# Geometric Arrangements: Substructures and Algorithms

Thesis submitted for the degree of “Doctor of Philosophy”
by

**Esther Ezra**

Submitted to the Senate of Tel-Aviv University
August 2007

The work on this thesis was carried out under the supervision of

Prof. Micha Sharir

To Yaron with much love.

To my parents, for their unconditional love.
To my brothers Alon and Hezi, my sister Shelly, and to Savta Shula.

# Abstract

In this thesis we study a variety of problems in computational and combinatorial geometry, which involve arrangements of geometric objects in the plane and in higher dimensions. Some of these problems involve the design and analysis of algorithms on arrangements and related structures, while others establish combinatorial bounds on the complexity of various substructures in arrangements.

Informally, an arrangement is the subdivision of space induced by a collection of geometric objects. For example, a collection of triangles in the plane subdivides it into polygonal regions, each being a maximal connected region contained in a fixed subset of the triangles and disjoint from all the others. This subdivision is the arrangement of the triangles, and each of these polygonal regions is a cell of the arrangement. A substructure in an arrangement is a collection of certain features of the arrangement. Two main substructures that we study in this thesis, under both combinatorial and algorithmic aspects, are the union of geometric objects and a single cell in an arrangement.

This thesis consists of two major parts, where in the first we discuss several algorithmic problems, and in the second we present combinatorial bounds on substructures in arrangements.

## Algorithmic problems.

**Constructing the union of geometric objects.** A central problem in computational and combinatorial geometry, with various applications, concerns the union of geometric objects. Given a collection $S$ of geometric objects in $d$-space, let $U = U(S)$ denote their union. Informally, the combinatorial complexity of the union is the overall number of features of the arrangement of $S$ that appear on its boundary. For example, if $S$ is a set of triangles as above, then their union boundary consists of all vertices (intersections between a pair of triangle boundaries, or an original vertex of a triangle) and edges (a maximal connected portion of a boundary of a triangle, that does not contain any vertex of the arrangement in its relative interior) that are not contained in the interior of any of the triangles in $S$. It is well known that in the worst case the combinatorial complexity of the union can be (asymptotically) the same as that of the entire arrangement. However, there are various special cases for which this complexity is considerably smaller (by, roughly,

one order of magnitude than the complexity of the full arrangement), and, as a result, the union boundary can be constructed much more efficiently (that is, without constructing the entire arrangement).

In Chapter 2 we present a subquadratic "output-sensitive" algorithm to construct the boundary of the union of a set of triangles in the plane, under the assumption that there exists a (relatively small) subset of triangles (unknown to us) such that their union is equal to the union of the entire set. 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 $d$-space, when the union is determined by a small subset of the bodies. The solution is based on a variant of the Brönnimann-Goodrich technique [40] for obtaining an approximate solution to the hitting-set problem.

**Counting triple intersections among triangles in 3-space.** Intersection problems are among the most basic problems in computational geometry, with many different application areas. The problem of *reporting* all intersections in a given collection of geometric objects has received considerable attention, and several efficient *output-sensitive* algorithms (algorithms whose running time depends on the output size) have been designed for this problem (mostly for planar instances). However, in some applications, we are only interested in *counting* the overall number of intersections, without reporting them explicitly. In this case, one prefers an algorithm whose running time does not depend on the number of intersections (which, in the worst case, is proportional to the size of the arrangement induced by the input objects).

In Chapter 3 we present an algorithm that efficiently counts all intersecting triples among a collection of $n$ triangles in 3-space in nearly-quadratic time. This solves a problem posed by Pellegrini [118]. Using a variant of the technique, the algorithm can also represent the set of all triple intersections in a compact form, as the disjoint union of complete tripartite hypergraphs, which requires nearly-quadratic construction time and storage. A compact representation of this form is useful for drawing a random vertex (triple intersection point) of the arrangement of the given triangles. Our approach also applies to any collection of planar objects of constant description complexity in 3-space, with the same performance bounds. We also prove that this counting problem belongs to the "3SUM-hard" family, and thus our algorithm is likely to be nearly optimal in the worst case.

**Analyzing the ICP algorithm.** The matching and analysis of geometric patterns and shapes is an important problem that arises in various application areas. In a typical scenario, we are given two objects $A$ and $B$, and we wish to determine how much they *resemble* each other (with respect to some *cost function*). Usually one of the objects may undergo certain transformations, like translation, rotation and/or scaling, in order to be matched with the other object as well as possible. In many cases, the objects are represented as finite sets of (sampled) points in two or three dimensions.

A heuristic matching algorithm that is widely used, due to its simplicity (and its good performance in practice), is the *Iterative Closest Point* algorithm, or the *ICP* algorithm for short, of Besl and McKay [35]. Given two point sets $A$ and $B$ in $d$-space, we wish to minimize a cost function $\phi(A + t, B)$, over all *translations* $t$ of $A$ relative to $B$. The algorithm starts with an arbitrary translation that aligns $A$ to $B$ (suboptimally), and then repeatedly performs local improvements that keep re-aligning $A$ to $B$, while decreasing the given cost function $\phi(A + t, B)$, until no improvement is possible. This algorithm has been identified and used as a practical heuristic solution over the past fifteen years. Many experimental reports on its performance, including additional heuristic enhancements of it have been published [35, 78, 120, 125]. Still, this technique has never before been subject to a serious and rigorous analysis of its worst-case behavior.

In Chapter 4 we analyze the performance of this algorithm, and present several of its structural geometric properties. In particular, we present upper and lower bounds for the number of iterations that it performs, where we consider two standard measures of resemblance that the algorithm attempts to optimize: The RMS (root mean squared distance) and the (one-sided) Hausdorff distance. We show that in both cases the number of iterations performed by the algorithm is polynomial in the number of input points. In particular, the upper bound is quadratic in the one-dimensional case, under the RMS measure, for which we present a lower bound construction that requires $\Omega(n \log n)$ iterations, where $n$ is the overall size of the input. Under the Hausdorff measure, this bound is only $O(n)$ for input point sets whose *spread* is polynomial in $n$, and this is tight in the worst case. Regarding the structural geometric properties of the algorithm, we show, for the RMS measure, that at each iteration of the algorithm the cost function monotonically and strictly decreases along the vector $\Delta t$ of the *relative translation*. As a result, we conclude that the polygonal path $\pi$, obtained by concatenating all the relative translations that are computed during the execution of the algorithm, does not intersect itself. In particular, in the one-dimensional problem all the relative translations of the ICP algorithm are in the same (left or right) direction. For the Hausdorff measure, some of these properties continue to hold (such as monotonicity in one dimension), whereas others do not.

## Combinatorial bounds on substructures in arrangements

**A single cell.** A natural problem that relates to substructures in arrangements is to analyze the complexity of a *single* cell in an arrangement of geometric objects. In Chapter 5 we study the case where the input consists of $k$ convex polyhedra in 3-space with $n$ facets in total. We show that in this case the combinatorial complexity of a single cell in the arrangement of the polyhedra is $O(nk^{1+\varepsilon})$, for any $\varepsilon > 0$, thus settling a conjecture of Aronov *et al.* [27], who presented a lower bound of $\Omega(nk\alpha(k))$, and conjectured that the upper bound is close to $O(nk)$. We also design an efficient deterministic algorithm that constructs a single cell of the arrangement, whose running time matches the combinatorial bound up to a polylogarithmic factor. We note that a nearly-quadratic bound on the complexity of a single cell in 3-dimensional arrangements is well known, for fairly general types of arrangements [24, 85]. The novelty of our bound is its dependence on $k$, which makes it linear in $n$ for any fixed $k$.

**Union of geometric objects.** As discussed above, constructing the union of geometric objects in $d$-space is a central problem in computational geometry with many applications. On the combinatorial front, it is interesting to seek for natural examples, for which the combinatorial complexity of the union is considerably smaller than that of the full arrangement. Besides yielding more efficient algorithms that construct the union in these special cases, these problems involve intricate combinatorial techniques, which we believe them to be of independent interest, and may find additional applications to related problems.

In Chapter 6 we study the case where the input consists of "fat" tetrahedra in 3-space, and derive a nearly-quadratic bound on the complexity of their union. Our bound is almost tight in the worst case, thus almost settling a conjecture of Pach *et al.* [112]. Our result extends, in a significant way, the result of Pach *et al.* [112] for the restricted case of nearly congruent cubes. As an immediate corollary, we obtain that the combinatorial complexity of the union of $n$ cubes in 3-space, having arbitrary side lengths, is $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$ (again, significantly extending the result of [112]). Our analysis can easily be extended to yield a nearly-quadratic bound on the complexity of the union of arbitrarily oriented fat triangular prisms (whose cross-sections have arbitrary sizes) in 3-space. Finally, we also show that a simple variant of our analysis implies a nearly-linear bound on the complexity of the union of fat triangles in the plane (this latter bound is known, even in a slightly sharper form [107, 116], but the new proof is considerably simpler). In spite of the steady progress during the past decade in the study of the union of objects in 3-space, the case of arbitrary fat tetrahedra, considered one of the major basic instances, has remained open.

**Regular vertices.** Another problem that belongs to this family is to obtain, for a collection of $n$ objects in the plane, a sharp bound on the number of *regular* vertices (intersection points of two object boundaries that intersect twice), which appear on the boundary of the union. In Chapter 7 we show that the number of regular vertices that appear on the boundary of the union of $n$ compact convex sets in the plane, such that the boundaries of any pair of these sets intersect in at most some constant number $s$ of points, is $O(n^{4/3+\varepsilon})$, for any $\varepsilon > 0$. Our bound is nearly tight in the worst case (already for $s = 4$), and improves earlier bounds due to Aronov *et al.* [20].

# Acknowledgments

First and foremost, I deeply thank my advisor Micha Sharir for his guidance and help during the past five years, and especially for his endless dedication to our joint work. Above all, working with Micha was a unique educational experience, and I am grateful to him for all that he did for me.

I wish to thank Dan Halperin, my M.Sc advisor, for introducing me the field of computational geometry, and for his kind help during my Ph.D studies.

I also wish to thank Pankaj Agarwal, for useful discussions and good advice, and to Boris Aronov and Joe Mitchel for their kind help. Many thanks to Boris Aronov, Ken Clarkson, Alon Efrat, Sariel Har-Peled, Zur Izhakian, Vladlen Koltun, Janos Pach, Rom Pinchasi, Evgenii Shustin, Shakhar Smorodinsky and Mark de Berg for many helpful and profound discussions, and their useful comments, some of which appear in this thesis.

I also would like to thank Boris Aronov, Bernard Chazelle, Alon Efrat, Vladlen Koltun, Nimrod Megiddo, Joe Mitchel, Janos Pach and Mark de Berg for hosting me at various institutes abroad.

Last, but not least, I thank all my colleagues and friends Angela Anosh, Adi Avidor, Alon Efrat, Leah Epstein, Nir Halman, Tali Kaufman, Eli Packer, Rom Pinchasi, Yossi Richter and Shakhar Smorodinsky.

# Contents

# Chapter 1

# Introduction

## 1.1 Overview

Given a finite collection $S$ of geometric surfaces in $\mathbb{R}^d$, the *arrangement* $\mathcal{A}(S)$ is the decomposition of $\mathbb{R}^d$ into connected open cells of dimensions $0, 1, \ldots, d$ induced by $S$. That is, each cell is a maximal connected region that is contained in the intersection of a fixed number of the surfaces and avoids all other surfaces; in particular, the 0-dimensional cells of $\mathcal{A}(S)$ are called "vertices", the 1-dimensional cells are referred to as "edges", and higher-dimensional cells are called "faces", where $(d-1)$-cells are sometimes referred to as "facets", and $d$-dimensional cells are simply referred to as "cells". Arrangements have emerged as the underlying structure of geometric problems in a variety of applications, such as robot motion planning, pattern matching, visibility problems in computer graphics and modelling, geographic information systems, bioinformatics, and more. In addition, they arise in most of the basic problems in computational geometry; see [124, 81] for surveys on arrangements.

Here is a simple application of arrangements to translational motion planning in robotics. Consider a rigid polyhedral robot $R$ that is moving around in a 2- or 3-dimensional environment with polyhedral obstacles $A_1, \ldots, A_k$ (which is also called the *workspace* of the robot). To put the discussion in focus, assume for now that the workspace is 3-dimensional. Assume that the robot is only allowed to translate within the workspace, where the placement of the robot is determined with respect to some reference point that we place, say, inside the robot. The goal is to determine the set of all *free* placements of the robot, that is, placements at which it does not collide with any of the obstacles in the workspace. The robot thus has three degrees of freedom, which correspond to the location of its reference point. The set of all these placements comprises the so called *configuration space* of the robot, and the free placements form a subset of this space, known as the *free configuration space*.

In this configuration space, each possible contact between a feature (vertex, edge or face) of an obstacle and a similar feature (face, edge or vertex) of the robot induces a surface (so-called *contact surface*) in the configuration space. In our particular problem, all the contact surfaces are piecewise-linear. The set $\mathcal{H}$ of the contact surfaces is constructed by computing the *Minkowski sum* $K_i = A_i \oplus (-R)$ of each of the obstacles $A_i$ and a reflected copy of $R$, for $i = 1, \ldots, k$; each of the $K_i$s is also called the *expanded obstacle* of $A_i$. The contact surfaces are then the boundaries $\partial K_i$ of the expanded obstacles. Indeed, each point that lies inside the expanded obstacle $K_i$ corresponds to a placement of the robot that results in a collision between $R$ and $A_i$, and vice versa; see [57, 98] for further details. Thus the free portion of the configuration space is the complement of the union of all the expanded obstacles. For each of the cells in the arrangement $\mathcal{A}(\mathcal{H})$, either all placements that it defines are free, or all of them are non-free. Thus the free configuration space is a substructure of the arrangement $\mathcal{A}(\mathcal{H})$, which consists of the union of all cells that contain free placements, and we can determine the set of all these placements by computing the union of the expanded obstacles.

Another application that concerns translational motion planning is to determine whether there exists a continuous motion of the robot between a given initial position and a given goal position. In the configuration space, this motion is a connected arc from the start to the goal placement, which must not cross any of the contact surfaces. We can use the arrangement $\mathcal{A}(\mathcal{H})$ to check whether there exists such a valid continuous motion. Such a decision is equivalent to deciding whether the two configurations lie in the same cell of the arrangement $\mathcal{A}(\mathcal{H})$. Thus deciding whether there exists a continuous collision-free motion between two robot placements, can be solved by computing the cell of the arrangement of the contact surfaces containing one of the placements, say the start placement, and by checking whether the other placement lies in the same cell. In fact, all free placements reachable from the start placement via a collision-free motion lie in this cell. As in the previous problem, this problem involves a substructure of the arrangement as well, which, in this case, is a single cell.

Consider a set $S$ of $n$ geometric objects of constant description complexity[1] in $d$-space. The *complexity* of the arrangement $\mathcal{A}(S)$ is the number of its cells (of all dimensions). This complexity might be $\Theta(n^d)$ in the worst case, and (some reasonable representation of) $\mathcal{A}(S)$ can be computed in $O(n^d \log n)$ time [124]. However, as discussed above, there are applications where one is only interested in parts of the arrangement. The hope is that substructures in arrangements should have smaller complexity than that of the full arrangement, and that therefore one should be able to compute them more efficiently (the complexity of a substructure is the number of cells of $\mathcal{A}(S)$ that it contains). See Section 1.1.1 for a detailed survey of related results, and Section 1.1.2 for a survey of *Voronoi diagrams*, whose combinatorial complexity is analyzed via substructures in arrangements.

On the algorithmic front, many algorithms for constructing substructures in arrange-

[1]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.

ments are based on randomization, that is, they use some source of random numbers ("coin-flips"), and their performance parameters, on any arbitrary input, are random variables. In these cases, we seek algorithms whose expected performance (over all possible coin flips) is good for *any* input. This approach is described in Section 1.1.3. In particular, a fundamental tool, based on a (randomized) divide-and-conquer mechanism, for manipulating arrangements, in both aspects, algorithmic and combinatorial, is $(1/r)$-*cutting*; see Section 1.1.4 for an overview.

### 1.1.1 Arrangements and their substructures

**Lower envelopes.** We first introduce a widely studied substructure in arrangements, the so-called *lower envelope* (resp., *upper envelope*). Let $\Gamma$ be a collection of $n$ surface patches in $d$-space of constant description complexity, each of which is the graph of some totally or partially defined function. The lower envelope $E_\Gamma$ of $\Gamma$ is the *pointwise minimum* of these functions. That is, $E_\Gamma$ is the graph of the following (possibly partially defined) function:

$$E_\Gamma(\mathbf{x}) = \min_{\gamma \in \Gamma} \gamma(\mathbf{x}), \qquad \text{for } \mathbf{x} \in \mathbb{R}^{d-1}.$$

If we project the lower envelope $E_\Gamma$ onto the $(d-1)$-dimensional space $\mathbf{x}_d = 0$, we obtain a subdivision of $\mathbb{R}^{d-1}$ into maximal connected relatively open cells of dimensions that range between 0 and $d-1$. For each cell $\Delta$ of this subdivision, a fixed subset of $\Gamma$ attains $E_\Gamma$ over all points $\mathbf{x} \in \Delta$, and no other function attains $E_\Gamma$ over any point in $\Delta$. This subdivision is called the *minimization diagram* of $\Gamma$, and we denote it by $M_\Gamma$. The complexity of $M_\Gamma$ is the number of cells (of all dimensions) of $M_\Gamma$, and the complexity of $E_\Gamma$ is defined similarly (and is equal to that of $M_\Gamma$). This complexity is known to be $O(n^{d-1+\varepsilon})$, for any $\varepsilon > 0$, [84, 123][2].

**A single cell.** As discussed in the overview above, one of the natural substructures in arrangements is a single cell. The complexity of a cell in an arrangement is the number of cells (of all dimensions) that appear on its boundary. For instance, the complexity of a 2-dimensional face is the number of edges and vertices along its boundary. In general, the complexity of a single cell of an arrangement $\mathcal{A}(\Gamma)$ (where $\Gamma$ is defined as above) is known to be $O(n^{d-1+\varepsilon})$, for any $\varepsilon > 0$; see [33, 85] and Section 1.5 for related results. In particular, we show in Chapter 5 that the combinatorial complexity of a single cell in an arrangement of $k$ convex polyhedra with a total of $n$ facets in 3-space is $O(nk^{1+\varepsilon})$, for any $\varepsilon > 0$; an improvement over the general bound $O(n^{2+\varepsilon})$ just cited.

A substructure that is related to a single cell is the *zone*. The zone of a surface $\sigma$ in $\mathcal{A}(\Gamma)$ is the collection of the cells crossed by $\sigma$. The complexity of the zone is the sum of

[2]In this thesis, a bound of the form $f(n) = O(n^{q+\varepsilon})$ implies that, for any arbitrarily small $\varepsilon > 0$, there exists a constant $c_\varepsilon$, such that $f(n) \le c_\varepsilon n^{q+\varepsilon}$. Generally, $c_\varepsilon$ tends to $\infty$ as $\varepsilon$ decreases to 0.

the complexities of these cells. When all the elements in $\Gamma$, as well as $\sigma$, have constant description complexity (though, sometimes, it suffices to require that $\sigma$ is convex), it can be shown that the zone of $\sigma$ is actually a single cell in another arrangement of $O(n)$ surfaces of constant description complexity (obtained by "cutting" the surfaces of $\Gamma$ at their intersections with $\sigma$), and thus the bound $O(n^{d-1+\varepsilon})$, for any $\varepsilon > 0$, applies to the complexity of the zone as well; see [33, 85].

**The planar case.** For planar arrangements, slightly sharper bounds are known on the complexities of lower envelopes, single cells, and zones. In the plane, it suffices to assume that $\Gamma$ is a collection of connected curves, each pair of which intersect at most $s$ times, for some constant $s$ (note that the curves need not have constant description complexity). In this setting, the maximum number of edges forming the lower envelope of the curves in $\Gamma$ is $\lambda_{s+2}(n)$, where $\lambda_q(n)$ is the maximum length of *Davenport-Schinzel sequences* of order $q$ on $n$ symbols (in fact, this maximum complexity becomes $\lambda_s(n)$ when the curves in $\Gamma$ are graphs of totally defined continuous functions); see [124] and below. The same asymptotic bounds also hold for the complexity of a single cell in $\mathcal{A}(\Gamma)$, as well as of the zone of a curve $\sigma$ in $\mathcal{A}(\Gamma)$, assuming that $\sigma$ intersects each of the curves in $\Gamma$ in a constant number of points (see [124] for details).

A Davenport-Schinzel sequence of order $s$ on $n$ symbols is a sequence composed of $n$ distinct symbols, such that **(i)** no two adjacent elements in the sequence are equal, and **(ii)** the sequence does not contain a (possibly non-contiguous) alternation $< a \ldots b \ldots a \ldots b \ldots >$ of length $s + 2$, for any two distinct symbols $a$ and $b$.

It is known that the maximum length $\lambda_s(n)$ of such a sequence is close to linear, for any constant value of $s$. In particular, $\lambda_1(n) = n$, $\lambda_2(n) = 2n - 1$, and $\lambda_3(n) = O(n\alpha(n))$, where $\alpha(n)$ is the (extremely slowly growing) inverse Ackermann function [124]. In general, for $s$ even,

$$\lambda_s(n) = n \cdot 2^{\Theta\left(\alpha(n)^{(s-2)/2}\right)},$$

which, as above, is nearly-linear, slightly super-linear, when $s$ is any fixed constant.

**Union of geometric objects.** Another substructure of arrangements that we widely study throughout this thesis is the union of geometric objects. We now let $\Gamma$ denote a set of $n$ geometric (full-dimensional) objects of constant description complexity in $\mathbb{R}^d$. The combinatorial complexity of their union is the overall number of cells (of all dimensions) of the arrangement of their boundaries (which, for simplicity, we denote by $\mathcal{A}(\Gamma)$) that appear on the union boundary. This complexity is known to be $\Theta(n^d)$ in the worst case, which, asymptotically, is identical to that of the entire arrangement $\mathcal{A}(\Gamma)$. Nevertheless, there are various cases for which this complexity can be shown to be roughly $O(n^{d-1})$ (and sometimes even smaller), such as the case of the union of the Minkowski sums arising in translational motion planning in 3-space, with a convex polyhedral robot and polyhedral obstacles (see the overview above). That is, in this case the complexity of the free configuration space is close to quadratic (in the overall complexity of the individual Minkowski sums of the robot and each of the obstacles); see [25]. For the analogous problem in the plane, this bound becomes only linear [93]. Another favorable case is when all the objects in $\Gamma$ are "fat" (see Section 1.6 for the formal definition). This case has attracted considerable attention in the past fifteen years. Most of the known earlier results involve planar instances, but considerable progress has been made on three-dimensional instances during the past few years; see Section 1.6 for a detailed survey of related results. In this thesis we make a significant contribution, by deriving, in Chapter 6, a nearly-quadratic bound on the complexity of the union of fat tetrahedra of arbitrary sizes in 3-space.

**Arrangements of linear objects.** Some of the above bounds are slightly (or significantly) sharper for arrangements of linear objects in three (and higher) dimensions.

Consider first an arrangement $\mathcal{A}(H)$ of $n$ hyperplanes in $d$-space, $d \geq 2$. In this case, the lower envelope of these hyperplanes is the boundary of a convex polyhedron in $d$-space (bounded by at most $n$ hyperplanes), and by the Upper Bound Theorem [108], it follows that its combinatorial complexity is $O\left(n^{\lfloor d/2 \rfloor}\right)$ (where the constant of proportionality depends on $d$). This bound is tight in the worst case. A single cell in the arrangement $\mathcal{A}(H)$ is also a convex polyhedron with the same combinatorial bound as above. Aronov *et al.* [22] have shown that the complexity of the zone in $\mathcal{A}(H)$ of a $(d-1)$-dimensional algebraic surface $\sigma$ of low degree is $O(n^{d-1} \log n)$, and the same bound applies when $\sigma$ is the boundary of a convex set in $\mathbb{R}^d$. In fact, when $\sigma$ has dimension $0 \leq \rho < d$, the more general bound established in [22] is $O\left(n^{\lfloor \frac{d+\rho}{2} \rfloor}\right)$, when $d - \rho$ is even, and $O\left(n^{\lfloor \frac{d+\rho}{2} \rfloor} \log n\right)$, when $d - \rho$ is odd. These bound are tight within a logarithmic factor.

When the given objects are $n$ $(d-1)$-dimensional simplices in $\mathbb{R}^d$, the complexity of their lower envelope is $O(n^{d-1}\alpha(n))$, and this bound is tight in the worst case; see [113]. The best known upper bound on the complexity of a single cell in an arrangement of such simplices is the slightly worse bound $O(n^{d-1} \log n)$ [24]. In Sections 1.5 and 1.6 we survey additional results that are related to substructures in such arrangements and also in three-dimensional arrangements of convex polyhedra.

### 1.1.2 Voronoi diagrams

The *Voronoi diagram* of a set of geometric objects is one of the central structures in computational geometry, and arises in many applications. It was first introduced by Voronoi [127] a hundred years ago, and became a major topic in computational geometry during the past thirty years. The reader is referred to Aurenhammer and Klein [32], Fortune [76], and Leven and Sharir [99] for comprehensive surveys of the subject; see also Boissonnat and Yvinec [37].

The Voronoi diagram of a set $S$ of disjoint objects, also referred to as "sites", in $d$-space, under a metric (or "convex distance function")[3] $\rho(\cdot,\cdot)$, is a subdivision of $\mathbb{R}^d$ into cells, one for each site, such that the cell associated with a site $s \in S$ consists of the points in space for which $s$ is *closer* (under the metric $\rho$) than all other sites of $S$. Formally, the Voronoi diagram $\mathcal{V}or(S)$ of $S$ is the subdivision of $\mathbb{R}^d$ into *Voronoi cells* $\mathcal{V}(s)$, for each $s \in S$, such that

$$\mathcal{V}(s) = \{\mathbf{x} \in \mathbb{R}^d \mid \rho(\mathbf{x}, s) \le \rho(\mathbf{x}, s'),\ \forall s' \in S, s' \neq s\}, \tag{1.1}$$

where $\rho(\mathbf{x}, s) = \min\{\rho(\mathbf{x}, \mathbf{y}) \mid \mathbf{y} \in s\}$. The combinatorial complexity of $\mathcal{V}or(S)$ is the overall number of vertices, edges and faces (of all dimensions) that appear on the boundaries of its cells.

Let us consider the special (and most common in computational geometry) case, where the set $S$ consists of $n$ points in $d$-space, and $\rho$ is the Euclidean metric. For each pair of distinct sites $s$, $s'$ in $S$, the condition $\rho(\mathbf{x}, s) \le \rho(\mathbf{x}, s')$ defines a halfspace, which contains $s$ and is bounded by the hyperplane bisecting (and perpendicular to) the segment $ss'$. Hence, $\mathcal{V}(s)$ is the intersection of $n-1$ such halfspaces, and is thus a convex (possibly unbounded) polyhedron. In particular, when $d = 2$, $\mathcal{V}or(S)$ is a convex subdivision of the plane into $n$ convex cells, and Euler's formula for planar graphs implies that the overall complexity of the diagram is $O(n)$ in this case; see, e.g., [57] for the full analysis. In fact, in almost all cases, planar Voronoi diagrams have linear complexity (see [124]).

Voronoi diagrams are strongly related to lower envelopes of "distance functions" in a straightforward manner, first noticed in [61]. Define a collection $\mathcal{F}_S = \{\rho_s \mid s \in S\}$ of $n$ $d$-variate functions, where $\rho_s(\mathbf{x}) = \rho(\mathbf{x}, s)$, for $s \in S$. It now follows by definition that $\mathcal{V}or(S)$ is the minimization diagram of $\mathcal{F}_S$. Hence the problem of bounding the combinatorial complexity of Voronoi diagrams, as well as constructing them efficiently, can be solved using the machinery developed for lower envelopes of $d$-variate functions.

In particular, for the case where $S$ is a set of points as above, it can be shown that $\mathcal{V}or(S)$ is the minimization diagram of $n$ hyperplanes in $\mathbb{R}^{d+1}$ (see, e.g., [124] for the full analysis), and thus their lower envelope is the boundary of a convex polyhedron in $\mathbb{R}^{d+1}$ (bounded by at most $n$ hyperplanes). Its combinatorial complexity is $O\left(n^{\lceil d/2 \rceil}\right)$ [108], which implies the same bound on $\mathcal{V}or(S)$.

In Chapter 4 we analyze the Iterative Closest Point (ICP) algorithm, and exploit the structure of Voronoi diagrams of a set of points in one and higher dimensions, as well as the overlay of Voronoi diagrams of this kind.

### 1.1.3 Randomized algorithms

Randomization has become a central tool for many geometric problems, both algorithmic and combinatorial. In particular, in order to achieve efficient (deterministic) solutions for many computational problems (in geometry and elsewhere), the resulting algorithms tend to be too complicated and difficult to design, and, consequently, their performance in

[3]In the simplest case, $\rho$ is the Euclidean metric.

practice tend to be poor. Here is where randomization plays an important role: It avoids the use of complicated data structures, and uses instead simple-minded procedures that might be prohibitively expensive in the worst case, but behave very well when the input is "shuffled" in a random manner. This makes randomized algorithms considerably simpler than their deterministic counterparts, and therefore easier to implement, which improves their performance in practice. The running time of such a randomized algorithm is a random variable, which depends on internal draws of random numbers ("coin flips") made by the algorithm. One then seeks an algorithm whose *expected* running time is small, no matter how "bad" is the input data. Although efficiency is guaranteed only on average, it can be shown, for many of these algorithms (using a more involved probabilistic analysis) that their actual running time is close to its expectation, with high probability.

Many of the randomized algorithms in geometry are based on (i) a divide-and-conquer approach, where the problem is partitioned into roughly equal-sized smaller subproblems, which are then solved recursively, or (ii) a randomized incremental construction, where we add the given input objects one by one in a random order, and, after each insertion, we update the structure that we wish to construct, keeping it representing the elements added so far. See a survey by Clarkson [50], presenting both approaches.

The field of randomized algorithms in computational geometry was introduced by Clarkson [49, 52] and by Clarkson and Shor [53], and has developed rapidly during the past twenty years. Mulmuley's book [109] is devoted to randomized approaches to computational geometry problems; see also a survey for randomized algorithms in geometric optimization by Agarwal and Sen [12].

In Chapter 2 we present an efficient randomized algorithm that constructs the union of geometric objects of constant description complexity.

### 1.1.4 Geometric cuttings

A natural approach for solving various problems in computational geometry is the divide-and-conquer paradigm. A typical application of this paradigm to, say, problems involving a set $\Gamma$ of $n$ curves in the plane of constant description complexity, is to fix a parameter $r > 0$, and to partition the plane into regions $R_1, \ldots, R_m$ (those regions are usually vertical pseudo-trapezoids), such that the number of curves of $\Gamma$ that intersect the interior of $R_i$ is at most $n/r$, for any $i = 1, \ldots, m$. This allows us, in many cases, to split the problem at hand into subproblems, each involving the subset of curves intersecting a region $R_i$. Such a partition is called a $(1/r)$-*cutting* of the plane, with respect to the set $\Gamma$. See [2] for a survey of algorithms that use cuttings, and [9] for further work related to cuttings. The notion of cuttings can be extended to higher-dimensional arrangements; see [47]. In this thesis we will be concerned with cuttings for planar arrangements and three-dimensional arrangements.

The first (though not optimal) construction of cuttings is due to Clarkson [49], where $m = O(r^2 \log^2 r)$. Chazelle and Friedman [47] showed the existence of $(1/r)$-cuttings with

$m = O(r^2)$ (a bound that is worst-case tight). Their results are extended to arrangements of hyperplanes in $d$-space, $d \geq 1$, with $m = O(r^d)$; see [47]. An optimal deterministic algorithm for generating cuttings of hyperplanes in $d$-space was given by Chazelle [43], with $m = O(r^d)$ (again, a tight bound), and construction time $O(nr^{d-1})$, though those constructions do not seem to produce a practically small number of regions. For other types of surfaces, in $d \geq 3$ dimensions, the bounds are not that tight. Since we will not be using them in the thesis, we omit the further details (for which see [94, 124]).

**Curve-sensitive cuttings.** Let $S$ be a set of $n$ surfaces in 3-space of constant description complexity, let $C$ be a set of $m$ curves in 3-space of constant description complexity, and let $1 \leq r \leq \min\{m, n\}$ be a given parameter. A $(1/r)$-cutting of $S$, which is *sensitive* to the curves in $C$, is a subdivision of 3-space into connected cells, each of constant description complexity, so that each cell is crossed by at most $n/r$ surfaces of $S$, and the number of cells in the cutting is nearly $O(r^3)$ (these are the standard requirements from a three-dimensional cutting), and, in addition, the overall number of crossings between the curves of $C$ and cells of the cutting is roughly $O(mr)$. This notion was recently introduced by Koltun and Sharir [95]. The first properties can be guaranteed, with high probability, by applying one of the standard sampling constructions of, e.g., Clarkson [49], or Chazelle and Friedman [47], as reviewed above. However, the second property is not necessarily guaranteed by the above techniques, as the number of crossings may be $\Omega(mr^2)$ in the worst case. See [95] for a specific construction of curve-sensitive cuttings.

Curve-sensitive cuttings find many applications in contexts that involve interactions between surfaces and curves in 3-space. For example, curve-sensitive cuttings can be used to obtain a bound of roughly $O(m^{1/2}n^2)$ on the complexity of the *multiple zone* of $m$ curves in an arrangement of $n$ surfaces in 3-space, all of constant description complexity [95]. The multiple zone is defined as the collection of all cells of the arrangement of the given surfaces that are crossed by at least one of the curves.

In Chapter 7 we use planar cuttings in order to bound the number of "regular" vertices that appear on the boundary of the union of planar convex sets, such that the boundaries of any pair of these sets intersect in a constant number of points. In Chapters 3, 6 we represent two major studies that incorporate curve-sensitive cuttings (albeit in more restricted settings): In Chapter 3 we derive an algorithm that efficiently counts and represents in a compact form all triples of intersections among triangles in three dimensions, and in Chapter 6 we present a nearly-quadratic bound on the complexity of the union of "fat" tetrahedra in 3-space.

## 1.2 Output-Sensitive Construction of the Union of Triangles

In this, and in each of the subsequent sections of the introduction, we present an overview of the results obtained in the corresponding Chapters 2–7.

### Overview

Many computational geometry problems deal with the task of constructing (the boundary of) the union of geometric objects. The most classical problem is perhaps the construction of the union of triangles in the plane. Problems of this kind arise in motion planning [98], where we wish to construct the forbidden portions of the configuration space (see Section 1.1), in hidden surface removal for visibility problems in three dimensions [109], in finding the minimal Hausdorff distance under translation between two sets of points (or of segments) in $\mathbb{R}^2$ [92], in applications in geographic information systems [56], and in many other areas. The goal is to design an efficient algorithm that constructs the union of the given triangles. Computing the union by constructing the full arrangement of the $n$ input triangles requires $\Theta(n^2)$ time in the worst case, which, in most cases, is wasteful, since the combinatorial complexity of the union boundary might be considerably smaller. Ideally, one would like to have an algorithm that computes the union in an "output-sensitive" fashion, so that the running time depends on the actual union complexity. Nevertheless, it is strongly believed that such an "output-sensitive" algorithm for this problem, which runs in subquadratic time, when the boundary of the union has subquadratic complexity, is unlikely to exist, since this problem belongs to the family of *3SUM-hard* problems [77], which are problems that are very likely to require $\Omega(n^2)$ time in the worst case. However, subquadratic algorithms exist in several special cases.

**Output sensitivity.** Expanding upon the preceding discussion, there are two obvious ways to define output sensitivity (other than just measuring the complexity of the boundary of the union). 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 1.1 for an illustration of the two measures. Note that if the output size is $\xi$, 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 in the sense discussed below. Consider the reduction, as presented in [77], 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) 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

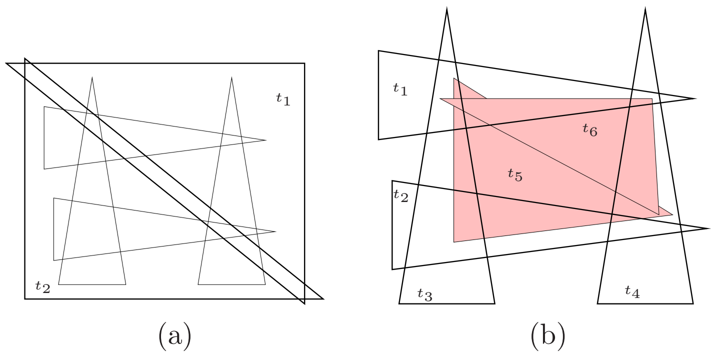


Figure 1.1: (a) An arrangement of six triangles, illustrating the first measure of output sensitivity. The triangles $t_1$ and $t_2$ cover the entire union, so the output size is $2$. (b) Illustrating the second measure of output sensitivity. The union boundary is determined only by the triangles $t_1, \ldots, t_4$, even though the triangles $t_5$ and $t_6$ cover the hole created by $\bigcup_{i \le 4} t_i$. The output size is $4$ according to the second measure, and $6$ according to the first one.

reduce this latter problem to our problem, as follows. Let $\mathcal{A}$ denote an algorithm that efficiently computes the union of $n$ triangles in the plane, in terms of the second measure, and let $T_{\mathcal{A}}(n, \xi)$ denote its running time, expressed as a function of $n$ and of the "output size" $\xi$. We assume that $T_{\mathcal{A}}(n, \xi) = o(n^2)$ 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 $\mathcal{A}$ 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 $\mathcal{A}$ will terminate in a predictable subquadratic time. We thus run $\mathcal{A}$. 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 $\mathcal{A}$, 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 presented in Chapter 2.

## Related work

The union of $n$ triangles can be trivially computed in time $O(n \log n + k \log n)$, where $k$ is the complexity of the whole arrangement of the triangles, using a line-sweeping algorithm (see [57]). There are several special cases in which the union of (any subfamily of) the given triangles has nearly-linear complexity, a property that leads to nearly-linear algorithms for constructing the union. One such case is that of *fat* triangles (namely, triangles all of whose angles are at least some constant positive angle). Another case involves 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 — see Section 1.1). In these cases, the union has only linear or nearly-linear complexity [93, 106, 107], 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 arbitrary triangles, then the union can have quadratic complexity in the worst case. The challenge here is to compute the union in subquadratic time if it has subquadratic complexity. As noted above, this is likely to be impossible, because even determining whether the union has a hole is a 3SUM-hard problem [77], and thus unlikely to be solvable in subquadratic time. Still, in certain favorable instances a subquadratic solution may be possible. For example, one can employ the randomized incremental construction (RIC) of Agarwal and Har-Peled [6], whose analysis is based on Mulmuley's *theta series* [109]. 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, the RIC algorithm has good performance, provided that the *depth* $d(v)$ of most of the vertices $v$ of the arrangement induced by the $n$ input triangles, is large enough (the depth $d(v)$ is the number of input triangles that contain $v$ in their interior). We refer to such vertices as being *deep*. Otherwise, when most of the vertices of the arrangement are *shallow*, the RIC algorithm performs poorly. In this case, one can employ the *Disjoint Cover* (DC) algorithm that we have earlier developed in [68], 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, that we have presented in [68]), 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.

## Contributions

In Chapter 2 we present an efficient algorithm that computes the union in an "output-sensitive" manner (according to the first measure). That is, 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 $\bigcup S = \bigcup T$. The running time of the algorithm is $O(n^{4/3} \log n + n\xi \log^2 n)$, which is subquadratic when $\xi = o(n/\log^2 n)$. In our solution, we use the method of Brönnimann and Goodrich [40] for finding a set cover in a set system of finite VC-dimension, such that the size of this cover is at most $O(\log \xi)$ times the optimal size $\xi$. In itself, the running time of the Brönnimann-Goodrich technique is rather inefficient, but we provide a careful implementation of (a certain modified variant of) it for the case at hand, and thereby obtain an algorithm with the aforementioned subquadratic running time. 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 (as in the first measure for the planar case).

The results of Chapter 2 appeared in [71].

## 1.3 Counting and Representing Intersections Among Triangles in Three Dimensions

### Overview

Intersection problems are among the most basic problems in computational geometry, with many different application areas, such as solid modeling [97], robotics and motion planning [98], geographic information systems [56], computer graphics [109], and many others. The problem of *reporting* all intersections in a given collection of geometric objects has received considerable attention, and several output-sensitive algorithms have been designed in order to solve this problem efficiently (mostly for planar instances). However, in some applications, we are only interested in *counting* the overall number of intersections, without reporting them explicitly. In this case, one prefers an algorithm whose running time does not depend on the number of intersections (which, in the worst case, is proportional to the size of the arrangement induced by the input objects, and can be $\Theta(n^d)$ in $d$ dimensions). A more general problem is to construct a compact representation of the intersection set, as an edge-disjoint union of complete $d$-partite hypergraphs (this problem is more general, since one can use the compact representation in order to efficiently count the overall number of intersections). One motivation for constructing such a compact representation is that it facilitates a simple mechanism for sampling a random element out of the entire set of intersections, without constructing this set explicitly.

While planar versions of the intersection counting problem have been successfully solved [3], no efficient technique was known for counting *triple* intersections in the 3-dimensional case. This subproblem actually arose in (one variant of) our study of the problem of computing efficiently the union of simplices in $\mathbb{R}^3$ (extending the planar result, reported above, to three dimensions). Another interesting application of this problem is to select the $k$-th vertex in a given direction $d$ in an arrangement of $n$ triangles in $\mathbb{R}^3$. This can be performed using a randomized scheme in three dimensions (similar to those in [54, 103], given for the two-dimensional case), that is based on drawing a random vertex in the arrangement, and counting the number of vertices in the arrangement that lie in a given halfspace, bounded by a plane through that vertex.

## Related work

Many intersection problems involving geometric objects in the plane have been investigated, such as reporting all intersections in a set of general arcs [34], detecting a *red-blue* intersection between two sets of "red" and "blue" Jordan arcs [15], and counting intersections in a set of segments [3, 42, 79, 118], or in a set of circular arcs [11]. The best algorithm for counting intersections among $n$ line segments in the plane takes $O(n^{4/3}\log^{1/3} n)$ time [43]. (Of course, if the number $k$ of intersections is smaller than $n^{4/3}\log^{1/3} n$, an output-sensitive algorithm for *reporting* the intersections will perform faster; see, e.g., [109].) In contrast, there exist much fewer studies of intersection problems involving objects in three dimensions. de Berg *et al.* [55] provide a procedure for constructing all intersecting pairs in a collection of $n$ triangles in $\mathbb{R}^3$ in an output-sensitive manner. The running time of this procedure is $O(n^{4/5+\varepsilon}\kappa^{4/5+\varepsilon})$, for any $\varepsilon > 0$, where $\kappa$ is the number of intersecting pairs. (This bound is meaningful only when $\kappa \ll n^{3/2}$.) Agarwal and Matoušek [8] provide an algorithm for segment intersection queries, in which we are given a collection of $n$ triangles in $\mathbb{R}^3$ and, for any query segment $e$, wish to report all the input triangles that $e$ intersects. It follows from their analysis that $m$ queries can be answered in time $O(m^{4/5+\varepsilon}n^{4/5+\varepsilon} + \kappa)$, for any $\varepsilon > 0$, where $\kappa$ is the total output size to all queries. Applying this algorithm to the $3n$ edges of $n$ given triangles, we obtain a procedure that reports all $\kappa$ intersecting pairs of the triangles, in time $O(n^{8/5+\varepsilon} + \kappa)$, for any $\varepsilon > 0$. A later work of Agarwal *et al.* [4] presents an algorithm for counting or reporting all intersecting pairs in a collection of convex polytopes in three dimensions. Counting requires time $O(n^{8/5+\varepsilon})$, for any $\varepsilon > 0$, where $n$ is the overall complexity of the input polytopes, and reporting takes $O(n^{8/5+\varepsilon} + \kappa)$ time, where $\kappa$ is the number of intersecting pairs of polytopes. (This is a potential improvement over the algorithm of [8], since $\kappa$ counts here the number of intersecting pairs of polytopes, and not of pairs of facets.)

