c'r} = O(1)$. Indeed, $(1+\lambda_0 p)^{2c'r} < e^{2c'\lambda_0 pr}$. Using the fact that $e^{\lambda} \leq 1 + 2\lambda$, for $0 \leq \lambda \leq 1$, and substituting $\lambda = \frac{r_0}{p\kappa_{_S}}$, $\lambda_0 = e^{\lambda} - 1$ we have

$$e^{2c'\lambda_0 pr} \leq e^{4c'\frac{rr_0}{\kappa_{_S}}}.$$

Since we assume in Lemma 2.1 that $r_0 = 2\sqrt{c_0\frac{r}{\kappa}\kappa_{_S}\log n}$, for some constant $c_0 \geq 1$, the latter expression is smaller than

$$e^{8c'\frac{r}{\kappa_{_S}}\sqrt{c_0\frac{r}{\kappa}\kappa_{_S}\log n}} = e^{O\left(r\sqrt{\frac{r\log n}{\kappa\kappa_{_S}}}\right)},$$

which is always upper bounded by

$$e^{O\left(\frac{r}{\kappa}\sqrt{tr\log n}\right)},$$

using the assumption of Lemma 2.1 that $\kappa_{_S} \geq \frac{\kappa}{t}$.

Substituting $r = ct \log n$, for some constant c, and $t = \max\left\{\frac{\sqrt{\kappa}}{\xi \log n}, 1\right\}$, as above, and using the assumption that $\kappa \gg \max\{\xi^2, n^{4/3}\}$ (see Section 3)

$$e^{2c'\lambda_0 pr} < \max\left\{e^{O\left(\frac{1}{\xi^2}\right)}, e^{O\left(\frac{\log^2 n}{\kappa}\right)}\right\} = O(1).$$

It thus remains to show that

$$(1+\lambda_0 q)^{r^*} \leq (1+\lambda_0 p)^{2c'r+\kappa_{_S}}\,. \tag{9}$$

We first assume that $\frac{c'r\binom{n}{2}}{\kappa} \leq \kappa_{_S}$. We thus show that

$$\left(1+\lambda_0\frac{\kappa_{_S}}{\kappa}\right)^{\frac{c'r\binom{n}{2}}{\kappa}} \leq (1+\lambda_0 p)^{2c'r+\kappa_{_S}}\,,$$

or that

$$\sum_{i=0}^{\frac{c'r\binom{n}{2}}{\kappa}} \binom{\frac{c'r\binom{n}{2}}{\kappa}}{i}\left(\lambda_0\frac{\kappa_{_S}}{\binom{n}{2}}\right)^i \leq \sum_{i=0}^{2c'r+\kappa_{_S}} \binom{2c'r+\kappa_{_S}}{i}(1+\lambda_0 p)^i\,.$$

Note that $2c'r+\kappa_{_S} > \frac{c'r\binom{n}{2}}{\kappa}$, due to the assumption that $\frac{c'r\binom{n}{2}}{\kappa} \leq \kappa_{_S}$. It thus sufficient to show that

$$\binom{\frac{c'r\binom{n}{2}}{\kappa}}{i}\left(\lambda_0\frac{\kappa_{_S}}{\binom{n}{2}}\right)^i \leq \binom{2c'r+\kappa_{_S}}{i}(1+\lambda_0 p)^i\,,$$

for each $0 \leq i \leq \frac{c'r\binom{n}{2}}{\kappa}$. Clearly, this inequality holds for $i = 0$, and using (5), for each $i > 0$, it implies that

$$\left(\frac{\kappa_{_S}}{2c'r+\kappa_{_S}}\right)^i \leq \left(1-\frac{c'r}{\kappa}\right)^i.$$

It therefore suffices to show that $\frac{\kappa_{_S}}{2c'r+\kappa_{_S}} \leq 1-\frac{c'r}{\kappa}$, or that $1-\frac{2c'r}{2c'r+\kappa_{_S}} \leq 1-\frac{c'r}{\kappa}$, or that

$$\kappa \geq c'r + \frac{\kappa_{_S}}{2}, \tag{10}$$

which clearly holds, since $\kappa_{_S} \leq \kappa$ and $r \ll \kappa$.

We now show that (9) holds when $\frac{c'r\binom{n}{2}}{\kappa} > \kappa_{_S}$. We thus show that

$$\left(1+\lambda_0\frac{\kappa_{_S}}{\binom{n}{2}}\right) \leq (1+\lambda_0 p)^{\frac{2c'r+\kappa_{_S}}{\kappa_{_S}}}\,.$$

Using the fact that $(1+\lambda_0 p)^{\frac{2c'r+\kappa_S}{\kappa_S}} \geq 1+\lambda_0 p\left(\frac{2c'r+\kappa_S}{\kappa_S}\right)$ and (5), it is sufficient to show that

$$\frac{\kappa_S}{\binom{n}{2}} \leq c'\frac{r}{\kappa}\left(1-\frac{c'r}{\kappa}\right)\left(1+\frac{2c'r}{\kappa_S}\right),$$

or that $\left(1-\frac{c'r}{\kappa}\right)\left(1+\frac{2c'r}{\kappa_S}\right) \geq 1$, using the assumption on κ_S. The latter implies that $\kappa - c'r - \frac{\kappa_S}{2} \geq 0$, which clearly holds due to (10).

We note that (9) holds for any value of $\lambda_0 > 0$, and the assumption on λ is used only when showing that $(1+\lambda_0 p)^{2c'r} = O(1)$. This completes the proof of (6) for the above considered values of λ.