We are not aware of any previous specific work on counting (efficiently) intersecting *triples* among $n$ objects in three dimensions, although known solutions for the planar case can be employed to solve (suboptimally) three-dimensional instances. (For example, by intersecting each fixed triangle $\Delta$ with all the other ones, by counting the number of intersections among the resulting segments on $\Delta$, and by summing these counts over all $\Delta$, we obtain an $O(n^{7/3}\log^{1/3} n)$ solution.)

## Contributions

In Chapter 3 we present an efficient algorithm that counts all intersecting triples among a collection $T$ of $n$ triangles in $\mathbb{R}^3$ in nearly-quadratic time. Our algorithm is recursive, and exploits (a restricted variant of) 3-dimensional curve-sensitive cuttings (see [95], Section 1.1.4, and below for more details). More specifically, we recursively partition $\mathbb{R}^3$ using such a cutting. Each triangle is decomposed into portions that lie in different cells of the cutting. We take each such portion $\Delta$, and intersect it with all other triangles, obtaining a system of segments within $\Delta$. We then count the number of intersections between these segments, applying standard techniques that count intersections between lines, and between line-segments and lines in the plane [3]. The recursion is handled in a careful manner that ensures that the algorithm indeed runs in nearly-quadratic time.

The main technical issue that our algorithm faces is the need to avoid direct computation of the intersections among the segments on each triangle. As already observed, doing so, using the best algorithms for this planar problem [43], would result in an overall running time $O(n \cdot n^{4/3} \log^{1/3} n) = O(n^{7/3} \log^{1/3} n)$.

Using a variant of this technique, it is possible to construct a representation of the triple-intersection hypergraph of the triangles in $T$ as an edge-disjoint union of complete tripartite subhypergraphs $\{A_i \times B_i \times C_i\}_{i=1}^{s}$ (where $s$ is the overall number of subhypergraphs), so that $\sum_{i=1}^{s}(|A_i| + |B_i| + |C_i|) = O(n^{2+\varepsilon})$, for any $\varepsilon > 0$. The construction takes $O(n^{2+\varepsilon})$ time as well.

We also extend our technique to count or represent all intersecting triples among $n$ planar objects of constant description complexity that lie in distinct planes. These extensions also run in nearly-quadratic time.

Finally, we show that it is unlikely that the triangle intersection counting problem has a subquadratic solution, since it belongs to the 3SUM-hard family [77], and thus our algorithm is likely to be nearly worst-case optimal.

The results of Chapter 3 appeared in [72].

## 1.4 Analysis of the ICP Algorithm

### Overview

The matching and analysis of geometric patterns and shapes is an important problem that arises in various application areas, in particular in computer vision and pattern recognition [18]. In a typical scenario, we are given two objects $A$ and $B$, and we wish to determine how much they *resemble* each other. Usually one of the objects may undergo certain transformations, like translation, rotation and/or scaling, in order to be matched with the other object as well as possible. In many cases, the objects are represented as finite sets of (sampled) points in two or three dimensions (they are then referred to as "point patterns" or "shapes"). In order to measure "resemblance", various *cost functions* have been used. Two prominent ones among them are the (one-sided) *Hausdorff distance* [18], and the *sum of squared distances* or *root mean square* [35, 88], which we also call RMS, for short. Under the first measure, the cost function is $\Phi_\infty(A, B) = \max_{a \in A} \|a - N_B(a)\|$, and under the second measure, it is $\Phi_2(A, B) = \frac{1}{m} \sum_{a \in A} \|a - N_B(a)\|^2$, where $\| \cdot \|$ denotes the Euclidean norm[4], $N_B(a)$ denotes the nearest neighbor of $a$ in $B$, and $m = |A|$.

[4]Of course, other norms or metrics can also be considered, but in this study we only treat the Euclidean case.

A heuristic matching algorithm that is widely used, due to its simplicity (and its good performance in practice), is the *Iterative Closest Point* algorithm, or the *ICP* algorithm for short, of Besl and McKay [35]. It typically aims to minimize the resemblance cost function under translations. That is, given two point sets $A$ and $B$ in $\mathbb{R}^d$ (also referred to as the *data shape* and the *model shape*, respectively), we wish to minimize a cost function $\phi(A+t, B)$, over all *translations* $t$ of $A$ relative to $B$. The algorithm starts with an arbitrary translation that aligns $A$ to $B$ (suboptimally), and then repeatedly performs local improvements that keep re-aligning $A$ to $B$, while decreasing the given cost function $\phi(A+t, B)$, until no improvement is possible. In the original version of the ICP algorithm, the only cost function used is the sum of squared distances (see [35, 78, 119, 120, 125]), but the algorithm also allows $A$ to be rotated in order to be matched with $B$. In Chapter 4 we analyze the ICP algorithm only under translations, but we also consider the (one-sided) Hausdorff distance cost function, as defined above, and analyze the algorithm according to either of these two measures of resemblance.

The ICP algorithm, under translations, proceeds as follows. At the $i$-th iteration, the set $A$ has already been translated by some vector $t_{i-1}$, where $t_0 = \overrightarrow{0}$ (there are many heuristics, which we ignore, to get a good starting translation $t_0$). We then apply the following two steps:

(i) We assign each (translated) point $a + t_{i-1} \in A + t_{i-1}$ to its nearest neighbor $b = N_B(a + t_{i-1}) \in B$ under the Euclidean distance. (ii) We then compute the new relative translation $\Delta t_i$ that minimizes the cost function $\phi$ (with respect to the above fixed assignment). Specifically, under the one-sided Hausdorff distance, we find the $\Delta t_i$ that minimizes

$$\phi_\infty(A + t_{i-1}, \Delta t_i, B) = \max_{a \in A} \|a + t_{i-1} + \Delta t_i - N_B(a + t_{i-1})\|,$$

and under the sum of squared distances, we minimize

$$\phi_2(A + t_{i-1}, \Delta t_i, B) = \frac{1}{m} \sum_{a \in A} \|a + t_{i-1} + \Delta t_i - N_B(a + t_{i-1})\|^2.$$

We then align the points of $A$ to $B$ by translating them by $\Delta t_i$, so the new (overall) translation is $t_i = t_{i-1} + \Delta t_i$.

In other words, in stage (i) of each iteration of the ICP algorithm we assign the points in (the current translated copy of) $A$ to their respective nearest neighbors in $B$, and in stage (ii) we translate the points of $A$ in order to minimize the value of the cost function with respect to the assignment computed in stage (i). This in turn may cause some of the points in the new translated copy of $A$ to acquire new nearest neighbors in $B$, which causes the algorithm to perform further iterations. If no point of $A$ changes its nearest neighbor in $B$, the value of the cost function does not change in the next iteration (in fact, the next relative translation equals $\overrightarrow{0}$) and, as a consequence, the algorithm terminates. (As a matter of fact, the ICP algorithm in its original presentation stops when the difference

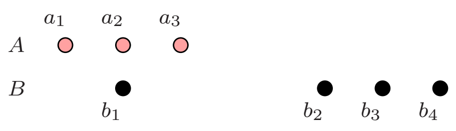


Figure 1.2: A local minimum in $\mathbb{R}^1$ of the ICP measures. The global minimum is attained when $a_1$, $a_2$, $a_3$ are aligned on top of $b_2$, $b_3$, $b_4$, respectively.

in the cost function falls below a given threshold $\tau > 0$; however, in our analysis, we assume that $\tau = 0$). It is shown by Besl and McKay [35] that, when $\phi(\cdot, \cdot)$ is the sum of squared distances, this algorithm always converges to a local minimum, moreover, the value of the cost function decreases at each iteration (we definitely decrease it with respect to the present nearest-neighbor assignment, and the revised nearest-neighbor assignment at the new placement can only decrease it further). An easy variant of their proof (noted in Chapter 4) establishes convergence also when the cost function is the (one-sided) Hausdorff distance.

Note that the pattern matching performed by the algorithm is *one-sided*, that is, it aims to find a translation of $A$ that places the points of $A$ near points of $B$, but not necessarily the other way around.

Since the value of the cost function is strictly reduced at each iteration of the algorithm, it follows that no nearest-neighbor assignment arises (in its entirely) more than once during the course of the algorithm, and thus it is sufficient to bound the overall number of nearest-neighbor assignments (or, NNA's, for short) that the algorithm reaches in order to bound the number of its iterations.

Which minimum the algorithm converges to depends on the initial position of the input points (see [35] for details and for a heuristic that "helps" the algorithm to converge in practice to the global minimum). There are simple constructions, such as the one depicted in Figure 1.2, that show that the algorithm may terminate at a local minimum that is quite different (and far) from the global one, under either of the resemblance measures that we use. (Nevertheless, as many practical experimentations indicate, the convergence to the (possibly local) minimum is rather fast in practice [35, 78, 119, 120].) Still, this is a disadvantage of the algorithm from a theoretical "worst-case" point of view, and the potential convergence to a local minimum raises several interesting questions. The most obvious question is to obtain sharp upper and lower bounds on the maximum possible number of local minima that the function can attain. Another is to analyze the decomposition of space into "influence regions" of the local minima, where each such region consists of all the translations from which the algorithm converges to a fixed local minimum.

## Related Work

The pattern matching problem is a central and important problem that arises in many applications, ranging from surveillance to structural bioinformatics, and the ICP algorithm has been identified and used as a practical heuristic solution over the past fifteen years. Many experimental reports on its performance, including additional heuristic enhancements of it (e.g., in finding a good initial translation and using various techniques for sampling points from the input model) have been published [35, 78, 120, 125]. Still, to the best of our knowledge, this technique has never before been subject to a serious and rigorous analysis of its worst-case behavior, which it definitely deserves.

Nevertheless, there are a few recent studies, which analyze the performance of related algorithms. One such algorithm is Lloyd's method for $k$-means clustering, which was first studied by Har-Peled and Sadri [88], who established upper and lower bounds on the number of iterations performed by this method. Their bounds are polynomial in the size of the input set and in its *spread* (i.e., the ratio between the diameter of the set and the distance between its closest pair of points). In particular, they presented a lower bound construction that yields $\Omega(n)$ iterations for point sets on the real line. The lower bound has later been improved by Arthur and Vassilvitskii [29] to $2^{\Omega(\sqrt{n})}$, when the dimension $d$ of the problem, as well as the number of clusters, are $\Omega(\sqrt{n})$.

## Contributions

In Chapter 4 we first show a (probably weak) upper bound of $O\left(m^d n^d\right)$ (where $n = |B|$) on the number of iterations of the algorithm in $\mathbb{R}^d$ under either of the two measures, for any $d \geq 1$. We then present several structural geometric properties of the algorithm under the RMS measure. Specifically, we show that at each iteration of the algorithm the (real) cost function monotonically and strictly decreases, in a continuous manner, along the vector $\Delta t$ of the relative translation; this is a much stronger property than the originally noted one, that the value at the end of the translation is smaller than that at the beginning. As a result, we conclude that the polygonal path $\pi$ obtained by concatenating all the relative translations that are computed during the execution of the algorithm, does not intersect itself. In particular, for $d = 1$, the ICP algorithm is *monotone* — all its translations are in the same (left or right) direction. Next, we present a lower bound construction of $\Omega(n \log n)$ iterations for the one-dimensional problem under the RMS measure (assuming $m \approx n$). The upper bound is quadratic. We also discuss the problem under the (one-sided) Hausdorff distance measure. In particular, we present for the one-dimensional problem an upper bound of $O\left((m+n) \log \delta_B / \log n\right)$ on the number of iterations of the algorithm, where $\delta_B$ is the spread of the input point set $B$. We then present a tight lower bound construction with $\Theta(n)$ moves, for the case where the spread of $B$ is polynomial in $n$. We also study the problem under the Hausdorff measure in two and higher dimensions, and show that some of the structural properties of the algorithm that hold for the RMS measure do not hold in this case.

The results of Chapter 4 appeared in [75]. We note that our work has triggered further study of the problem by Arthur and Vassilvitskii [30], who have improved our lower bound construction, and have shown that, for the RMS measure in one dimension, $\Omega(n^2)$ moves might be required in the worst case.

## 1.5 A Single Cell in an Arrangement of Convex Polyhedra in $\mathbb{R}^3$

### Overview

Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be a collection of $k$ convex polyhedra in 3-space having $n$ facets in total, and let $\mathcal{A}(\mathcal{P})$ denote the three-dimensional arrangement induced by the polyhedra in $\mathcal{P}$. The problem that we study in Chapter 5 is to obtain a sharp upper bound on the combinatorial complexity of a single cell in such an arrangement, and to construct such a cell efficiently.

Repeating and extending some of the details discussed in Section 1.1, a major application of this problem is to *translational motion planning*, where a rigid polyhedral robot $R$ (*not necessarily convex*) translates in a fixed polyhedral environment, and one wishes to compute all free placements of $R$ (at which it avoids collision with any obstacle) which are reachable via a continuous collision-free motion from a given initial free placement. Assume that $R$ can be represented as the union of $k_1$ convex polyhedra $R_1, \ldots, R_{k_1}$, and that the obstacles consist of $k_2$ convex polyhedra $A_1, \ldots, A_{k_2}$. (The case when $R$ is a single convex robot admits a different treatment; see [25] and below.) For each $1 \le i \le k_1$ and $1 \le j \le k_2$, set $K_{i,j}$ to be $A_j \oplus (-R_i)$, the Minkowski sum of $A_j$ and a reflected copy of $R_i$. Then the free portion of the configuration space is the complement of $\bigcup_{i,j} K_{i,j}$, and the subset of all placements reachable from a given initial free placement $q_0$ via a collision-free motion is a single cell (connected component) of this complement, namely, the cell containing $q_0$, which is also a single cell in the arrangement of the polyhedra $K_{i,j}$. See [25, 121] for more details.

### Related work

In two dimensions, Guibas *et al.* [80] showed that the complexity of a single face in an arrangement of $n$ Jordan arcs, each pair of which intersect in at most some constant number $s$ of points, is $O(\lambda_{s+2}(n))$ (see also Section 1.1.1). Efficient algorithms for computing a single face in two dimensions are given in [80] and [46]. Aronov and Sharir [26] showed that the combinatorial complexity of a single face in an arrangement of $k$ convex polygons in the plane, having $n$ edges in total, is $O(n\alpha(k))$, and this bound is tight in the worst case. This (ever so slightly) improves the bound $O(\lambda_3(n)) = O(n\alpha(n))$ obtained from the general results of [80].

In higher dimensions, it was shown by Aronov and Sharir [24] that the combinatorial complexity of a single cell in an arrangement of $n$ $(d-1)$-simplices in $\mathbb{R}^d$ is $O(n^{d-1}\log n)$, and that this bound also holds for the complexity of the zone of an additional algebraic surface $\sigma$ of constant degree, or any convex surface, in such an arrangement. They also presented a randomized nearly-quadratic algorithm that constructs a single cell in an arrangement of $n$ triangles in 3-space (see also [23]). Finally, Halperin and Sharir [85] presented a nearly-quadratic bound on the combinatorial complexity of a single cell (or a zone) in an arrangement of $n$ low-degree algebraic surface patches in 3-space. These results have later been extended to any dimension $d \geq 3$ by Basu [33], where the bounds are $O(n^{d-1+\varepsilon})$, for any $\varepsilon > 0$. We note (see also Section 1.1.1) that the case of a single cell is a generalization of the simpler problem involving the lower envelope of the given surfaces. This simpler problem has been studied earlier [84, 123], with the same asymptotic bounds on its complexity, as those mentioned above.

Applying these results to the case of convex polyhedra, we obtain bounds that depend only on the total number $n$ of facets, and do not exploit the fact that these facets are organized in a (potentially) smaller number of convex structures. Previous results that do exploit this fact mostly study the complexity of the (boundary of the) *entire union* of $\mathcal{P}$. A detailed survey of related results is given in Section 1.6, however, for the sake of completeness, we discuss some of these results below. Aronov and Sharir [26] showed that the complexity of the union of $k$ convex polygons with a total of $n$ edges in the plane is $O(k^2 + n\alpha(k))$. Aronov *et al.* [27] showed that the complexity of the union of $k$ convex polyhedra with a total of $n$ facets in $\mathbb{R}^3$ is $O(k^3 + nk\log k)$, and it can be $\Omega(k^3 + nk\alpha(k))$ in the worst case. The bound was improved by Aronov and Sharir [25] to $O(nk\log k)$ (and $\Omega(nk\alpha(k))$) when the given polyhedra are Minkowski sums of a fixed convex polyhedron with $k$ *pairwise-disjoint* convex polyhedra. In the motion-planning context mentioned above, this problem arises in the case of a *convex* translating robot $R$. Our new result (see below) yields a comparable bound (for only one cell, though) when the translating robot is not convex.

As noted above, the single-cell problem is a generalization of the related problem of bounding the complexity of the lower envelope of the polyhedra in $\mathcal{P}$. Specifically, except for vertical discontinuities of the envelope, each of its features also appears on the boundary of the unbounded cell; the number of extra features of discontinuity is also asymptotically bounded by the complexity of the unbounded cell. See, e.g., [124] for further details. The case of lower envelopes is easier to analyze, and the special case of convex polyhedra was settled by Huttenlocher *et al.* [92], who presented an upper bound of $O(nk\alpha(nk))$, which can easily be improved to $O(nk\alpha(k))$.

On the algorithmic front, Aronov *et al.* [27] presented a randomized incremental algorithm for computing the boundary of the union of $k$ convex polyhedra in 3-space having

$n$ facets in total, whose expected running time is $O(k^3 + nk \log k \log n)$. Huttenlocher *et al.* [92] presented a deterministic algorithm that constructs the lower envelope in an arrangement of this kind in time $O(nk^{1+\varepsilon})$, for any $\varepsilon > 0$. Aronov and Sharir [24] showed that a single cell in an arrangement of $n$ triangles in 3-space can be computed in randomized expected time $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$. They also showed that if all the triangles lie in planes having only a constant number of orientations, then a single cell in an arrangement of this kind can be constructed deterministically in $O(n^2 \log n)$ time. For the general case of algebraic surface patches of constant description complexity, Schwarzkopf and Sharir [122] showed that a single cell can be constructed in randomized expected $O(n^{2+\varepsilon})$ time, for any $\varepsilon > 0$.

## Contributions

In Chapter 5 we show that the combinatorial complexity of a single cell of $\mathcal{A}(\mathcal{P})$ is $O(nk^{1+\varepsilon})$, for any $\varepsilon > 0$, thus settling a conjecture of Aronov *et al.* [27], who presented a lower bound of $\Omega(nk\alpha(k))$, and conjectured that the upper bound is close to $O(nk)$. Thus, when $k \ll n$, our bound is a significant improvement over the general bound mentioned above. In particular, we improve significantly the solution of the above motion planning problem, when the number $k_1 k_2$ of polyhedra $K_{i,j}$ is much smaller than the overall number of their facets.

We present a detailed proof for the *unbounded cell* of $\mathcal{A}(\mathcal{P})$, under the additional assumption that the polyhedra in $\mathcal{P}$ are bounded, and then argue that this implies the same asymptotic bound on the complexity of any other cell, and also extends to the case where the polyhedra may be unbounded. We apply a variant of the *charging scheme* of Halperin and Sharir [85], which relies on the randomized technique of Clarkson and Shor [53], and is based on the proof technique of [84] and [123]. The main difference between the proof in [85] and ours is that we treat each polyhedron in $\mathcal{P}$ as a *single surface*, and therefore a triple of our surfaces may intersect in a large non-constant number of points, which the analysis in [85] cannot handle. We thus have to modify this analysis and adapt it to our scenario. We also extend our analysis and show that the overall complexity of the zone in $\mathcal{A}(\mathcal{P})$ of a low-degree algebraic surface, or of the boundary of an arbitrary convex set in 3-space, is $O(nk^{1+\varepsilon})$ we well.

**Constructing a single cell.** We also design an efficient deterministic algorithm that constructs a single cell of $\mathcal{A}(\mathcal{P})$, in time $O(nk^{1+\varepsilon} \log^3 n)$, for any $\varepsilon > 0$. The algorithm adapts the general recursive approach of [92], but the implementation details of the “merge step” are quite different and more involved. In contrast, the $O(n^{2+\varepsilon})$-algorithm of [24] is randomized (ours is deterministic) and not $k$-sensitive.

The results of Chapter 5 appeared in [73].

## 1.6 The Union of Fat Tetrahedra in Three Dimensions

### Overview

Let $\mathcal{T}$ be a collection of $n$ (arbitrarily oriented) tetrahedra of arbitrary sizes in 3-space, and let $U$ denote their union. The problem that we study in Chapter 6 is to obtain a nearly-quadratic upper bound on the combinatorial complexity of $U$, in the special case where all the given tetrahedra are *fat*, meaning that the solid angles at their vertices are all at least some fixed constant $\alpha > 0$. (Recall that the complexity of $U$ is the number of faces of the arrangement $\mathcal{A}(\mathcal{T})$ that appear on $\partial U$.)

### Related work

As already discussed, the problem of determining the combinatorial complexity of the union of geometric objects has received considerable attention in the past twenty years, although most of the earlier work has concentrated on the planar case. See [10] for a recent comprehensive survey of the area.

The case involving *pseudo-discs* (that is, a collection of simply connected planar regions, where the boundaries of any two distinct objects intersect at most twice), arises for Minkowski sums of a fixed convex object with a set of pairwise disjoint convex objects (which is the problem one faces in translational motion planning of a convex robot, as reviewed in Section 1.1), and has been studied by Kedem *et al.* [93]. In this case, the union has only linear complexity. Matoušek *et al.* [106, 107] proved that the union of $n$ $\alpha$*-fat* triangles (where each of their angles is at least $\alpha$) in the plane has only $O(n)$ holes, and its combinatorial complexity is $O(n \log\log n)$. The constant of proportionality, which depends on the fatness factor $\alpha$, has later been improved by Pach and Tardos [116]. Extending the study to the realm of curved objects, Efrat and Sharir [65] studied the union of planar convex fat objects. Here we say that a planar convex object $c$ is $\alpha$*-fat*, for some fixed $\alpha > 1$, if there exist two concentric disks, $D \subseteq c \subseteq D'$, such that the ratio between the radii of $D'$ and $D$ is at most $\alpha$. In this case, the combinatorial complexity of the union of $n$ such objects, such that the boundaries of each pair of objects intersect in a constant number of points, is $O(n^{1+\varepsilon})$, for any $\varepsilon > 0$. See also Efrat and Katz [63] and Efrat [62] for related (and slightly sharper) nearly-linear bounds.

In three and higher dimensions, it was shown by Aronov *et al.* [27] (see also Section 1.5) that the complexity of the union of $k$ convex polyhedra with a total of $n$ facets in $\mathbb{R}^3$ is $O(k^3 + nk \log k)$, and it can be $\Omega(k^3 + nk\alpha(k))$ in the worst case. The bound was improved by Aronov and Sharir [25] to $O(nk \log k)$ (and $\Omega(nk\alpha(k))$) when the given polyhedra are Minkowski sums of a fixed convex polyhedron with $k$ pairwise-disjoint convex polyhedra. Boissonnat *et al.* [36] proved that the maximum complexity of the union of $n$ axis-parallel hypercubes in $\mathbb{R}^d$ is $\Theta\left(n^{\lceil d/2 \rceil}\right)$, and that the bound improves to $\Theta\left(n^{\lfloor d/2 \rfloor}\right)$ if all hypercubes have the same size. Pach *et al.* [112] showed that the combinatorial complexity of the union of $n$ nearly congruent arbitrarily oriented cubes in three dimensions is $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$ (see also [110] for a subcubic bound on the complexity of the union of fat wedges in 3-space). Agarwal and Sharir [13] have shown that the complexity of the union of $n$ congruent infinite cylinders is $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$. In fact, the more general problem studied in [13] involves the union of the Minkowski sums of $n$ pairwise disjoint triangles with a ball (where congruent infinite cylinders are obtained when the triangles become lines), and the nearly-quadratic bound is extended in [13] to this case as well. Finally, Aronov *et al.* [21] showed that the union complexity of $n$ $\kappa$*-round* objects in $\mathbb{R}^3$ is $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$, where an object $c$ is $\kappa$-round if for each $p \in \partial c$ there exists a ball $B \subset c$ that touches $p$ and its radius is at least $\kappa \cdot diam(c)$. The bound is $O(n^{3+\varepsilon})$, for any $\varepsilon > 0$, for $\kappa$-round objects in $\mathbb{R}^4$. Each of the nearly-quadratic bounds mentioned above (for the three-dimensional case) is almost tight in the worst case.

To recap, all of the above results indicate that the combinatorial complexity of the union of fat objects is roughly “one order of magnitude” smaller than the complexity of the entire arrangement that they induce. While considerable progress has been made on the analysis of unions in three dimensions, the case of the union of fat *polyhedra* has so far been lagging behind, where only very few nearly-quadratic bounds are known.

## Contributions

In Chapter 6 we derive a nearly-quadratic bound on the combinatorial complexity of the union of fat tetrahedra. Our bound, which is the first known subcubic bound for this general problem, is almost tight in the worst case.

Specifically, a tetrahedron is called $\alpha$*-fat* if each of its four solid angles (at its four respective vertices) is at least $\alpha$. We show that, for any fixed $\alpha > 0$, the complexity of the union of $n$ $\alpha$-fat tetrahedra is $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$, where the constant of proportionality depends on $\varepsilon$ and on $\alpha$. Our proof technique relies on the nearly-quadratic bound of the union of $\alpha$-fat dihedral wedges, established by Pach *et al.* [112]; a dihedral wedge is called $\alpha$-fat if its dihedral angle is at least $\alpha$. An immediate application of our result is a nearly-quadratic bound on the complexity of the union of arbitrarily oriented cubes of arbitrary sizes (a problem left open in [112]). Moreover, even for the special case of nearly congruent cubes, studied in [112], the second part of the analysis in that paper is not needed any more, since it is subsumed by our analysis, which does not rely on that part (and applies in a much wider context).

The analysis is based on cuttings, which incorporate the Dobkin-Kirkpatrick hierarchical decomposition scheme for convex polytopes [58], in order to partition space into subcells (simplices), so that, on average, the overwhelming majority of the tetrahedra intersecting a subcell $\Delta$ behave as $\alpha'$-fat *dihedral wedges* within $\Delta$, where $\alpha'$ is another constant that depends on $\alpha$. Since, as shown in [112] (and mentioned above), the complexity of the union of $\alpha'$-fat dihedral wedges is nearly-quadratic, it only remains to analyze the number of other types of vertices, namely, those incident to at least one ”real” tetrahedron in $\Delta$. This latter task is handled by the cutting-based divide-and-conquer mechanism (see Chapter 6 for details).

Our analysis can also be applied when the given objects are arbitrarily oriented *$\alpha$-fat* triangular prisms (that is, all the dihedral angles in each prism are at least $\alpha$) having cross-sections of arbitrary sizes. In this case, the complexity of the union is nearly-quadratic as well, and the bound is nearly worst-case tight. We are not aware of any previous known subcubic bound in this case, except for the nearly-quadratic bound of Aronov and Sharir [25], for the special case where the prisms are Minkowski sums of lines in 3-space with a fixed (not necessarily fat) polyhedron.

An immediate consequence of our results is a bound $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$, on the complexity of the union of any family of polyhedral objects, so that each of them is the union of some number of $\alpha$-fat tetrahedra (or triangular prisms), and the total number of these tetrahedra or prisms is $n$.

The problem that we study is a natural extension of the two-dimensional problem of bounding the complexity of the union of fat triangles [106, 107, 116]. We show that a simple specialization of our analysis to the two-dimensional case yields the bound $O(n^{1+\varepsilon})$, for any $\varepsilon > 0$, on this complexity. The analysis, based on the new approach, is almost immediate (albeit yielding a slightly suboptimal bound), and is significantly simpler than the analysis in [106, 107, 116].

By now, the arsenal of techniques for analyzing the complexity of the union of geometric objects in 3-space is quite rich: It includes, for example, the technique of Aronov *et al.* [21], for bounding the combinatorial complexity of the union of $n$ $\kappa$-round objects in $\mathbb{R}^3$, by reducing the problem to subproblems involving sandwich regions between upper and lower envelopes (see also [13]), and the technique of Pach *et al.* [112] for bounding the complexity of the union of $n$ nearly congruent cubes in 3-space by bounding the number of "special cubes" in the arrangement of these cubes (see also [27]). However, we were unable to extend any of these alternative techniques to our context, and had to develop new machinery. We believe it to be of independent interest, and hope that it will find additional applications to related problems.

The results of Chapter 6 appear in [74]. A full version of the paper has been recently submitted, and is available online.

## 1.7 Regular Vertices of the Union of Planar Convex Objects

### Overview

Let $\mathcal{C}$ be a collection of $n$ compact convex sets in the plane, such that the boundaries of any pair of sets in $\mathcal{C}$ intersect in at most $s$ points, for some constant $s$. Let $U$ denote the union of $\mathcal{C}$. If the boundaries of a pair of sets in $\mathcal{C}$ intersect exactly twice, we refer to their

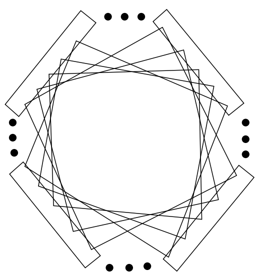


Figure 1.3: A construction with $R = \Theta(n^2)$ regular vertices on $\partial U$.

two intersection points as *regular* intersections; all other boundary intersections are called *irregular*. The problem at hand is to bound the number of regular vertices that appear on the boundary of $U$.

## Related work

Several recent papers have considered the problem of obtaining sharp bounds on the number of regular intersection points that can appear on the boundary of the union $U$. In the simplest instance of this problem, we assume that the boundaries of any pair of sets in $\mathcal{C}$ intersect at most twice, but make no other assumption on the shape of these sets; we then face the problem of bounding the complexity of the union of pseudo-discs (see also Section 1.6 for a short discussion about pseudo-discs in the context of translational motion planning). In an early paper [93], Kedem *et al.* show that in this case the boundary of the union contains at most $6n - 12$ intersection points, and this bound is tight in the worst case. Pach and Sharir [114] have shown that, for the special case where $\mathcal{C}$ consists of convex sets, one always has $R \leq 2I + 6n - 12$, where $R$ (resp., $I$) denotes the number of regular (resp., irregular) intersection points on $\partial U$, thus generalizing the result of Kedem *et al.*, in which $I = 0$.

The bound of Pach and Sharir is tight in the worst case, but since $I$ can be large, it does not provide a good "absolute" upper bound (a bound that depends only on $n$) on $R$. In fact, $I$ can be $\Omega(n^2)$ in the worst case, and there exist constructions in which both $I$ and $R$ are $\Theta(n^2)$ (see Figure 1.3). However, in these lower bound constructions, some pairs of boundaries of the sets in $\mathcal{C}$ intersect in an arbitrarily large number of points (that is, our assumption on $\mathcal{C}$ does not hold). It is therefore interesting to seek bounds on $R$ that are independent of $I$ and depend only on $n$, in cases where each pair of boundaries intersect in a constant number of points. This has been done by Aronov *et al.* [20]. Under similar assumptions on the input sets, they obtained the upper bound $R = O(n^{3/2+\varepsilon})$, for any $\varepsilon > 0$. For the more general case, where the sets in $\mathcal{C}$ are not necessarily convex, they show the existence of a positive constant $\delta$, which depends only on $s$, so that $R = O(n^{2-\delta})$.

## Contributions

In Chapter 7 we consider the case where $\mathcal{C}$ satisfies the above assumptions, and derive an improved bound on $R$. Specifically, we show that $R = O(n^{4/3+\varepsilon})$, for any $\varepsilon > 0$. This improves the first bound of [20]. Moreover, this bound is nearly tight in the worst case, since one can easily construct $n$ rectangles and disks which generate $\Theta(n^{4/3})$ regular vertices on the boundary of their union; see [114] for details.

The results of Chapter 7 appeared in [69]. A full version of the paper has been recently submitted, and is available online.

# Chapter 2

# Output-Sensitive Construction of the Union of Triangles

In this chapter 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 $\bigcup S = \bigcup T$. The running time of the algorithm 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 [40]. Their method is based on a randomized *natural selection* technique used by Clarkson [51, 52], Littlestone [100], and Welzl [128]. In our case, the objects are the triangles of $T$, and any point $v$ in the plane defines a set $T_v = \{\Delta \in T \mid v \in int(\Delta)\}$. The collection $\{T_v\}_{v \in \mathbb{R}^2}$ forms a set system for which a *hitting set* $H \subset T$ is a subset satisfying $\bigcup H = \bigcup T$, and thus a minimum-size hitting set is the object that we wish to compute. (It is well known, and easy to verify, that this set system has finite VC dimension; see below for details.) Note that this set system is the same as the one generated by sampling one point $v$ in the interior of each cell of the arrangement $\mathcal{A}(T)$. 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 (of positive depth) 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 2.2 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 in the plane and in $\mathbb{R}^d$. These extensions are discussed in Section 2.3. We give concluding remarks and suggestions for further research in section 2.4.

## 2.1 The Union Construction as a Set Cover Problem

### 2.1.1 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 [40]; for the sake of completeness, we provide a brief overview of this approach, in the context of the union construction problem. Recently, Even *et al.* [67] have proposed an alternative technique, based on a linear-programming formulation of the problem, that appears to be somewhat simpler and more efficient.

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). A hitting set for the set system induced by $\{T_v\}_{v \in V}$, where $T_v$ consists of all the triangles $\Delta \in T$ that contain $v$ in their interior, is a subset of triangles $H \subset T$ such that $\bigcup H$ covers all the vertices in $V$. It needs not necessarily cover $\bigcup T$ entirely, but the pieces left uncovered are easily computable, and will be computed in the final stages of the algorithm. Thus we consider the set system $(T, V^*)$, where

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

Since this set system is dual to $(V, T)$, which has some finite VC-dimension $d$ (see, e.g., [19]), it follows that the VC-dimension of $(T, V^*)$ is also finite; as a matter of fact, it does not exceed $2^{d+1}$ [28]. As already mentioned, our 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 $T_v \in V^*$, $v \in V$.

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 [40] for further details.) Since, by definition, the size of the optimal cover is assumed to be $\xi$, 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 $(T, V^*)$ [19]. 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 $\xi$ (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 [40], 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 $\xi$ 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 2.2. 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 $R$ (actually, it suffices that $H$ covers most of $R$ — see below), 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 $\bigcup T$.

### 2.1.2 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 2.1.1** *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 $\kappa$ denote the size of $V$, and suppose that there are only $\xi$ 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 prespecified 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*

$$\kappa_S \le \max\left\{\frac{\kappa}{t}, \beta\frac{\kappa}{r}r_S\right\}, \tag{2.1}$$

*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 2.2 and Appendix 2.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 (2.1) 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 [19, Theorem A.13], it follows that

$$Pr[A_S] < e^{-\frac{r_0{}^2}{2\frac{r}{\kappa}\kappa_S}}. \tag{2.2}$$

Putting $r_0 = \sqrt{2c_0\frac{r}{\kappa}\kappa_S \log n}$, for some constant $c_0 \ge 1$, (2.2) implies that the probability that the event $A_S$ occurs is at most $\frac{1}{n^{c_0}}$. Hence, w.o.p.,

$$r_S - \frac{r}{\kappa}\kappa_S \ge -\sqrt{2c_0\frac{r}{\kappa}\kappa_S \log n},$$

or

$$r_S \ge \sqrt{\frac{r}{\kappa}\kappa_S}\left[\sqrt{\frac{r}{\kappa}\kappa_S} - \sqrt{2c_0 \log n}\right].$$

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.,

$$\sqrt{\frac{r}{\kappa}\kappa_S} - \sqrt{2c_0 \log n} > \alpha\sqrt{\frac{r}{\kappa}\kappa_S}, \tag{2.3}$$

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

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

and thus the lemma follows. $\square$

**Remarks:** 1) Note that Lemma 2.1.1, 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 2.3.

2) We re-emphasize that Lemma 2.1.1 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 2.2 for further details.)

Lemma 2.1.1 implies that if the triangles in $S$ cover all but at most $\frac{r}{t}$ of the elements of $R$ (and thus $r_S = O\left(\frac{r}{t}\right)$), then, w.o.p., $\kappa_S \le \frac{\kappa}{t}$. 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 2.1.2** *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 $V$, $\kappa$ and $t$ be as in Lemma 2.1.1. 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 $\xi$ 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 2.2 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.

## 2.2 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 (most of the elements of) 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 $\xi$ and $\kappa$ (see below for details concerning these guesses). Let $\kappa^*$ 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 [109]. We may thus assume that $\kappa > \beta\kappa^*$, for some absolute constant $\beta > 1$. We also may assume that $\kappa > \beta \max\{\xi^2, n^{4/3}\}$, for the same constant. 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 $\frac{9c'r\binom{n}{2}}{\kappa}$ sampling steps, where in each step we choose, uniformly and independently, a pair of edges of distinct triangles in $T$, for an appropriate constant $c' > 1$ (we first select a random pair of triangles, and then randomly choose a pair of triangle edges out of the 9 triangle edge pairs induced by the two chosen triangles). Clearly, an intersection vertex of the arrangement $\mathcal{A}(T)$ (that is formed by a pair of intersecting triangle edges) 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

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

Applying the same deviation bound used in Lemma 2.1.1, it can be shown that, w.o.p., the actual number of such pairs satisfies

$$r' \geq \mathbf{E}(r') - \sqrt{\gamma \frac{\kappa}{9\binom{n}{2}} \frac{9c'r\binom{n}{2}}{\kappa} \log n} = \mathbf{E}(r') - \sqrt{\gamma c' r \log n},$$

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

Not all sampled vertices have positive depth. However, since $\kappa > \beta\kappa^*$, 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 (most of the elements of 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 covers all but at most $\frac{r}{t}$ elements of $R$. The analysis in [40] 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 in [40], the overall number of such iterations does not exceed $4\xi \log(n/\xi)$.

We start with the description of the net finder $\mathcal{F}$. We use a simple method, presented by Matoušek [102], and briefly reviewed in [40], 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 $\Delta$). 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 [102] that an $\varepsilon$-net for (the unweighted set) $T'$ is also an $\varepsilon$-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 [19], 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 happen with an overwhelmingly small probability for any $\xi \geq \log n$; see, e.g., [90]), 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. The fact that the process fails with an overwhelmingly small probability ensures that, w.o.p., the number of such trials is not larger than some constant. (When $\xi < \log n$, the number of trials that guarantees success, w.o.p., is at most $O(\log n)$. However, this does not affect the asymptotic running time of the algorithm; see Section 2.2 for the specific bound on the running time of the algorithm.)

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 $R_H$ contains at least $\frac{r}{t}$ vertices (otherwise, the first part of the algorithm terminates). Rather than just picking any $v \in R_H$, we sample a random vertex $v$ from $R_H$, 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, if $T_v \neq \emptyset$, double their weights. The reason for sampling is that $R$ may in general also contain zero-depth vertices, and $T_v$ will be empty for such vertices $v$. To accommodate this case, we use the sampling technique, and stop when we find a positive-depth vertex in $R_H$. Since (i) $|R_H| \geq \frac{r}{t} = \Omega(\log n)$, (ii) $\kappa > \beta\kappa^*$, and (iii) $R$ is sampled after $H$ has been constructed, it follows that a constant positive fraction of the elements of $R_H$ have positive depth, and that, w.o.p., such an element will be found after at most $O(\log n)$ samplings. 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, w.o.p.,

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

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 $\mathcal{A}(T)$.

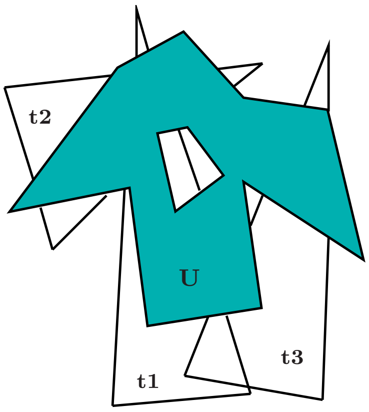


Figure 2.1: The second stage of the actual construction of the union. $U$ denotes the union of the $h$ triangles in the hitting set $H$, and $t_1, t_2$ and $t_3$ denote the remaining triangles to be inserted into the union. Only the portions of $t_1, t_2$ and $t_3$ that lie outside $U$ are relevant.

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 [109]). Next, we efficiently find the intersections of the boundary of each of the remaining triangles $\Delta$ 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 2.1 for an illustration.)

In order to find those portions efficiently, we use the algorithm of Bentley and Ottmann [34] 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 $\bigcup H$ [34]. 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 $\xi$ is an input parameter. Since $\xi$ is not known a priori, we run this procedure with the values $\xi = 1, 2, 4, \ldots, 2^i, \ldots$ (where $i < \log n$), thereby guaranteeing a constant approximation of the actual value of $\xi$. The choice of $r$ (that is, of the parameter $t$) in this procedure will be specified later.

```
Procedure ConstructUnion($T$, $\xi$)
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 all but at most $\frac{r}{t}$ vertices of $R$ 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 $c > 0$, and for the parameter $t$ that we still need to fix. Since the size $h$ 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 (including the exponential search of the actual value of $\xi$) 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 $\xi \le n$, 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 $\kappa$ 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 2.2.1** *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 $\xi = o\left(\frac{n}{\log^2 n}\right)$.*

## 2.3 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 [40] 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., [105]). 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 2.1.1 continues to hold in this case, since it does not make any assumptions on the input shapes. It thus follows that Theorem 2.1.2 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 2.3.1 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 2.2. 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 2.2. 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 [34], which can be applied for any set of Jordan arcs of constant description complexity, with the same asymptotic time bound, as stated in Section 2.2.

We can thus easily derive the following theorem:

**Theorem 2.3.1** *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 $\xi = o\left(\frac{n}{\log^2 n}\right)$.*

### 2.3.1 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 $\kappa > \beta\max\{\xi^3, n^2\}$, for some absolute constant $\beta > 1$. 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 $\kappa_B$ 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 [34]. 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, 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 2.2 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 2.2, 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 $\bigcup H$.

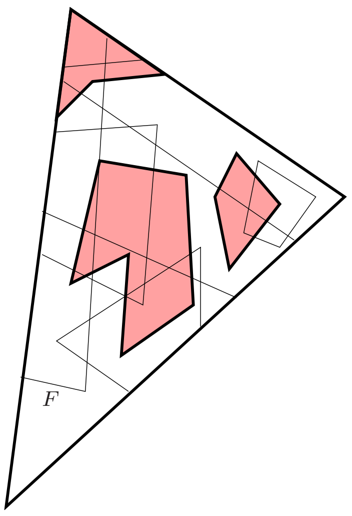


Figure 2.2: The case where the input bodies are simplices in three dimensions. The facet $F$ belongs to one of the $h$ simplices in $H$. The thick lines are the boundaries of $\bigcup H$ on $F$. The thin lines are the intersections of the $n-h$ remaining simplex boundaries with $F$. The intersections appearing in the shaded regions lie in the interior of the union of the $n$ simplices, and need not be computed explicitly.

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 2.2 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 boundaries $F$, is less than $\frac{\kappa}{t}$, the overall time needed to report all these intersections, over all these boundaries, 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 outside $\bigcup H$, 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 $E_F$. This constructs the entire 2-dimensional arrangements that these portions induce, from which the complete union boundary is easy to extract. 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 $\xi \le n$, 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 $\kappa$ 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 $\xi = o\left(\frac{n}{\log^2 n}\right)$.

Consider next the union problem in $d \ge 4$ dimensions. 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 $\xi$ 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 $\partial B$ 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 $\partial B$ is equal to the union of the intersections with the bodies of $\mathcal{S}$. In either case, with an appropriate parameterization of the boundaries, we obtain $n$ $(d-1)$-dimensional instances of the union construction problem, each with output size $\le \xi$, 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 2.3.2** *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 $\xi = o\left(\frac{n}{\log^2 n}\right)$.*

## 2.4 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 $\xi$ 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 [40] for finding an approximate 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 $\xi = o(\frac{n}{\log^2 n})$. Derandomization of our implementation seems nontrivial, and an open problem that thus arises is to make the worst-case running time of the algorithm subquadratic and deterministic. The algorithm does not have to know the value of $\xi$ in advance. Instead, it runs an exponential search on $\xi$, which approximates well the correct value of $\xi$, up to a constant factor. However, this approximation concerns only the size $h$ of the subset $H$ computed in the first stage of the algorithm, whereas the number of the remaining triangles whose union covers $\bigcup T \setminus \bigcup H$ may be much larger than $h$. An open problem raised by this study is to compute (in subquadratic time) a subset $H' \subset T$ such that $\bigcup H' = \bigcup T$ and $|H'|$ is within a constant (or even $O(\log n)$) factor off the optimum size $\xi$, or, alternatively, to show that this problem is 3*SUM-hard*.

In addition, the subset $H$ of triangles that the algorithm computes is not a hitting set for the weighted system $(T, V^*)$ but is rather a $(1/2\xi)$-net for that system. Thus another question that arises is whether the Brönnimann–Goodrich algorithm can be transformed to an algorithm that finds a small $\varepsilon$-net in a general setting (with finite VC-dimension), as it is believed that finding the smallest $\varepsilon$-net is NP-complete. This might lead to an approximation algorithm for finding minimum-size $\varepsilon$-nets.

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 $\xi$ 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 $\xi = o(\frac{n}{\log^2 n})$. 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 $O(n^{7/3} \log n + n^2 \xi \log^2 n)$ expected running time, which is worse than the bound that we have obtained when $\xi = o(\frac{n^{1/3}}{\log n})$. (Note, however, that the two-dimensional approach is used in the actual construction of the union, because a global approach, in this case, would yield an inefficient solution, since it involves the vertical decomposition of simply shaped bodies in three dimensions, which may become quadratic in the number of bodies in the worst case [55, 94, 126].)

A direction for further research is to determine whether there exist simpler efficient approaches to the union construction problem that we study. We note that the standard randomized incremental construction (RIC) of [109] may fail in the case that we have considered. 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 $\xi = o(n)$ 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 nearly-quadratic (see [13, 21, 74] for known instances of this kind). Our approach runs in *subcubic* time, when $\xi$ is small, but does not improve upon standard, output-insensitive techniques when the union complexity is always nearly-quadratic. One of the simplest instance of such a problem would be the following: Given a collection of $n$ balls in $\mathbb{R}^3$ whose union is equal to the union of some $\xi = o(n)$ of the balls, can the union be constructed in subquadratic time?

Finally, we note that in an earlier version of the algorithm [70], we used a different approach, based on a careful implementation of the DC algorithm of [68]. The previous approach is more complicated, yields a somewhat less efficient solution (which is subquadratic for only a smaller range of the values of the parameter $\xi$), and is more difficult to extend to other geometric shapes and to higher dimensions (in this previous approach, the implementation was based on the techniques of [8, 11, 16, 47, 72]). Our new approach, based on the technique of Brönnimann and Goodrich, is simpler and more generic, improves our previous result, and extends to other shapes and to higher dimensions.

## 2.A Appendix - The Actual Model for Sampling $R$

In this appendix we show that Lemma 2.1.1 continues to hold under the actual model of sampling $R$.

As described in Section 2.2, we draw the elements of $R$ by randomly making $\frac{9c'r\binom{n}{2}}{\kappa}$ 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 triangle edges from distinct triangles) is chosen with probability $\frac{1}{9\binom{n}{2}}$ (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

$$p = 1 - \left(1 - \frac{1}{9\binom{n}{2}}\right)^{9c'r\frac{\binom{n}{2}}{\kappa}}. \tag{2.4}$$

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

$$p > c'\frac{r}{\kappa} - \frac{(c'r)^2}{\kappa^2}. \tag{2.5}$$

In this model, the variables $X_1, \ldots, X_{\kappa_S}$ (as defined in Lemma 2.1.1) are no longer independent. Nevertheless, examining the proof of the deviation bound given in [19, 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

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

for any $\lambda$. Moreover, the analysis in [19] only uses the value $\lambda = \frac{r_0}{p\kappa_S}$, where $r_0$ is defined as in Lemma 2.1.1. An inspection of the derivation of these bounds in [19] 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 2.1.1 continues to hold when the weaker inequality

$$\mathbf{E}\left[e^{\sum_{i=1}^{\kappa_S} \lambda X_i}\right] \leq \gamma\Pi_{i=1}^{\kappa_S}\mathbf{E}\left[e^{\lambda X_i}\right] \tag{2.6}$$

holds, for some positive constant $\gamma$. This has the effect of multiplying the probability that (2.1) fails by $\gamma$, which implies that (2.1) still holds, w.o.p. Hence, it suffices to show that (2.6) holds for the above value of $\lambda$. More precisely, as in the original proof of Lemma 2.1.1, it suffices to establish this under the assumption that $\kappa_S > \frac{\kappa}{t}$.

In our model,

$$\Pi_{i=1}^{\kappa_S}\mathbf{E}\left[e^{\lambda X_i}\right] = \left(e^{\lambda}p + (1-p)\right)^{\kappa_S} = \left(1 + p(e^{\lambda} - 1)\right)^{\kappa_S}, \tag{2.7}$$

and

$$\mathbf{E}\left[e^{\sum_{i=1}^{\kappa_S}\lambda X_i}\right] = \sum_{m=0}^{r^*} Pr\left[r_S = m\right] e^{\lambda m}, \tag{2.8}$$

where $r^* = \min\left\{\frac{9c'r\binom{n}{2}}{\kappa}, \kappa_S\right\}$. (Note that $Pr\left[r_S = m\right] = 0$, for any $m > r^*$.)

In each of the $\frac{9c'r\binom{n}{2}}{\kappa}$ drawing trials, the probability that we have selected a vertex $v$, and that it is not covered by $S$, is $q = \frac{\kappa}{9\binom{n}{2}} \cdot \frac{\kappa_S}{\kappa} = \frac{\kappa_S}{9\binom{n}{2}}$. Since these trials are independent, we have

$$Pr\left[r_S = m\right] = \binom{r^*}{m} q^m (1-q)^{r^*-m}.$$

Hence the expression in (2.8) becomes

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

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

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

$$\left(1+\lambda_0 q\right)^{r^*} \le \gamma\left(1+\lambda_0 p\right)^{\kappa_S},$$

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

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

which implies the preceding inequality because $\left(1+\lambda_0 p\right)^{2c'r} = O(1)$. Indeed, $\left(1+\lambda_0 p\right)^{2c'r} < e^{2c'\lambda_0 pr}$. Using the fact that $e^{\lambda} \le 1 + 2\lambda$, for $0 \le \lambda \le 1$, and substituting $\lambda = \frac{r_0}{p\kappa_S}$ (which is indeed $\le 1$ when $r_0$ is chosen as in Lemma 2.1.1 and $c'$ is sufficiently large, as is easily verified), and $\lambda_0 = e^{\lambda} - 1$, we have $\lambda_0 \le 2\lambda$, and thus

$$e^{2c'\lambda_0 pr} \le e^{4c'\frac{rr_0}{\kappa_S}}.$$

Since we choose $r_0 = \sqrt{2c_0\frac{r}{\kappa}\kappa_S\log n}$ in Lemma 2.1.1, for some constant $c_0 \ge 1$, the latter expression is smaller than

$$e^{4c'\frac{r}{\kappa_S}\sqrt{2c_0\frac{r}{\kappa}\kappa_S\log n}} = e^{O\left(r\sqrt{\frac{r\log n}{\kappa\kappa_S}}\right)},$$

which, since we assume that $\kappa_S > \frac{\kappa}{t}$, is upper bounded by

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

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 $\kappa > \beta \max\{\xi^2, n^{4/3}\}$, for an absolute constant $\beta > 1$ (see Section 2.2), we have

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

It thus remains to show that

$$(1+\lambda_0 q)^{r^*} \le (1+\lambda_0 p)^{2c'r+\kappa_S}. \tag{2.9}$$

Let us first assume that $\frac{9c'r\binom{n}{2}}{\kappa} \le \kappa_S$. We thus need to show that

$$\left(1+\lambda_0 \frac{\kappa_S}{9\binom{n}{2}}\right)^{\frac{9c'r\binom{n}{2}}{\kappa}} \le (1+\lambda_0 p)^{2c'r+\kappa_S},$$

or that

$$\left(1+\lambda_0 \frac{\kappa_S}{9\binom{n}{2}}\right)^{\frac{9\binom{n}{2}}{\kappa_S}} \le (1+\lambda_0 p)^{\frac{\kappa(2c'r+\kappa_S)}{c'r\kappa_S}}.$$

Note that the function $(1+\lambda_0 x)^{\frac{1}{x}}$ is monotone decreasing, and since we have assumed that $\frac{9c'r\binom{n}{2}}{\kappa} \le \kappa_S$, it follows that $\frac{\kappa_S}{9\binom{n}{2}} \ge \frac{c'r}{\kappa} > p$. We thus have

$$\left(1+\lambda_0 \frac{\kappa_S}{9\binom{n}{2}}\right)^{\frac{9\binom{n}{2}}{\kappa_S}} \le (1+\lambda_0 p)^{\frac{1}{p}}.$$

It therefore suffices to show that $\frac{1}{p} \le \frac{\kappa(2c'r+\kappa_S)}{c'r\kappa_S}$. Using (2.5), $p > c'\frac{r}{\kappa} - \frac{(c'r)^2}{\kappa^2}$, and thus it suffices to show that $\frac{c'r}{\kappa}\left(1-\frac{c'r}{\kappa}\right) \ge \frac{c'r\kappa_S}{\kappa(2c'r+\kappa_S)}$, or that

$$1-\frac{c'r}{\kappa} \ge 1-\frac{2c'r}{2c'r+\kappa_S}, \tag{2.10}$$

or that

$$\kappa \ge c'r + \frac{\kappa_S}{2},$$

which clearly holds, since $\kappa_S \le \kappa$ and $r = o(\kappa)$.

We next assume that $\frac{9c'r\binom{n}{2}}{\kappa} > \kappa_S$. We thus need to show that

$$\left(1+\lambda_0 \frac{\kappa_S}{9\binom{n}{2}}\right) \le (1+\lambda_0 p)^{\frac{2c'r+\kappa_S}{\kappa_S}}.$$

Using the facts that $(1+\lambda_0 p)^{\frac{2c'r+\kappa_S}{\kappa_S}} \geq 1+\lambda_0 p\left(\frac{2c'r+\kappa_S}{\kappa_S}\right)$, $\frac{\kappa_S}{9\binom{n}{2}} < \frac{c'r}{\kappa}$, and (2.5), it is sufficient to show that

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

or that $\left(1-\frac{c'r}{\kappa}\right)\left(1+\frac{2c'r}{\kappa_S}\right) \geq 1$, which is identical to (2.10), as is easily checked, and thus follows by the preceding argument.

We note that (2.9) holds for any value of $\lambda_0 > 0$, and the assumption on $\lambda$ is used only when showing that $(1+\lambda_0 p)^{2c'r} = O(1)$. This completes the proof of (2.6) for the value of $\lambda$ that we use, and therefore shows that Lemma 2.1.1 also holds for the sampling model used by our algorithm.

# Chapter 3

# Counting and Representing Intersections Among Triangles in Three Dimensions

In this chapter we present an efficient algorithm that counts and represents in a compact form all intersecting triples among triangles in $\mathbb{R}^3$. In the next section we present a nearly quadratic algorithm that counts all intersecting triples among a collection of $n$ triangles in $\mathbb{R}^3$. In Section 3.2 we show how these intersections can be represented as the disjoint union of complete tripartite hypergraphs, with an overall storage (and construction time) that is nearly quadratic in the size of the input. In Section 3.3 we show that the triangle intersection counting problem belongs to the 3SUM-hard family. In Section 3.4 we extend our algorithm to count intersecting triples among a collection of planar objects of constant description complexity that lie in distinct planes in $\mathbb{R}^3$. We give concluding remarks and suggestions for further research in Section 3.5.

## 3.1 Counting Intersecting Triples Among Triangles in $\mathbb{R}^3$

Given a collection $T$ of $n$ triangles in $\mathbb{R}^3$, we present an algorithm that efficiently counts all intersecting triples among the triangles in $T$. For simplicity of presentation, we assume that the triangles in $T$ are in general position.

#### Preliminary alternative stage

If the number $\kappa$ of pairs of intersecting triangles is significantly smaller than $n^{5/3}$ then the problem can be solved in subquadratic time as follows. We first report all $\kappa$ intersection segments of pairs of triangles in $O(n^{8/5+\varepsilon}+\kappa)$ time, for any $\varepsilon > 0$, using segment intersection queries in 3-space [8], where we preprocess the input triangles, and query with the triangle edges. These queries produce the set of all intersection points between the triangle edges and other triangles. The intersection segments between the triangles are delimited by these points, and can then be easily reconstructed. We count, on each triangle $t$, the number of intersections among the $\kappa_t$ intersection segments that lie on $t$ (where we have $\sum_t \kappa_t = 2\kappa$). Using the algorithm of [1, 102], this takes $O(\kappa_t^{4/3} \log n)$ time, for each triangle $t$, for a total of $O\left(\sum_t \kappa_t^{4/3} \log n\right) = O(\kappa n^{1/3} \log n)$ time. Thus the overall running time of this algorithm is $O(n^{8/5+\varepsilon} + \kappa n^{1/3} \log n)$, for any $\varepsilon > 0$, which is subquadratic when $\kappa = o(n^{5/3}/\log n)$.

To detect such favorable situations, we first run this algorithm, as a preliminary stage. If the running time of this step becomes quadratic, we abandon it, and run the main algorithm, presented in detail below.

### 3.1.1 Ingredients of the algorithm

#### Curve-sensitive cuttings

We use a recent result of Koltun and Sharir [95] on the existence of "curve-sensitive" cuttings. In our context, it implies the following result (see also Section 1.1.4). For any $r \leq n$ there exists a $(1/r)$-cutting $\Xi$ for $T$ of size $O(r^{3+\varepsilon})$, for any $\varepsilon > 0$, which is a partition of $\mathbb{R}^3$ into $O(r^{3+\varepsilon})$ simplices, such that every simplex (also referred to as a *cell* of $\Xi$) is crossed by at most $\frac{n}{r}$ triangles of $T$, with the additional property that the number of crossings between the edges of the triangles and the cells of $\Xi$ is $O(n^{1+\varepsilon} r)$. The time needed to construct such a cutting, when $r$ is at most $O(n^{\varepsilon})$, is $O(n^{1+\varepsilon'})$, for any $\varepsilon' > 0$ that is sufficiently larger than $\varepsilon$.

We note that, for the case of triangles, one can obtain such a cutting using a simpler construction than that in [95]. Specifically, the cutting is constructed from a random sample $R$ of $O(r \log r)$ of the planes containing the triangles of $T$. We form the arrangement $\mathcal{A}(R)$ of $R$ and triangulate each of its cells using the Dobkin-Kirkpatrick hierarchical decomposition of convex polytopes [58], which has the property that a line that crosses a cell $C$ crosses only $O(\log r)$ of its simplices (see also Chapter 6 for further details). The random sampling theory of [49, 53, 90] implies that, with high probability, the resulting decomposition is indeed a $(1/r)$-cutting for $T$.[1] Since a line (or, rather, an edge of a triangle) crosses at most $O(r \log r)$ cells of $\mathcal{A}(R)$ (it has to cross a plane of $R$ to move from one cell to another), it crosses at most $O(r \log^2 r)$ simplices, so the total number of edge-cell crossings is $O(nr \log^2 r)$. A cutting of this kind, together with the distribution of the input triangles among its simplices, can be constructed in time $O(nr^2 \log^2 r)$: We construct $\mathcal{A}(R)$ and

[1] Here, and in several subsequent steps of our algorithms, we construct cuttings based on a random sample of the input objects. For simplicity of presentation, we bypass issues such as verifying that our sample is indeed a good sample, or reducing the size of the sample by the 2-stage sampling method of [47], or replacing the randomized approach by a deterministic one. All these issues are fairly standard by now, and can be applied in our settings as well.

hierarchically decompose each of its cells into simplices, in a total of $O(r^3 \log^3 r)$ time [58]. We next compute, for every resulting simplex $\Delta$, the subset $T^\Delta$ of the triangles of $T$ that cross $\Delta$, in overall time $O(nr^2 \log^2 r)$ [47]. To do so, we trace, for each triangle $t$ of $T$, all the cells of $\mathcal{A}(R)$ that $t$ crosses. By the Zone Theorem (see [124]), the overall complexity of these cells is $O(r^2 \log^2 r)$, which also bounds (i) the number of simplices into which these cells are decomposed, and (ii) the cost of tracing all these cells (see [124] for similar algorithms). The resulting procedure has the asserted time bound.

However, for more general planar objects that we will consider in Section 3.4, this simpler approach does not work, and the more general curve-sensitive cutting of [95] is needed.

### The recursive decomposition–an overview

We construct an "edge-sensitive" $(1/r)$-cutting $\Xi$, as described above, with a value of $r$ that will be specified later, and count the intersecting triples in each cell of $\Xi$ separately. Fix a cell $\Delta$ of $\Xi$. We classify each triangle $t \in T$ that intersects $\Delta$ as being either *long* in $\Delta$, if $\partial t \cap \Delta = \emptyset$, or *short*, otherwise. Each intersecting triple in $\Delta$ is consequently classified as **LLL**, if all three triangles that form the intersection are long in $\Delta$, **LLS**, if two of these triangles are long and one is short, **LSS**, if one of these triangles is long and two are short, or **SSS**, if all three triangles are short.

In what follows we assume that each triangle (long or short) that crosses $\Delta$ is clipped to within $\Delta$. In particular, for any long triangle $t$, $t \cap \Delta$ is a triangle or a quadrilateral. For short triangles, $t \cap \Delta$ is at most a 7-gon: Since $\Delta$ is a simplex, the plane containing $t$ intersects $\Delta$ in at most a quadrilateral, and the edges of $t$ contribute at most three additional edges to the cross-section.

We count the number of intersecting triples within each cell $\Delta_0$ by further partitioning $\Delta_0$ into smaller subcells $\Delta$, and recursively derive from each such subcell new subproblems. We partition $\Delta_0$ using the same kind of sensitive $(1/r)$-cutting, for the same $r$, with respect to the set of long and short triangles in $\Delta_0$, and the set of edges bounding the short triangles in $\Delta_0$ (and crossing $\Delta_0$). Initially, $\Delta_0$ is the entire three-dimensional space, and all triangles are short in $\Delta_0$, but they may become long in further recursive steps.

Let us denote by $N_S = N_S^{\Delta_0}$ the overall number of short input triangles (within a cell $\Delta_0$) and by $N_L = N_L^{\Delta_0}$ the overall number of long input triangles (within $\Delta_0$). During each step of the recursion, after partitioning $\Delta_0$ into smaller subcells $\Delta$, we immediately dispose of any new LLL and LLS intersections within each subcell $\Delta$, using two respective simple algorithms that run in time $O\left((N_L^\Delta)^2 \log N_L^\Delta\right)$ and $O(N_S^\Delta N_L^\Delta \log N_L^\Delta)$, respectively. These intersections are not recounted during any further recursive substep. At the bottom of the recursion (when $N_S < \max\left\{\sqrt{N_L}, c\right\}$, for some constant $c \geq 3$), we use two additional simple algorithms that count intersecting triples of types LSS and SSS, which run in time $O({N_S}^3 + N_S N_L \log N_S)$. We note that the goal of the recursive step is only to count

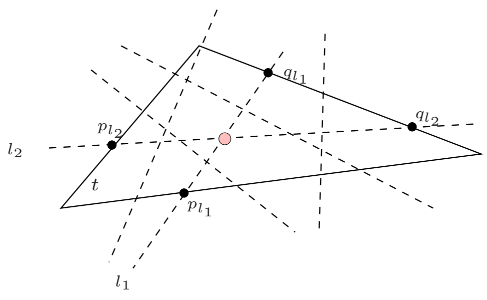


Figure 3.1: The long triangles that intersect the triangle $t$, drawn as lines crossing $t$. Two lines $l_1$ and $l_2$ intersect within $t$ if and only if their intersection points $p_{l_1}$, $p_{l_2}$, $q_{l_1}$, $q_{l_2}$ with $\partial t$ interleave along $\partial t$.

efficiently intersecting triples of type LSS and SSS; the (new) intersecting triples of types LLL and LLS are counted before entering the recursive step. (Of course, each recursive step may generate its own LLL and LLS intersections, involving triangles that were short in the input but became long in some of its subproblems. In other words, some of the LSS and SSS intersections within a cell $\Delta$ may be counted as LLL or LLS intersections in various recursive subproblems within $\Delta$.)

We first describe these four simple intersection counting algorithms, and then present in detail the complete recursive algorithm, which uses these simple algorithms as subroutines.

## Counting intersections of type LLL

Let $\Delta$ be a simplex cell of (some recursive cutting) $\Xi$ and let $L^\Delta$ denote the set of clipped long triangles in $\Delta$. Let $N_L = N_L^\Delta = |L^\Delta|$ denote, as above, the total number of long triangles in $\Delta$. We apply the planar algorithm of Agarwal [3] to each clipped triangle $t \in L^\Delta$. That is, we intersect $t$ with all the other (clipped) triangles in $L^\Delta$, and count all intersecting pairs within $t$. Since the boundary of every triangle $t' \in L^\Delta$ lies outside $\Delta$, $t'$ must cross $t$ in a line segment, both of whose endpoints lie on $\partial t$; see Figure 3.1. As shown in [3], this problem can be solved in time

$$O\left(\left|L^\Delta\right| \log \left|L^\Delta\right|\right) = O(N_L \log N_L),$$

by sorting the intersection points of these lines with $\partial t$ along $\partial t$ in a clockwise direction, say, and by counting all pairs whose intersection points appear along $\partial t$ in an *interleaved order*, as illustrated in Figure 3.1. It follows that the overall running time needed for counting all LLL intersections over all the clipped long triangles within $\Delta$ is $O(N_L^2 \log N_L)$. (If we follow this approach verbatim, we need to divide the final count by 3, since each intersection is counted three times; see below for our modified approach.)

Note that once a triangle has become long in a cell $\Delta$, it will remain long in all recursive steps involving subcells of $\Delta$. Since we need to ensure that each LLL intersection is counted only once, we count only intersections that involve at least one *new* long triangle (a triangle that is short in the parent cell of $\Delta$ but long in $\Delta$). To do so, we take only new long triangles as the base triangles $t$, within which the planar counting algorithm is applied. Moreover, we enumerate the new long triangles as $t_1, \ldots, t_{N_L}$, and apply the algorithm, within each $t_i$, only to the new long triangles $t_j$, for $j > i$, and to all the old long triangles. With these modifications, the running time of the algorithm just presented is $O(N_L N_L^0 \log N_L)$, where $N_L^0$ is the number of new long triangles (which are also included among all $N_L$ long triangles). Note that, with all these modifications, each intersection point is now counted only once.

### Counting intersections of type LLS

We use a similar approach as in the LLL case. Let $N_S = N_S^\Delta$ denote the number of short triangles in $\Delta$. We apply the preceding two-dimensional scheme within each short triangle. That is, we intersect each short triangle with all the long triangles, obtaining $O(N_L)$ lines on each such (clipped) triangle. Then we count all intersecting pairs within each short triangle, using the preceding algorithm. The overall running time is thus $O(N_S N_L \log N_L)$. Here too we need to ensure that no intersection is recounted in further recursive substeps. This is done as follows: On each short triangle in $\Delta$, we solve a *bichromatic* version of the problem, which counts all intersections between the new long triangles and the old long triangles. The algorithm for solving this problem is similar to the preceding one, and runs in time $O(N_L \log N_L)$; see [3] for further details. Then we count the intersections involving only new long triangles, using the two-dimensional procedure described in the LLL case. It thus follows that the running time of the modified algorithm remains $O(N_S N_L \log N_L)$.

### Counting intersections of type LSS

Let $S^\Delta$ denote the set of (clipped) short triangles in $\Delta$, so $N_S = N_S^\Delta = |S^\Delta|$. Intersect each (clipped) triangle $t \in S^\Delta$ with all the other triangles of $S^\Delta$ and $L^\Delta$. We thus face the problem of counting intersecting pairs involving a long segment (whose endpoints lie on the boundary of $t$) and a short segment, within every triangle $t \in S^\Delta$. Note that each such problem has an input of $O(N_S)$ short segments and $O(N_L)$ long segments. Since the short segments are confined to within $t$, we may replace the long segments by their containing lines, without affecting the set of intersecting pairs. The problem can then be solved in $O({N_S}^2 + N_L \log N_S)$ time, using an approach presented in [3], in which we construct the arrangement of the lines dual to the endpoints of the primal segments (representing short triangles), and then locate in this arrangement all points that are dual to the primal lines (representing long triangles). Since each face of the arrangement consists of points dual to lines that cross a fixed set of segments, this easily yields the count of the intersections between the (primal) segments and the (primal) lines.

To make sure that each intersection is counted only once, we enumerate the short triangles as $t_1, \ldots, t_{N_S}$, and make each triangle $t_i$ process only short segments that are formed by its intersections with triangles $t_j$ with $j > i$. Thus, the overall running time of the algorithm, for a fixed cell $\Delta$, is $O\left({N_S}^3 + N_S N_L \log N_S\right)$.

### Counting intersections of type SSS

We count all intersecting triples of type SSS using a brute-force algorithm which examines all triples in time $O(N_S^3)$. Note that this bound is subsumed by the bound on the time needed to count LSS intersections.

## The overall recursive algorithm

Each step of the algorithm involves a simplex $\Delta_0$, which, in the initial step, is the entire 3-space, and in further recursive steps is a cell of a cutting of some larger simplex. The algorithm receives as input a set of $N_S$ short triangles and a set of $N_L$ long triangles clipped to within $\Delta_0$.

If $N_S \le \max\left\{\sqrt{N_L}, c\right\}$, for some constant $c \ge 3$, we stop the recursion and compute the number of LSS and SSS intersections, using the explicit algorithms described above. Note that, in this case, there is no need to count intersecting triples of type LLL and LLS, since all intersecting triples of these types have already been counted in the preceding step that has processed the parent cell of $\Delta_0$.

If $N_S > \max\left\{\sqrt{N_L}, c\right\}$, we first compute a $(1/r)$-cutting $\Xi$ of the arrangement of all long and short triangles within $\Delta_0$, which is also sensitive to the edges of the short triangles, in the sense defined above. For technical reasons, we need to construct a sensitive $(1/r)$-cutting of $\Delta_0$ that has the property that each subcell $\Delta$ of $\Delta_0$ is crossed by at most $N_S^{\Delta_0}/r$ short triangles in $\Delta_0$ and by at most $N_L^{\Delta_0}/r$ long triangles in $\Delta_0$. This problem can be solved by sampling two subsets of $O(r \log r)$ triangles each, one from the long triangles in $\Delta_0$ and one from the short ones. The standard $\varepsilon$-net theory [90] implies that the resulting cutting has the desired property with high probability.

Next we count all LLL and LLS intersections within each subcell $\Delta$ of $\Delta_0$, that involve at least one new long triangle (a triangle that is short in $\Delta_0$ but long in $\Delta$), using the algorithms described above. We then continue to solve the problem recursively in every cell $\Delta \in \Xi$, applying the analysis presented in detail below.

Since there are only $O(N_S^{1+\varepsilon} r)$ crossings between short triangles and the cells of $\Xi$,[2] it follows that, for any $r^2 \le s \le r^{3+\varepsilon}$, the number of cells in $\Xi$ that are crossed by at least $\frac{N_S^{1+\varepsilon} r}{s}$ short triangles is at most $O(s)$. (The case $s < r^2$ cannot arise, since each cell of $\Xi$ is intersected by at most $\frac{N_S}{r}$ short triangles of $T$, due to the sampling that we have used.)

[2]We use here the more general and slightly weaker bound of [95] for the number of edge-cell crossings, rather than the slightly improved bound that can be obtained from the Dobkin-Kirkpatrick hierarchical decomposition. This does not affect the asymptotic running time bound, and allows us to extend the analysis essentially verbatim to the case of general planar objects, which is undertaken in Section 3.4.

We partition the set of all cells in $\Xi$ into at most $\log\left(\frac{M}{r^2}\right)$ subsets, where $M$ is the overall number of cells in $\Xi$ (note that $\log\left(\frac{M}{r^2}\right) = O(\log r)$), so that the $i$-th subset $\Xi_i$ contains $O(2^i r^2)$ cells of $\Xi$, each of which satisfies (recall that $S^\Delta$ denotes the set of short triangles in $\Delta$)

$$\frac{N_S^{1+\varepsilon}}{2^i r} \le |S^\Delta| \le \frac{2N_S^{1+\varepsilon}}{2^i r},$$

for $i = 0, \ldots, \log\left(\frac{M}{r^2}\right)$. Note that $|S^\Delta| = O\left(\frac{N_S^{1+\varepsilon}}{r^{2+\varepsilon}}\right)$ for each of the $O(r^{3+\varepsilon})$ cells $\Delta$ in the last subset.

We now create recursive subproblems, one in each cell $\Delta \in \Xi_i$, over all $i = 0, \ldots, \log\left(\frac{M}{r^2}\right)$, where the number of short triangles in $\Delta$ is $O\left(\frac{N_S^{1+\varepsilon}}{2^i r}\right)$, and the number of long triangles in $\Delta$ is at most $\frac{N_S+N_L}{r}$, because of the cutting property, and because some of the short triangles in the parent cell $\Delta_0$ may have become long in $\Delta$.

We estimate the cost of computing the LLL and LLS intersections within each cell $\Delta$ of $\Xi$ in the following crude manner. The number of new long triangles in $\Delta$ is at most $\frac{N_S}{r}$, the overall number of long triangles in $\Delta$ is at most $\frac{N_S+N_L}{r}$, and the number of short triangles in $\Delta$ is at most $\frac{N_S}{r}$. Hence the cost of computing the LLL and LLS intersections within $\Delta$ is $O\left(\frac{N_S(N_S+N_L)\log(N_S+N_L)}{r^2}\right)$. Summing over all cells $\Delta$, the overall running time is

$$O\left(\frac{r^{3+\varepsilon}N_S(N_S+N_L)\log(N_S+N_L)}{r^2}\right) = O\left(r^{1+\varepsilon}N_S(N_S+N_L)\log(N_S+N_L)\right).$$

Let $F(N_S, N_L)$ denote the maximum time needed to count all intersecting triples at a recursive step involving $N_S$ short triangles and $N_L$ long triangles. Then $F$ satisfies the following recurrence:

$$F(N_S,N_L) \le \begin{cases} O\left(r^{1+\varepsilon}N_S(N_S+N_L)\log(N_S+N_L) + (N_S+N_L)^{1+\varepsilon'}\right) & \\ + \sum_{i=0}^{\log\left(\frac{M}{r^2}\right)} O(2^i r^2) F\left(\frac{2{N_S}^{1+\varepsilon}}{2^i r}, \frac{N_S+N_L}{r}\right), & \text{if } N_S > \max\left\{\sqrt{N_L}, c\right\} \\ \\ O(N_S^3 + N_S N_L \log N_S), & \text{if } N_S \le \max\left\{\sqrt{N_L}, c\right\}, \end{cases}$$

where $c \ge 3$ is constant. The first term is the time needed to count the LLL and LLS intersections in the current recursive step, and the term $O\left((N_S+N_L)^{1+\varepsilon'}\right)$ is the time needed to construct the curve-sensitive cutting, for any $\varepsilon' > 0$, whose choice is correlated with the choice of $r$.

To solve the recurrence, for a given $\varepsilon > 0$, we substitute $r = (N_S+N_L)^{c'\varepsilon''}$, for an appropriate constant $c' > 0$ and for $\varepsilon'' \ll \varepsilon$. It is then easy to see, using induction on $N_S$ and $N_L$, that the solution is

$$F(N_S,N_L) = O(N_S(N_S+N_L)^{1+\varepsilon}), \text{ for any } \varepsilon > 0, \tag{3.1}$$

with a constant of proportionality that depends on $\varepsilon$. (Note that in the case $N_S \le \sqrt{N_L}$, the term $O(N_S^3)$, that appears in the bound for the cost of counting all intersecting triples of types LSS and SSS, is dominated by the term $O(N_S N_L \log N_S)$, so (3.1) does hold in this case too.)

The algorithm begins with $\Delta_0$ equal to the entire three-dimensional space, and $N_S = n$, $N_L = 0$. Note that at this point there are only intersecting triples of type SSS, so the preceding algorithm will count all of them. (As already remarked, the recursive process will generate the other types of intersections as the space is progressively cut up into subcells.)

In summary, we have shown:

**Theorem 3.1.1** *The number of intersecting triples in a set of $n$ triangles in $\mathbb{R}^3$ can be counted in time*

$$\min\left\{O(n^{8/5+\varepsilon} + \kappa n^{1/3}\log n), O(n^{2+\varepsilon})\right\},$$

*for any $\varepsilon > 0$, where $\kappa$ is the number of pairs of intersecting triangles.*

**Remark.** We note that by slightly modifying this algorithm, we can solve the following trichromatic variant of the problem in nearly quadratic time:

**Theorem 3.1.2** *Given three sets, $T_r$ of $n_r$ "red" triangles, $T_b$ of $n_b$ "blue" triangles, and $T_g$ of $n_g$ "green" triangles, all in $\mathbb{R}^3$, one can count the number of triples in $T_r \times T_b \times T_g$ with nonempty intersection, in time*

$$\min\left\{O(N^{8/5+\varepsilon} + \kappa N^{1/3}\log N), O(N^{2+\varepsilon})\right\},$$

*for any $\varepsilon > 0$, where $N = n_r + n_b + n_g$, and $\kappa$ is the number of bichromatic pairs of intersecting triangles.*

## 3.2 Compact Representation of All Intersecting Triples

Given a collection $T = \{t_1, \ldots, t_n\}$ of $n$ triangles in $\mathbb{R}^3$, we represent the set of all intersecting triples among the triangles of $T$ as a 3-*uniform hypergraph* [19] $H = (T, E)$, where

$$E = \{\{t_i, t_j, t_k\} \quad | \quad 1 \le i < j < k \le n \text{ and } t_i \cap t_j \cap t_k \ne \emptyset\}.$$

The size of the above representation is $\Theta(n^3)$ in the worst case. Our goal is to provide a compact representation for $H$ of nearly quadratic size, that stores the intersection triples only implicitly. As noted in the introduction, one application of such a compact representation is for sampling a random element out of the set of all intersecting triples in $T$, without having to list all these intersections explicitly. We discuss this and another application, already mentioned in the introduction, at the end of this section.

The compact representation for $H$ that we seek (defined analogously to that in [17]) is a collection $\mathcal{H} = \{H_i = (T_i, E_i)\}_{i=1}^{s}$, of $s$ subhypergraphs of $H$, such that

1. Each $H_i$ is a complete tripartite hypergraph, that is, the set $T_i$ of its vertices can be partitioned into three disjoint subsets $A_i$, $B_i$ and $C_i$, such that $E_i = A_i \times B_i \times C_i$, and $E_i \subseteq E$.
2. $E = \bigcup_{i=1}^{s} E_i$.
3. $E_i \bigcap E_j = \emptyset$, for $i \neq j$.

Clearly, the storage needed for such a compact representation is $\sum_{i=1}^{s} |T_i| = \sum_{i=1}^{s}(|A_i| + |B_i| + |C_i|)$, since the edges of $H$ are now defined implicitly. We show that the algorithm described in Section 3.1 can be modified to produce such a compact representation of $H$, with $\sum_{i=1}^{s} |T_i| = O(n^{2+\varepsilon})$, for any $\varepsilon > 0$, and that the running time also remains $O(n^{2+\varepsilon})$. (We remark that our compact representation is somewhat degenerate, in the sense that in each output subhypergraph $H_i$, one of the three sets of vertices is a singleton.)

As in the preceding section, we first report, as a preliminary stage, all $\kappa$ intersection segments of pairs of triangles in $O(n^{8/5+\varepsilon} + \kappa)$ time, for any $\varepsilon > 0$. Then we run the algorithm of Agarwal [3], in a manner similar to that described in Section 3.1, but slightly modified, so that it produces all intersecting triples as the disjoint union of complete tripartite hypergraphs composed of precomputed canonical subsets of triangles, so that the total size of all these subsets (as well as the running time needed to construct them) is $O(\kappa n^{1/3+\varepsilon})$, for any $\varepsilon > 0$. (With some care, we can ensure that no intersection point is implicitly constructed more than once — the simple modifications applied in the main algorithm can be used here as well.) Hence, when the number $\kappa$ of intersection segments is significantly smaller than $n^{5/3}$, this method will yield a compact representation of subquadratic size and the construction time will also be subquadratic. As above, we abandon this alternative computation when its output size (as well as its running time) becomes more than quadratic, and resort to the main algorithm.

The main algorithm follows the same recursive mechanism as in the preceding section, but the four simple counting algorithms (that the recursive algorithm uses as subroutines) are now modified, so that they construct a compact representation for all relevant intersecting triples (instead of counting them). We first describe these simple algorithms.

### Representing intersections of types LLL and LLS

We describe the compact representation of intersecting triples of type LLS; the intersecting triples of type LLL are handled similarly. Fix a cell $\Delta$. For each (clipped) short triangle $t$ in $\Delta$, let $L_t$ denote the set of lines obtained by intersecting $t$ with all the long triangles in $\Delta$. (Recall that the actual algorithm is slightly modified, to account only for intersections that involve new long triangles; for simplicity of presentation, we ignore this issue in what follows.) It is sufficient to obtain a separate compact representation of all intersecting pairs of lines in $L_t$, for each short triangle $t$ in $\Delta$, a task that proceeds as follows. Sort the intersection points of the lines in $L_t$ with $\partial t$, and turn the resulting circular sequence into a linear sequence $\sigma_t$ by breaking it at some arbitrary point. For each original triangle $t_i$, for $i = 1, \ldots, N_L = N_L^\Delta$, that intersects $t$ in a line $l^i$, we write $l_1^i$ (resp., $l_2^i$) for its first (resp., second) intersection point in $\sigma_t$. We want to represent compactly all pairs $\{t_i, t_j\}$, $i \neq j$, that satisfy $l_1^i < l_1^j < l_2^i < l_2^j$ (in the order within $\sigma_t$).

This is an instance of standard 2-dimensional orthogonal range searching. For the sake of completeness, we spell out the details. We use a 2-level tree-like structure. The first-level structure $T_1$ stores the points $l_1^i$ in sorted order. For each node (subtree) $v$ of $T_1$, we construct a secondary structure $T_2(v)$ that stores all the points $l_2^i$ whose matching points $l_1^i$ are stored at $v$. We now query with each triangle $t_j$. We first search with $l_1^j$ in $T_1$, and find all elements that (strictly) precede $l_1^j$, represented as the disjoint union of $O(\log N_L)$ subtrees. For each subtree (rooted at some node) $v$, we go to $T_2(v)$ and search there for all elements that lie (strictly) between $l_1^j$ and $l_2^j$, again, obtaining them as a collection of $O(\log N_L)$ subtrees. Altogether, $t_j$ "lands" in $O(\log^2 N_L)$ subtrees of the secondary structures. For each such subtree $\tau$, we collect the set $A_\tau$ of all triangles of $T$ in $\Delta$ that reach it as queries, and output the complete bipartite graph $A_\tau \times B_\tau$, where $B_\tau$ is the set of triangles that are stored at $\tau$ (more precisely, those triangles $t_i$ whose second intersection points $l_2^i$ are stored there).

The overall size of the vertex sets of the output graphs is $O(N_L \log^2 N_L)$. Indeed, the overall size of all the secondary trees is $O(N_L \log N_L)$, and the overall size of all subtrees of a secondary tree $\tau$ with $k$ vertices is $O(k \log k)$, which implies that $\sum_\tau |B_\tau| = O(N_L \log^2 N_L)$. Similarly, since each triangle reaches as a query $O(\log^2 N_L)$ subtrees, we also have $\sum_\tau |A_\tau| = O(N_L \log^2 N_L)$. We now add, for each subtree $\tau$, the tripartite hypergraph $\{t\} \times A_\tau \times B_\tau$ to the output representation. Hence, the overall size of the sets of the compact representation, over all short triangles $t$ in $\Delta$, is $O(N_S^\Delta N_L^\Delta \log^2 N_L^\Delta)$. The time for constructing such a representation has the same upper bound. Note that the output consists of *edge-disjoint* complete tripartite 3-uniform hypergraphs, due to the fact that each triple intersection is represented exactly once—see below, with one of the three vertex sets in each hypergraph being a singleton.

We need to ensure that each triple intersection is represented only once. Proceeding as in Section 3.1, the algorithm can be modified so that it represents only bichromatic intersections between the new long triangles and all the long triangles. This can be done by constructing the two-level tree-like structure for all the long triangles as above, but querying only with the points obtained by the new long triangles.

Handling LLL intersections is done similarly. We ensure that each triple intersection is represented only once, using similar arguments to those described in the LLL counting algorithm of Section 3.1. The size of the resulting representation is $O(N_L N_L^0 \log^2 N_L)$, where $N_L^0$ is defined as in Section 3.1, and it can be constructed in $O(N_L N_L^0 \log^2 N_L)$ time.

### Representing intersections of type LSS

Here too we adapt the corresponding algorithm of the preceding section, so that it represents (rather than counts) all intersecting pairs between the $O(N_S)$ short segments and the $O(N_L)$ long segments within every short triangle $t$ in $\Delta$. As in the LSS counting algorithm, to make sure that each intersection is represented only once, we enumerate the short triangles as $t_1, \ldots, t_{N_S}$, and make each triangle $t_i$ process only short segments that are formed by its intersections with triangles $t_j$ with $j > i$. By repeating the following procedure for all short triangles, we obtain a compact representation of all LSS intersections.

Fix a short triangle $t$, denote by $S_t$ the set of short segments within $t$, obtained by intersecting $t$ with all the short triangles in $\Delta$ (that succeed $t$ in the above enumeration), and by $L_t$ the set of lines within $t$, obtained by intersecting $t$ with all the long triangles in $\Delta$. Denote by $S_t^*$ the set of the double wedges dual to the segments in $S_t$, and by $L_t^*$ the set of points dual to the lines in $L_t$. We wish to obtain a compact representation of the set of all pairs $(p, w)$, where $p \in L_t^*$, $w \in S_t^*$, and $p \in w$. This can be done using standard range searching techniques, which, for the sake of completeness, we spell out next.

We construct a $(1/r)$-cutting $\Pi$ for the double wedges in $S_t^*$, for a sufficiently large constant parameter $r$, and locate all the points of $L_t^*$ in the cells of $\Pi$. We subdivide, if needed, each cell in $\Pi$ into smaller subcells, each containing at most $|L_t^*|/r^2$ points of $L_t^*$ in its interior. Let $\Pi'$ denote this new set of cells (it is easily seen that this decomposition does not asymptotically increase the number of cells in $\Pi'$, and thus $|\Pi'| = O(r^2)$). We then compute, for each cell $\pi \in \Pi'$, the set of all the double wedges of $S_t^*$ that fully contain $\pi$. The overall running time of this step is $O(r^2|S_t^*| + |L_t^*| \log r)$. Let us denote by $L_t(\pi)$ the set of lines of $L_t$ whose dual points lie in the interior of $\pi$, and by $S_t(\pi)$ the set of segments in $S_t$ whose dual double wedges contain $\pi$. We add $\{t\} \times L_t(\pi) \times S_t(\pi)$ to the output representation. Since each double wedge may contain $O(r^2)$ cells in its interior, and since $\sum_{\pi \in \Pi'} |L_t(\pi)| = |L_t^*|$, it follows that the overall size of the vertex sets of this compact representation, at this stage, is

$$\sum_{\pi \in \Pi'} (1 + |L_t(\pi)| + |S_t(\pi)|) = O(r^2|S_t^*| + |L_t^*|)$$

(we add 1 for each cell of $\Pi'$, since $\{t\}$ is also part of the representation). We now recursively continue to construct such a compact representation within each cell $\pi$ of $\Pi'$, where the subproblem at $\pi$ involves the at most $|L_t^*|/r^2$ dual points in $\pi$ and the at most $|S_t^*|/r$ double wedges whose boundaries cross $\pi$. The recursion is stopped when either $N_S$ or $N_L$ becomes smaller than $r$. We then report all intersecting pairs in a brute-force manner, as a collection of single-edge hypergraphs. The complexity of the representation, at any such bottom step, is $O(r(N_L + N_S))$.

Let $G(N_S, N_L)$ denote the maximum size of the compact representation of all intersecting pairs at a recursive step involving $N_S$ segments and $N_L$ lines. Then $G$ satisfies the following recurrence:

$$G(N_S, N_L) \leq \begin{cases} O\left(r^2 N_S + N_L\right) + O(r^2) G\left(\frac{N_S}{r}, \frac{N_L}{r^2}\right), & \text{if } N_S, N_L > r \\ O\left(r\left(N_S + N_L\right)\right), & \text{if } N_S \leq r \quad \text{or } N_L \leq r. \end{cases}$$

The solution of this recurrence (for a sufficiently large but constant value of $r$) is easily seen to be

$$G(N_S, N_L) = O(N_S^{2+\varepsilon} + N_L^{1+\varepsilon}),$$

for any $\varepsilon > 0$. We note that the same bound applies for the time needed to construct this representation. Thus the overall size of the compact representation of all intersecting triples of type LSS within a cell $\Delta$ is $O((N_S^{\Delta})^{3+\varepsilon} + N_S^{\Delta}(N_L^{\Delta})^{1+\varepsilon})$, for any $\varepsilon > 0$.

### Representing intersections of type SSS

The compact representation for all intersecting triples of type SSS is constructed in a brute-force manner, by examining all triples, and reporting separately each intersecting triple, as a separate single-edge tripartite hypergraph. The overall size of the representation, and the time needed to compute it, are both $O(N_S^3)$. This bound is subsumed by the bound on the size and the construction time of the representation of the intersecting triples of type LSS.

### The overall compact representation

We use the same recursive mechanism as in Section 3.1. To analyze its performance, we let $F(N_S, N_L)$ denote the time needed to construct the compact representation of all intersecting triples at a recursive step involving $N_S$ short triangles and $N_L$ long triangles. (The storage cost is analyzed in essentially the same manner.) Then $F$ satisfies the following recurrence:

$$F(N_S, N_L) \leq \begin{cases} O\left(r^{1+\varepsilon} N_S (N_S + N_L) \log^2 (N_S + N_L) + (N_S + N_L)^{1+\varepsilon'}\right) & \\ + \quad \sum_{i=0}^{\log\left(\frac{M}{r^2}\right)} O(2^i r^2) F\left(\frac{{N_S}^{1+\varepsilon}}{2^i r}, \frac{N_S + N_L}{r}\right), & \text{if } N_S > \max\left\{\sqrt{N_L}, c\right\} \\ O(N_S^{3+\varepsilon} + N_S N_L^{1+\varepsilon}), & \text{if } N_S \leq \max\left\{\sqrt{N_L}, c\right\}, \end{cases}$$

for any $\varepsilon > 0$, where $c \geq 3$ is constant, and $M$ and $\varepsilon'$ are defined as in Section 3.1.

Applying arguments similar to those in Section 3.1, we conclude that the solution of this recurrence is

$$F(N_S, N_L) = O\left(N_S (N_S + N_L)^{1+\varepsilon}\right), \text{ for any } \varepsilon > 0.$$

A slightly simpler version of this recurrence, that results in the same bound, applies for the *size* of the compact representation; we omit the straightforward details. We have thus shown:

**Theorem 3.2.1** *Given a collection $T$ of $n$ triangles in $\mathbb{R}^3$, the set of all intersecting triples among the triangles of $T$ can be represented in compact form, as the disjoint union of complete tripartite hypergraphs, with an overall size of*

$$\min\left\{O(\kappa n^{1/3+\varepsilon}), O(n^{2+\varepsilon})\right\},$$

*where $\kappa$ is the overall number of pairs of intersecting triangles. The time needed to construct this representation is*

$$\min\left\{O(n^{8/5+\varepsilon} + \kappa n^{1/3+\varepsilon}), O(n^{2+\varepsilon})\right\},$$

*for any $\varepsilon > 0$.*

### 3.2.1 Applications

**Drawing random intersections.** We now present two applications of the results obtained so far. In the first application, we wish to draw at random an element from the set of all intersecting triples among a set $T$ of $n$ triangles in $\mathbb{R}^3$. This is easy to do, in $O(\log n)$ time, using the compact representation of this set. [3] We first count the number $X$ of all the intersecting triples among the triangles of $T$ in $\min\left\{O(n^{8/5+\varepsilon} + \kappa n^{1/3}\log n), O(n^{2+\varepsilon})\right\}$ time, for any $\varepsilon > 0$, where $\kappa$ denotes, as usual, the overall number of intersecting pairs. If $X = O(n^2)$, we construct all intersecting triples explicitly in time $O(n^2 \log n)$, using the reporting algorithm mentioned in the introduction. In this case, a random intersection can then be drawn in $O(1)$ time. Otherwise, let the compact representation of all intersecting triples be given as $\bigcup_{i=1}^{s}(A_i \times B_i \times C_i)$. We first compute, as a preprocessing step, all the "prefix sums" $X_i = \sum_{i'<i} |A_{i'}| \cdot |B_{i'}| \cdot |C_{i'}|$, for $i = 1, \ldots, s$. We store these sums in a (sorted) array. The cost of this step is $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$. Next, to draw a random intersecting triple, we draw a random number $j$ between 1 and $X$, and find in $O(\log n)$ time the index $i$ that satisfies $X_i < j \le X_{i+1}$. We then pick the $(j - X_i)$-th edge of the hypergraph $A_i \times B_i \times C_i$, according to some obvious lexicographical order, and output the corresponding intersecting triple of triangles. Thus, drawing a random intersecting triple takes $O(\log n)$ time, with $O(n^{2+\varepsilon})$ preprocessing time and storage, for any $\varepsilon > 0$.

**Finding the $k$-th highest vertex.** We have mentioned in the introduction another application of our machinery: Given a set $T$ of $n$ triangles in $\mathbb{R}^3$, and a parameter $k$, find the $k$-th highest vertex of $\mathcal{A}(T)$. This problem can be solved in nearly quadratic time as follows. We construct the compact representation of all triple intersections in $T$, and draw a random subset $V$ of $O(n^2)$ intersections. We sort the points in $V$ in increasing order of their $z$-coordinates, and run a binary search through them. At each step of the search, involving a vertex $v$, we count the number of triple intersections between the triangles

[3]This algorithm is a variant of another algorithm that the authors have used for drawing a random element of the set of all intersecting pairs of $n$ given segments in the plane [70].

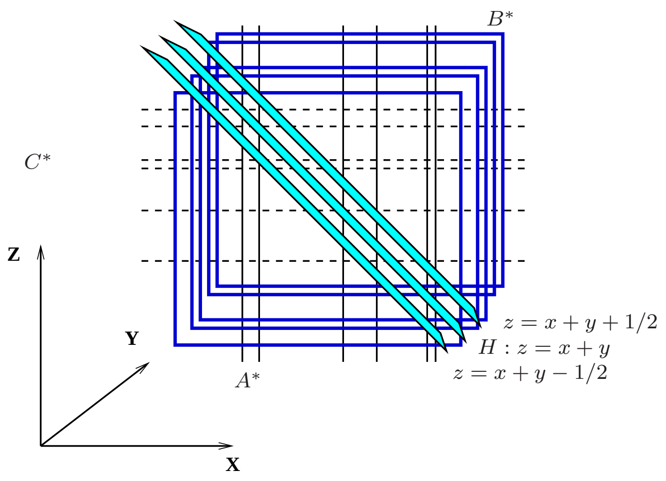


Figure 3.2: The construction used to reduce 3SUM' to 3COUNTING. The vertical lines are the planes representing the elements of $A$, the rectangles are the planes representing the elements of $B$, and the horizontal lines are the planes representing the elements of $C$. We exclude the region $x + y - 1/2 < z < x + y + 1/2$.

of $T$ that lie above $v$, and use this count to guide the binary search. This counting can be done by clipping each triangle to the halfspace above $v$, and by applying our counting algorithm to the clipped triangles. When the search terminates, we obtain a horizontal slab $\Sigma$, which is known to contain the desired $k$-th highest vertex, and which contains, with high probability, only $O\left(\frac{n^3}{n^2}\log n\right) = O(n \log n)$ triple intersections. We can then enumerate all of them explicitly, by applying an intersection reporting algorithm to the portions $t \cap \Sigma$, for $t \in T$, in $O(n^2 \log n)$ time, and select from among these points the desired vertex. The total cost of this algorithm is, with high probability, $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$. Note that when the number $\kappa$ of intersecting pairs of triangles is small, we can get a subquadratic algorithm that runs in time $O(n^{8/5+\varepsilon} + \kappa n^{1/3+\varepsilon})$, for any $\varepsilon > 0$.

## 3.3 Counting Intersecting Triples is 3SUM-Hard

In this section we show that the problem of counting all intersecting triples among triangles in $\mathbb{R}^3$, a problem that we denote as 3COUNTING, belongs to the 3SUM-hard family (see [77]), and thus, the best solution to this problem is likely to require $\Omega(n^2)$ time in the worst case, thus making our algorithm nearly worst-case optimal. Let us consider the following problem, which is among the basic 3SUM-hard problems [77]:

**Problem 3SUM':**

Given three sets of integers $A$, $B$, and $C$ of total size $n$, are there $a \in A$, $b \in B$, $c \in C$ with $a + b = c$?

We show that the 3SUM' problem is linear-time reducible to 3COUNTING. Given three sets $A$, $B$ and $C$ of integers, we transform each element $a \in A$ into the plane $h_a : x = a$, each element $b \in B$ into the plane $h_b : y = b$, and each element $c \in C$ into the plane $h_c : z = c$. We denote the three resulting sets of planes by $A^*$, $B^*$ and $C^*$, respectively. Every triple of planes $h_a \in A^*$, $h_b \in B^*$ and $h_c \in C^*$ intersects at the point $(a, b, c)$, and the overall number of such intersecting triples is $|A||B||C|$. We now add to the scene the plane $H : z = x + y$. See Figure 3.2 for an illustration.

The (obvious but) key observation is that there is a triple $a \in A$, $b \in B$, $c \in C$ such that $a + b = c$ if and only if the plane $H$ contains the intersection point of the three planes $h_a$, $h_b$ and $h_c$. We thus split each plane $h \in A^* \cup B^* \cup C^*$ into two halfplanes at the intersection line $h \cap H$, resulting in six sets of *open* halfplanes, such that the halfplanes in three subfamilies lie above $H$ and the halfplanes in the other three subfamilies lie below $H$. We now count all triple intersections among the open halfplanes that lie above $H$ and all triple intersections among the open halfplanes that lie below $H$. It now follows that the overall number of intersections on both sides of $H$ is strictly smaller than $|A||B||C|$ if and only if there are three planes $h_a \in A^*$, $h_b \in B^*$ and $h_c \in C^*$, such that $H$ contains their intersection point $(a, b, c)$, which is equivalent to the existence of three numbers $a \in A$, $b \in B$, $c \in C$ such that $a + b = c$.

To make the reduction compatible with the type of input assumed by 3COUNTING, we next modify each open halfplane into a bounded closed triangle. We first intersect each plane in $A^*$, $B^*$ and $C^*$ with a box bounding all vertices of the arrangement of these planes, obtaining three corresponding sets of rectangles $A^\square$, $B^\square$ and $C^\square$. Next, we observe that the intersection points among all triples $r_a \in A^\square$, $r_b \in B^\square$ and $r_c \in C^\square$ have integer coordinates (and thus the distance between each pair of such points is at least 1). Hence, the region $\{x + y - 1/2 < z < x + y + 1/2\} \setminus H$ does not contain any of these intersection points (see Figure 3.2 for an illustration). Hence, it is sufficient to intersect each rectangle in $A^\square$, $B^\square$ and $C^\square$ with the two halfspaces $z \le x+y-1/2$ and $z \ge x+y+1/2$, obtaining six families of bounded closed triangles, quadrilaterals, and pentagons, which can be further refined into families of closed bounded triangles, with the same properties as the families of halfplanes constructed earlier.

We note that computing the bounding box of all vertices of the arrangement of the planes in $A^*$, $B^*$ and $C^*$ is trivially done in $O(n)$ time ($n = |A| + |B| + |C|$), and that the construction of the six families of triangles takes additional $O(n)$ time. Thus we have shown that 3SUM' is linear-time reducible to 3COUNTING, implying that 3COUNTING is a 3SUM-hard problem.

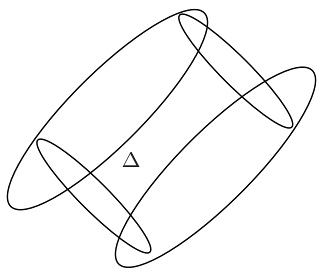


Figure 3.3: A view from above of four ellipses in $\mathbb{R}^3$. After the vertical walls from their boundaries are erected, nonconvex cells, such as $\Delta$, are generated. Thus, the intersection of $\Delta$ with any ellipse that crosses $\Delta$ is not convex.

## 3.4 Extensions

In this section we extend the algorithms presented in Sections 3.1 and 3.2 to count or represent all intersecting triples among $n$ planar simply shaped objects in $\mathbb{R}^3$.

Let $\mathcal{S}$ be a collection of $n$ planar objects in $\mathbb{R}^3$, such that each object $s \in \mathcal{S}$ is bounded by a closed planar curve $c \in \mathbb{R}^3$ of *constant description complexity*; recall that this means that each bounding curve is defined as a Boolean combination of a constant number of polynomial equalities and inequalities of constant maximum degree. We also assume that the objects in $\mathcal{S}$ are in general position, and in particular that *no two of them are coplanar*. In this case, as already discussed in Section 3.1, we can construct for $\mathcal{S}$ a $(1/r)$-cutting $\Xi$ of size $O(r^{3+\varepsilon})$, for any $\varepsilon > 0$, which is sensitive to the set of the bounding curves of the elements in $\mathcal{S}$, in the sense that the number of crossings between these bounding curves and the cells of $\Xi$ is $O(n^{1+\varepsilon}r)$. The time needed to construct this cutting, when $r$ is at most $O(n^{\varepsilon})$, is $O(n^{1+\varepsilon'})$, for any $\varepsilon' > 0$ that is sufficiently larger than $\varepsilon$.

Note that the cells of $\Xi$, and the clippings of objects of $\mathcal{S}$ to cells of $\Xi$, need not be convex, even when the original objects in $\mathcal{S}$ are convex. Indeed, as part of the construction of $\Xi$ (as presented in [95]), we draw a random sample $R$ of the bounding curves, and erect vertical walls up and down from each such curve. Thus, a clipped object $s \in \mathcal{S}$ to within a cell $\Delta \in \Xi$ need not be convex; see Figure 3.3. Nevertheless, since the given objects have constant description complexity (and hence so does each cell $\Delta \in \Xi$), it follows that each clipped object $s$ has constant description complexity, and each element $s' \in \mathcal{S}$ intersects the (clipped) object $s$ in $O(1)$ line-segments (recall that they are assumed to lie in different planes). A clipped object need not be connected, but it has at most $O(1)$ connected components, and we treat each of them separately. In what follows we abuse the notation of $s$ to denote a (connected component of a) clipped object to within a cell $\Delta$ of $\Xi$.

These properties allow us to apply a similar algorithm to that presented in Section 3.1, in order to count all intersecting triples among the elements of $\mathcal{S}$. More specifically, the recursive mechanism remains the same, and the four simple algorithms can be applied with

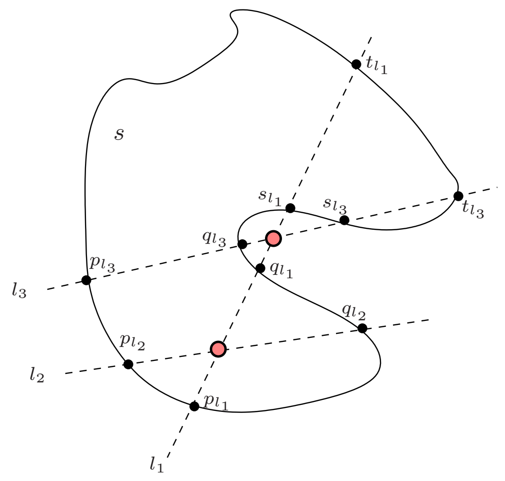


Figure 3.4: The lines $l_1$, $l_2$, $l_3$ are cross sections within $s$ of corresponding long objects of $\mathcal{S}$. The lines $l_1$ and $l_2$ intersect within $s$ since the intersection points $p_{l_1}$, $p_{l_2}$, $q_{l_1}$, $q_{l_2}$ of $\partial s$ with two of their contained segments interleave along $\partial s$. The lines $l_1$ and $l_3$ do not intersect within $s$ since they do not contain segments that are clipped to within $s$ and have interleaving endpoints.

slight modifications. In the case of counting intersecting triples of type LLL (or LLS), the input to the planar algorithm, that we apply within each (clipped) object $s$, is the set of all clipped segments that are generated by the intersections of $s$ with an appropriate subset of the long objects. Note that, since we intersect $s$ only with long objects, the endpoints of each intersection segment lie on $\partial s$. In this case, as in the case of triangles, two clipped segments $l_1$, $l_2$ intersect within $s$ if and only if their endpoints interleave along $\partial s$; see Figure 3.4 for an illustration. Since each long object $s'$ intersects $s$ in a constant number of segments, the number of input segments to the two-dimensional algorithm is $O(N_L)$, and thus the running time of the planar algorithm remains $O(N_L \log N_L)$.

In the case of counting intersecting triples of type LSS, the input to the two-dimensional algorithm, that we apply within each clipped object, is the set of all clipped short segments and the set of the containing lines of all the clipped long segments. Here however we face a new difficulty. Since the cells $\Delta$ need not be convex, extending a clipped long segment into a full line, as we did in the case of triangles, may produce new intersections with short (or long) segments, a situation that is illustrated in Figure 3.5. We overcome this difficulty as follows. We extend each short segment $e$ to the (unique) longer segment $e'$ that contains $e$, which is contained in $s$, and has endpoints lying on $\partial s$. Let $S'_s$ denote the set of the resulting extended segments, and let $L_s$ denote the set of long segments in $s$. We first apply an appropriate variant of the *representation* algorithm for LLS (or LLL) intersections (see Section 3.2), that reports the set of all bichromatic intersecting pairs $(e', \lambda) \in S'_s \times L_s$ as the disjoint union of complete bipartite graphs $S'_i \times L_i$, $i = 1, \ldots, m$.

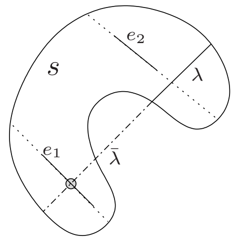


Figure 3.5: Extending a long segment $\lambda$ into a full line may create new intersections with short segments (as the intersection with $e_1$). However, if the extended short segment (such as the extension of $e_2$) and $\lambda$ meet, extending $\lambda$ to a line is safe.

Recall that $\sum_{i=1}^{m}|S'_i| = O(N_S^\Delta \log^2 N_L^\Delta)$ and $\sum_{i=1}^{m}|L_i| = O(N_L^\Delta \log^2 N_L^\Delta)$. We now process each subgraph $S'_i \times L_i$ separately. For any pair $(e', \lambda) \in S'_i \times L_i$, we have $e' \cap \lambda \neq \emptyset$. This is easily seen to imply that if we replace $\lambda$ by its containing line $\bar{\lambda}$, then the original short segment $e$ whose extension is $e'$ intersects $\lambda$ if and only if $e$ intersects $\bar{\lambda}$; see Figure 3.5. Hence we are now in the scenario of the LSS algorithm of Section 3.1, in which we need to count intersections between segments and lines, which can be accomplished using the same duality-based algorithm. The running time is, as in Section 3.1,

$$O\left(\sum_{i=1}^{m}(|S'_i|^2 + |L_i|\log|S_i|)\right) = O((N_S^\Delta)^2 \log^2 N_L^\Delta + N_L^\Delta \log^2 N_L^\Delta \log N_S^\Delta),$$

where the first term in the bound is obtained by noting that $|S'_i| = O(N_S^\Delta)$, for each $i = 1, \ldots, m$.

In the case of counting intersecting triples of type SSS, we use a brute-force algorithm as in Section 3.1. The running time of this algorithm is $O(N_S^3)$, because the objects in $\mathcal{S}$ have constant description complexity, and thus each triple is examined in constant time.

With all the four intersection counting routines (for LLL, LLS, LSS, and SSS intersections) properly extended, the algorithm can proceed in essentially the same manner as in Section 3.1. The slight increase in the running time of the LSS routine is subsumed in the final bound $O(n^{2+\varepsilon})$, as is easy to check.

Arguing as in Section 3.2, we can use the same mechanism, with appropriate modifications, to derive an algorithm for constructing a compact representation of these intersections, with the same nearly quadratic bound on the storage and the running time.

Note that the preliminary alternative algorithm described in Section 3.1 is not applicable to general planar objects, since it operates only on polygonal objects. We can replace this step by a more complicated procedure based on the technique of [8] for range searching with semi-algebraic sets, to obtain, in time $O(n^{2-\gamma} + \kappa)$, all the $\kappa$ intersecting objects in $S$, where $\gamma > 0$ is some constant that depends on the complexity of the objects in $S$. This yields a total of $O(\kappa)$ intersection segments on the individual objects in $S$, and we can apply the algorithm of [1, 102] to count the number of (or represent) their intersections, exactly as in Sections 3.1 and 3.2. If $\kappa$ is relatively smaller than $n^{5/3}$, we end up with alternative subquadratic counting algorithms that run in time $O(n^{2-\gamma} + \kappa n^{1/3} \log n)$, for any $\varepsilon > 0$, (or $O(n^{2-\gamma} + \kappa n^{1/3+\varepsilon})$ when constructing the compact representation). We thus conclude:

**Theorem 3.4.1** *The number of intersecting triples in a set of $n$ planar objects of constant description complexity that lie in different planes in $\mathbb{R}^3$ can be counted in time*

$$\min\left\{O(n^{2-\gamma} + \kappa n^{1/3} \log n), O(n^{2+\varepsilon})\right\},$$

*where $\gamma$ is a positive constant that depends on the complexity of the objects in $S$. Moreover, these intersecting triples can be represented in a compact form, as the disjoint union of complete tripartite hypergraphs, whose total size is*

$$\min\left\{O(\kappa n^{1/3+\varepsilon}), O(n^{2+\varepsilon})\right\}.$$

*The time needed to construct this representation is*

$$\min\left\{O(n^{2-\gamma} + \kappa n^{1/3+\varepsilon}), O(n^{2+\varepsilon})\right\},$$

*for any $\varepsilon > 0$.*

**Applications.** It follows, as in Section 3.2, that, with $O(n^{2+\varepsilon})$ preprocessing time and storage, we can draw a random intersecting triple of objects of $\mathcal{S}$ in $O(\log n)$ time. Similarly, we can find in $O(n^{2+\varepsilon})$ time the $k$-highest vertex in an arrangement of $n$ such objects. As in the case of triangles, subquadratic solutions can be obtained when the number $\kappa$ of intersecting pairs of objects is significantly smaller than $n^{5/3}$.

## 3.5 Concluding Remarks and Open Problems

The results obtained in this study raise several open problems. A challenging open problem is to obtain comparably efficient algorithms for the case where the input objects are not necessarily planar, but are curved surface patches (or closed surfaces) in $\mathbb{R}^3$ of constant description complexity. The three subtasks of counting LLL, LLS, and LSS intersections become considerably harder, because they call for counting the number of intersections between curves and arcs on some curved surface, and the best known algorithms for these tasks are much less efficient than those for lines and line-segments, which we have used above.

Consider, for example, the problem of counting all intersecting triples among $n$ spheres in $\mathbb{R}^3$. In this case, in the LLL subroutine, we need to solve the problem of counting all intersecting pairs among circles and/or long circular arcs (that is, arcs within a patch of a sphere, that completely cross this patch). However, we are not aware of any algorithm for this task that is faster than (an appropriate variant of) the standard algorithm that counts intersections between circular arcs in the plane, and runs in time $O(n^{3/2+\varepsilon})$ [11]. Thus, in this case, our algorithm is not better than a simple-minded algorithm that intersects each sphere with all the others spheres, and uses the two-dimensional algorithm of [11] on each sphere. The running time of this algorithm is thus $O(n^{5/2+\varepsilon})$.

Finally, another challenging problem is to count $d$-wise intersections among $(d-1)$-simplices in $\mathbb{R}^d$, for $d \geq 4$. Note that the reduction presented in Section 3.3 can be extended to $d$-space and thus the $d$-dimensional problem is dSUM-hard [66], which implies that the best solution to this problem is likely to require $\Omega(n^{\lfloor (d+1)/2 \rfloor})$ time in the worst case. However, we are not aware of any algorithm whose running time is close to this bound. A simple minded algorithm can be performed, for example, by induction on the dimension $d$, as follows. Given $n$ $(d-1)$-simplices in $\mathbb{R}^d$, we intersect the facets of each simplex $s$ with all the other $n-1$ input simplices, obtaining $O(n)$ subproblems in one dimension lower (with a constant of proportionality that depends on $d$). We now continue to solve each such subproblem recursively. We stop the recursion when we reach three-dimensional problems, and then solve each of them in nearly quadratic time. It thus follows that the overall running time of this algorithm is $O(n^{d-1+\varepsilon})$, for each $d \geq 3$, where the constant of proportionality depends on $d$ and $\varepsilon$. An open problem is to prove that this bound is nearly optimal, or, alternatively, design an improved algorithm for this problem.

We also note that in higher dimensions there is a wider range of problems, in which one might wish to count (or represent) the number of all $k$-wise intersections among $n$ $(d-1)$-simplices in $d$-space, where $k$ can vary from 2 to $d$. Each of these variants is a challenging open problem.

# Chapter 4

# Analysis of the ICP Algorithm

In this chapter we analyze the performance of the ICP algorithm, as reviewed in Section 1.4.

In the next section we first show a (probably weak) upper bound of $O\left(m^d n^d\right)$ on the number of iterations of the algorithm in $\mathbb{R}^d$ under either of the two measures, for any $d \geq 1$. We then present, in Section 4.2, several structural geometric properties of the algorithm under the RMS measure. Specifically, we show that at each iteration of the algorithm the (real) cost function monotonically and strictly decreases, in a continuous manner, along the vector $\Delta t$ of the relative translation; this is a much stronger property than the originally noted one, that the value at the end of the translation is smaller than that at the beginning. As a result, we conclude that the polygonal path $\pi$ obtained by concatenating all the relative translations that are computed during the execution of the algorithm, does not intersect itself. In particular, for $d = 1$, the ICP algorithm is *monotone* — all its translations are in the same (left or right) direction. Next, in Section 4.3 we present a lower bound construction of $\Omega(n \log n)$ iterations for the one-dimensional problem under the RMS measure (assuming $m \approx n$). The upper bound is quadratic, and closing the substantial gap between the bounds remains a major open problem. In Section 4.4 we discuss the problem under the (one-sided) Hausdorff distance measure. In particular, we present for the one-dimensional problem an upper bound of $O\left((m+n) \log \delta_B / \log n\right)$ on the number of iterations of the algorithm, where $\delta_B$ is the *spread* of the input point set $B$ (i.e., the ratio between the diameter of the set and the distance between its closest pair of points). We then present a tight lower bound construction with $\Theta(n)$ moves, for the case where the spread of $B$ is polynomial in $n$. We also study the problem under the Hausdorff measure in two and higher dimensions, and show that some of the structural properties of the algorithm that hold for the RMS measure do not hold in this case. We present open problems and give concluding remarks in Section 4.5.

## 4.1 An Upper Bound for the Number of Iterations

We use the notation introduced in Section 1.4. Let $A = \{a_1, \ldots, a_m\}$ and $B = \{b_1, \ldots, b_n\}$ be two point sets in $d$-space, for $d \geq 1$, and, as described in Section 1.4, suppose that the ICP algorithm aligns $A$ to $B$; that is, $B$ is fixed and $A$ is translated to best fit $B$. In what follows, we use the above notation (unless stated otherwise).

**Theorem 4.1.1** *The maximum possible overall number of nearest-neighbor assignments, over all translated copies of $A$, is $\Theta\left(m^d n^d\right)$.*

**proof**: Let $\mathcal{V}(B)$ denote the *Voronoi diagram* of $B$, that is, the partition of $\mathbb{R}^d$ into $d$-dimensional cells $\mathcal{V}(b_i)$, for $i = 1, \ldots, n$, such that each point $p \in \mathcal{V}(b_i)$ satisfies $\|p - b_i\| \leq \|p - b_j\|$, for each $j \neq i$.

The global NNA (nearest-neighbor assignment) changes at critical values of the translation $t$, in which the nearest-neighbor assignment of some point $a + t$ of the translated copy of $A$ is changed; that is, $a + t$ crosses into a new Voronoi cell of $\mathcal{V}(B)$. For each $a \in A$ (this denotes the *initial* location of this point) consider the shifted copy $\mathcal{V}(B) - a = \mathcal{V}(B - a)$ of $\mathcal{V}(B)$; i.e., the Voronoi diagram of $B - a = \{b - a \mid b \in B\}$. Then a critical event that involves the point $a_i$ occurs when the translation $t$ lies on the boundary of some Voronoi cell of $\mathcal{V}(B - a_i)$, for $i = 1, \ldots, m$. Hence we need to consider the *overlay* $M(A, B)$ of the $m$ shifted diagrams $\mathcal{V}(B - a_1), \ldots, \mathcal{V}(B - a_m)$. Each cell of the overlay consists of translations with a common NNA, and the number of assignments is in fact equal to the number of cells in the overlay $M(A, B)$. A recent result of Koltun and Sharir [95] implies that the complexity of the overlay is $O(m^d n^d)$. It is straightforward to give constructions that show that this bound is tight in the worst case, for any $d \geq 1$. $\Box$

**Corollary 4.1.2** *For any cost function that guarantees convergence (in the sense that the algorithm does not reach the same NNA more than once), the ICP algorithm terminates after $O(m^d n^d)$ iterations.*

**Remark:** A major open problem is to determine whether this bound is tight in the worst case. So far we have been unable to settle this question, under the RMS measure, even for $d = 1$ (but subsequent progress on this question has been made in [30]).

## 4.2 General Structural Properties under the RMS Measure

We first present a simple but crucial property of the relative translations that the algorithm generates.

**Lemma 4.2.1** *At each iteration $i \geq 2$ of the algorithm, the relative translation vector $\Delta t_i$ satisfies*

$$\Delta t_i = \frac{1}{m}\sum_{a\in A}\Big(N_B(a+t_{i-1}) - N_B(a+t_{i-2})\Big), \tag{4.1}$$

*where $t_j = \sum_{k=1}^{j} \Delta t_k$.*

**Proof**: Follows using easy algebraic manipulations, based on the well-known fact that, for a fixed nearest-neighbor assignment, the RMS cost is minimized when the two centroids $\frac{1}{|A|}\sum_{a\in A}(a+t_{i-1})$, $\frac{1}{|A|}\sum_{a\in A} N_B(a+t_{i-1})$ coincide, and thus

$$\Delta t_i = \frac{1}{m}\sum_{a\in A}\Big(N_B(a+t_{i-1}) - (a+t_{i-1})\Big). \tag{4.2}$$

(See [88, Lemma 5.2] for similar considerations.) Applying (4.2) also to $\Delta t_{i-1}$, and subtracting the two equations, yields (4.1). □

**Remark:** The expression in (4.1) implies that the next relative translation is the average of the differences between the new $B$-nearest neighbor and the old $B$-nearest neighbor of each point of (the current and preceding translations of) $A$. This property does not hold for the *first* relative translation of the algorithm.

**Theorem 4.2.2** *Let $\Delta t$ be a move of the ICP algorithm from translation $t_i$ to $t_i + \Delta t$. Then* $\mathrm{RMS}(t_i + \xi\Delta t)$ *is a strictly decreasing function of $\xi \in [0, 1]$.*

**First Proof:** We present two (related) proofs. In the first proof, put

$$\mathrm{RMS}_0(\xi) := \frac{1}{m}\sum_{a\in A}\|a + t_i + \xi\Delta t - N_B(a+t_i)\|^2.$$

Note that, by the definition of the ICP algorithm, the graph of $\mathrm{RMS}_0(\xi)$ is a parabola that attains its minimum at $\xi = 1$. Hence, its derivative is negative for $\xi \in [0, 1)$. That is,

$$\frac{1}{2}\mathrm{RMS}_0'(\xi) = \frac{1}{m}\sum_{a\in A}\Big(a + t_i + \xi\Delta t - N_B(a+t_i)\Big)\cdot\Delta t < 0,$$

or

$$\frac{1}{2}\mathrm{RMS}_0'(\xi) = \xi\|\Delta t\|^2 + \frac{1}{m}\sum_{a\in A}\Big(a + t_i - N_B(a+t_i)\Big)\cdot\Delta t < 0.$$

On the other hand, for any $\xi \in [0, 1]$, the function

$$\mathrm{RMS}_1(\xi) := \mathrm{RMS}(t_i + \xi\Delta t) = \frac{1}{m}\sum_{a\in A}\|a + t_i + \xi\Delta t - N_B(a + t_i + \xi\Delta t)\|^2$$

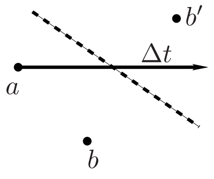


Figure 4.1: The new nearest neighbor lies ahead of the old one in the direction $\Delta t$.

is the (real) RMS-distance from $A$ to $B$ at the translation $t_i + \xi\Delta t$, i.e., the distance with the real nearest-neighbor assignment at $t_i+\xi\Delta t$, rather than the "frozen" assignment at $t_i$. Our goal is to show that $\mathrm{RMS}_1'(\xi) < 0$, for any $\xi \in [0,1]$, in which the function $\mathrm{RMS}_1(\xi)$ is smooth (note that $\mathrm{RMS}_1(\xi)$ is non-smooth exactly at points where some $a$ changes its nearest neighbor in $B$). As above, we have, at points $\xi$ where $\mathrm{RMS}_1(\xi)$ is smooth,

$$\frac{1}{2}\mathrm{RMS}_1'(\xi) = \xi\|\Delta t\|^2 + \frac{1}{m}\sum_{a\in A}\Big(a + t_i - N_B(a+t_i+\xi\Delta t)\Big)\cdot\Delta t.$$

It follows that

$$\mathrm{RMS}_0'(\xi) - \mathrm{RMS}_1'(\xi) = \frac{2}{m}\sum_{a\in A}\Big(N_B(a+t_i+\xi\Delta t) - N_B(a+t_i)\Big)\cdot\Delta t.$$

We claim that each of the terms in the latter sum is non-negative. Indeed, consider a fixed point $a$. When $a$ changes its nearest neighbor from some $b$ to another $b'$, it has to cross the bisector of $b$ and $b'$ from the side of $b$ to the side of $b'$. This is easily seen to imply that (see also Figure 4.1)

$$(b'-b)\cdot\Delta t \geq 0.$$

Adding up all these inequalities that arise at bisector crossings during the motion of $a$, we obtain the claimed inequality. Hence $\mathrm{RMS}_0'(\xi) \geq \mathrm{RMS}_1'(\xi)$ throughout the motion, and since $\mathrm{RMS}_0'(\xi)$ is negative, so must be $\mathrm{RMS}_1'(\xi)$.

**Second Proof:** (This can be regarded as a geometric interpretation of the first proof.) The function

$$\mathrm{RMS}(t) = \frac{1}{m}\sum_{a\in A}\|a+t-N_B(a+t)\|^2 =$$

$$\frac{1}{m}\sum_{a\in A}\Big(\|t\|^2 + 2t\cdot(a-N_B(a+t)) + \|a-N_B(a+t)\|^2\Big)$$

is the average of $m$ Voronoi surfaces $\mathcal{S}_{B-a}(t)$, whose respective minimization diagrams are $\mathcal{V}(B-a)$, for each $a\in A$. That is,

$$\mathcal{S}_{B-a}(t) = \min_{b\in B}\|a+t-b\|^2 = \min_{b\in B}\Big(\|t\|^2 + 2t\cdot(a-b) + \|a-b\|^2\Big),$$

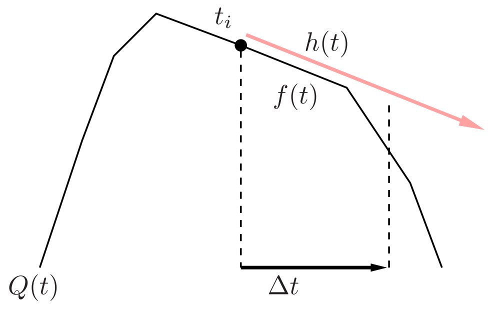


Figure 4.2: Illustrating the proof that $\mathrm{RMS}(t_i + \xi\Delta t)$ is a strictly decreasing function of $\xi \in [0, 1]$.

for each $a \in A$. Subtracting the term $\|t\|^2$, we obtain that each resulting Voronoi surface $\mathcal{S}_{B-a}(t) - \|t\|^2$ is the lower envelope of $n$ hyperplanes, and its graph is thus the boundary of a concave polyhedron. Hence $Q(t) := \mathrm{RMS}(t) - \|t\|^2$ is equal to the average of these concave polyhedral functions, and is thus itself the boundary of a concave polyhedron (see also the proof of Theorem 4.1.1).

Consider the NNA that corresponds to the translation $t_i$. It defines a facet $f(t)$ of $Q(t)$, which contains the point $(t_i, Q(t_i))$. We now replace $f(t)$ by the hyperplane $h(t)$ containing it, and note that $h(t)$ is tangent to the polyhedron $Q(t)$ at $t_i$; see Figure 4.2 for an illustration. The graph of $\mathrm{RMS}_0(\xi)$, as defined above, is the image of the relative translation vector $\Delta t$ on the paraboloid $\|t\|^2 + h(t)$. Since $Q(t) \le h(t)$, for any $t \in \mathbb{R}^d$, the concavity of $Q(t)$ implies that for any $0 \le \xi_1 < \xi_2 \le 1$, $Q(t_i + \xi_1\Delta t) - Q(t_i + \xi_2\Delta t) \ge h(t_i + \xi_1\Delta t) - h(t_i + \xi_2\Delta t)$. Since $\|t\|^2 + h(t)$ is (strictly) monotone decreasing along $\Delta t$ (by definition, $\Delta t$ moves from $t_i$ to the minimum of the fixed paraboloid $\|t\|^2 + h(t)$), we obtain

$$\mathrm{RMS}(t_i + \xi_1\Delta t) - \mathrm{RMS}(t_i + \xi_2\Delta t) =$$

$$\|t_i + \xi_1\Delta t\|^2 + Q(t_i + \xi_1\Delta t) - \|t_i + \xi_2\Delta t\|^2 - Q(t_i + \xi_2\Delta t) \ge$$

$$\|t_i + \xi_1\Delta t\|^2 - \|t_i + \xi_2\Delta t\|^2 + h(t_i + \xi_1\Delta t) - h(t_i + \xi_2\Delta t) > 0,$$

which implies that $\mathrm{RMS}(t_i + \xi\Delta t)$ is a strictly decreasing function of $\xi \in [0, 1]$. $\square$

Let $\pi$ be the connected polygonal path obtained by concatenating the ICP relative translations $\Delta t_j$. That is, $\pi$ starts at the origin and its $j$-th edge is the vector $\Delta t_j$. Theorem 4.2.2 implies:

**Theorem 4.2.3** *The ICP path $\pi$ does not intersect itself.*

In particular, Theorem 4.2.3 implies that, on the line, the points of $A$ are always translated in the same direction at each iteration of the algorithm. We thus obtain:

**Corollary 4.2.4 (Monotonicity)** *In the one-dimensional case, the ICP algorithm moves the points of $A$ always in the same (left or right) direction. That is, either $\Delta t_i \geq 0$ for each $i \geq 0$, or $\Delta t_i \leq 0$ for each $i \geq 0$.*

**Corollary 4.2.5** *In any dimension $d \geq 1$, the angle between any two consecutive edges of $\pi$ is obtuse.*

**Proof:** Consider two consecutive edges $\Delta t_i$, $\Delta t_{i+1}$ of $\pi$. Using Lemma 4.2.1 we have $\Delta t_{i+1} = \frac{1}{m}\sum_{a\in A}\Big(N_B(a+t_i) - N_B(a+t_{i-1})\Big)$. As follows from the first proof of Theorem 4.2.2 (see once again Figure 4.1),

$$\Big(N_B(a+t_i) - N_B(a+t_{i-1})\Big) \cdot \Delta t_i \geq 0,$$

for each $i \geq 1$, where equally holds if and only if $a$ does not change its $B$-nearest neighbor. Hence $\Delta t_{i+1} \cdot \Delta t_i \geq 0$. It is easily checked that equality is possible only after the last step (where $\Delta t_{i+1} = 0$). □

**Lemma 4.2.6** *At each iteration $i \geq 1$ of the algorithm,* $\mathrm{RMS}(t_{i-1}) - \mathrm{RMS}(t_i) \geq \|\Delta t_i\|^2$.

**Proof**: As in the proof of Theorem 4.2.2, consider the function

$$\mathrm{RMS}_0(\xi) := \frac{1}{m}\sum_{a\in A}\|a + t_{i-1} + \xi\Delta t_i - N_B(a+t_{i-1})\|^2.$$

This is a parabola, with minimum at $\xi = 1$ (i.e., at $\Delta t_i$), whose quadratic term is $\xi^2\|\Delta t_i\|^2$. Hence, its value at $\xi = 0$ is $\|\Delta t_i\|^2$. That is,

$$\mathrm{RMS}(t_{i-1}) - \mathrm{RMS}(t_i) \geq \mathrm{RMS}_0(0) - \mathrm{RMS}_0(1) = \|\Delta t_i\|^2,$$

where the first inequality follows from $\mathrm{RMS}(t_i) \leq \mathrm{RMS}_0(1)$, since both expressions are computed at $t_i$, where $\mathrm{RMS}_0(1)$ uses the old NNA, and $\mathrm{RMS}(t_i)$ uses the new NNA, which makes its value smaller. □

**Corollary 4.2.7** *If the relative translations computed by the algorithm are $\Delta t_1, \ldots, \Delta t_k$, then*

$$\frac{1}{k}\left(\sum_{i=1}^{k}\|\Delta t_i\|\right)^2 \leq \sum_{i=1}^{k}\|\Delta t_i\|^2 \leq \mathrm{RMS}(t_0) - \mathrm{RMS}(t_k). \tag{4.3}$$

**Proof**: Use the Cauchy-Schwarz inequality, applied to the result of Lemma 4.2.6. □

**Lemma 4.2.8** *At each iteration $i \geq 0$ of the algorithm*

$$\mathrm{RMS}(t_0) - \mathrm{RMS}(t_i) \leq \|t_{i+1}\|^2 - \|\Delta t_{i+1}\|^2. \tag{4.4}$$

**Proof**: We have

$$\mathrm{RMS}(t_i) - \mathrm{RMS}(t_0) = \frac{1}{m}\sum_{a\in A}\Big(\|N_B(a+t_i) - a - t_i\|^2 - \|N_B(a) - a\|^2\Big) =$$

$$\frac{1}{m}\sum_{a\in A}\Big(\|N_B(a+t_i) - a - t_i\|^2 - \|N_B(a+t_i) - a\|^2\Big) + \frac{1}{m}\sum_{a\in A}\Big(\|N_B(a+t_i) - a\|^2 - \|N_B(a) - a\|^2\Big).$$

The second sum is non-negative, since $\|N_B(a) - a\|^2 \le \|N_B(a+t_i) - a\|^2$, for each $a \in A$, and the first sum is

$$\frac{1}{m}\sum_{a\in A}\Big(-t_i \cdot (2(N_B(a+t_i) - a - t_i) + t_i)\Big) = -\|t_i\|^2 - 2t_i \cdot \Delta t_{i+1},$$

by Equation 4.2. That is, we have

$$\mathrm{RMS}(t_i) - \mathrm{RMS}(t_0) \ge -\|t_i\|^2 - 2t_i\cdot\Delta t_{i+1} = -\|t_i + \Delta t_{i+1}\|^2 + \|\Delta t_{i+1}\|^2 = -\|t_{i+1}\|^2 + \|\Delta t_{i+1}\|^2,$$

as asserted. $\square$

Combining inequalities (4.3) and (4.4), we obtain,

**Corollary 4.2.9** *For any $k \ge 1$,*

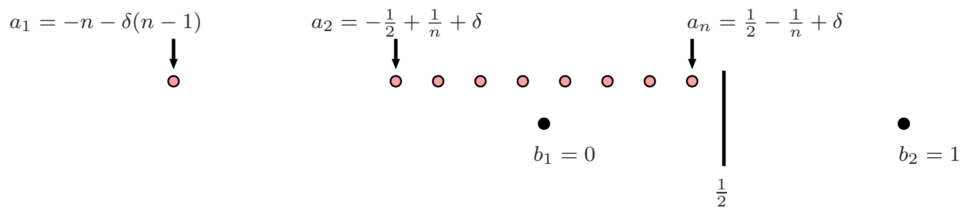
$$\sum_{i=1}^{k}\|\Delta t_i\|^2 \le \mathrm{RMS}(t_0) - \mathrm{RMS}(t_k) \le \|t_{k+1}\|^2 - \|\Delta t_{k+1}\|^2.$$

*In particular, we have, rearranging terms and replacing $k+1$ by $k$, $\sum_{i=1}^{k}\|\Delta t_i\|^2 \le \|t_k\|^2$.*

**Remarks:** (1) Note that, for $d = 1$, this inequality is trivial (and weak), due to the monotonicity of the ICP translations. For $d \ge 2$, the inequality means, informally, that as the ICP is rambling around, the path $\pi$ that it traces does not get too close to itself. In particular, if each $\Delta t_i$ is of length at least $\delta$ then, after $k$ steps, the distance between the initial and final endpoints of the ICP path is at least $\delta\sqrt{k}$. This also holds for any pair of intermediate translations, $k$ apart in the order.
(2) Specializing Remark (1) to the case $k = 1$, we obtain $\|\Delta t_1\|^2 \le \mathrm{RMS}(t_0) - \mathrm{RMS}(t_1) \le \|t_2\|^2 - \|\Delta t_2\|^2$. This provides an alternative proof that the angle between $\Delta t_1$ and $\Delta t_2$ is non-acute. Moreover, the closer this angle is to $\pi/2$ the sharper is the estimate on the decrease in the RMS function.

## 4.3 The ICP Algorithm on the Line under the RMS Measure

In this section we consider the special case $d = 1$, and analyze the performance of the ICP algorithm on the line under the RMS measure. Theorem 4.1.1 implies that in this case the



$a_1 = -n - \delta(n-1)$
$a_2 = -\frac{1}{2} + \frac{1}{n} + \delta$
$a_n = \frac{1}{2} - \frac{1}{n} + \delta$
$b_1 = 0$
$\frac{1}{2}$
$b_2 = 1$

Figure 4.3: The lower bound construction. Only the two leftmost cells of $\mathcal{V}(B)$ are depicted.

number of NNA's, and thus the number of iterations of the algorithm, is $O(mn)$. We show below that the number of iterations can be superlinear in the worst case:

**Theorem 4.3.1** *There exist point sets $A$, $B$ on the real line of arbitrarily large common size $n$, for which the number of iterations of the ICP algorithm, under the* RMS *measure, is $\Theta(n \log n)$.*

**Proof:** We construct two point sets $A$, $B$ on the real line, where $|A| = |B| = n$. The set $A$ consists of the points $a_1 < \cdots < a_n$, where $a_1 = -n - \delta(n-1)$, $a_i = \frac{2(i-1)-n}{2n} + \delta$, for $i = 2, \ldots, n$, and $\delta = o\left(\frac{1}{n}\right)$ is some sufficiently small parameter. The set $B$ consists of the points $b_i = i - 1$, for $i = 1, \ldots, n$. See Figure 4.3.

Initially, all the points of $A$ are assigned to $b_1$. As the algorithm progresses, it keeps translating $A$ to the right. The first translation satisfies

$$\Delta t_1 = \frac{1}{n} \sum_{i=1}^{n} (b_1 - a_i) = \frac{1}{n}(b_1 - a_1) - \frac{n-1}{n}\delta = 1,$$

which implies that after the first iteration of the algorithm all the points of $A$, except for its leftmost point, are assigned to $b_2$. Using (4.1), we have $\Delta t_2 = \frac{1}{n}\sum_{i=1}^{n-1}(b_2 - b_1) = \frac{n-1}{n}$, which implies that the $n-1$ rightmost points of $A$ move to the next Voronoi cell $\mathcal{V}(b_3)$ after the second iteration, so that the distance between the new position of $a_n$ from the right boundary of $\mathcal{V}(b_3)$ is $\frac{2}{n} - \delta$, and the distance between the new position of $a_2$ and the left boundary of $\mathcal{V}(b_3)$ is $\delta$, as is easily verified.

In the next iteration $\Delta t_3 = \frac{n-1}{n}$ (arguing as above). However, due to the current position of the points of $A$ in $\mathcal{V}(b_3)$, only the $n-2$ rightmost points of $A$ cross the right Voronoi boundary of $\mathcal{V}(b_3)$ (into $\mathcal{V}(b_4)$), and the nearest neighbor of $a_2$ remains unchanged (equal to $b_3$).

We next show, using induction on the number of Voronoi cells the points of $A$ have crossed so far, the following property. Assume that the points of $A$, except for the leftmost one, are assigned to $b_{n-j+1}$ and $b_{n-j+2}$, for some $1 \le j \le n$ (clearly, these assignments can involve only two consecutive Voronoi cells), and consider all iterations of the algorithm, in which some points of $A$ cross the common Voronoi boundary $\beta_{n-j+1}$ of the cells $\mathcal{V}(b_{n-j+1})$,

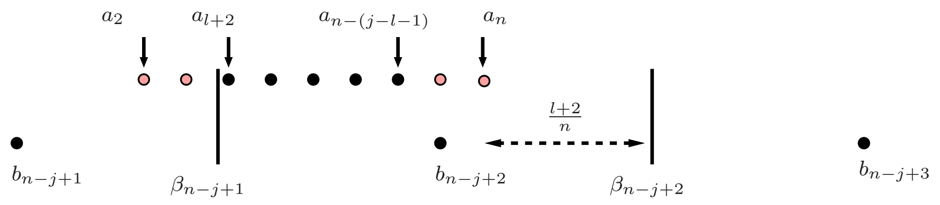


Figure 4.4: At the last iteration of round $j$, after shifting the points of $A$ by $\Delta t = \frac{j}{n}$ to the right, the points $a_{l+2}, \ldots, a_{n-(j-l-1)}$ (represented in the figure as black bullets) still remain in $\mathcal{V}(b_{n-j+2})$.

$\mathcal{V}(b_{n-j+2})$. We call the sequence of these iterations *round* $j$ of the algorithm. Then, (i) at each such iteration the relative translation is $\frac{j}{n}$, (ii) at each iteration in this round, other than the last one, the overall number of points of $A$ that cross $\beta_{n-j+1}$ is exactly $j$, and no point crosses any other boundary, and (iii) at the last iteration of the round, the overall number of points of $A$ that cross either $\beta_{n-j+1}$ or $\beta_{n-j+2}$ is exactly $j-1$. In fact, in the induction step we assume that properties (i), (ii) hold, and then show that property (iii) follows, for $j$, and that (i) and (ii) hold for $j-1$.

To prove this property, we first note, using (4.1), that the relative translation at each iteration of the algorithm is $\frac{k}{n}$, for some integer $1 \le k \le n$. The preceding discussion shows that the induction hypothesis holds for $j = n$ and $j = n-1$. Suppose that it holds for all $j' \ge j$, for some $2 \le j \le n-1$, and consider round $j-1$ of the algorithm, during which points of $A$ cross $\beta_{n-j+2}$ (that is, we consider all iterations with that property). Thus, at each iteration of round $j$ (except for the last one), in which there are points of $A$ that remain in the cell $\mathcal{V}(b_{n-j+1})$, the $j$ rightmost points of $A$ (among those contained in $\mathcal{V}(b_{n-j+1})$) cross $\beta_{n-j+1}$. Let us now consider the last such iteration. In this case, all the points of $A$, except $l$ of them, for some $0 \le l < j$ (and the leftmost point, which we ignore), have crossed $\beta_{n-j+1}$ in previous iterations. The key observation is that the distance from the current position of $a_n$ to the next Voronoi boundary $\beta_{n-j+2}$ is $\frac{l+2}{n} - \delta$ (this follows since we shift in total $n-1$ points of $A$ that are equally spaced apart by $\frac{1}{n}$), and since the next translation $\Delta t$ satisfies $\Delta t = \frac{j}{n}$ (using the induction hypothesis and (4.1)), it follows that only $j-1$ points of $A$ cross a Voronoi boundary in the next iteration. Moreover, the points $a_2, \ldots, a_{l+1}$ cross the boundary $\beta_{n-j+1}$, and the points $a_{n-(j-l-2)}, \ldots, a_n$ cross the boundary $\beta_{n-j+2}$ (this is the first move in which this boundary is crossed at all); see Figure 4.4 for an illustration.

Thus, at the next iteration, since only $j-1$ points have just crossed between Voronoi cells, (4.1) implies that the next translation is $\frac{j-1}{n}$, and, as is easily verified, at each further iteration, as long as there are at least $j-1$ points of $A$ to the left of $\beta_{n-j+2}$, this property must continue to hold, and thus $j-1$ points will cross $\beta_{n-j+2}$. This establishes the induction step.

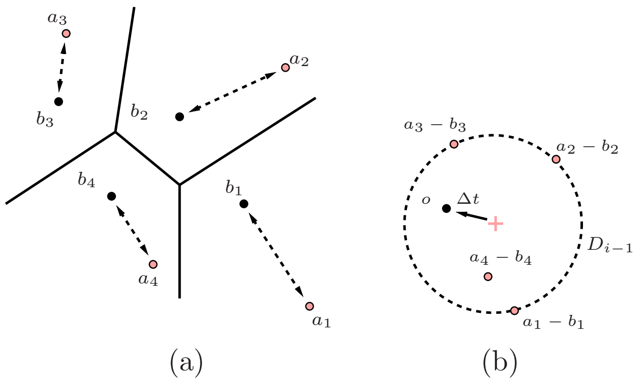


Figure 4.5: Proof of Lemma 4.4.2.

.

It now follows, using the above properties, that the number of iterations required for all the points of $A$ to cross $\beta_{n-j+1}$ is $\lceil \frac{n}{j} \rceil$, where in the first (last) such iteration some of the points may cross $\beta_{n-j}$ ($\beta_{n-j+2}$) as well. This implies that the number of such iterations, in which the points of $A$ cross only $\beta_{n-j+1}$ (and none of the two neighboring Voronoi boundaries), is at least $\left\lceil \frac{n}{j} \right\rceil - 2$ (but not more than $\left\lceil \frac{n}{j} \right\rceil$). Thus the overall number of iterations of the algorithm is $\Theta\left(\sum_{j=1}^{n} \left\lceil \frac{n}{j} \right\rceil\right) = \Theta(n \log n)$. $\square$

## 4.4 The Problem Under the Hausdorff Distance Measure

### 4.4.1 General Structural Properties of the ICP Algorithm

**Lemma 4.4.1** *The ICP algorithm converges under the (one-sided) Hausdorff distance measure in at most $O(m^d n^d)$ steps.*

**Proof**: At each iteration $i$, we compute $\Delta t_i$ that minimizes $\max_{a \in A} \|a + t_{i-1} + \Delta t_i - N_B(a + t_{i-1})\|$. Since $\|a + t_i - N_B(a + t_i)\| \le \|a + t_i - N_B(a + t_{i-1})\|$, for each $a \in A$, the cost function decreases after each iteration (the algorithm terminates if there is no decrease). The lemma then follows from Corollary 4.1.2. $\square$

The following lemma provides a simple tool to compute the relative translations that the algorithm executes.

**Lemma 4.4.2** *Let $D_{i-1}$ be the smallest enclosing ball of the points $\{a + t_{i-1} - N_B(a + t_{i-1}) \mid a \in A\}$. Then the next relative translation $\Delta t_i$ of the ICP algorithm is the vector from the center of $D_{i-1}$ to the origin.*

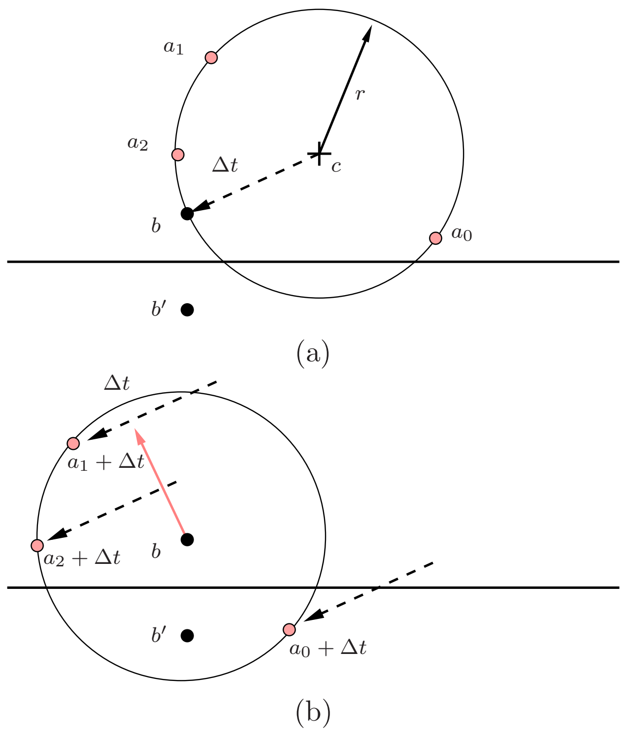


Figure 4.6: Proof of Lemma 4.4.3. The point $b$ is placed at the origin, the center of the minimum enclosing disc of the points $a_0$, $a_1$, $a_2$ is $c$, and its radius is $r$. Initially, $\|a_0 - b\| = \max \|a_i - b\| > r$, for $i = 0, 1, 2$ (a), and after translating by $\Delta t$, $\|a_0 + \Delta t - b'\| < r$ (b).

**Proof**: The proof follows from the (easy) observation that since $D_{i-1}$ is a minimum enclosing ball, all points appearing on its boundary are not contained in the same halfspace bounded by a hyperplane that passes through its center, and thus any further infinitesimal translation of the points $a + t_{i-1} + \Delta t_i$, for $a \in A$, from their current position causes at least one of the points on the boundary of (the translated ball) $D_{i-1} + \Delta t_i$ to get further from the origin (which is also the center of $D_{i-1} + \Delta t_i$). Therefore the Hausdorff distance measure is minimized (with respect to the above fixed NNA) after translating by $\Delta t_i$. Note that it follows by definition that the cost obtained after the relative translation by $\Delta t_i$ is smaller than (or equal to) the radius of $D_{i-1}$ (it may become strictly smaller, when the NNA changes after translating by $\Delta t_i$). See Figure 4.5 for an illustration. $\square$

In contrast with Theorem 4.2.2, we have:

**Lemma 4.4.3** *In any dimension $d$, there exist finite point sets $A$, $B$ with the following property. Define the cost function $H(t) = \max_{a \in A} \|a + t - N_B(a + t)\|$. Then $H(t_0 + \xi \Delta t)$, for $\xi \in [0, 1]$, is not monotonically decreasing along the relative translation vector $\Delta t$ that the algorithm executes from translation $t_0$.*

**Proof**: A planar example (which can be lifted to any dimension $d \geq 3$) is depicted in Figure 4.6. Initially, all three points $a_0$, $a_1$, $a_2$, are closer to $b$. By Lemma 4.4.2, the translation $\Delta t$ moves the center $c$ of the circumcircle of $\Delta a_0 a_1 a_2$ to $b$, so the final distance of all three $a_i$'s from $b$ is equal to the radius $r$ of this circle. As we translate each of them by $\Delta t$, $a_0$ crosses into $\mathcal{V}(b')$, its distance to its nearest neighbor (first $b$ and then $b'$) keeps decreasing, and its final value is strictly smaller than $r$. In contrast, the distances of $a_1$, $a_2$ from $b$ (their nearest neighbor throughout the translation) both *increase* towards the end of the translation, and their final values are both $r$. Hence, towards the end of the translation $H(t_0 + \xi\Delta t)$ is *increasing.* □

**Remark:** We do not know whether non-monotonicity can arise at *any* step of the algorithm. Perhaps only the first step might have this property.

**Lemma 4.4.4** *Let $H(t)$ be as above. At each iteration $i \geq 1$ of the algorithm*

$$H(t_{i-1})^2 - H(t_i)^2 \geq \|\Delta t_i\|^2.$$

**Proof**: Using Lemma 4.4.2, the next relative translation $\Delta t_i$ is the vector $c_{i-1}o$, where $c_{i-1}$ is the center of the minimum enclosing ball $D_{i-1}$ of the set $A^* = \{a + t_{i-1} - N_B(a + t_{i-1}) \mid a \in A\}$, and $o$ is the origin.

The argument in the proof of Lemma 4.4.2 implies that the cost $H(t_i)$ (obtained after the relative translation by $\Delta t_i$) is smaller than or equal to the radius of $D_{i-1}$. (It is equal the radius under the former NNA, and can only become smaller under the new NNA, after the translation.) Let $A_0^*$ denote the set of all points $a^* \in A^*$ that appear on $\partial D_{i-1}$, and let $a_0^*$ be the point of $A_0^*$ farthest from the origin.

As above, since $D_{i-1}$ is a minimum enclosing ball, it follows that all points of $A_0^*$ cannot be contained in the same halfspace bounded by a hyperplane through $c_{i-1}$, which, in particular, implies that $a_0^*$ and $o$ are separated by the hyperplane $\lambda$, perpendicular to the segment connecting $c_{i-1}$ and $o$, and passing through $c_{i-1}$; see Figure 4.7 for an illustration. Clearly, the cost $H(t_{i-1})$ is at least $\|a_0^*\|$ (the maximum distance may be obtained by another point of $A^*$ that lies in the interior of $D_{i-1}$). Hence the angle $\measuredangle a_0^* c_{i-1} o$ is at least $\pi/2$, and thus

$$H(t_{i-1})^2 - H(t_i)^2 \geq \|a_0^*\|^2 - \|a_0^* - c_{i-1}\|^2 \geq \|c_{i-1} - o\|^2 = \|\Delta t_i\|^2.$$

□

Using the Cauchy-Schwarz inequality, we obtain the following corollary, which is the analogue of Corollary 4.2.7:

**Corollary 4.4.5** *If the relative translations computed by the algorithm are $\Delta t_1, \ldots, \Delta t_k$, then*

$$\frac{1}{k}\left(\sum_{i=1}^{k} \|\Delta t_i\|\right)^2 \leq \sum_{i=1}^{k} \|\Delta t_i\|^2 \leq H(t_0)^2 - H(t_k)^2. \tag{4.5}$$

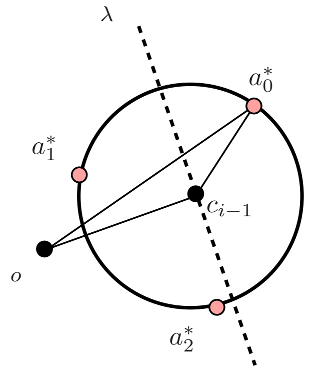


Figure 4.7: The angle $\measuredangle a_0^* c_{i-1} o$ is obtuse.

### 4.4.2 The one-dimensional problem

Let $A$, $B$ be two point sets on the real line, with $|A| = m$, $|B| = n$.

**Lemma 4.4.6 (Monotonicity)** *The points of $A$ are always translated in the same direction, over all iterations of the algorithm. That is, either $\Delta t_i \geq 0$ for each $i \geq 1$, or $\Delta t_i \leq 0$ for each $i \geq 1$.*

**Proof**: Let $a^* \in A$, $b^* = N_B(a^*)$, be the pair (which is unique if we assume initial general position[1]) that satisfies initially $\xi = |b^* - a^*| = \max_{a \in A} |N_B(a) - a|$. Suppose without loss of generality that $a^* < b^*$. By Lemma 4.4.2, the initial "ball" (i.e., interval) $D_0$ has $a^* - b^* = -\xi$ as its left endpoint, and its right endpoint is smaller than $\xi$ (otherwise, the algorithm terminates; following the observations of Lemma 4.4.2, the next relative translation is $\overrightarrow{0}$). Hence the center (midpoint) of $D_0$ is *negative*, so the first translation $\Delta t_1$ of the algorithm is to the right. See Figure 4.8.

After translating, $a^* + \Delta t_1$ is still to the left of $b^*$ (since $\Delta t_1 < \xi$) and is *closer* to $b^*$, so $b^*$ is still the nearest neighbor of $a^* + \Delta t_1$, and $|a^* + \Delta t_1 - b^*| = \max_{a \in A}\{|a + \Delta t_1 - N_B(a)|\} \geq \max_{a \in A}\{|a + \Delta t_1 - N_B(a + \Delta t_1)|\}$, since right after the translation by $\Delta t_1$, the left and the right endpoints of $D_0$ are at the same distance from the origin, but then the reassignment may modify the right endpoint of $D_0$. Thus $a^* + \Delta t_1 - b^*$ is still the left endpoint of the new interval $D_1$, whose right endpoint is *closer* to the origin (or at the same distance, in which case the algorithm terminates). Hence, the preceding argument implies that $\Delta t_2$ will also be to the right, and, using induction, the lemma follows. □

**Remarks:** (1) The proof implies that the pair $a^*$, $b^*$, which attains the maximum value of the cost function at the initial position of $A$ continues to do so over *all* iterations of the

[1]If there are several such pairs then either (i) some of the differences $b^* - a^*$ are positive and some are negative, and then the algorithm terminates right away, or (ii) all the differences $b^* - a^*$ have the same sign, and then the same argument given in the proof continues to apply.

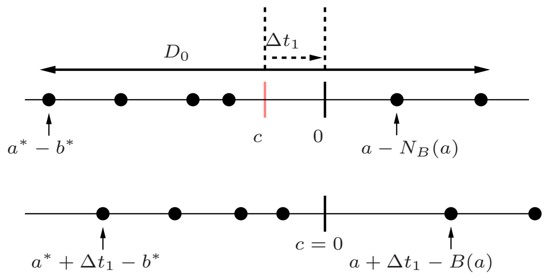


Figure 4.8: Proof of Lemma 4.4.6. The points $a - N_B(a)$, for $a \in A$, before translating by $\Delta t_1$ (top), and after the translation (bottom).

algorithm. The point $a^*$ gets closer to $b^*$, and can never exit its cell $\mathcal{V}(b^*)$ (actually, it never passes over $b^*$).
(2) The relative translation $\Delta t_i$ is always determined by $a^*$, $b^*$, and by another pair of points $a'$, $b'$, which determine the other endpoint of $D_{i-1}$. Note that in the next iteration $N_B(a')$ must change, or else the algorithm terminates.
(3) While monotonicity holds in $\mathbb{R}^1$, we do not know (in view of Lemma 4.4.3) whether the analog of Theorem 4.2.3 holds for the Hausdorff distance measure in two (and higher) dimensions.

Recall that the spread of a point set $P$ is the ratio between the diameter of $P$ and the distance between its closest pair of points. Our main result on the ICP algorithm under the Hausdorff distance measure is given in the following theorem.

**Theorem 4.4.7** *Let $A$ and $B$ be two point sets on the real line, with $|A| = m$, $|B| = n$, and let $\delta_B$ be the spread of $B$. Then the number of iterations that the ICP algorithm executes is $O((m+n)\log \delta_B / \log n)$.*

**Proof**: Let the elements of $A$ be $a_1 < a_2 < \cdots < a_m$, and those of $B$ be $b_1 < b_2 < \cdots < b_n$. Put $\Delta_A = a_m - a_1$, $\Delta_B = b_n - b_1$. Assume, without loss of generality, that, initially, $\max_{a \in A} |N_B(a) - a| \le \Delta_A$ (otherwise, this is the case after the first translation), and that $b_1 - a_1 = \max_{a \in A} |N_B(a) - a|$ (in particular, $a_1 < b_1$). The initial interval $D_0$ (in the notation of Lemma 4.4.2) is $[a_1 - b_1, 0]$. As shown in Lemma 4.4.6, all translations will be to the right, and $a_1$ will stay to the left of $b_1$. Thus the overall length of all translations is at most $b_1 - a_1 \le \Delta_A$. Put $I_{k-1} = b_1 - (a_1 + t_{k-1})$, for each iteration $k \ge 1$ of the algorithm.

A relative translation $\Delta t_k$, computed at the $k$-th iteration of the algorithm, for $k \ge 0$, is said to be *short* if $\Delta t_k < \frac{I_{k-1}}{2n/\log n}$, otherwise, $\Delta t_k$ is *long*. We first claim that the overall number of (short and long) relative translations that the algorithm executes is $O\left(m \log\left(\frac{\Delta_A}{\Delta_B}\delta_B\right) / \log n\right)$.

We say that a pair $(a', b')$ of points, $a' \in A$, $b' \in B$, $a' \ne a_1$, is a *configuration* of the algorithm, if, at some iteration $k$, $a' - b'$ is the right endpoint of $D_{k-1}$ (so $(a_1, b_1)$,

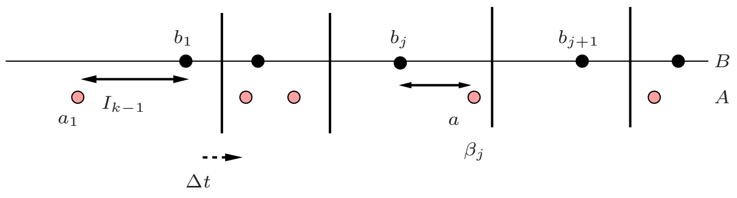


Figure 4.9: Proof of Theorem 4.4.7.

$(a', b')$ determine the $k$-th relative translation of the algorithm). Due to monotonicity, each configuration can arise at most once, and thus an upper bound on the overall number of such configurations also applies to the actual number of iterations performed by the algorithm.

The idea of the proof is as follows. The overall number of long relative translations is relatively small, since, after performing each of them, the distance between $b_1$ and the translated copy of $a_1$, which measures the cost function, significantly decreases. As to the number of short relative translations, if there are at least two configurations involving the *same* point $a' \neq a_1$ in $A$, which determine short relative translations, then the cost function must significantly decrease (since $a'$ has changed its nearest neighbor, and becomes significantly further from its previous nearest neighbor), and, as a result, each such point $a'$ cannot be involved in too many configurations that determine short relative translations.

Let $\mathcal{S}$ be the sequence of all configurations produced by the algorithm (sorted by the "chronological" order of their creation), which determine short relative translations. We next bound the number of *a-configurations* in $\mathcal{S}$, namely, those that involve the same point $a \in A$.

Fix some $a \neq a_1 \in A$. Let $(a, b_j)$, $(a, b_l)$, $1 \leq j \neq l \leq n$, be two consecutive $a$-configurations in $\mathcal{S}$, so each configuration that appears between $(a, b_j)$, $(a, b_l)$ does not involve $a$. Due to the monotonicity of the relative translations, we must have $j < l$. Suppose that $(a, b_j)$ arises at the $k$-th iteration, and $(a, b_l)$ arises at the $k'$-th iteration $(k' > k)$. Since $(a, b_j)$ determines a short relative translation, (the translated copy of) $a$ must lie to the right of $b_j$ before the $k$-th step, for otherwise $\Delta t_k$ would be at least $\frac{I_{k-1}}{2}$, and thus would not be short. Furthermore, we have, by construction,

$$\Delta t_k = \frac{1}{2}\left(I_{k-1} + (b_j - (a + t_{k-1}))\right) < \frac{I_{k-1}}{2n/\log n},$$

and thus

$$|b_j - (a + t_{k-1})| \geq I_{k-1} - \frac{I_{k-1}}{n/\log n}.$$

Since $a + t_{k-1} \in \mathcal{V}(b_j)$, we have

$$|b_{j+1} - (a + t_{k-1})| \geq |b_j - (a + t_{k-1})| \geq I_{k-1} - \frac{I_{k-1}}{n/\log n}.$$

Thus $a$ can pass over $b_{j+1}$ only if we further translate it by at least $I_{k-1} - \frac{I_{k-1}}{n/\log n}$; see Figure 4.9 for an illustration. Since $(a, b_l)$ determines a short relative translation at the $k'$-th iteration (and thus $a$ lies to the right of $b_l$ at that time), it follows that $\sum_{r=k}^{k'-1} \Delta t_r > I_{k-1} - \frac{I_{k-1}}{n/\log n}$. But then, $|b_1 - (a_1 + t_{k'-1})| < \frac{I_{k-1}}{n/\log n}$. Thus the cost function is reduced by a factor of at least $n/\log n$ between each two consecutive configurations of $\mathcal{S}$ that involve the same point $a \neq a_1$ of $A$.

We now show that the overall number of such configurations is $O\left(\log\left(\frac{\Delta_A}{\Delta_B}\delta_B\right)/\log n\right)$, for a fixed point $a \neq a_1 \in A$. Let $C_B$ be the distance between the closest pair in $B$; that is, $C_B = \frac{\Delta_B}{\delta_B}$. We claim that when $I_{k-1}$ becomes smaller than $\frac{C_B}{4}$ (at some iteration $k \geq 1$), the algorithm terminates. Indeed, since $I_{k-1} = \max_{a\in A}\{|N_B(a+t_{k-1})-(a+t_{k-1})|\}$, this implies that the next relative translation satisfies $|\Delta t_k| < \frac{C_B}{4}$. On the other hand, the distance between each (translated) point $a + t_{k-1}$, $a \in A$, to its nearest Voronoi boundary is at least $\frac{C_B}{4}$ (since the distance between any $b \in B$ and the (left or right) boundary of its Voronoi cell $\mathcal{V}(b)$ is at least $\frac{C_B}{2}$), and thus, after shifting the points by $\Delta t_k$, the nearest-neighbor assignments do not change. This easily implies that the overall number of iterations, in which $I_0$ is reduced by a factor of at least $n/\log n$ until it becomes smaller than $\frac{C_B}{4}$, is

$$O(\log_n \Delta_A - \log_n C_B) = O\left(\log\left(\frac{\Delta_A}{\Delta_B}\delta_B\right)/\log n\right),$$

as asserted. Thus the overall number of iterations of the algorithm that involve short relative translations, over all points of $A$, is $O\left(m\log\left(\frac{\Delta_A}{\Delta_B}\delta_B\right)/\log n\right)$.

We next show that the overall number of long relative translations is $O\left(n\log\left(\frac{\Delta_A}{\Delta_B}\delta_B\right)/\log n\right)$. A long relative translation $\Delta t_k$ reduces $I_{k-1}$ by a factor of at least $1 - \frac{1}{2n/\log n}$, so if $j$ long relative translations occur before the $k$-th iteration then $I_{k-1} \leq \Delta_A\left(1 - \frac{1}{2n/\log n}\right)^j$. Arguing as above, and, using the fact that $(1-x) < e^{-x}$, for $0 < x < 1$, the largest value of $j$ for which $\Delta_A\left(1 - \frac{\log n}{2n}\right)^j \geq \frac{C_B}{4}$ satisfies $j = O\left(n\log\left(\frac{\Delta_A}{\Delta_B}\delta_B\right)/\log n\right)$.

In order to remove the factor $\log\frac{\Delta_A}{\Delta_B}$ from the bound, we argue that when $\Delta_A \geq 5\Delta_B$, the algorithm terminates after at most two iterations. Indeed, after the first iteration of the algorithm, the next relative translation is determined by $(a_1, b_1)$, $(a_m, b_n)$, and these two pairs of points maintain this property in any further iteration, so the algorithm will terminate at the next iteration, as claimed. Hence, the actual bound on the overall number of iterations is $O\left((m+n)\log\delta_B/\log n\right)$, which completes the proof of the theorem. $\Box$

**Corollary 4.4.8** *The number of iterations of the ICP algorithm is $O(m+n)$ when the spread of the point set $B$ is polynomial in $n$, where the constant of proportionality is linear in the degree of that polynomial bound.*

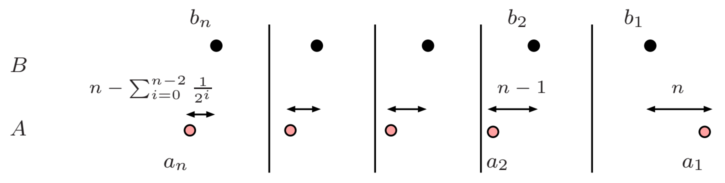


Figure 4.10: The lower bound construction.

Our second main result of this section is a matching linear lower bound construction, for the case where the spread of $B$ is linear in $n$.

**Theorem 4.4.9** *There exist point sets $A$, $B$ of arbitrarily large common size $n$, such that the spread of $B$ is linear, and the number of iterations of the algorithm is $\Theta(n)$.*

**Proof**: We construct two point sets $A$, $B$ on the real line, with $|A| = |B| = n$. For simplicity of the analysis, we implicitly define the two point sets by the following relations:

(1) $a_1 = 0$,

(2) $a_1 - b_1 = n$,

(3) $a_j - b_j = -\left(n - \sum_{k=0}^{j-2} \frac{1}{2^k}\right)$, for each $2 \le j \le n$,

(4) $a_1 - \frac{b_1+b_2}{2} = 2n$, $a_j - \frac{b_j+b_{j+1}}{2} = \sum_{k=1}^{j-1} \frac{1}{2^k} - \varepsilon$, for each $2 \le j \le n-1$, where $\varepsilon = o\left(\frac{1}{2^n}\right)$.

It is easy to verify that the above conditions determine uniquely the sets $A$ and $B$, and that $2(n-1) < |b_j - b_{j+1}| \le 2n$, for each $j = 1, \ldots, n-1$, and thus the spread of $B$ is $O(n)$. Note that in this construction each point $a_j \in A$ is initially located in the respective Voronoi cell $\mathcal{V}(b_j)$, for $j = 1, \ldots, n$; see Figure 4.10 for an illustration. (Note that in this notation the points are indexed in increasing order from right to left.)

We now claim, using induction on the number of iterations of the algorithm, that the relative translation at the $i$-th iteration $\Delta t_i$ is $-\frac{1}{2^i}$, for $i = 1, \ldots, n-2$. As a consequence, each point $a_j \in A$ progresses to the left towards $\mathcal{V}(b_{j+1})$, and, in particular, $a_{i+1}$ crosses, at the $i$-th iteration, the Voronoi boundary common to $\mathcal{V}(b_{i+1})$ and $\mathcal{V}(b_{i+2})$, as follows easily from property (4). In addition, all the remaining nearest neighbors remain the same at that iteration, and the nearest neighbor of $a_{i+1}$ remains $b_{i+2}$ at any subsequent iteration — see below. This would imply that the overall number of iterations is $n-2$, which establishes our bound.

The pair $a_1$, $b_1$ satisfies $b_1 = N_B(a_1)$ and $|a_1 - b_1| = \max_{a \in A} |a - N_B(a)|$, as is easily verified, and, by Lemma 4.4.6, this pair attains the maximum value of the cost function at every subsequent iteration of the algorithm. Thus at the first iteration of the algorithm $a_1 - b_1$ is the right endpoint of the interval $D_0$, and $a_2 - b_2$ is its left endpoint. Hence

$$\Delta t_1 = \frac{(b_1 - a_1) + (b_2 - a_2)}{2} = -\frac{1}{2},$$

and, as a consequence, all the points of $A$ move to the left. Moreover, due to property (4) of the construction, the nearest neighbor of $a_2$ becomes $b_3$, and the nearest neighbors of all the remaining points do not change. Suppose now, for the induction hypothesis, that at the $(i-1)$-th iteration $\Delta t_{i-1} = -\frac{1}{2^{i-1}}$, and, as a consequence, the overall translation so far $t_{i-1}$ is $-\sum_{j=1}^{i-1} \frac{1}{2^j}$. It can be easily verified, using property (4), that the current nearest neighbor of each point $a_j$, $j = 2, \ldots, i$, is now $b_{j+1}$, and that $a_j$ is located to the right of $b_{j+1}$. We next claim, using properties (3) and (4), that each of these points satisfies

$$a_j + t_{i-1} - b_{j+1} = n - 2\varepsilon - \sum_{k=1}^{i-1} \frac{1}{2^k} < a_1 + t_{i-1} - b_1 = n - \sum_{k=1}^{i-1} \frac{1}{2^k}. \qquad (4.6)$$

In addition, due to property (3),

$$a_{i+1} + t_{i-1} - b_{i+1} = -(n-1) < a_j + t_{i-1} - b_j,$$

for each $j = i+2, \ldots, n$. That is, $a_{i+1} + t_{i-1} - b_{i+1}$ is the left endpoint of the interval $D_{i-1}$. Thus, at the $i$-th step we have

$$\Delta t_i = \frac{(b_1 - (a_1 + t_{i-1})) + (b_{i+1} - (a_{i+1} + t_{i-1}))}{2} = -\frac{1 - \sum_{k=1}^{i-1} \frac{1}{2^k}}{2} = -\frac{1}{2^i},$$

as asserted, which, using property (4), implies that the new nearest neighbor of $a_{i+1}$ is $b_{i+2}$. Note that it can be easily verified, using (4.6) and properties (3), (4), that all the remaining points remain in their previous cells, and, in particular, that none of the points $a_j$, for $j = 2, \ldots, i$ can exit the cell $\mathcal{V}(b_{j+1})$ in any further iteration (since the overall translation length is less than 1). This completes the induction step. Note that, the nearest neighbors of the points $a_1$, $a_n$ do not change during the execution of the algorithm, and thus the overall number of iterations is $n-2$, as asserted. $\square$

**Remark:** In the above construction, the number of bits that is required in order to represent each input point is $\Theta(n)$. We are not aware of any construction in which this number is $O(\log n)$ and the number of iterations is $\Omega(n)$. We would therefore like to conjecture that in the latter case the overall number of iterations that the algorithm performs is sublinear.

## 4.5 Concluding Remarks

One major open problem that this study raises is to improve the upper bound, or, alternatively, present a tight lower bound construction, on the number of iterations performed by the algorithm under each of the above measures. In a further study, this problem was settled by Arthur and Vassilvitskii for the one-dimensional case [30], who showed a lower bound construction of $\Omega(n^{d+1})$ iterations in any dimension $d \geq 1$ (where the constant of proportionality depends on $d$). Still, it leaves a substantial gap between our upper bound and the above lower bound, for $d \geq 2$.

Another problem concerns the running time of the algorithm. The algorithm has to reassign the points in $A$ to their (new) nearest neighbors in $B$ at each iteration. This can be done by searching with each point of $A$ in $\mathcal{V}(B)$, but this will take time that is more than linear in $m$ for each iteration. Thus, for points on the real line, when the number of iterations is linear or super-linear, we face a super-quadratic running time. The irony is that we can solve the pattern matching problem (for the RMS measure) directly, without using the ICP algorithm, in $O(mn \log m)$ time, as follows. (i) Compute the overlay $M(A, B)$ of the Voronoi diagrams $\mathcal{V}(B - a)$, for $a \in A$, in $O(mn \log m)$ time. (ii) Process the intervals of $M(A, B)$ from left to right. (iii) For each interval $I$, compute the corresponding NNA by updating, in $O(1)$ time, the NNA of the previous interval (only one point changes its nearest neighbor). (iv) Obtain, in $O(1)$ time, the minimizing translation of this NNA, using (4.2) for the leftmost interval, and (4.1) for any subsequent interval, and the corresponding value of the cost function. (v) Collect those $I$ for which the minimizing translation lies in $I$; these are the *local minima* of the cost function. (vi) Output the global minimum from among those minima. The problem can also be solved for the Hausdorff distance measure in $O(mn \log m)$ time, by computing the upper envelope of the $m$ Voronoi surfaces $\mathcal{S}(B - a)$, for $a \in A$, and reporting its global minimum (see, e.g., [124]).

Of course, in practice the ICP algorithm tends to perform much fewer steps, so it performs much faster than this worst case bound. We remark that a variant of the preceding algorithm (for points on the real line) can be employed in the ICP algorithm, so that the overall cost of updating the NNA's remains $O(mn \log n)$, regardless of how many iterations it performs. Many interesting open problems arise in this connection, such as finding a faster procedure to handle the NNA updates, analyzing the performance under the Hausdorff distance and in higher dimensions, and so on.

Moreover, inspired by a comment of D. Kozlow, if we contend ourselves with finding a *local minimum* of the cost function, this can be found in nearly-linear time, using binary search over the intervals of $M(A, B)$, which we keep *implicit*. Specifically, we proceed as follow. At each step of the search, there are three previously tested translations $t_1 < t_2 < t_3$, for which we have computed the corresponding values $\mathrm{RMS}(t_i)$, and the indices $j_i$ (numbering from left to right) of the intervals of $M(A, B)$ containing $t_i$, for $i = 1, 2, 3$, and the translation we are looking for lies in the interval $[t_1, t_3]$. We inductively assume that $\mathrm{RMS}(t_2) \leq \min\{\mathrm{RMS}(t_1), \mathrm{RMS}(t_3)\}$. Assume, without loss of generality, that $j_2 - j_1 \geq j_3 - j_2$. We compute $j^- = \lfloor (j_1 + j_2)/2 \rfloor$, and find a point $t^-$ in the $j^-$-th interval of $M(A, B)$. By definition, this is a point $t^-$ that has exactly $j^-$ Voronoi boundaries to its left, that is, exactly $j^-$ differences of the form $\frac{b_i + b_{i+1}}{2} - a_l$ are smaller than $t^-$. Finding such a $t^-$ is a special case of the *slope selection* problem (see [54]), and can thus be solved in $O((m + n)\log(m+n))$ time. We now compute $\mathrm{RMS}(t^-)$. If $\mathrm{RMS}(t^-) \le \mathrm{RMS}(t_2)$, we continue the search with the triple $(t_1, t^-, t_2)$; otherwise, we continue the search with $(t^-, t_2, t_3)$. Clearly, the process converges after logarithmically many steps, to a local minimum of $\mathrm{RMS}(T)$, in overall time $O((m+n)\log^2(m+n)$ time.

Clearly, one expects the algorithm to converge faster (say, under the RMS measure) when the initial placement of $A$ is sufficiently close to $B$, in the sense that $\mathrm{RMS}(t_0)$ is small. Attempts to exploit such heuristics in practice are reported in [78, 120]. It would be interesting to quantify this "belief", and show that when $\mathrm{RMS}(t_0)$ is smaller than some threshold that depends on the layout of $B$, the algorithm converges after very few iterations.

Finally, we note that recent variants of the ICP technique [78, 120] cater to situations where the point sets $A$ and $B$ are samples of points on two respective curves (or surfaces) $\gamma_A$, $\gamma_B$. Then each point of $A$ finds its nearest neighbor along $\gamma_B$ (rather than in $B$), using some polygonal (or polyhedral) approximation of $\gamma_B$. This tends to speed up the algorithm in practice, as reported e.g. in [78, 120]. It would be interesting to extend the worst-case analysis of this study to this scenario.

# Chapter 5

# A Single Cell in an Arrangement of Convex Polyhedra in $\mathbb{R}^3$

In this chapter we show that the combinatorial complexity of a single cell in an arrangement of $k$ convex polyhedra in 3-space, having $n$ facets in total, is $O(nk^{1+\varepsilon})$, for any $\varepsilon > 0$; the main analysis is presented in Section 5.1, where we also show that the number of "special quadrilaterals" on the unbounded cell of $\mathcal{A}(\mathcal{P})$ is $O(nk)$, which is one of the main ingredients of the derivation of the above bound on the combinatorial complexity of the unbounded cell of $\mathcal{A}(\mathcal{P})$. We then present an extension of our main result, and show that the overall complexity of the zone of an algebraic surface patch of constant description complexity, or the boundary of an arbitrary convex set in 3-space, in an arrangement of $k$ convex polyhedra in 3-space with $n$ facets in total is also $O(nk^{1+\varepsilon})$, for any $\varepsilon > 0$. In Section 5.2 we present a deterministic algorithm (based on a divide-and-conquer scheme) that constructs a single cell of such an arrangement in time $O(nk^{1+\varepsilon}\log^3 n)$, for any $\varepsilon > 0$. We give concluding remarks and suggestions for further research in Section 5.3.

## 5.1 The Complexity of a Single Cell

**Preliminaries.** Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be a collection of $k$ convex polyhedra in 3-space having $n$ facets in total. For simplicity of the analysis, we assume that the given polyhedra are in *general position*. This excludes degenerate configurations, and allows us to assume that no four polyhedron boundaries meet at a common point, no vertex of one polyhedron lies on the boundary of another, no edges of two distinct polyhedra meet, and no edge of a polyhedron meets the polygonal curve of intersection of the surfaces of any two other polyhedra. As claimed in [27], this assumption involves no loss of generality, since we can slightly perturb the vertices of the given polyhedra, so as to move them to general position, and verify that the number of vertices, edges, and faces appearing on the boundary of a given cell $\mathcal{C}$ of $\mathcal{A}(\mathcal{P})$ does not decrease.

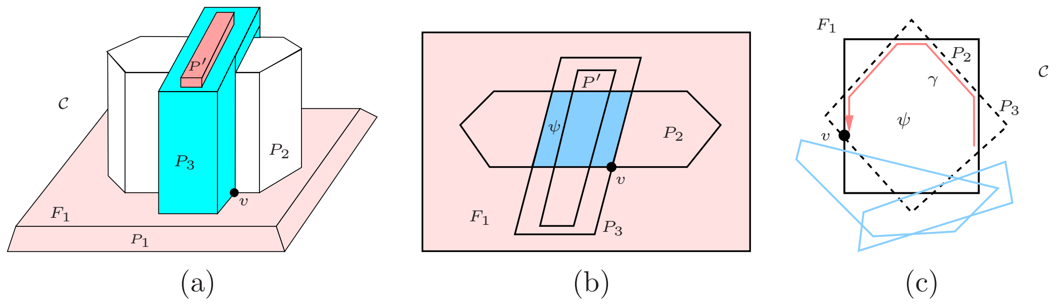


Figure 5.1: An inner vertex $v$. (a) and (b) The outer exterior of the four polyhedra $P_1$, $P_2$, $P_3$, $P'$, coincides with the (unbounded) cell $\mathcal{C}$. The polyhedra $P_1$, $P_2$, $P_3$ create a quadrilateral $\psi$, that lies on the facet $F_1$ of $P_1$ and is intersected by the fourth polyhedron $P'$. All four vertices of $\psi$ lie on the boundary of $\mathcal{C}$. (b) A cross-section through the facet $F_1$ of $P_1$, its intersections with $P_2$, $P_3$, $P'$, and the quadrilateral $\psi$. (c) The convex polygon $\psi$ is formed on the cross-section through the facet $F_1$ (represented by the plane of the figure) and the two polyhedra $P_2$ and $P_3$. As depicted in the figure, $\psi$ is an octagon, and three of its edges are intersected by the two bottom polyhedra. The five remaining edges form the exposed convex chain $\gamma$, all of whose six vertices appear on $\partial\mathcal{C}$.

We classify the vertices of $\mathcal{A}(\mathcal{P})$ as in [27]: An intersection vertex $v$ of $\mathcal{A}(\mathcal{P})$ (i.e., not a vertex of one of the polyhedra of $\mathcal{P}$) is said to be an *outer* vertex if it is the intersection of an edge of one polyhedron and the relative interior of a facet of another polyhedron. Otherwise $v$ is an *inner* vertex (that is, $v$ is the intersection of the relative interiors of three facets of three distinct polyhedra). The number of outer vertices in the entire arrangement $\mathcal{A}(\mathcal{P})$ is $O(nk)$, since each of the $O(n)$ edges of the polyhedra of $\mathcal{P}$ intersects at most two facets of any other polyhedron (see also [27]), so our main goal is to bound the number of inner vertices of any given (or, as in our analysis, the unbounded) cell $\mathcal{C}$ of $\mathcal{A}(\mathcal{P})$. We note that the total number of inner vertices of the entire arrangement $\mathcal{A}(\mathcal{P})$ is $O(nk^2)$; see [27] for the easy proof.

**Reducing to the case of the unbounded cell in an arrangement of bounded polyhedra.** In this section we show that the combinatorial complexity of $\mathcal{C}$ is $O(nk^{1+\varepsilon})$, for any $\varepsilon > 0$. In what follows, we assume that the polyhedra in $\mathcal{P}$ are *bounded* and that $\mathcal{C}$ is the *unbounded cell* of $\mathcal{A}(\mathcal{P})$. These assumptions involve no loss of generality, since the problem of bounding the combinatorial complexity of any single cell of $\mathcal{A}(\mathcal{P})$ can be reduced to the above case. Indeed, let $\Delta$ be a cell of $\mathcal{A}(\mathcal{P})$, and assume first that $\Delta$ lies in the complement of the union of the polyhedra in $\mathcal{P}$, and that the polyhedra are bounded. Let $h$ denote a plane that intersects the interior of $\Delta$ and does not pass through any vertex of $\Delta$; such a plane is easily seen to exist. We cut each polyhedron $P$ of $\mathcal{P}$ by $h$, and leave an arbitrarily small gap between the two resulting pieces of $P$, thereby splitting $P$ into two smaller polyhedra ($P$ remains unchanged if it misses $h$). Let us denote the resulting collection of these polyhedra by $\mathcal{P}'$. It is easy to verify from the construction that $\Delta$ becomes a portion of the unbounded cell $\mathcal{C}'$ in the new arrangement $\mathcal{A}(\mathcal{P}')$, and that the number of vertices, edges, and faces of $\mathcal{C}'$ is not smaller than the number of vertices, edges, and faces of the original cell $\Delta$. The new collection $\mathcal{P}'$ contains at most $2k$ polyhedra that have at most $2(n+k)$ facets in total. Thus our asserted asymptotic upper bound on the combinatorial complexity of the unbounded cell of $\mathcal{A}(\mathcal{P})$ also holds for any cell of $\mathcal{A}(\mathcal{P})$ (in the complement of the union of $\mathcal{P}$).

We next consider the case where the polyhedra in $\mathcal{P}$ may be unbounded. We note that the notion of the unbounded cell of $\mathcal{A}(\mathcal{P})$ may not be well defined in this case. However, we can transform the given polyhedra into a set of bounded polyhedra, by intersecting each unbounded polyhedron $P \in \mathcal{P}$ with a large box enclosing all vertices of $\mathcal{A}(\mathcal{P})$ in its interior, thereby obtaining a new set $\mathcal{P}''$ of $k$ bounded convex polyhedra with $O(n+k)$ facets in total. It is easy to see that any bounded cell in the original arrangement $\mathcal{A}(\mathcal{P})$ appears unchanged in the resulting arrangement $\mathcal{A}(\mathcal{P}'')$, and that any unbounded cell of $\mathcal{A}(\mathcal{P})$ is contained in the unbounded cell of $\mathcal{A}(\mathcal{P}'')$. In the latter case, the number of features of the new unbounded cell is not smaller that those of the original cell. (To keep the polyhedra in general position, we can intersect each polyhedron with a slightly shifted copy of the bounding box; the complexity of the resulting unbounded cell may increase in this way.)

So far, we have only considered cells in the complement of the union of $\mathcal{P}$. Suppose next that the cell $\Delta$ lies in the intersection $K$ of $l$ polyhedra of $\mathcal{P}$ (and outside all the remaining ones). We first replace these $l$ polyhedra by $K$. Clearly, this does not change the combinatorial complexity of $\Delta$. We now observe that the vertices of $\Delta$ are either (i) vertices of $\partial K$, (ii) outer vertices that lie on the edges of $\partial K$ (some of which might have been inner vertices before replacing the $l$ original polyhedra by $K$), (iii) inner vertices on $\partial K$, or (iv) inner vertices of $\Delta$ in the interior of $K$. Since the complexity of $K$ is clearly only $O(n)$, there are at most $O(n)$ vertices of type (i), at most $O(nk)$ vertices of type (ii), and all the vertices of type (iv) lie on the boundary of (one component of) the complement of the union of the $k-l$ remaining polyhedra. We now bound the number of vertices of type (iii). Each such vertex $v$ is created by the intersection of a facet $F_1$ of $K$ and the intersection edge $e$ of a pair $P_2$, $P_3$ of polyhedra (neither of which is $K$). Clearly, each such edge $e$ can intersect $\partial K$ in at most two points, due to the convexity of $K$. It thus follows that the overall number of such vertices is $O(nk)$. We can thus ignore the $l$ polyhedra containing $\Delta$, since the cell containing $\Delta$ in the resulting subarrangement contains all but at most $O(nk)$ vertices, edges, and faces of $\Delta$.

To summarize, we have shown that it suffices to analyze the complexity of the unbounded cell in an arrangement of $k$ convex bounded polyhedra with a total of $n$ facets.

In what follows, with a slight abuse of notation, we denote by $\mathcal{C}$ the closure of the unbounded cell of $\mathcal{A}(\mathcal{P})$. In the analysis, we will be using subsets $\mathcal{P}' \subseteq \mathcal{P}$, and will denote by $\mathcal{C}(\mathcal{P}')$ the (closure of the) unbounded cell of $\mathcal{A}(\mathcal{P}')$.

**The number of inner vertices.** Let $v$ be an inner vertex of $\mathcal{C}$, which is incident to three facets $F_1$, $F_2$, and $F_3$ of three distinct respective polyhedra $P_1$, $P_2$, and $P_3$. Then $v$ is incident to a convex polygon $\psi$ obtained by the intersection of one of the facets, say facet $F_1$, and the two other polyhedra $P_2$, $P_3$. In this case, the polygon $\psi = F_1 \cap P_2 \cap P_3$ lies fully inside the union of $\mathcal{P}$, except for (some of) its vertices, one of which is $v$ (since $v$ lies on the boundary of the unbounded cell of $\mathcal{A}(\mathcal{P})$). The cell $\mathcal{C}$ may contain other vertices of $\psi$ in addition to $v$. See Figure 5.1(a)–(b) for an illustration of the case where $\psi$ is a quadrilateral, and Figure 5.1(c) for the general case.

Rather than dealing with $\psi$ as a single entity, we break it into a collection of pairwise disjoint *exposed chains*. Such a chain is a maximal contiguous sequence of *inner* vertices $v_0, v_1, \ldots, v_j$ of $\psi$ in counterclockwise order, such that each $v_i$ lies on $\partial\mathcal{C}$, and each edge $v_i v_{i+1}$ does not intersect any other polyhedron of $\mathcal{P}$. The *length* of a chain is the number $j$ of its edges (in particular, when $j = 0$ only one vertex of the chain lies on $\partial\mathcal{C}$). Note that we may have $v_j = v_0$ (when $j > 2$), that is, the chain may be the entire polygon $\psi$. Exposed chains with $v_0 \neq v_j$ are called *open*, and chains with $v_0 = v_j$ (with $j > 2$) are called *closed*. See Figure 5.1(c). The length of a chain is not necessarily bounded by a constant, since two polyhedra and a facet of a third polyhedron may intersect in a convex polygon with up to $\Omega(n)$ vertices. However, we show below that the overall number of vertices of exposed chains of length at least 5, as well as of all exposed *closed* chains, is $O(nk)$. It therefore suffices to consider only exposed *open* chains of length at most 4. We define $V_0^{(j)}(\mathcal{P})$, for $j \geq 0$, to be the maximum number of inner vertices of the unbounded cell of $\mathcal{A}(\mathcal{P})$ that lie on exposed open chains of length at least $j$.

Our approach is to derive a recurrence relationship for the number of inner vertices, by bounding each of the functions $V_0^{(j)}$ in terms of $V_0^{(j+1)}$ (with a special handling of $V_0^{(5)}$), and the solution of the resulting system of recurrences will yield the asserted bounds. Note that we actually seek a bound on the quantity $V_0^{(0)}(\mathcal{P})$, which bounds the overall number of inner vertices of $\mathcal{C}$ (that lie on exposed open chains).

For the analysis, we define, analogously to [85], the *level* of a vertex $w$ of $\mathcal{A}(\mathcal{P})$ to be $l$ if, by removing $l$ polyhedra from $\mathcal{P}$, none of which is incident to $w$, we make $w$ a vertex of the unbounded cell in the resulting subarrangement, and if $l$ is the smallest number with that property. Clearly, all vertices of $\mathcal{C}$ are at level 0. (In general, the set of $l$ polyhedra whose removal "exposes" $w$ need not be unique.)

**Lemma 5.1.1** *For each $j = 0, \ldots, 4$, and for any parameter $\xi \leq k$, we have*

$$V_0^{(j)}(\mathcal{P}) = O\left(\xi^2 \mathbf{E}[V_0^0(\mathcal{R})] + \xi^3 \mathbf{E}[V_0^{(j+1)}(\mathcal{R})] + nk\right), \tag{5.1}$$

*where $\mathcal{R}$ is a random sample of $\frac{k}{\xi}$ polyhedra of $\mathcal{P}$, and $\mathbf{E}[\cdot]$ denotes expectation with respect to the choice of $\mathcal{R}$.*

**Proof**: We fix $0 \leq j \leq 4$, and let $\gamma$ be an exposed open chain of length $j' \geq j$ but less than 5, that is contained in an intersection polygon $\psi$ (if $j' \geq 5$ then, as we will

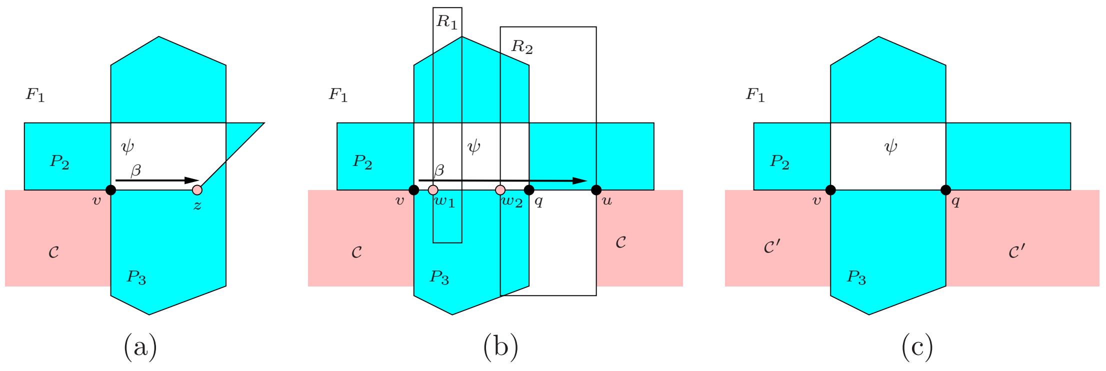


Figure 5.2: The charging scheme. (a) We trace the segment $\beta = vz$ from a vertex $v$ of a polygon $\psi$, and charge $v$ to the outer vertex $z$. (b) We trace the segment $\beta = vu$ from a vertex $v$ of a polygon $\psi$, and encounter the polyhedra $R_1$, $R_2$ before reaching $u \in \partial\mathcal{C}$. We charge the first $\xi$ vertices $w_i$, assuming there are at least $\xi$ of them. (c) When we remove these polyhedra, $q$ becomes a vertex of $\mathcal{C}$, and the edge $vq$ becomes a (portion of an) exposed chain.

show in Lemma 5.1.2, the overall size of these chains is $O(nk)$). Let $v \in \partial\mathcal{C}$ be the *last* (most counterclockwise) vertex of $\gamma$, let $e$ be the edge of $\psi$ that emanates from $v$ in counterclockwise direction, and let $q$ denote the other endpoint of $e$.

We traverse the line containing $e$ from $v$ towards $q$, and stop as soon as one of the critical events listed below is encountered. In each case we charge $v$ to certain features encountered along the traced portion $\beta$. (We may stop either before or after reaching $q$, or at $q$ itself.)

**Case (a):** $\beta$ ends at an outer vertex $z$ (in this case $z = q$), before reaching any vertex of $\mathcal{C}$. We charge $v$ to $z$. Note that $z$ is charged in this manner only a constant number of times, since along each edge emanating from $z$ it is charged at most twice. Thus the overall number of vertices $v$ of this kind is $O(nk)$; see Figure 5.2(a).

**Case (b):** $\beta$ ends at a vertex $u$ of $\mathcal{C}$. Clearly, the relative interior of $vq$ is disjoint from $\mathcal{C}$. If $u = q$, then, since $v$ is the last vertex on the chain, the edge $vq$ must intersect a fourth polyhedron (other than the three that create the chain). Otherwise, if $q$ is not a vertex of $\mathcal{C}$, $\beta$ must contain $q$ as an interior point; see Figure 5.2(b). Let $t$ denote the number of distinct polyhedra of $\mathcal{P}$ that intersect $\beta$ (excluding the three defining $\psi$). We consider the following two subcases:

**Case (b.1):** $t \geq \xi$. In this case we charge $v$ to a block of $\xi$ points of intersection between $\beta$ and the first $\xi$ polyhedra that we encounter during the traversal. Note that each polyhedron intersects $\beta$ in at most two points, and we choose the point of intersection that lies nearer to $v$ along $\beta$ (at which $\beta$ *enters* the corresponding polyhedron). Any such point $w$ is an inner vertex of $\mathcal{A}(\mathcal{P})$, and it can be charged up to three times (we omit the easy details). By construction, each of the charged vertices $w$ lies at level at most $\xi$, as is easily verified.

Next we obtain an upper bound for the number of inner vertices of $\mathcal{A}(\mathcal{P})$ that lie at level

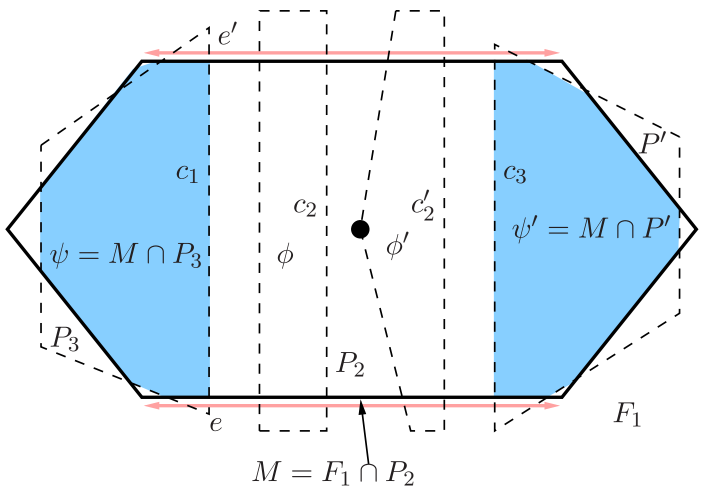


Figure 5.3: Proof of Lemma 5.1.2(a). The convex polygon $M = F_1 \cap P_2$ is drawn in the plane of the facet $F_1$. The two chords $c_1$, $c_3$ lying on the boundaries of the two respective polyhedra $P_3$, $P'$, connect the two edges $e$, $e'$ of $M$ along the two respective intersection polygons $\psi$, $\psi'$. The additional chord $c_2$, connecting $e$ and $e'$, lies on an intersection quadrilateral, and the chord $c_2'$ lies on a pentagon having an outer vertex (in the interior of $M$).

at most $\xi$, by applying the probabilistic analysis technique of Clarkson and Shor [53], in a manner similar to that in [85]. That is, we choose a random sample $\mathcal{R}$ of $r = \frac{k}{\xi}$ polyhedra of $\mathcal{P}$, and construct the arrangement $\mathcal{A}(\mathcal{R})$. Let $w$ be an inner vertex of $\mathcal{A}(\mathcal{P})$ at level $l \le \xi$, and let $\mathcal{L}$ be a collection of $l$ polyhedra whose removal makes $w$ a vertex of $\mathcal{C}(\mathcal{R})$. The probability that $w$ shows up as a vertex of $\mathcal{C}(\mathcal{R})$ is at least $\frac{\binom{k-l-3}{r-3}}{\binom{k}{r}}$, since among all samples of $r$ polyhedra, those that contain the three polyhedra that form $w$ and do not contain any of the polyhedra of $\mathcal{L}$ make $w$ a vertex of $\mathcal{C}(\mathcal{R})$. Hence, we have $\sum_{l=0}^{\xi} \frac{\binom{k-l-3}{r-3}}{\binom{k}{r}} V_l \le \mathbf{E}[V_0^0(\mathcal{R})]$, where $V_l = V_l(\mathcal{P})$ is the number of vertices $w$ of $\mathcal{A}(\mathcal{P})$ at level $l$. As in [53] and [85], this implies that, for $r = \frac{k}{\xi}$, we have $\sum_{l=0}^{\xi} V_l = O\left(\xi^3 \mathbf{E}[V_0^0(\mathcal{R})]\right)$. Since we charge $v$ to a block of $\xi$ of these points, and none of these points is charged more than three times, this implies that the number of inner vertices of $\mathcal{C}$ that fall into this subcase is $O\left(\xi^2 \mathbf{E}[V_0^0(\mathcal{R})]\right)$.

**Case (b.2):** $t < \xi$. In this case, if we remove the $t$ encountered polyhedra from the arrangement (while retaining the three polyhedra that define $\psi$), the next vertex $q$ of $\psi$ that we meet during the traversal must appear on the boundary of the unbounded cell $\mathcal{C}'$ in the reduced arrangement, and the edge $e = vq$ of $\psi$ does not intersect in its interior any of the remaining polyhedra. Thus $\gamma \cup e$ becomes (possibly a prefix of) an exposed chain of length at least $j + 1$, and we charge $v$ to $q$ (note that $q$ is charged up to six times in this manner); see Figure 5.2(c) for an illustration.

Applying standard arguments, as in [53], [85], and above, one can easily show that the number of vertices $v$ that fall into this case is $O\left(\xi^3\mathbf{E}[V_0^{(j+1)}(\mathcal{R})]\right)$, where $\mathcal{R}$ is, as above, a random sample of $\frac{k}{\xi}$ polyhedra of $\mathcal{P}$.

Since we deal with chains of bounded length, considering only the *last* vertex in a chain affects the bound by only a constant factor. Combining cases (a), (b.1), and (b.2), the lemma follows. □

**Lemma 5.1.2** **(a)** *$V_0^{(j)}(\mathcal{P}) = O(nk)$, for $j \geq 5$.*
**(b)** *The number of vertices on exposed closed chains is $O(nk)$.*

**Proof**: **(a)** Let $\gamma$ be an exposed open chain of at least five edges, lying on an intersection polygon $\psi = F_1 \cap P_2 \cap P_3$, where $F_1$ is a facet of $P_1$, and $P_1$, $P_2$, $P_3$ are three distinct polyhedra in $\mathcal{P}$. By definition, $\gamma$ consists of only inner vertices, and thus its edges must alternate between edges incident to $\partial P_2$ and edges incident to $\partial P_3$; without loss of generality, assume that at least three of the edges of $\gamma$ lie on $\partial P_2$. That is, the convex polygon $M = F_1 \cap P_2$ has at least three edges that lie on $\partial P_2$, and $\gamma$ contains at least two *chords* (edges of $\gamma$ lying in the interior of $M$ whose endpoints lie on $\partial\mathcal{C}$) that connect pairs of these edges. We now argue that, once we fix a pair $e$, $e'$ of edges of $M$, they can be connected by at most two chords, each of which belongs to a distinct exposed chain $\gamma$ of length at least five, with at least three edges that lie on $\partial P_2$ (two of which are $e$, $e'$). Indeed, suppose that there are three chords $c_1$, $c_2$, $c_3$ of this kind that connect $e$ and $e'$. Since these are edges of exposed chains, they do not cross each other, so one of them, say $c_2$, lies in between the other two. Moreover, the intersection polygon $\phi$ that contains $c_2$ as an edge must also be fully contained in the region bounded by $e$, $e'$, $c_1$, $c_3$. See Figure 5.3. This implies that either $\phi$ is a quadrilateral, or $\phi$ is a closed polygon with at least five edges, all of whose vertices that lie in the interior of $M$ are outer vertices of $\mathcal{A}(\mathcal{P})$. In either case, $c_2$ does not lie on an exposed chain with at least five edges (as assumed above).

We now fix a facet $F$ and a polyhedron $P \in \mathcal{P}$, different from the one containing $F$ in its boundary, and define the graph $G_M = (V_M, E_M)$, associated with the polygon $M = F \cap P$, as follows. The set $V_M$ of vertices of $G_M$ consists of all edges of the polygon $M$ that lie on $\partial P$. For each exposed chain $\gamma$ that visits at least three edges of $M$ as above, we connect two vertices $v_1, v_2 \in V_M$ by an edge in $E_M$, if they represent two distinct respective edges of $M$ that are connected by a chord along $\gamma$. In addition, if there are two chords connecting $v_1$, $v_2$, we arbitrarily choose one of them to represent the edge $(v_1, v_2)$, ensuring that $G_M$ is simple. Our construction implies that the number of edges $|E_M|$ is proportional to the overall size of all exposed chains $\gamma$ that involve at least three edges of $M$. The "natural" drawing of $G_M$ shows that it is planar. Hence, by Euler's formula, $|E_M| = O(|V_M|)$. That is, the overall size of the chains under consideration is proportional to $|V_M|$. Since $\sum_{F,P} |V_M|$, summed over all facets $F$ and polyhedra $P \in \mathcal{P}$, is $O(nk)$, we obtain that the overall size of all the exposed chains of length at least five is $O(nk)$, as asserted.
**(b)** In view of (a), it suffices to consider exposed closed chains of length 3, 4, and 5 (for longer chains, we can pretend that they are open, by discarding one edge, and then apply

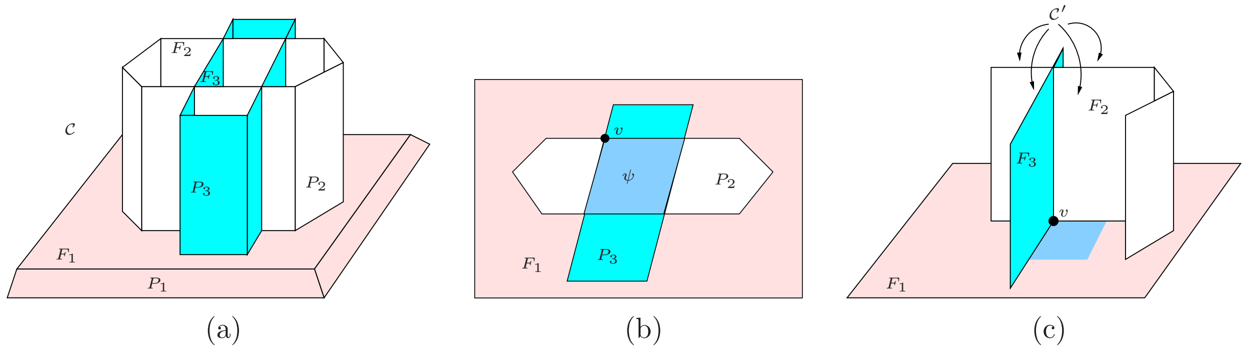


Figure 5.4: From special quadrilaterals to special vertices. (a) The outer exterior of the three polyhedra $P_1$, $P_2$ and $P_3$ coincides with the unbounded cell $\mathcal{C}$. These polyhedra create a special quadrilateral $\psi$ of $\mathcal{C}$. (b) The cross-section through the facet $F_1$ of $P_1$, $P_2$, $P_3$, and $\psi$. (c) The only surviving facets that defined $\psi$ are the base $F_1$ of $P_1$, and the two adjacent walls $F_2$ of $P_2$ and $F_3$ of $P_3$, and thus $v$ is a special vertex of the unbounded cell $\mathcal{C}'$. All four "upper" sides of $v$ are accessible in $\mathcal{C}'$.

(a)). We first note that such a chain, i.e., the polygon $\psi$ containing it, cannot be a triangle or a pentagon: As is easily verified, one of the vertices of $\psi$ must then be the intersection of two adjacent edges that lie on the boundary of the same polyhedron, implying that this vertex is an outer vertex of $\mathcal{A}(\mathcal{P})$, contrary to the definition of chains.

Let us assume then that $\psi$ is a quadrilateral. Thus either $\psi$ is empty (i.e., does not meet any polyhedron other than $P_1$, $P_2$, $P_3$), or else it must contain an outer vertex of $\mathcal{A}(\mathcal{P})$ in its interior (that is obtained by the intersection of the interior of $\psi$ and another polyhedron $P'$). In the latter case, we simply charge $\psi$ to the outer vertex that it contains (note that this vertex is charged only once in this manner), and thus the overall number of charges in this case is $O(nk)$. In the former case, adapting a term from [27], which was also used in [25] and [112], we call $\psi$ a *special quadrilateral of* $\mathcal{C}$; see Figure 5.4(a)–(b).

We claim that the number of special quadrilaterals of $\mathcal{C}$ is $O(nk)$. In the analysis we "open up" the polyhedra of $\mathcal{P}$, by sampling only some of their facets. As a consequence, we now have to regard $\mathcal{C}$ as an *open* cell. We first define the following notions by modifying similar definitions in [85]: (i) An $x$-*extreme* vertex $v$ of $\mathcal{C}$ is a vertex whose $x$-coordinate is the smallest or largest in the closure of some connected component of $N \cap \mathcal{C}$, where $N$ is a sufficiently small ball centered at $v$; see Figure 5.5(a) for an example (in two dimensions). (ii) A point $p \in \partial\mathcal{C}$, with $x$-coordinate $x_0$, is said to be *critical* (with respect to the $x$-direction) if for every sufficiently small ball $N$ centered at $p$ there exists a connected component $K$ of $N \cap \mathcal{C}$, such that $K \cap \pi_{x_0}$ is disconnected, where $\pi_\xi$ denotes the plane $x = \xi$, but $K \cap \pi_x$ is connected either for all $x < x_0$, or for all $x > x_0$ sufficiently close to $x_0$; see Figure 5.5(b) for an example.

We distinguish between different *sides* of an inner vertex $v$: Suppose that $v$ is created by

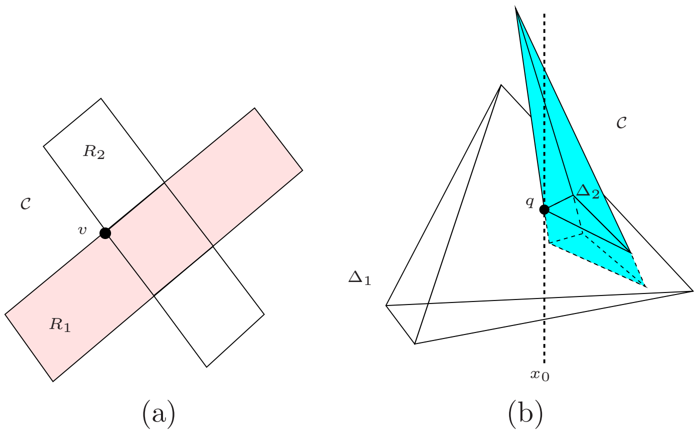


Figure 5.5: (a) The outer exterior of the two rectangles $R_1$, $R_2$ coincides with the unbounded cell $\mathcal{C}$ of their arrangement. The intersection vertex $v$ is a locally $x$-extreme vertex of $\mathcal{C}$. (b) The outer exterior of the two tetrahedra $\Delta_1$, $\Delta_2$ coincides with the unbounded cell $\mathcal{C}$ of their arrangement. The intersection point $q$ of an edge of $\Delta_2$ and a facet of $\Delta_1$ is a critical point.

the intersection of three facets $F_1$, $F_2$ and $F_3$. The planes spanning these facets subdivide space into eight open octants. A *side* of $v$ is a pair $(v, O)$, where $O$ is one of these octants. Let $\mathcal{F}$ denote the set of all the $n$ facets of the polyhedra in $\mathcal{P}$. Consider any subset $\mathcal{F}' \subset \mathcal{F}$ and its arrangement $\mathcal{A}(\mathcal{F}')$. We say that $(v, O)$ is *incident* to (an open) cell $\mathcal{C}'$ of $\mathcal{A}(\mathcal{F}')$, if $v$ is a vertex of the closure of $\mathcal{C}'$, and the intersection of $O$ with a sufficiently small neighborhood of $v$ is contained in $\mathcal{C}'$.

Now let $\psi = F_1 \cap P_2 \cap P_3$ be a special quadrilateral (with four vertices on the boundary of $\mathcal{C}$), where $F_1$ is a facet of some polyhedron $P_1$, and $P_2$, $P_3$ are two other polyhedra in $\mathcal{P}$. Suppose that $\partial\psi$ is formed by the intersection of $F_1$ with the facets $F_2$ and $F_2'$ of $P_2$, and $F_3$ and $F_3'$ of $P_3$. We refer to the facet $F_1$ as the *base* of $\psi$, and to the facets $F_2$, $F_2'$, $F_3$, $F_3'$ as the *walls* bounding $\psi$. See Figure 5.4.

Following the above notation, and continuing to assume that $\psi = F_1 \cap P_2 \cap P_3$ is a special quadrilateral, let $\mathcal{F}'$ be a subset of $\mathcal{F}$ that contains $F_1$, $F_2$, and $F_3$, and does not contain $F_2'$, $F_3'$. Let $\mathcal{C}'$ denote the unbounded (open) cell of $\mathcal{A}(\mathcal{F}')$. We then say that the vertex $v = F_1 \cap F_2 \cap F_3$ is a *special vertex* of $\mathcal{C}'$. Clearly, $v$ is a vertex of $\mathcal{C}'$. The halfspace bounded by the plane through $F_1$ and not containing $P_1$ contains four of the eight sides of $v$, and all these four sides are incident to $\mathcal{C}'$. Indeed, removal of $F_2'$, $F_3'$ exposes all these sides of $v$ to the unbounded cell. See Figure 5.4.

Following the approach of Clarkson and Shor [53], we choose a random subset $\mathcal{F}' \subset \mathcal{F}$ of expected size $\frac{n}{2}$, by selecting each facet of $\mathcal{F}$ independently with probability $\frac{1}{2}$. Let $\phi'(\mathcal{F}')$ be the overall number of special vertices of $\mathcal{C}'$. We then bound the expected number $\mathbf{E}\left[\phi'(\mathcal{F}')\right]$ of special vertices of the unbounded cell of $\mathcal{A}(\mathcal{F}')$, as defined above, and show that the actual number $\phi(\mathcal{P})$ of special quadrilaterals of $\mathcal{C}$ (before the sampling) is at most proportional to $\mathbf{E}[\phi'(\mathcal{F}')]$.

We claim that each special vertex $v$ of $\mathcal{C}'$ is a locally $x$-extreme point of $\mathcal{C}'$. Indeed, let $F_1$, $F_2$, $F_3$ be the facets incident to $v$, and let $O_1$, $O_2$, $O_3$, $O_4$ be the four octants of $v$ that lie in the exterior halfspace of $F_1$ (the one which does not contain $P_1$). All four sides $(v, O_i)$ of $v$, for $i = 1, \ldots, 4$, are incident to $\mathcal{C}'$. The plane $\pi$ that passes through $v$ and is orthogonal to the $x$-axis misses exactly one of these octants, showing that $v$ is a locally $x$-extreme vertex of $\mathcal{C}'$.

We now apply the analysis of [85], which shows that the number of $x$-extreme vertices in a single cell of $\mathcal{A}(\mathcal{F}')$ is proportional to one plus the number of critical points of that cell. Thus it is sufficient to bound the number of critical points of $\mathcal{C}'$. Each critical point must be either a vertex of a polyhedron in $\mathcal{P}$, or an outer vertex of $\mathcal{A}(\mathcal{P})$, because no inner vertex can be critical, as is easily verified. Hence, the overall number of critical points of $\mathcal{C}'$, and thus also the number of special vertices of $\mathcal{C}'$, is only $O(nk)$.

We now bound the number $\phi(\mathcal{P})$ of special quadrilaterals in terms of $\mathbf{E}[\phi'(\mathcal{F}')]$. A special quadrilateral $\psi$ becomes a special vertex if its base and an adjacent pair of its walls are chosen in $\mathcal{F}'$ and neither of the two other walls defining $\psi$ is chosen in $\mathcal{F}'$. Thus the probability that $\psi$ becomes a special vertex is at least $4\left(\frac{1}{2}\right)^5 = \frac{1}{8}$, since there are four pairs of adjacent walls defining $\psi$, and the four corresponding events are pairwise disjoint. Hence $\mathbf{E}[\phi'(\mathcal{F}')] \geq \frac{1}{8}\phi(\mathcal{P})$, which implies that $\phi(\mathcal{P}) = O(nk)$, as asserted. This completes the proof of part (b). $\Box$

**Remarks:** 1) The proof of (a) is similar to the one given in [112], although the context in which we apply it, and some of the details, are quite different.
2) The number of special quadrilaterals is tight in the worst case, as follows from a construction of Aronov *et al.* [27], where the number of special quadrilaterals in the complement of the union of the polyhedra in the resulting collection is $\Omega(nk)$. In their construction, the complement of the union is connected, and thus coincides with the unbounded cell in the arrangement of these polyhedra, so the lower bound immediately follows; see [27] for further details.
3) The analysis of [85], which shows that the number of $x$-extreme vertices in a single cell is proportional to one plus the number of critical points of that cell, can be extended to any number $m$ of cells (see [85] for the technical details). Thus the analysis in (b) applies to any number $m$ of cells in the arrangement, and yields a bound of $O(m + nk)$ on the overall number of special quadrilaterals of these cells (where all four vertices of each special quadrilateral appear on the boundary of a common cell).

The solution of the recurrences derived in Lemmas 5.1.1 and 5.1.2 is $V_0^{(j)}(\mathcal{P}) \leq A_j(\varepsilon)nk^{1+\varepsilon}$, for any $\varepsilon > 0$ and $0 \leq j \leq 4$, where the constants $A_j(\varepsilon)$ depend only on $\varepsilon$ and $j$. In particular, the number of inner vertices of the unbounded cell of $\mathcal{A}(\mathcal{P})$ is at most $A_0(\varepsilon)nk^{1+\varepsilon}$, for any $\varepsilon > 0$. The proof of this claim is routine but technical, and is thus given in Appendix 5.A.

Since the number of outer vertices is only $O(nk)$, routine arguments imply that the overall complexity (number of vertices, edges, and faces) of $\mathcal{C}$ is $O(nk^{1+\varepsilon})$, for any $\varepsilon > 0$, and that, as asserted above, this carries over to any cell of $\mathcal{A}(\mathcal{P})$. The lower bound follows from a construction of Aronov *et al.* [27]. In summary, we have obtained our main result:

**Theorem 5.1.3** *Let $\mathcal{P}$ be a collection of $k$ convex polyhedra in $\mathbb{R}^3$ with $n$ facets in total. The combinatorial complexity of a single cell of $\mathcal{A}(\mathcal{P})$ is $O(nk^{1+\varepsilon})$, for any $\varepsilon > 0$, where the constant of proportionality depends on $\varepsilon$. This bound is almost tight in the worst case, since there are constructions where the complexity of a single cell is $\Omega(nk\alpha(k))$.*

**Zone complexity.** The preceding analysis can be extended to derive an upper bound on the overall complexity of the *zone* of a surface $\sigma$ in $\mathcal{A}(\mathcal{P})$. (Recall that the zone of $\sigma$ in $\mathcal{A}(\mathcal{P})$ is the set of all open cells of $\mathcal{A}(\mathcal{P})$ that are intersected by $\sigma$, and the complexity of the zone is the sum of the complexities of these cells.) We obtain:

**Theorem 5.1.4** *Let $\mathcal{P}$ be a collection of $k$ convex polyhedra in $\mathbb{R}^3$ with $n$ facets in total, and let $\sigma$ be an algebraic surface of constant degree, or the boundary of an arbitrary convex set in 3-space. Then the combinatorial complexity of the zone of $\sigma$ in $\mathcal{A}(\mathcal{P})$ is $O(nk^{1+\varepsilon})$, for any $\varepsilon > 0$, where the constant of proportionality depends on $\varepsilon$, and on the degree of $\sigma$ in the algebraic case.*

**Proof:** Extending the technique presented in [60] and in [85], we cut the boundary of each polyhedron $P \in \mathcal{P}$ along its curve of intersection with $\sigma$, thus obtaining a new collection $\mathcal{P}'$ of $O(n)$ subpatches whose overall complexity is $O(n+k)$. We leave an arbitrarily small gap between the resulting subpatches of $P$, so that all cells of the zone of $\sigma$ form a single cell $\mathcal{C}'$ in the arrangement $\mathcal{A}(\mathcal{P}')$. However, each of these patches does not necessarily have the shape of a convex polyhedron, and thus the result of Section 5.1 is not directly applicable in this case. Nevertheless, we can extend the preceding analysis and show that the combinatorial complexity of $\mathcal{C}'$ is $O(nk^{1+\varepsilon})$, for any $\varepsilon > 0$.

We first claim that the overall number of intersection vertices on $\sigma$ itself is $O(nk)$. Any such vertex is the intersection of $\sigma$ with some edge $e = F_1 \cap F_2$, where $F_1$, $F_2$ are two facets of two distinct polyhedra $P_1, P_2 \in \mathcal{P}$. Since either $\sigma$ has constant description complexity or is the boundary of a three-dimensional convex set, it intersects $e$ in a constant number of points (two points in the latter case). Since the overall number of edges $e$ of this kind is $O(nk)$, the asserted bound follows. We now extend the charging scheme described in Section 5.1, as follows. We trace a segment $\beta$ starting at a vertex $v$ on $\partial\mathcal{C}'$ (that does not lie on $\sigma$), and stop when one of the critical events listed in Lemma 5.1.1 is encountered along $\beta$. Note that in case (b) $\beta$ ends at a vertex $u$ of $\mathcal{C}'$ (see Lemma 5.1.1), which may lie on $\sigma$. In this case, we charge $v$ to $u$, and, as is easily verified, $u$ is charged at most twice in this manner, and thus the overall number of charges of this kind is $O(nk)$. Combining this case with the cases listed in Lemma 5.1.1, we obtain the same recurrences as those given in Lemma 5.1.1. In these recurrences we also use the fact that the number of locally $x$-extreme vertices of $\mathcal{C}'$ is $O(nk)$. This follows by bounding the number of locally $x$-extreme vertices by 1 plus the number of critical points of $\mathcal{C}'$, using the general relationship of [85]. In this case, each critical point is either an outer vertex, as above, or a vertex on $\sigma$, so the number of these points is still $O(nk)$. We thus conclude that the combinatorial complexity of $\mathcal{C}'$ is $O(nk^{1+\varepsilon})$, for any $\varepsilon > 0$. $\square$

## 5.2 Constructing a Single Cell

In this section we present a divide-and-conquer algorithm that constructs the unbounded cell in an arrangement of *bounded* convex polyhedra. Specifically, let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be a collection of $k$ bounded convex polyhedra in 3-space having $n$ facets in total, and let $\mathcal{C}$ be the unbounded cell of $\mathcal{A}(\mathcal{P})$. We present below an efficient algorithm that constructs $\mathcal{C}$, whose overall running time is $O(nk^{1+\varepsilon} \log^3 n)$, for any $\varepsilon > 0$. At the end of this section we present a slight modification of the algorithm so that it constructs any specified cell in $\mathcal{A}(\mathcal{P})$, e.g., the cell containing a given marking point $p$ not lying on any polyhedron boundary.

We apply a variant of the divide-and-conquer scheme presented in [92]. We partition the given polyhedra into $r$ roughly equal subsets $\mathcal{P}_1, \ldots, \mathcal{P}_r$, each of which consists of $\lceil \frac{k}{r} \rceil$ polyhedra, where $r$ is a sufficiently large constant that we will fix shortly. Let $n_i$ be the overall number of facets of the polyhedra of $\mathcal{P}_i$, for $i = 1, \ldots, r$, and let $\mathcal{P}_{i,j} = \mathcal{P}_i \cup \mathcal{P}_j$, for $1 \le i < j \le r$. We recursively compute the unbounded cells $\mathcal{C}_{i,j}$ of the arrangements $\mathcal{A}(\mathcal{P}_{i,j})$, and "merge" them (that is, superimpose them) to extract from them the unbounded cell $\mathcal{C}$ in the entire arrangement $\mathcal{A}(\mathcal{P})$. As we will show, the merge step can be performed in overall time $O\left(rnk^{1+\varepsilon} \log^3 n\right) = O\left(nk^{1+\varepsilon} \log^3 n\right)$, for any $\varepsilon > 0$, which implies the following recurrence for the maximum time $T(k, n)$ for constructing the unbounded cell in an arrangement of $k$ convex polyhedra with $n$ facets in total:

$$T(k, n) \le \sum_{1 \le i < j \le r} T\left(\frac{2k}{r}, n_i + n_j\right) + cnk^{1+\varepsilon} \log^3 n, \tag{5.2}$$

where $c$ is a sufficiently large constant that depends on $r$, and where $\sum_{i=1}^{r} n_i = n$. Using induction on $k$, it is easy to see that $T(k, n) = O(nk^{1+\varepsilon} \log^3 n)$, for any $\varepsilon > 0$, where the constant of proportionality depends on our choice of $r$ and $\varepsilon$.

The merge step of the algorithm uses the following ray shooting technique (similar ideas were used in [23]). Suppose we are at a vertex $v \in \partial\mathcal{C}$ that lies on three facets $F$, $F'$, $F''$ of three respective polyhedra $P \in \mathcal{P}_i$, $P' \in \mathcal{P}_j$, $P'' \in \mathcal{P}_l$, for some triple of distinct indices $i, j, l$. (Vertices that lie on facets of polyhedra in only two subcollections $\mathcal{P}_i$, $\mathcal{P}_j$, are easier to handle.) We shoot from $v$ along the intersection segment $e = F \cap F'$ in the direction that proceeds from $v$ along $\partial\mathcal{C}$ (that is, away from $P''$), until we hit a new polyhedron, thereby reaching a new vertex $v'$ of $\partial\mathcal{C}$. We then shoot from $v'$ along its two other incident

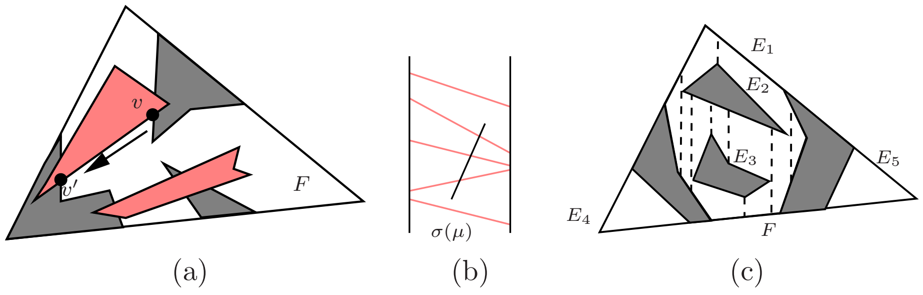


Figure 5.6: The merge step. (a) Shooting along an edge of $\partial\mathcal{C}$ (the white portion) within $F$. (b) The short blue segment crosses a contiguous subsequence of long red segments within $\sigma(\mu)$. (c) Vertical extensions that connect different components of $E$ on $F$.

edges along $\partial\mathcal{C}$, and keep exploring $\partial\mathcal{C}$ in this manner until no new vertices are found. See Figure 5.6(a). At this point we have traced a complete connected component of *1-skeleton* $E$ of $\partial\mathcal{C}$, i.e., the union of the edges of $\partial\mathcal{C}$. We now look for a new vertex of $\partial\mathcal{C}$ that lies on a different connected component of $E$, and repeat the above procedure until the entire 1-skeleton $E$ is constructed. Completing the representation of $\mathcal{C}$ is then quite routine (and thus omitted).

To complete the presentation of the algorithm, we need to describe two steps: (i) How to perform the ray shootings. (ii) How to find a point on each connected component of $E$.

We note that a variant of both of these steps was previously introduced by Aronov and Sharir [23], where they present an algorithm that constructs $m$ cells in an arrangement of $n$ convex plates that lie in planes with only a constant number of orientations. However, we improve and generalize both steps of the algorithm: (i) Our ray-shooting machinery supports $O(\log^3 n)$ query time in more general setups, whereas the query time in the ray-shooting machinery of [23] is $O(\log n)$ times the number of distinct orientations (which can be very expensive in our case). (ii) We simplify the second step of the algorithm of [23] by exploiting the topological properties stated in Lemma 5.2.1 — see below.

Consider task (i). Fix a facet $F$ of some polyhedron $P \in \mathcal{P}_i$, and consider the problem of shooting within $F$ from a vertex $v$ along an edge $e$ of some $\partial\mathcal{C}_{i,j}$. Fix a third subcollection $\mathcal{P}_l$, and consider the subtask of finding the first point (if any) where the portion of $e$ along which we shoot hits some polyhedron of $\mathcal{P}_l$. This is equivalent to asking for the first segment of $\partial\mathcal{C}_{i,l}$ lying on $F$ that this portion of $e$ hits. We need to repeat this step for each $l \neq i, j$, and take among all the (at most) $r-2$ output vertices the one that is nearest to $v$, or the other endpoint of $e$ if it does not hit any other segment.

We thus face the following "trichromatic" subproblem. We have three fixed subcollections, $\mathcal{P}_i$, $\mathcal{P}_j$, $\mathcal{P}_l$, to which we refer as the *green collection*, the *red collection*, and the *blue collection*, respectively. We fix a green facet $F$, and denote by $\mathcal{R} = \mathcal{R}_F$ the set of all edges of $\partial\mathcal{C}_{i,j}$ on $F$ (green-red edges) and by $\mathcal{B} = \mathcal{B}_F$ the set of all edges of $\partial\mathcal{C}_{i,l}$ on $F$ (green-blue edges). To simplify the notation, we refer to the edges of $\mathcal{R}$ (resp., $\mathcal{B}$) simply as *red* (resp., *blue*). We thus need to preprocess these two sets to support efficient ray-shooting queries that involve the red and the blue edges — see below.

**Efficient ray-shooting machinery.** Our ray-shooting machinery uses a *hereditary segment tree* data structure $\mathcal{T}$ (see [45, 117]) that stores the set $\mathcal{S} = \mathcal{R} \cup \mathcal{B}$ of the red and the blue segments on $F$, using their projections on the $x$-axis (in an appropriate generic two-dimensional coordinate frame attached to $F$). Put $N = |\mathcal{S}|$. Each interior node $\mu$ of $\mathcal{T}$ represents an interval along the $x$-axis, which is the union of the "atomic intervals" associated with the leaves of the subtree rooted at $\mu$. Let $\sigma(\mu)$ be the vertical slab containing all points whose $x$-coordinates lie in the interval that $\mu$ represents. A segment $s \in \mathcal{S}$ is said to be *long* in a slab $\sigma(\mu)$ if $s$ crosses $\sigma(\mu)$ from side to side, otherwise, if at least one endpoint of $s$ lies in $\sigma(\mu)$, $s$ is *short* in $\sigma(\mu)$. We store in each node $\mu$ of $\mathcal{T}$ two respective *long lists* of red and of blue long segments, and two additional respective *short lists* of red and of blue short segments, as follows. If a segment $s$ is long in $\sigma(\mu)$ but not in the parent slab of $\sigma(\mu)$, then we store $s$ in the appropriate red or blue long list of $\mu$ (the standard segment tree rule). The short red (blue) list of $\mu$ contains all red (blue) segments stored in all long red (blue) lists of the *proper descendants* of $\mu$. As shown in [45, 117], the overall size of the lists stored in all the nodes of $\mathcal{T}$, and the time to construct them, is $O(N \log N)$. As observed in [45, 117], each red-blue intersection point arises as the intersection of a long segment and another (short or long) segment in exactly one slab $\sigma(\mu)$. We sort each of the long red lists within each slab $\sigma(\mu)$ in their (well-defined total) ascending vertical order (i.e., the order in the $y$-direction, in the $F$-coordinate frame), and store them, in this order, at the leaves of a balanced binary tree $\mathcal{T}_\mu$. Since the red long segments are pairwise disjoint, the set of the red long segments that a blue (short or long) segment $b$ crosses (within $\sigma(\mu)$) is a *contiguous* subsequence of the red long list, and can be represented as the disjoint union of $O(\log N)$ subtrees of $\mathcal{T}_\mu$; see Figure 5.6(b) and [45, 117]. We thus query $\mathcal{T}_\mu$ with each blue segment $b$ that is stored at $\mu$, and store $b$ at each of the $O(\log N)$ subtrees of $\mathcal{T}_\mu$ that constitute the output to the query. We thus obtain in each node $\nu$ of $\mathcal{T}_\mu$ a *complete bipartite graph* $\mathcal{R}_\nu \times \mathcal{B}_\nu$, where $\mathcal{R}_\nu$ is the set of red long segments stored at the leaves of the subtree rooted at $\nu$, and $\mathcal{B}_\nu$ is the set of all blue segments (long and short) that reach $\nu$ in their query. Every pair of segments $e \in \mathcal{R}_\nu$, $e' \in \mathcal{B}_\nu$ intersect each other (within $\sigma(\mu)$). We construct an analogous structure for the blue long list and the red segments in $\sigma(\mu)$. The overall running time needed to construct the trees $\mathcal{T}_\mu$ and the corresponding bipartite graphs, over all nodes $\mu$ of $T$, is $O(N \log^2 N)$.

Suppose now, that we query with a ray that starts at some point $u$, and proceeds from $u$ along a portion $\rho'$ of a red segment $\rho$. For each node $\mu$ of $\mathcal{T}$ that stores $\rho$ in one of its lists we do the following. If $\rho$ is stored in the short list of $\mu$, we locate the two endpoints of $\rho'$ in the blue long list of $\mu$, and report the first blue long segment (nearest to $u$) lying between these two endpoints. If $\rho$ is stored in the long list of $\mu$, then we visit each of the

$O(\log N)$ nodes $\nu$ of $\mathcal{T}_\mu$ on the search path to $\rho$, and, searching in the corresponding $\mathcal{B}_\nu$, find the first blue segment in that list hit by $\rho'$. The overall query time in $\mathcal{T}_\mu$ is $O(\log^2 N)$, thus the query takes a total of $O(\log^3 N)$ time. The final output of the query is the first blue segment that the ray hits, among all reported blue segments.

In summary, we maintain for each facet $F$ of a polyhedron $P \in \mathcal{P}_i$, for $1 \leq i \leq r$, $r-1$ trees $\mathcal{T}$ as above, which correspond to all $r-1$ sets of edges of $\partial\mathcal{C}_{i,j}$, for $j \neq i$, that lie on $F$, and each ray-shooting query accesses $r-2$ such trees. Since $r$ is a constant, a ray-shooting query can be performed in overall time $O(\log^3 n)$.

**Tracing all the components of $E$.** Consider next task (ii), of finding a point on each connected component of 1-skeleton $E$ of $\partial\mathcal{C}$. We first show

**Lemma 5.2.1** *Let $\mathcal{K}$ be a connected component of $\bigcup\mathcal{P}$. Then the intersection of $\partial\mathcal{K}$ and $\partial\mathcal{C}$ (i.e., the common boundary of $\mathcal{K}$ and $\mathcal{C}$) is connected.*

**Proof**: In the proof, we replace $\mathcal{C}$ by its closure, and thus regard $\mathcal{C}$ as a closed set. Recall also that we assume that the polyhedra in $\mathcal{P}$ are in general position. We use the three-dimensional version of the Jordan curve theorem, which asserts that a closed surface in $\mathbb{R}^3$ (a connected embedded 2-manifold without boundary) separates the three-dimensional space into two connected (disjoint) components (see, e.g., [89, Proposition 2B.1]). Specifically, we proceed as follows.

Let $H$ be a connected component of the common boundary of $\mathcal{K}$ and $\mathcal{C}$. We claim that $H$ is a closed 2-manifold (without boundary). Indeed, being a manifold is a local condition. Let $p$ be a point on $H$, and let $N$ be a sufficiently small neighborhood of $p$. Then, due to the general position assumptions, $N \cap \mathcal{K}$ is either (locally) (i) a halfspace, in case $p$ lies in the interior of a two-dimensional face of $H$, or (ii) a convex dihedral wedge (or its complement), if $p$ appears on an edge of $H$, or (iii) a trihedral cone, if $p$ coincides with a vertex of $H$ (the cone is the complement of a convex cone in case $p$ coincides with an inner vertex, and non-convex (with a non-convex complement) if $p$ coincides with an outer vertex). Therefore in all the above cases $N \cap H$ is locally a disk, and thus $\mathcal{K}$ is a connected polyhedral set whose boundary is a (possibly not connected) polyhedral 2-manifold without boundary, and so is $\mathcal{C}$.

Assume now, for the sake of contradiction, that the common boundary of $\mathcal{K}$ and $\mathcal{C}$ has at least two connected components. Let $H_1$, $H_2$ be two of these boundary components. Then, applying the three-dimensional variant of the Jordan curve theorem, we obtain that $H_1$ separates $\mathbb{R}^3$ into two disjoint open connected regions $R_1$, $R_2$, where $R_1$ contains $\mathcal{K}$ and $R_2$ contains $\mathcal{C}$. Since $\mathcal{C}$ is connected, the second boundary component must be contained in $R_2$. But then it is clear that $\mathcal{K}$ cannot be connected, which results in a contradiction. $\Box$

We first describe how we construct, for each connected component $\mathcal{K}$ of $\bigcup\mathcal{P}$, the entire portion $E_\mathcal{K}$ of $E$ that lies on $\partial\mathcal{K}$. Let $F$ be a fixed facet of some $P \in \mathcal{P}_i$. $F$ contains $r-1$ planar subdivisions $S_j$, each formed by the edges of $\partial\mathcal{C}_{i,j}$ on $F$, for some $j \neq i$. We first remove those vertices of each union $\mathcal{C}_{i,j}$ on $F$ that lie inside other unions $\mathcal{C}_{i,l}$ (this can be accomplished by $r-2$ point locations of each vertex in the other subdivisions). We then draw one or two vertical extensions (upwards and/or downwards in the $y$-direction of the $F$-frame) from each surviving vertex of each subdivision $S_j$, into the interior of the corresponding regions $F \cap \partial\mathcal{C}_{i,j}$, and stop as soon as these extensions meet another segment (that belongs to one of the $r-1$ subdivisions), or reach the boundary of $F$; see Figure 5.6(c) for an example. This can be easily done in time $O(r|F|\log n) = O(|F|\log n)$, where $|F|$ is the total number of edges of the unions $\partial\mathcal{C}_{i,j}$ on $F$, for $j \neq i$.

Consider the graph $\mathcal{G}$ whose vertices are the vertices of $E_{\mathcal{K}}$ and the endpoints of the vertical extensions that lie on edges of $E_{\mathcal{K}}$, and whose edge set is the union of the edges of $E_{\mathcal{K}}$ and the above vertical extensions (where edges that contain vertical extension endpoints are partitioned into sub-edges at these points). We claim that $\mathcal{G}$ is connected. Indeed, we observe, using Lemma 5.2.1, that each pair $u$, $v$ of vertices of $\mathcal{G}$ is connected along a path $\pi$ on the common boundary of $\mathcal{C}$ and $\mathcal{K}$. Consider any connected portion $\pi'$ of $\pi$ that intersects some facet $F$. It is easily checked that the endpoints of $\pi'$ lie on edges of $\partial\mathcal{C}$ (and of $\partial\mathcal{K}$). Since the vertical extensions on $F$ decompose $F \cap \partial\mathcal{C}$ into simply connected regions, we can replace $\pi'$ by the concatenation of appropriate boundary portions of the regions intersected by $\pi'$, and thus obtain a new path $\pi^*$ that connects $u$ to $v$ and is composed only of edges of $\mathcal{G}$. (Technically, some portions of $\pi^*$ may not traverse complete edges of $\mathcal{G}$, in which case we simply discard these portions.) Hence $\mathcal{G}$ is connected.

Let $\mathcal{K}$ be a connected component of $\bigcup\mathcal{P}$, and let $E_{\mathcal{K}}$ denote $E \cap \partial\mathcal{K} \subseteq \partial\mathcal{C}$, as above. Suppose we are given a starting point vertex of $E_{\mathcal{K}}$. Having found such a vertex, we start the repeated ray-shooting procedure along $\partial\mathcal{C}$, in the manner described above, but with the following modification. When the shooting reveals a new edge $e$ of $E_{\mathcal{K}}$, we check whether $e$ contains any endpoint of some vertical extension. For any such extension, we add its other (one or two) contacts with $E_{\mathcal{K}}$ to the set of points from which further ray shootings should be attempted. See Figure 5.6(c). In other words, the modified ray-shooting mechanism traces the graph $\mathcal{G}$ defined above. Since $\mathcal{G}$ is connected and the union of its edges contains $E_{\mathcal{K}}$, we construct the entire $E_{\mathcal{K}}$ in this manner. Finding a starting point on each $E_{\mathcal{K}}$ is done as follows. The highest vertex $v$ of a polytope that has not yet been traversed is a candidate for a new starting point, provided that the upward vertical ray from $v$ either does not hit any other polyhedra or hits a polyhedron at a point that belongs to the already traced boundary of $\mathcal{C}$. This shooting can be accomplished in overall time $O(k^2\log n)$, using standard techniques involving vertical ray-shooting in 3-space (see, e.g., [23] for a variant of this technique).

In conclusion, the overall running time of the merge step is easily seen to be $O(nk^{1+\varepsilon}\log^3 n)$, for any $\varepsilon > 0$. As argued above, this also bounds the running time of the entire algorithm.

**Constructing any specified cell in $\mathcal{A}(\mathcal{P})$.** We now show how to slightly modify the previous algorithm so that it constructs the cell of $\mathcal{A}(\mathcal{P})$ containing a given marking point

$p$ (not lying on any polyhedron boundary). With a slight abuse of notation, let $\mathcal{C}$ denote this cell. Let us denote by $\mathcal{P}'$ the subset of the polyhedra of $\mathcal{P}$ that contain $p$ in their interior (as in the case of the unbounded cell, we may have $\mathcal{P}' = \emptyset$); clearly, $\mathcal{C}$ is contained in $\bigcap \mathcal{P}'$.

We now construct $\bigcap \mathcal{P}'$ in $O(n \log n)$ time, as the intersection of $O(n)$ halfspaces. Next, we pass some plane $\pi_0$ through $p$, and split any polyhedron $P \in \mathcal{P} \setminus \mathcal{P}'$ that $\pi_0$ crosses into two subpolyhedra $P^-$, $P^+$, both bounded by $\pi_0$, which the algorithm regards symbolically as slightly separated from each other. Let $\mathcal{P}^*$ denote the new set of polyhedra (excluding those in $\mathcal{P}'$). We next construct the unbounded cell of $\mathcal{A}(\mathcal{P}^* \setminus \mathcal{P}')$ in $O(nk^{1+\varepsilon} \log^3 n)$ time, for any $\varepsilon > 0$. Finally, we merge these two cells using our merge procedure (in this case we only merge two cells rather than $O(r^2)$ cells as in the original description of this procedure).

We summarize below our analysis:

**Theorem 5.2.2** *Let $\mathcal{P}$ be a collection of $k$ bounded convex polyhedra in $\mathbb{R}^3$ with $n$ facets in total, and let $p$ be a given marking point not lying on any polyhedron boundary. Then the cell $\mathcal{C}$ of the arrangement $\mathcal{A}(\mathcal{P})$ that contains $p$ can be constructed in time $O(nk^{1+\varepsilon} \log^3 n)$, for any $\varepsilon > 0$, where the constant of proportionality depends on $\varepsilon$.*

**Discussion.** We note that our algorithm, though efficient, is not output-sensitive, since the number of vertices that appear on the boundary of some of the recursively computed unbounded cells $\mathcal{C}_{i,j}$ may be significantly larger than those that appear on the boundary of the (final) unbounded cell $\mathcal{C}$. In this case, the merge step eliminates most of these vertices when $\mathcal{C}$ is constructed.

## 5.3 Concluding Remarks

We have presented a nearly tight bound on the combinatorial complexity of a single cell in an arrangement of $k$ convex polyhedra in $\mathbb{R}^3$ with $n$ facets in total, thus settling a conjecture of Aronov *et al.* [27], which asserts that the complexity of a single component of the complement of the union of these polyhedra is close to $O(nk)$. We have also extended our result to derive an upper bound on the overall complexity of the zone of a low-degree algebraic surface patch, or the boundary of an arbitrary convex set, in an arrangement of $k$ convex polyhedra in 3-space with $n$ facets in total. Finally, we presented a deterministic algorithm that constructs a single cell of an arrangement of this kind in overall running time $O(nk^{1+\varepsilon} \log^3 n)$, for any $\varepsilon > 0$.

We note that the combinatorial upper bound that we obtained in this study can be used to improve other combinatorial bounds on substructures in arrangements of convex polyhedra in higher dimensions. Consider the problem of bounding the combinatorial complexity of the lower envelope of $k$ convex polyhedra in $\mathbb{R}^4$ with $n$ facets in total. The three-dimensional version of this problem was solved by Huttenlocher *et al.* [92], who presented an upper bound of $O(nk\alpha(nk))$. In their technique, they divide the polyhedra into two subcollections of approximately equal size, referred to as the "red" collection $R$ and the "blue" collection $B$. Then they bound the *increase* in the number of red faces caused by the addition of the facets of the blue polyhedra to the already existing red envelope, and vice versa, and, using an appropriate recursion scheme, show that the overall combinatorial complexity of the lower envelope is proportional to the increase (when each of $R$, $B$ contains roughly $\frac{k}{2}$ polyhedra). As shown in [92], the increase is bounded by the number of reflex vertices in the arrangement and the overall complexity of the *zone* of the boundary of each blue facet $F$ in the three-dimensional arrangement $\mathcal{A}_F$ induced by intersecting $F$ (currently being added to the envelope) with all original red facets and previously added blue facets. An inspection of the proof given in [92] shows that it can be extended to four dimensions. The number of reflex vertices in the four-dimensional arrangement is easily seen to be $O(n^2k)$. The overall complexity of all these zones is bounded as follows. Let $F$ be a blue facet. The intersection of $F$ with each previously added polyhedron (or with an added portion of a blue polyhedron) is a three-dimensional convex polyhedron (possibly bounded by a portion of the boundary of $F$). Let $\mathcal{P}_F$ be the collection of these polyhedra, and let $t_F$ be the overall number of their facets. Applying the bounds stated in this study, the zone complexity of the boundary of $F$ in the arrangement $\mathcal{A}(\mathcal{P}_F)$ is $O(t_F k^{1+\varepsilon})$, for any $\varepsilon > 0$, and since $\sum_{P\in B}\sum_{F\in P} t_F = O(n^2)$ (as can be easily verified), it follows that the overall complexity of these zones, over all blue facets $F$, is $O(n^2k^{1+\varepsilon})$, for any $\varepsilon > 0$. Thus the overall increase is bounded by $O(n^2k^{1+\varepsilon})$, which implies that the combinatorial complexity of the lower envelope is $O(n^2k^{1+\varepsilon})$, for any $\varepsilon > 0$. That is, we have

**Theorem 5.3.1** *The combinatorial complexity of the lower envelope of $k$ convex polyhedra in $\mathbb{R}^4$ with $n$ facets in total is $O(n^2k^{1+\varepsilon})$, for any $\varepsilon > 0$, where the constant of proportionality depends on $\varepsilon$.*

An open problem that this study leaves is to tighten the small remaining gap between the upper and the lower bounds on the complexity of a single cell, or at least to improve the bound to $O(nk \cdot \text{polylog}(k))$. We note that in an attempt to derive a tighter bound, we tried to apply the techniques of Aronov and Sharir [24] and Tagansky [126] for upper bounding the complexity of a single cell in an arrangement of $n$ $(d-1)$-simplices in $\mathbb{R}^d$, as well as the technique of Aronov *et al.* [27] for bounding the complexity of the union of $k$ convex polyhedra in $\mathbb{R}^3$ with $n$ facets in total. Nevertheless, we did not manage to adapt these techniques so that they yield bounds that depend also on $k$. However, for small values of $k$, we can use the bound $O(k^3 + nk\log k)$, presented by Aronov *et al.* [27], on the complexity of the union of the polyhedra. In this case, one can easily conclude that the complexity of the unbounded cell in the arrangement of the polyhedra is $O(nk\log k)$, for any $k \le \sqrt{n}$.

It would be intersecting to extend our results to higher dimensions. An easy lower bound in $d$-dimensions, where $d > 2$ is even, is obtained by the construction presented by Aronov *et al.* [27], in which we consider a planar collection of $\frac{k}{d/2}$ convex polygons with a total of $\frac{n}{d/2}$ edges, so that the unbounded face in the two-dimensional arrangement induced by these polygons has $\Omega(n\alpha(k))$ vertices on its boundary (see [26]). Next, we consider a family of $d/2$ mutually orthogonal 2-flats in $d$-space, and place one copy of the two-dimensional configuration into each 2-flat. We then extend each polygon into a prism in the remaining $d-2$ coordinates. As shown in [27], the number of vertices on the boundary of the unbounded cell in the arrangement induced by the $k$ prisms, having a total of $n$ facets, is at least $\Omega(n^{d/2}\alpha^{d/2}(k))$. The construction for odd $d > 3$ is similar and yields $\Omega(kn^{\lfloor d/2 \rfloor}\alpha^{\lfloor d/2 \rfloor}(k))$ vertices on the boundary of the unbounded cell in the arrangement (see [27] for further details). The best known upper bound in $d > 3$ dimensions is $O(n^{d-1+\varepsilon})$, for any $\varepsilon > 0$, presented by Basu [33]. Nevertheless, obtaining a sharp upper bound seems at the moment elusive, and requires developing new topological arguments that help to bound the number of extreme points and critical points that appear on the boundary of a single cell in the arrangement.

Finally, another technical problem that this study raises is to improve the dependence of the running time on $n$ by a logarithmic factor. It would seem that this could be done using fractional cascading [44] in the ray-shooting machinery, but so far we have not been able to apply this enhancement.

## 5.A Appendix - Solving the recurrence

In this appendix we solve the recurrence inequalities given in Lemma 5.1.1, using induction on the number $k$ of input polyhedra, and show that $V_0^{(j)}(\mathcal{P}) \le A_j(\varepsilon)nk^{1+\varepsilon}$, for any $\varepsilon > 0$ and $0 \le j \le 4$, where the constants $A_j(\varepsilon)$ depend only on $\varepsilon$ and $j$. The proof is a straightforward generalization of the analysis of [85]. For the sake of completeness, we repeat some of the details given in [85], and modify the analysis to fit into the context of our problem.

Lemma 5.1.1 clearly holds for all $k \le k_0$, for any fixed constant threshold $k_0$, by choosing the coefficients $A_j = A_j(\varepsilon)$ sufficiently large. Consider the case where $k > k_0$, and suppose that the claim holds for all $k' < k$. Using the induction hypothesis, it is easily verified that, for $0 \le j \le 4$,

$$\mathbf{E}[V_0^{(j)}(\mathcal{R})] \le A_j \frac{n}{\xi}\left(\frac{k}{\xi}\right)^{1+\varepsilon},$$

and that $\mathbf{E}[V_0^{(5)}(\mathcal{R})] = O\left(\frac{nk}{\xi^2}\right)$. Indeed, consider the case $j \le 4$. Since the size of $\mathcal{R}$ is fixed and equal to $\frac{k}{\xi}$, we have $\mathbf{E}[V_0^{(j)}(\mathcal{R})] \le A_j \mathbf{E}[n_{\mathcal{R}}]\left(\frac{k}{\xi}\right)^{1+\varepsilon}$, where $n_{\mathcal{R}}$ is the overall number of facets of the polyhedra in $\mathcal{R}$. Setting $r = \frac{k}{\xi}$, each original facet $f$ is chosen with probability $\frac{\binom{k-1}{r-1}}{\binom{k}{r}} = \frac{r}{k} = \frac{1}{\xi}$ (this is the probability that the polyhedron containing $f$ is chosen), from which the claim follows easily. The case $j = 5$ is handled in exactly the same manner.

We first rewrite (5.1), using a different parameter $\xi_j$ for each $0 \le j \le 4$, as follows

$$V_0^{(j)}(\mathcal{P}) \le c\left(\xi_j^2 \mathbf{E}[V_0^{(0)}(\mathcal{R})] + \xi_j^3 \mathbf{E}[V_0^{(j+1)}(\mathcal{R})] + nk\right), \tag{5.3}$$

for appropriate positive constants $c > 1$ and $1 < \xi_j \le k$, for $0 \le j \le 4$, that are sufficiently large. In particular, we have for $j = 4$,

$$V_0^{(4)}(\mathcal{P}) \le c\left(\xi_4^2 \mathbf{E}[V_0^{(0)}(\mathcal{R})] + (\xi_4 + 1)nk\right).$$

We put $\xi_j = \xi_{j+1}^{\varepsilon}$, for $0 \le j \le 4$. We now apply the induction hypothesis in the right-hand side of (5.3), and conclude that the asserted bounds continue to hold for $k$, $n$ as well, provided that the following inequalities are satisfied:

$$A_4 \ge \frac{cA_0}{\xi_4^{\varepsilon}} + c\xi_4^{1-\varepsilon} + \frac{c}{\xi_4^{\varepsilon}}, \tag{5.4}$$

$$A_j \ge \frac{cA_0}{\xi_j^{\varepsilon}} + c\xi_j^{1-\varepsilon} A_{j+1} + \frac{c}{\xi_j^{\varepsilon}}, \tag{5.5}$$

for $j = 0, \ldots, 3$.

We choose

$$A_4 > 2c\left(\xi_4^{1-\varepsilon} + \frac{1}{\xi_4^{\varepsilon}}\right) \quad \text{and} \quad A_j = \frac{(5-j)c^{4-j}}{\xi_j^{\varepsilon}} \xi_4^{\varepsilon} A_4,$$

and require that

$$\frac{5c^5}{\xi_4^{\varepsilon^5}} < \frac{1}{2}. \tag{5.6}$$

Using simple algebraic arguments, the inequality in (5.4) is equivalent to

$$A_4 \geq \frac{5c^5 A_4}{\xi_4^{\varepsilon^5}} + c\left(\xi_4^{1-\varepsilon} + \frac{1}{\xi_4^{\varepsilon}}\right),$$

and due to (5.6), the choice of $A_4$ satisfies the inequality, as is easily verified. Applying again inequality (5.6), it follows that the general inequality in (5.5) is implied by

$$c \leq \left(c^{4-j} - \frac{1}{2}\right) A_4 \xi_4^{\varepsilon}.$$

Since the right-hand side increases as $j$ decreases, it suffices to verify the above inequality for $j = 4$, which trivially holds by our choice of $A_4$. This completes the induction step, and thus establishes the solution of the recurrence for the functions $V_0^{(j)}(\mathcal{P})$. $\square$

# Chapter 6

# The Union of Fat Tetrahedra in Three Dimensions

In this chapter we show that the combinatorial complexity of the boundary of the union of $n$ fat tetrahedra in 3-space is $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$, and that this bound is almost tight in the worst case. The main results and analysis are given in Section 6.1, and then we show the (relatively easy) extension to the case of fat arbitrarily oriented triangular prisms. Using a simple specialization of our technique, we derive in Section 6.2 an almost linear bound on the combinatorial complexity of the union of fat triangles in the plane. We give concluding remarks and present open problems in Section 6.3.

## 6.1 The Union of Fat Tetrahedra

### 6.1.1 Preliminaries and overview

We borrow the following notation from Pach *et al.* [112] (some of which has already been mentioned in the introduction). A *dihedral* (resp., *trihedral*) *wedge* is the intersection of two (resp., three) halfspaces. A dihedral (resp., trihedral) wedge is $\alpha$-*fat* if its dihedral (resp., solid) angle is at least $\alpha$. A trihedral wedge is also associated with the three dihedral angles at its edges. It is easily verified that there exists constant $\alpha' > 0$, which depends only on $\alpha$, such that, for any $\alpha$-fat trihedral wedge, each of its three dihedral angles is at least $\alpha'$ [1].

Similar definitions and observations apply to $\alpha$-fat tetrahedra, namely, tetrahedra all of whose solid angles are at least $\alpha$. In particular, there exist (the same) constant $\alpha' > 0$, such that, for any $\alpha$-fat tetrahedron, each of its six dihedral angles is at least $\alpha'$.

Let $\mathcal{T} = \{T_1, \ldots, T_n\}$ be a collection of $n$ $\alpha$-fat tetrahedra in 3-space, and let $\mathcal{U} = \bigcup \mathcal{T}$ denote their union. For simplicity of the analysis, we assume that the given tetrahedra are

[1] By intersecting the wedge with a sphere centered at it apex, this amounts to asserting that if the (normalized) area of a spherical triangle is at least $\alpha$ then each of its angles is at least $\alpha'$.

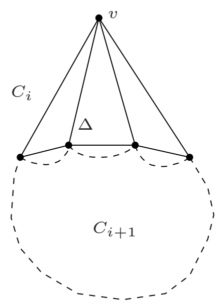


Figure 6.1: Illustrating a step in the construction of the Dobkin-Kirkpatrick hierarchical decomposition: The vertex $v$ is peeled off $C_i$, the convex hull of the "hole" that it leaves is constructed, and the new facets are connected to $v$ by tetrahedra that fill up this portion of $C_i \setminus C_{i+1}$.

in *general position* (see Chapter 5 for similar definitions, and [27] for an argument that this involves no loss of generality). This general position assumption implies that each vertex of the arrangement $\mathcal{A}(\mathcal{T})$ of the (facets of the) tetrahedra of $\mathcal{T}$ lies on exactly three tetrahedra facets, and is thus incident upon only a constant number of edges and faces. This is easily seen to imply that the combinatorial complexity of $\mathcal{U}$ is $O(|V(\mathcal{T})|)$, where $V(\mathcal{T})$ is the set of vertices of $\mathcal{A}(\mathcal{T})$ that appear on the boundary of the union.

We classify the vertices of $\mathcal{A}(\mathcal{T})$ as in Chapter 5, that is, an intersection vertex $v$ of $\mathcal{A}(\mathcal{T})$ (i.e., not a vertex of one of the tetrahedra of $\mathcal{T}$) is either *outer* or *inner*. Trivially, the number of outer vertices in the entire arrangement $\mathcal{A}(\mathcal{T})$ is $O(n^2)$, so our main goal is to bound the number of inner vertices that appear on $\partial\mathcal{U}$. The main result of this chapter is:

**Theorem 6.1.1** *The complexity of the union of $n$ $\alpha$-fat tetrahedra in $\mathbb{R}^3$ is $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$, where the constant of proportionality depends on $\varepsilon$ and $\alpha$. The bound is almost tight in the worst case.*

It is relatively easy (using standard techniques; see, e.g., [124]) to construct a set of $n$ $\alpha$-fat tetrahedra that yield $\Omega(n^2\alpha(n))$ vertices on the boundary of their union (see also [110, 112] for further details). We thus devote the remainder of this section to deriving the upper bound stated in Theorem 6.1.1.

**Curve-sensitive cuttings.** We use a divide-and-conquer approach, based on a simple variant of curve-sensitive cuttings (see Section 1.1.4, Chapter 3, and [95]). Specifically, let $\mathcal{F}$ be the set of all facets of the tetrahedra in $\mathcal{T}$. For any $r \le n$ there exists a $(1/r)$-cutting $\Xi$ for $\mathcal{F}$, which is a partition of $\mathbb{R}^3$ into $O(r^3 \log^3 r)$ simplices, such that every simplex (also referred to as a *cell* of $\Xi$) is crossed by at most $n/r$ facets of $\mathcal{F}$, with the additional property that any edge of a tetrahedron in $\mathcal{T}$ crosses at most $O(r \log^2 r)$ cells of $\Xi$.

One can obtain such a cutting using the following (simple) construction (see also Chapter 3 for a similar construction)[2]. We first draw a random sample $R$ of $O(r \log r)$ of the planes containing the facets of $\mathcal{F}$, and add to that collection four additional planes that define a sufficiently large simplex $\sigma_0$ that encloses all the vertices of $\mathcal{A}(\mathcal{F})$. We form the arrangement $\mathcal{A}(R)$ of $R$, consider only its portion within $\sigma_0$, and triangulate each of its cells $C$ contained in $\sigma_0$, using the Dobkin-Kirkpatrick hierarchical decomposition of convex polytopes [58].

The number of simplices is proportional to the overall complexity of $\mathcal{A}(R)$, and is thus $O(r^3 \log^3 r)$. The $\varepsilon$-net theory [49, 90] implies that, with high probability, each simplex of the resulting decomposition is crossed by at most $n/r$ (planes containing) facets of $\mathcal{F}$. We pick one sample $R$ for which this property holds, and fix it in the foregoing analysis.

So far, the use of the Dobkin-Kirkpatrick hierarchy is not essential—many other triangulation schemes for the cells of $\mathcal{A}(R)$ (e.g., bottom-vertex triangulation) would do equally well. However, the Dobkin-Kirkpatrick hierarchy *is* crucial for our divide-and-conquer approach in a manner that will be described later on.

Here is a brief review of the technique, given for the sake of completeness, and also because we will exploit several features of the construction in our analysis. Let $C \subseteq \sigma_0$ be a fixed (bounded) cell of $\mathcal{A}(R)$, which is a convex polytope. The hierarchical decomposition of $C$ is a sequence $C_1, \ldots, C_k$ of $k \geq 1$ convex polytopes, such that (i) $C_1 = C$ and $C_k$ is a simplex, (ii) $C_{i+1} \subset C_i$, for $1 \leq i < k$, (iii) $V(C_{i+1}) \subset V(C_i)$, for $1 \leq i < k$, where $V(P)$ is the set of the vertices of a polytope $P$, and (iv) the vertices in $V(C_i) \setminus V(C_{i+1})$ form an independent set in the planar skeleton graph of $\partial C_i$, for $1 \leq i < k$.

It is shown in [58] that there always exists a hierarchical decomposition for $C$ that satisfies $k = O(\log |V(C_i)|)$, $\sum_{i=1}^{k} |V(C_i)| = O(|V(C)|)$, and $\max_i \max_{v \in V(C_i) \setminus V(C_{i+1})} deg(v, C_i) \leq c$, for some absolute constant $c \geq 3$, where $deg(v, C_i)$ is the degree of $v$ in the skeleton graph of $\partial C_i$. Specifically, we obtain $C_{i+1}$ from $C_i$ by the following steps: (a) Find an independent subset $V_i^* \subseteq V(C_i)$ of vertices of degree at most $c$, whose size is $\Theta(|V(C_i)|)$ (e.g., an independent set of size $|V(C_i)|/24$ whose vertices have degree at most 11 can be shown to exist, as a simple consequence of Euler's polyhedral formula). (b) For each $v \in V_i^*$, remove $v$ and its adjacent edges and facets from $C_i$. (c) The removal of such a vertex $v$ leaves a "hole" in $C_i$. The convex hull of the set of neighboring vertices of $v$ is constructed, and its outer (triangular) facets are added as new facets of $C_{i+1}$, thereby closing the hole that $v$ has left [3]. (d) Finally, the gap between $\partial C_i$ and $C_{i+1}$ in the neighborhood of $v$ (formally, the connected component of $\mathrm{int}(C_i) \setminus C_{i+1}$ whose closure contains $v$) is triangulated into $O(1)$ simplices by connecting $v$ with each of the new facets of $C_{i+1}$ that bound the gap; see Figure 6.1 for an illustration.

[2]For simplicity, we do not use the refined technique of [47, 102], which improves the size of the cutting down to $O(r^3)$, since it does not affect the asymptotic bound that we obtain on the complexity of the union, and since the analysis is cleaner without this refinement.

[3]The independence of $V_i^*$ guarantees that the holes, and their hulls, are openly pairwise disjoint.

A simplicial subcell $\Delta$ is said to be generated at step $i$ if it has a vertex $v$ that is removed from $C_i$; that is, $\Delta$ is one of the simplices that fill up the gap formed by the removal of $v$. Note that the three other vertices of $\Delta$ belong to $C_{i+1}$.

The Dobkin-Kirkpatrick decomposition has several useful properties that we will exploit. One of these properties is that a line that crosses a cell $\tau$ of $\mathcal{A}(R)$ crosses only $O(\log r)$ of its simplices (it can visit at most two gaps of $C_i \setminus C_{i+1}$, for each of the logarithmically many indices $i$). Since a line (or, rather, an edge of a tetrahedron in $\mathcal{T}$) crosses at most $O(r \log r)$ cells of $\mathcal{A}(R)$ (it has to cross a plane of $R$ to move from one cell to another), it crosses at most $O(r \log^2 r)$ simplices, as claimed.

**The problem decomposition—an overview.** We construct the cutting $\Xi$, as just described, with a sufficiently large constant value of $r$, and bound the number of inner vertices of the union in each cell of $\Xi$ separately. Fix a cell $\Delta$ of $\Xi$. We classify each facet $F \in \mathcal{F}$ that intersects $\Delta$ as being either *long* in $\Delta$, if $\partial F \cap \Delta = \emptyset$, or *short*, otherwise (similar definitions have been used in Chapter 3). As just discussed, the number of cells $\Delta$ in which $F$ is short is $O(r \log^2 r)$.

Let us fix a tetrahedron $T \in \mathcal{T}$. For each cell $\Delta$ of $\Xi$, either (i) $\Delta$ is disjoint from $T$, or (ii) $\Delta$ is fully contained in $T$, or (iii) $\Delta$ intersects only one or two facets of $T$, or (iv) $\Delta$ intersects at least three facets of $T$. In case (i) $T$ has no effect on the union within $\Delta$. In case (ii) $\Delta$ is fully covered and does not contain any portion of the boundary of the union. In case (iii) we say that $T$ meets $\Delta$ as a dihedral wedge (which can also be a halfspace), and call $T$ a *D-tetrahedron* in $\Delta$, and in case (iv) we say that $T$ meets $\Delta$ as a tetrahedron or a trihedral wedge, and call $T$ a *T-tetrahedron* in $\Delta$.

If $\Delta$ meets only one facet $F$ of $T$, we replace $T$ by the halfspace bounded by that facet and containing $T$. Similarly, if $\Delta$ meets two facets of $T$, we replace $T$ by the dihedral wedge formed by the planes supporting these facets and containing $T$. Clearly, these replacements do not affect the union of the tetrahedra within $\Delta$. The case where at least three facets of $T$ meet $\Delta$ (case (iv)) is more involved—this is after all the situation we started with. What saves us is the property that the number of T-tetrahedra is small on average. This is one of the main technical insights in our analysis, and is established below in Lemmas 6.1.3 and 6.1.4.

Each inner intersection vertex $v$ of the union that appears in $\Delta$ is consequently classified as either *DDD*, if all three facets that are incident to $v$ belong to three respective D-tetrahedra in $\Delta$, *DDT*, if two of these facets belong to two respective D-tetrahedra and one belongs to a T-tetrahedron, *DTT*, if one of these facets belongs to a D-tetrahedron and two belong to two respective T-tetrahedra, or *TTT*, if all three facets belong to three respective T-tetrahedra. In all four cases, the three relevant tetrahedra are distinct.

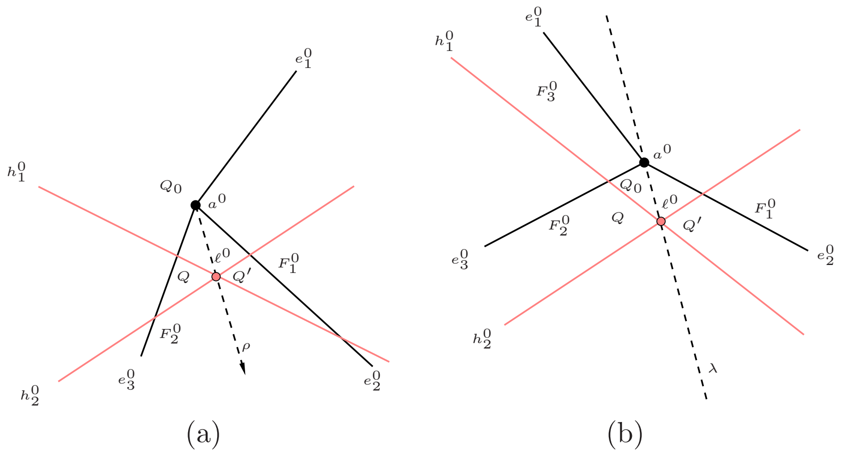


Figure 6.2: The proof of Lemma 6.1.2.

### 6.1.2 T-tetrahedra are scarce

Our next goal is to show that, for each tetrahedron $T \in \mathcal{T}$, the overall number of simplices $\Delta$ of $\Xi$, such that $T$ crosses $\Delta$ and is a T-tetrahedron in $\Delta$, is only $O(r \log^2 r)$. We emphasize that this part of the analysis does not use the fatness of $T$ — the bound holds for *any* tetrahedron $T$.

Let us fix a tetrahedron $T$ of $\mathcal{T}$, and consider the set of simplices $\Delta$ in $\Xi$ that meet at least three facets of $T$. It suffices to consider only simplices $\Delta$ in which all facets of $T$ are long: The edge-sensitivity of the cutting implies that the overall number of simplices $\Delta$ that are crossed by an edge of $T$ is $O(r \log^2 r)$.

We establish the above bound in two steps, in the respective Lemmas 6.1.3 and 6.1.4. In the first step (Lemma 6.1.3) we bound the number of cells of the untriangulated arrangement $\mathcal{A}(R)$ that meet at least three facets of $T$; a crucial ingredient of the analysis is established in Lemma 6.1.2. Then we fix such a cell $C$, and bound (in Lemma 6.1.4) the number of sub-simplices of $\Xi$ in $C$ that have this property. We first prove the following geometric property:

**Lemma 6.1.2** *Let $W$ be a trihedral wedge with apex $a$, let $h_1, h_2$ be two planes, whose intersection line crosses $W$. For $i = 1, 2$, denote by $h_i^+$ the halfspace bounded by $h_i$ and containing $a$, and by $h_i^-$ the complementary halfspace. Then at least one of the wedges $C = h_1^+ \cap h_2^-$, $C' = h_1^- \cap h_2^+$ is crossed by at most two facets of $W$.*

**Proof:** Project everything onto some plane $h_0$ orthogonal to both $h_1, h_2$, and denote the projection of object $u$ by $u^0$. The line $\ell = h_1 \cap h_2$ projects to a point $\ell^0$, and $h_1^0, h_2^0$ are two lines passing through $\ell^0$ and partitioning $h_0$ into four quadrants, so that one of them, $Q_0$, contains $a^0$, and the two quadrants $Q, Q'$ adjacent to $Q_0$ are the projections of $C$, $C'$, respectively.

Let $F_1, F_2, F_3$ be the facets of $W$, and let $e_i$ denote the edge incident to $F_i$ and $F_{i+1}$, for $i = 1, 2, 3$ (where $e_3$ is incident to $F_3$ and $F_1$, and we also denote it by $e_0$). The edges $e_i$ project to three respective rays $e_i^0$ that emanate from $a^0$. Note that, due to the assumption that $\ell$ crosses $W$, $\ell^0$ must be contained in the projection of $W$. We next consider the following two possibilities:
(a) $e_1^0, e_2^0, e_3^0$ are contained in a common halfplane, bounded by a line $\lambda$ through $a^0$. Assume, without loss of generality, that $e_2^0$ lies between $e_1^0$ and $e_3^0$. Then $F_1^0$ and $F_2^0$ are openly disjoint, so at least one of them, say $F_1^0$, does not contain $\ell^0$; see Figure 6.2(a). In this case the ray $\rho$ from $a^0$ through $\ell^0$ is disjoint from $F_1^0$ (except for its apex). We next claim that $F_1^0$ lies fully in one of the halfplanes bounded by (the line containing) $\rho$. It will then easily follow that $F_1^0$ cannot meet both $Q$ and $Q'$, because each of them is fully contained on a different side of the line containing $\rho$. Indeed, if the ray opposite to $\rho$ is contained in $F_1^0$, then it implies that $\ell^0$ and $F_1^0$ lie on different sides of $\lambda$, but then $\ell^0$ is disjoint from the projection of $W$, contradicting our assumption.
(b) $e_1^0, e_2^0, e_3^0$ are not contained in a common halfplane. In this case, all three projections $F_1^0$, $F_2^0$, $F_3^0$ are openly disjoint and cover $h_0$, so $\ell^0$ is contained in exactly one of them, say $F_2^0$. See Figure 6.2(b). The line $\lambda$ through $a^0$ and $\ell^0$ fully contains one of the two other facets, say $F_3^0$, on one side. As in (a), each of the quadrants $Q$, $Q'$ lies fully on one (distinct) side of this line. Hence, one of these quadrants cannot meet $F_3^0$, a contradiction that completes the proof. $\square$

**Lemma 6.1.3** *Let $T$ be an arbitrary tetrahedron. The overall number of cells $C$ of $\mathcal{A}(R)$, for which at least three facets of $T$ meet $C$, each as a long facet in $C$, is $O(r \log r)$.*

**Proof**: Let $F_1$, $F_2$, $F_3$ be a triple of facets of $T$, let $W$ be the trihedral wedge induced by these facets, and let $a$ denote its apex (which is a vertex of $T$). We show below that the overall number of cells $C$ of $\mathcal{A}(R)$, for which all three facets of $W$ meet $C$, each as a long facet in $C$, is $O(r \log r)$ (note that if $F_i$ is long in $C$ then the extended facet of $W$ is also long). By repeating this argument for each triple of facets of $T$, the lemma follows.

Let $\mathcal{H}_0$ denote the set of all the planes in $R$ that intersect $W$. Each such plane intersects $W$ in either a wedge or a triangle (which might be unbounded). We first dispose of all planes that intersect $W$ in a wedge. Each such plane $h$ is disjoint from one of the facets of $W$, and thus one of the halfspaces that it induces, say, the positive side $h^+$ of $h$, meets only two facets of $W$. Thus all the cells of $\mathcal{A}(R)$ under consideration are contained in $h^-$. Hence all these cells lie in the convex polyhedron $K$, which is the intersection of the respective halfspaces $h^-$ induced by the above "good" planes $h$.

Let $\mathcal{H}$ denote the set of "bad" planes in $\mathcal{H}_0$; each of them intersects $W$ in a (possibly unbounded) triangle. Let $\mathcal{C}$ denote the collection of all cells of $\mathcal{A}(\mathcal{H})$ that meet all three facets of $W$ but do not meet any edge of $W$. Fix a facet $F_1$ of $W$, form the intersections $C \cap F_1$, over all cells $C \in \mathcal{C}$, and denote by $\mathcal{C}_1$ the resulting collection of polygons. Note that $\mathcal{C}_1$ is a collection of cells of the 2-dimensional arrangement, within the plane $h_{F_1}$ containing

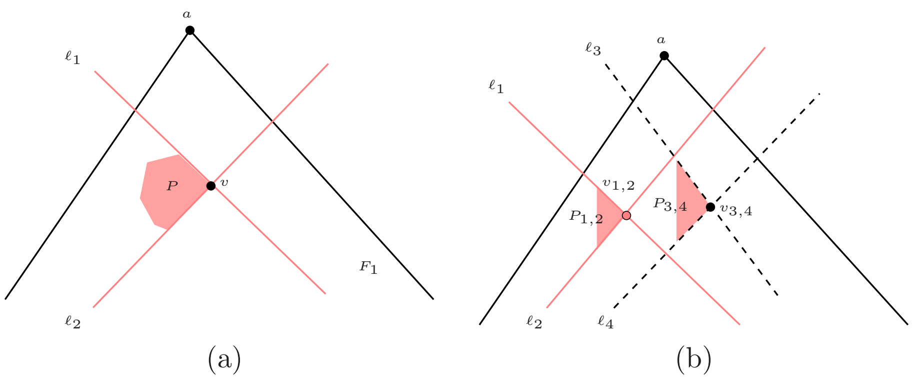


Figure 6.3: (a) The cross section of a cell of $\mathcal{A}(\mathcal{H})$ on $F_1$, and the pair of lines $\ell_1$, $\ell_2$ that it charges. (b) The vertices $v_{1,2}$ and $v_{3,4}$ cannot both be edges of $G$, because the polygon $P_{3,4}$ (partially shaded) is fully contained in the wedge spanned by the respective planes containing $\ell_1$, $\ell_2$ (and opposite to the wedge that contains $P_{1,2}$), which meets only two facets of $W$.

$F_1$, of the set $\mathcal{L}_1$ of the intersection lines between the planes of $\mathcal{H}$ and $h_{F_1}$. The number of unbounded polygons in $\mathcal{C}_1$ is thus $O(|\mathcal{H}|) = O(r \log r)$, so we focus on the bounded elements of this collection. Fix such a polygon $P$, and let $v$ be its vertex which is the most counterclockwise as seen from the apex $a$ (from some fixed side of $h_{F_1}$). Denote the two intersection lines that are incident to $v$ and bound $P$ by $\ell_1, \ell_2$, where $\ell_1$ separates $P$ and $a$ within $h_{F_1}$, and $\ell_2$ does not separate them; see Figure 6.3(a). We then charge $P$ to the pair $(\ell_1, \ell_2)$; clearly, the charge is unique (the two respective planes $h_1$, $h_2$, which intersect $h_{F_1}$ in $\ell_1$, $\ell_2$, can intersect only once on $F_1$).

Let $G$ be the graph whose vertices are the intersection lines $\ell \equiv h \cap h_{F_1}$, for $h \in \mathcal{H}$, and whose edges are all the charged pairs $(\ell_1, \ell_2)$ just defined. We claim that $G$ is planar. It will then follow that the number of edges of $G$ is at most $3|\mathcal{H}| - 6 = O(r \log r)$, which thus also bounds the number of cells of $\mathcal{A}(\mathcal{H})$ that meet all three facets of $W$. Any such cell $C$ induces at most one cell of $\mathcal{A}(R)$ that can touch all three facets of $W$, namely the intersection $C \cap K$. Hence the number of such cells of $\mathcal{A}(R)$ is also $O(r \log r)$.

To establish the claim, assume, without loss of generality, that $a$ is the origin in $h_{F_1}$, and apply the standard duality transform that maps points $(u, v)$ to the respective lines $ux + vy + 1 = 0$ and vice versa (where lines through $a$ are ignored). This duality maps the lines $\ell$ in $\mathcal{L}_1$ to points $\ell^*$, and each of the above pairs $(\ell_1, \ell_2)$ is mapped to the segment connecting the points $\ell_1^*$, $\ell_2^*$ dual to the respective lines $\ell_1, \ell_2$. By construction, and by the properties of this duality, any point $q$ within the polygon $P \in \mathcal{C}_1$ which is represented by $(\ell_1, \ell_2)$, is mapped to a line $q^*$ that separates the origin $o$ and $\ell_1^*$, and has $\ell_2^*$ on the same side as the origin. That is, $q^*$ intersects the segment $\ell_1^*\ell_2^*$. Conversely, for any line $q^*$ that separates $\ell_1^*$ and $\ell_2^*$ as above, its primal point $q$ must lie in the wedge between $\ell_1$ and $\ell_2$ that contains $P$. Moreover, if $q^*$ separates $\ell_1^*$ and $\ell_2^*$ in the opposite way (i.e., so that the origin and $\ell_1^*$ lie on the same side of $q^*$), $q$ lies in the opposite wedge between $\ell_1$ and $\ell_2$.

The collection of dual segments, as constructed above, defines a straight-line embedding of $G$ in the dual plane, and we claim that this drawing is crossing-free. Indeed, suppose to the contrary that two edges $\ell_1^*\ell_2^*$, $\ell_3^*\ell_4^*$ of the drawing cross each other. The preceding discussion then implies that, back in the primal plane $h_{F_1}$, each of the resulting vertices $v_{1,2} = \ell_1 \cap \ell_2$, $v_{3,4} = \ell_3 \cap \ell_4$ lies in the double wedge of the other vertex that does not contain $a$. Denote by $P_{1,2}$ (resp., $P_{3,4}$) the polygon of $\mathcal{C}_1$ whose most counterclockwise vertex is $v_{1,2}$ (resp., $v_{3,4}$). In particular, one of these vertices, say $v_{1,2}$ lies clockwise to the other vertex $v_{3,4}$, in which case $v_{1,2}$ must lie in the wedge of $\ell_3, \ell_4$ that contains $P_{3,4}$, and $v_{3,4}$ must lie in the wedge of $\ell_1, \ell_2$ opposite to the one containing $P_{1,2}$. Let $C_{1,2}$ (resp., $C_{3,4}$) denote the cell of $\mathcal{A}(\mathcal{H})$ that contains $P_{1,2}$ (resp., $P_{3,4}$); also, for $i = 1, \ldots, 4$, let $h_i$ denote the plane of $\mathcal{H}$ containing $\ell_i$. Then, since $C_{1,2}$ meets all three facets of $W$, it follows by Lemma 6.1.2 that the wedge spanned by $h_1$, $h_2$, and opposite to the wedge containing $P_{1,2}$ (and $C_{1,2}$), meets only two facets of $W$, but then $C_{3,4}$ (which is clearly contained in this wedge, since it meets the wedge, and, being a cell of $\mathcal{A}(\mathcal{H})$, cannot cross $h_1$ or $h_2$) cannot meet all three facets of $W$, contrary to assumption; see Figure 6.3(b). This contradiction implies that $G$ is planar, and this, as argued above, implies the assertion of the lemma. $\square$

**Remark:** As already noted, Lemma 6.1.3 is fairly general, and makes no assumption about fatness of $T$. In fact, we believe that in certain circumstances it might also be generalized to situations where $T$ is the boundary of a non-polyhedral convex shape. In this case, the assertion would be that the number of cells of $\mathcal{A}(R)$ that touch at least three pairwise disjoint connected sub-regions on $T$ is $O(r \log r)$ (perhaps with some additional restrictions on these sub-regions, or with a larger number of sub-regions). We consider the lemma to be of independent interest, and believe that it (and/or extensions of it of the kind just suggested) will find additional applications in related problems.

**Lemma 6.1.4** *Let $T \in \mathcal{T}$ be a fixed tetrahedron, and let $C$ be a cell of $\mathcal{A}(R)$ that meets at least three facets of $T$, but not any vertex of $T$. Then the number of simplicial subcells $\Delta$ of $C$ that meet at least three facets of $T$, each as a long facet in $\Delta$, is $O(\log r)$.*

**Proof**: As in Lemma 6.1.3, it is sufficient to assume that $T$ is a trihedral wedge, and to show that the number of simplicial subcells $\Delta$ of $C$ that meet all three facets of $T$, each as a long facet in $\Delta$, is $O(\log r)$.

We first claim that if all the three facets $F_1$, $F_2$, $F_3$ of $T$ are long in $\Delta$, there must be one (triangular) facet $F^\Delta$ of $\Delta$ that meets all these facets. This easily follows from the fact that each of these facets intersects $\Delta$ in either a triangle or a quadrilateral, which yields at least 9 intersections between facets of $T$ and facets of $\Delta$. Since $\Delta$ has four facets, at least one of them must meet all three facets of $T$, as claimed. In addition, each of $F_1$, $F_2$, $F_3$ intersects $F^\Delta$ in a distinct pair of edges, as is easily verified; see Figure 6.4.

Let $C_i$ denote the convex polytope obtained from $C$ after $i-1$ steps of the Dobkin-Kirkpatrick hierarchical decomposition, for $i \geq 1$ (see [58] and earlier in this section).

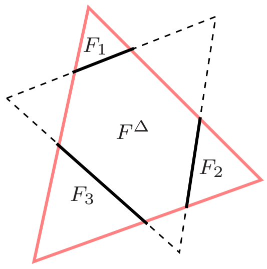


Figure 6.4: One facet $F_\Delta$ of $\Delta$ meets all three facets of $T$, in the depicted manner.

Recall that a simplicial subcell $\Delta$ is said to be generated at step $i$ if it has a vertex $v$ that is removed from $C_i$; that is, $v$ belongs to the independent set of vertices of $C_i$ collected at the $i$-th step. Recall also that $v$ and its adjacent edges and facets are removed, they leave a hole in $C_i$. The convex hull of the other vertices of that hole is constructed, and its (triangular) facets are connected to $v$ to form $O(1)$ simplices that fill up the corresponding gap between $C_i$ and $C_{i+1}$, and $\Delta$ is one of these simplices. Note that the three other vertices of $\Delta$ belong to $C_{i+1}$, and that all three edges of $\Delta$ incident to $v$ lie on the boundary of $C_i$. See Figure 6.1.

In what follows, we fix a decomposition step $i$, and show that there are only $O(1)$ simplices $\Delta$ of $C$ that are generated at step $i$ and have the properties in the lemma. The discussion above implies that for each such simplex $\Delta$, the corresponding facet $F^\Delta$ appears either on the boundary of $C_i$, or on the boundary of $C_{i+1}$, or as an "internal" facet of a hole of $C_i$ that is connected to the peeled-off vertex $v$ of $\Delta$, as described above.

Let $u$ denote the apex of $T$. By assumption, $u \notin C$. Let $\mathcal{F}^{(i)}$ denote the collection of all facets $F^\Delta$ of simplicial subcells $\Delta$ of $C$ that are generated at step $i$, such that $F^\Delta$ meets all three facets of $T$ and such that these facets are all long in $\Delta$. To simplify the analysis, we first prune away facets from $\mathcal{F}^{(i)}$, until $\mathcal{F}^{(i)}$ has the property that, for each peeled-off vertex $v$ of $C_i$ there is at most one simplex $\Delta$ incident to $v$, generated at step $i$, and contributing a facet to $\mathcal{F}^{(i)}$. By construction, this reduces the size of $\mathcal{F}^{(i)}$ by at most a constant factor.

We partition $\mathcal{F}^{(i)}$ into the following seven subcollections:

- $\mathcal{F}_1^{(i)}$, which consists of all facets of $C_i$ in $\mathcal{F}^{(i)}$ that are visible from $u$ (regarding $C_i$ itself as opaque); that is, the relative interiors of all the segments connecting $u$ to points on any $F^\Delta \in \mathcal{F}_1^{(i)}$ do not meet $\partial C_i$;
- $\mathcal{F}_2^{(i)}$, which consists of all facets of $C_i$ in $\mathcal{F}^{(i)}$ that are invisible from $u$; that is, all the segments connecting $u$ to any $F^\Delta \in \mathcal{F}_2^{(i)}$ cross $\partial C_i$ (once) before reaching $F^\Delta$;
- $\mathcal{F}_3^{(i)}$, which consists of all facets of $C_{i+1}$ in $\mathcal{F}^{(i)}$ that are not facets of $C_i$ and are visible from $u$ (regarding $C_{i+1}$ itself as opaque); that is, the relative interiors of all the segments connecting $u$ to points on any $F^\Delta \in \mathcal{F}_3^{(i)}$ do not meet $\partial C_{i+1}$; any such segment crosses $\partial C_i$ (once) before reaching $F^\Delta$;

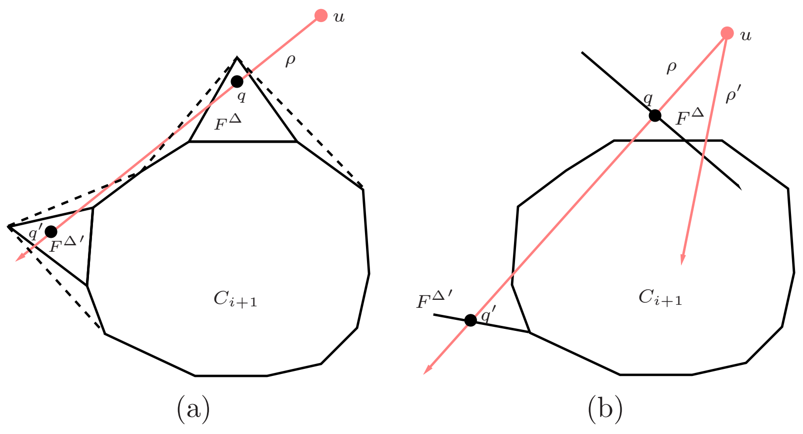


Figure 6.5: (a) The facets $F^{\Delta}$, $F^{\Delta'}$ are internal to two distinct holes generated at step $i$, and are fully visible from $u$ (with $C_{i+1}$ opaque). The ray $\rho$ that hits $F^{\Delta}$ at $q$ and then $F^{\Delta'}$ at $q'$ must cross $\partial C_i$ at least twice between $q$ and $q'$, which is impossible. (b) Illustrating the proof that no ray from $u$ can cross two distinct facets in $\mathcal{F}_7^{(i)}$.

• $\mathcal{F}_4^{(i)}$, which consists of all facets of $C_{i+1}$ in $\mathcal{F}^{(i)}$ that are not facets of $C_i$ and are invisible from $u$; that is, all the segments connecting $u$ to any $F^{\Delta} \in \mathcal{F}_4^{(i)}$ cross $\partial C_{i+1}$ (once) before reaching $F^{\Delta}$; as in the previous case, any such segment also crosses $\partial C_i$ (once) before reaching $F^{\Delta}$;

• $\mathcal{F}_5^{(i)}$, which consists of all facets in $\mathcal{F}^{(i)}$ that are internal to the holes (components of $C_i \setminus C_{i+1}$) generated at step $i$, and are fully visible from $u$ (in the presence of $C_{i+1}$ as an opaque object);

• $\mathcal{F}_6^{(i)}$, same as $\mathcal{F}_5^{(i)}$, but consisting of facets that are fully invisible from $u$ (fully occluded by $C_{i+1}$); and

• $\mathcal{F}_7^{(i)}$, same as $\mathcal{F}_5^{(i)}$, $\mathcal{F}_6^{\Delta}$, but consisting of facets that are partially visible from $u$ (partially occluded by $C_{i+1}$).

We next claim that each subset $\mathcal{F}_k^{(i)}$ consists of at most one facet. This implies that $\mathcal{F}^{(i)}$ has constant size, which, since the decomposition has only $O(\log r)$ steps, implies the bound stated in the lemma. We first need the following easy technical claim.

**Claim:** If we project the triangles of $\mathcal{F}_k^{(i)}$, for any fixed $1 \le k \le 7$, centrally from $u$, the projected triangles are pairwise disjoint.

**Proof:** The claim easily follows for $\mathcal{F}_1^{(i)}$, $\mathcal{F}_2^{(i)}$, $\mathcal{F}_3^{(i)}$, $\mathcal{F}_4^{(i)}$ by definition and by the convexity of $C_i$, $C_{i+1}$. Consider $\mathcal{F}_5^{(i)}$, and assume to the contrary that it contains two facets $F^{\Delta}$, $F^{\Delta'}$, such that a ray $\rho$ emanating from $u$ meets both of them, hitting, say, first $F^{\Delta}$ and then $F^{\Delta'}$ at two respective points $q$, $q'$. By the initial pruning process, $F^{\Delta}$ and $F^{\Delta'}$ lie in different holes of $C_i \setminus C_{i+1}$. By definition of $\mathcal{F}_5^{(i)}$, $qq'$ is disjoint from $C_{i+1}$, and is fully

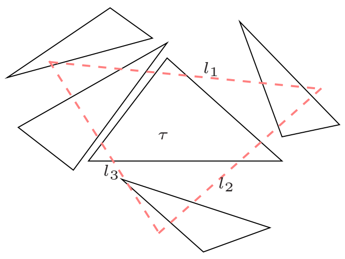


Figure 6.6: The centrally projected facets of some $\mathcal{F}_k^{(i)}$, and the central projections $l_1$, $l_2$, $l_3$ of the three respective facets $F_1$, $F_2$, $F_3$ of $T$. The triangle $\tau$ is the unique triangle that meets all three edges $l_1$, $l_2$, $l_3$.

contained in $C_i$, by convexity. This, however, is impossible, because $qq'$ has to cross from some hole of $C_i \setminus C_{i+1}$ to a different one, and the boundary of such a hole is contained in $\partial C_i \cup \partial C_{i+1}$, and thus $qq'$ must cross $\partial C_i$ (at least twice), a contradiction; see Figure 6.5(a) for an illustration.

The case of $\mathcal{F}_6^{(i)}$ is argued similarly. Here again $qq'$ is disjoint from $C_{i+1}$, because $uq$ must have already crossed $\partial C_{i+1}$ twice. Finally, for $\mathcal{F}_7^{(i)}$, we argue as follows. As above, the segment $qq'$ is fully contained in $C_i$ and crosses from one hole of $C_i \setminus C_{i+1}$ to another hole, so it must cross $\partial C_{i+1}$ twice. Since $F^\Delta$ is partially occluded by $C_{i+1}$, there exists another ray $\rho'$ from $u$ that first crosses $\partial C_{i+1}$ and then hits $F^\Delta$. This, however, is impossible, since it would have implied that $F^\Delta$ and $\partial C_{i+1}$ cross each other; this is proved by continuity, moving $\rho$ towards $\rho'$, within the plane that they span, and is illustrated in Figure 6.5(b). This completes the proof of the claim. $\square$

Let us now fix one of the subsets $\mathcal{F}_k^{(i)}$. The central projection of $\partial T$ from $u$ is a triangle whose three edges $l_1$, $l_2$, $l_3$ are the "head-on" projections of the respective facets $F_1$, $F_2$, $F_3$. Each facet $F^\Delta \in \mathcal{F}_k^{(i)}$ projects to a triangle that meets all three edges $l_1$, $l_2$, $l_3$. However, since the projections of the facets $F^\Delta$ of $\mathcal{F}_k^{(i)}$ are pairwise disjoint, at most one of them can touch all three edges $l_1$, $l_2$, $l_3$, as is easily checked; see Figure 6.6.

As argued above, this completes the proof of the lemma. $\square$

We have thus established the following theorem:

**Theorem 6.1.5** *For any tetrahedron $T$, the overall number of simplicial cells $\Delta$ of $\Xi$ that meet at least three facets of $T$ is $O(r \log^2 r)$.*

**Proof**: Lemma 6.1.3 shows that only $O(r \log r)$ cells $C$ of $\mathcal{A}(R)$ meet three facets of $T$. Of those, at most four contain an apex of $T$, and they have a total of $O(r \log r)$ sub-simplices. For any other cell, only $O(\log r)$ of its simplices have this property, as shown in Lemma 6.1.4. $\square$

**Remark:** With some additional care, the proof of Lemma 6.1.4 can be extended to the case where $C$ contains an apex of $T$. However, as just argued, the validity of Theorem 6.1.5

does not require this stronger property.

## 6.1.3 The overall recursive analysis

We now apply the following recursive scheme (a similar scheme has been presented in Chapter 3 (see also [72]), and for the sake of completeness we repeat some of these details). Each step of the analysis involves a simplex $\Delta_0$, which, in the initial step, is the entire 3-space, (or, rather, a sufficiently large simplex that contains all the vertices in the arrangement of the tetrahedra), and in further recursive steps is a cell of a cutting of some larger simplex, from the preceding recursive level.

We construct a $(1/r)$-cutting $\Xi$ of the arrangement of the planes that support facets of the tetrahedra that cross $\Delta_0$, using the Dobkin-Kirkpatrick hierarchical decomposition of each cell of the corresponding arrangement, as described above. Let $\Delta$ be a simplicial cell of $\Xi$ and let $\mathcal{D}^\Delta$ (resp., $\mathcal{T}^\Delta$) denote the set of D-tetrahedra (resp., T-tetrahedra) within $\Delta$. Put $N_D = N_D^\Delta := |\mathcal{D}^\Delta|$, and $N_T = N_T^\Delta := |\mathcal{T}^\Delta|$. (In the actual construction of the cutting we draw two respective random samples of $O(r \log r)$ planes, where the first sample is taken from the facets of the D-tetrahedra in $\Delta_0$, and second one is taken from the facets of the T-tetrahedra in $\Delta_0$. This guarantees, with high probability, the property that the number of D-tetrahedra (resp., T-tetrahedra) in each subcell of $\Delta_0$ is at most $\frac{|\mathcal{D}^{\Delta_0}|}{r}$ (resp., $\frac{|\mathcal{T}^{\Delta_0}|}{r}$)).

During each step of the recursion, after partitioning $\Delta_0$ into smaller subcells $\Delta$, we immediately dispose of any *new* DDD and DDT vertices within each subcell $\Delta$, and show that the overall number of these vertices is $O\left((N_D + N_T){N_T}^{1+\varepsilon}\right)$, for any $\varepsilon > 0$. These vertices are not considered during any further recursive substep, but only the remaining DTT and TTT vertices.

The recursion bottoms out when $N_T \le c$, for some absolute constant $c \ge 3$. In this case we bound the number of the remaining inner DTT and TTT vertices of the union in a brute-force manner, and thus obtain an overall bound of $O({N_T}^2 N_D + {N_T}^3) = O(N_D + 1)$ on the number of these vertices.

To bound the number of DDD vertices in $\Delta$, we replace each tetrahedron in $\mathcal{D}^\Delta$ by the equivalent halfspace or dihedral wedge, and face the problem of bounding the overall number of vertices appearing on the boundary of the union of $N_D$ halfspaces and $\alpha'$-fat dihedral wedges[4] (where, as argued earlier, $\alpha' > 0$ is a constant that depends only on $\alpha$). As shown in [112], the number of such vertices is $O(N_D^{2+\varepsilon})$, for any $\varepsilon > 0$, where the constant of proportionality depends on $\varepsilon$ and $\alpha$. In an additional major step of the analysis, we derive, in Section 6.1.4 below, a similar bound on the number of DDT-vertices.

**The recursive scheme.** With all this machinery at hand, we can now proceed to the proof of Theorem 6.1.1. The analysis begins with the initial cell $\Delta_0$, with $N_T^{\Delta_0} = n$,

[4]Clearly, any DDD-vertex of the full union in $\Delta$ is also a vertex of the union of $\mathcal{D}^\Delta$.

$N_D^{\Delta_0} = 0$, and recursively subdivides $\Delta_0$, using a $(1/r)$-cutting of the preceding kind, for some sufficiently large constant parameter $r$. (See Chapter 3 for similar considerations.)

For simplicity, write in what follows $n_T = N_T^{\Delta_0}$, and $n_D = N_D^{\Delta_0}$. Theorem 6.1.5 implies that there are only $O(rn_T \log^2 r)$ crossings between the cells of $\Xi$ and their T-tetrahedra. It thus follows that, for any $r^2 \log^2 r \le s \le M = O(r^3 \log^3 r)$, where $M$ is the size of $\Xi$, the number of cells in $\Xi$ that are crossed by at least $\frac{rn_T \log^2 r}{s}$ T-tetrahedra is at most $O(s)$. (The case $s < r^2 \log^2 r$ cannot arise, since each cell of $\Xi$ is intersected by at most $\frac{n_T}{r}$ tetrahedra of $\mathcal{T}^{\Delta_0}$.) We partition the set of cells of $\Xi$ into at most $\log\left(\frac{M}{r^2 \log^2 r}\right) = \Theta(\log r)$ subsets, so that the $i$-th subset $\Xi_i$ contains $O(2^i r^2 \log^2 r)$ cells $\Delta$ of $\Xi$, each of which satisfies

$$\frac{n_T}{2^i r} \le N_T^{\Delta} = |\mathcal{T}^{\Delta}| \le \frac{2n_T}{2^i r},$$

for $i = 1, \ldots, \log\left(\frac{M}{r^2 \log^2 r}\right)$. Note that $|\mathcal{T}^{\Delta}| = O\left(\frac{n_T}{r^2}\right)$ for each of the $O(r^3 \log^3 r)$ cells $\Delta$ in the last subset. For the number of D-tetrahedra in any cell $\Delta$, we use the bound $N_D^{\Delta} = |\mathcal{D}^{\Delta}| \le \frac{n_D + n_T}{r}$, for each $\Delta \in \Xi$.

As in Chapter 3 (and [72]), we recurse in each cell $\Delta$ of $\Xi$, where the goal of the recursive step at $\Delta$ is to obtain an upper bound for the number of DTT and TTT vertices in $\Delta$ (including vertices of these kinds that appear on $\partial\Delta$). Thus, before entering the recursion, we need to bound the number of *new* DDD and DDT vertices within $\Delta$ (or on its boundary). These are vertices $v$ that were DTT or TTT vertices at the parent cell $\Delta_0$ of $\Delta$, but have become DDT or DDD vertices at $\Delta$. Partition $\mathcal{D}^{\Delta_0}$ into $k = \lceil n_D / n_T \rceil$ subsets $\mathcal{D}_1, \ldots, \mathcal{D}_k$, each consisting of at most $n_T$ D-tetrahedra. The preceding observations imply that any new DDD or DDT vertex in $\Delta$ must be a DDD or a DDT vertex of the union of the wedges and tetrahedra of $\mathcal{D}_j \cup \mathcal{T}^{\Delta_0}$, clipped to $\Delta$, for some $j = 1, \ldots, k$ (because it was a DTT or a TTT-vertex in $\Delta_0$, and so involves at most one wedge of $\mathcal{D}^{\Delta_0}$). By the results of [112] and of Section 6.1.4, the number of such vertices, for any fixed $j$, is thus $O({n_T}^{2+\varepsilon})$, for any $\varepsilon > 0$. Hence, summing over $j = 1, \ldots, k$, the overall number of new DDD and DDT vertices in $\Delta$ is $O(k{n_T}^{2+\varepsilon}) = O((n_D + n_T){n_T}^{1+\varepsilon})$, for any $\varepsilon > 0$. Repeating the analysis to each subcell $\Delta$ of $\Delta_0$, and recalling that $r$ is a constant, the overall number of new DDD and DDT vertices within the children of $\Delta_0$ is

$$O(r^3 \log^3 r \cdot (n_D + n_T){n_T}^{1+\varepsilon}) = O((n_D + n_T){n_T}^{1+\varepsilon}),$$

for any $\varepsilon > 0$.

Let $U(N_T, N_D)$ denote the maximum number of DTT- and TTT-vertices that appear on the boundary of the union at a recursive step involving up to $N_T$ T-tetrahedra and $N_D$

D-tetrahedra. Then $U$ satisfies the following recurrence:

$$U(N_T, N_D) \le \begin{cases} O\left((N_D + N_T)N_T^{1+\varepsilon}\right) \\ + \sum_{i=0}^{\log\left(\frac{M}{r^2 \log^2 r}\right)} O(2^i r^2 \log^2 r) U\left(\frac{2N_T}{2^i r}, \frac{N_D + N_T}{r}\right), & \text{if } N_T > c, \\ \\ O(N_D + 1), & \text{if } N_T \le c, \end{cases}$$

where $\varepsilon > 0$ is arbitrary, $c \ge 3$ is an appropriate constant, and where the constant of proportionality in the first expression depends on ($\varepsilon$, $\alpha$ and on) $r$. It is straightforward to verify (see also Chapter 3 and [72]), that the solution of this recurrence is $U(N_T, N_D) = O(N_T(N_T + N_D)^{1+\varepsilon})$, for any $\varepsilon > 0$, with a constant of proportionality that depends on $\varepsilon$ and on $\alpha$.

Substituting the initial values $N_T = n$, $N_D = 0$, we conclude that the overall combinatorial complexity of the union is $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$, as asserted. This completes the proof of Theorem 6.1.1. modulo the still missing analysis of the number of DDT-vertices.

Since a cube in 3-space can be partitioned into a constant number of $\alpha$-fat tetrahedra, for some appropriate constant parameter $\alpha > 0$, we obtain the following extension of the bound in [112]:

**Corollary 6.1.6** *The complexity of the union of $n$ arbitrarily oriented cubes in $\mathbb{R}^3$, of arbitrary side lengths, is $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$, where the constant of proportionality depends on $\varepsilon$.*

Similar results can be obtained for any collection of polyhedral objects that can be decomposed into, or covered by, a total of $n$ $\alpha$-fat tetrahedra.

A similar, almost verbatim, analysis yields the bound $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$, for the complexity of the union of $n$ $\alpha$-fat trihedral wedges. For the sake of completeness, we state this result explicitly:

**Corollary 6.1.7** *The complexity of the union of $n$ $\alpha$-fat trihedral wedges is $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$, where the constant of proportionality depends on $\varepsilon$ and $\alpha$.*

Our analysis can easily be extended to the problem of bounding the complexity of the union of $n$ $\alpha$-fat arbitrarily oriented triangular prisms, with cross-sections of arbitrary sizes. In this case, we apply a similar decomposition scheme as in the case of $\alpha$-fat tetrahedra, exploiting similar properties to those stated in Lemmas 6.1.3 and 6.1.4 for the case where $T$ is a triangular prism (rather than a trihedral wedge) — simply think of a prism as a wedge with apex at infinity. A lower bound construction that yields $\Omega(n^2\alpha(n))$ vertices on the boundary of the union can be supplied using similar techniques as in the original problem. We thus obtain:

**Corollary 6.1.8** *The complexity of the union of $n$ arbitrarily oriented $\alpha$-fat triangular prisms, with cross-sections of arbitrary sizes, in $\mathbb{R}^3$, is $O(n^{2+\varepsilon})$, for any $\varepsilon > 0$, where the constant of proportionality depends on $\varepsilon$ and $\alpha$. The bound is almost tight in the worst case.*

Similar results can be obtained for any collection of polyhedral prisms that can be decomposed into, or covered by a total of $n$ $\alpha$-fat triangular prisms.

### 6.1.4 The number of DDT-vertices

In this section we provide the missing ingredient of the preceding analysis, showing that the number of DDT-vertices is nearly-quadratic.

We thus have, at each step of the analysis, a simplex $\Delta$, a set $\mathcal{D} = \mathcal{D}^{\Delta}$ of $N_D$ $\alpha'$-fat dihedral wedges, and a set $\mathcal{T} = \mathcal{T}^{\Delta}$ of $N_T$ $\alpha$-fat tetrahedra[5]. Our goal is to obtain a nearly-quadratic bound on the number of DDT vertices on the boundary of the union of $\mathcal{D} \cup \mathcal{T}$ within $\Delta$. We may assume that $N_T \leq N_D$. Otherwise, we apply (in "reverse") the partitioning trick used in the preceding analysis. That is, we partition $\mathcal{T}$ arbitrarily into $k = \lceil N_T/N_D \rceil$ subsets $\mathcal{T}_1, \ldots, \mathcal{T}_k$, of size at most $N_D$ each, establish a bound $O(N_D^{2+\varepsilon})$ on the number of DDT-vertices on each union $\mathcal{D} \cup \mathcal{T}_i$, $i = 1, \ldots, k$, and add up the bounds, to obtain an overall bound of $O(N_T N_D^{1+\varepsilon})$. In other words, the goal is to establish the upper bound $O(N_D^{2+\varepsilon})$ on the number of DDT-vertices, assuming that $N_T \leq N_D$.

Let $L$ denote the set of the $xy$-projections of the edges (lines) of the wedges in $\mathcal{D}$. (We assume that the coordinate system is generic, so none of these lines projects to a single point.) We construct a $(1/r)$-cutting $\Xi$ of the planar arrangement $\mathcal{A}(L)$, by taking a random sample $R$ of $O(r \log r)$ lines of $L$, for some sufficiently large constant parameter $r$, constructing the planar arrangement $\mathcal{A}(R)$, and triangulating each of its cells using the two-dimensional version of the Dobkin-Kirkpatrick hierarchical decomposition of a convex polygon [58]. We obtain $O(r^2 \log^2 r)$ triangles, and we may assume that the sample $R$ is such that each triangle is crossed by at most $N_D/r$ lines of $L$ (this indeed happens with high probability). We lift each cell of $\mathcal{A}(R)$, and each of its sub-triangles, into a vertical prism (or rather its portion within the current simplex $\Delta$ ). Each triangular prism $\sigma$ is crossed by at most $N_D/r$ edges of the wedges of $\mathcal{D}$. Any other wedge either misses $\sigma$ altogether, or each of its bounding halfplanes that meets $\sigma$ crosses $\sigma$ completely, cutting it into two disconnected pieces (as if it were a plane).

We now claim that, given a fixed tetrahedron $T \in \mathcal{T}$, the overall number of vertical triangular prisms $\sigma$, erected over the cells of $\Xi$, such that $\sigma$ meets at least three facets of $T$, is $O(r \log^2 r)$. Indeed, a facet $F$ of $T$ whose bounding edges do not meet $\sigma$ must intersect $\sigma$ in a triangle whose boundary is contained in $\partial\sigma$. Since $T \cap \sigma$ is convex, there can be

[5]Some of these tetrahedra may be trihedral wedges, in case only three of the facets of a tetrahedron $T$ appear in $\Delta$. However, to simplify the presentation, we will refer in what follows to all the elements of $\mathcal{T}$ as tetrahedra.

at most two such facets. Hence $\sigma$ meets at least one of the edges $e$ of $T$. That is, the projection $e^*$ of $e$ crosses the triangular cell of $\Xi$ which is the base of $\sigma$. However, applying a simplified version of the argument used above for 3-dimensional arrangements, $e^*$ can cross only $O(r \log r)$ cells of $\mathcal{A}(R)$, and only $O(\log r)$ triangles within each cell, from which the claim follows.

Summing up, we have so far $O(r^2 \log^2 r)$ subproblems, each defined within a triangular prism $\sigma$, and involves the following sets of objects: (i) The set $\mathcal{D}^\sigma$ of dihedral wedges of $\mathcal{D}$ whose edges cross $\sigma$; (ii) the set $\mathcal{P}^\sigma$ of dihedral wedges of $\mathcal{D}$ that cross $\sigma$ but whose edges do not cross $\sigma$ (we can think of each member of $\mathcal{P}^\sigma$ as either a halfspace or a region enclosed between a pair of planes crossing $\sigma$); (iii) the set $\mathcal{T}^\sigma$ of tetrahedra such that at least three of their facets cross $\sigma$; and (iv) the set $\mathcal{W}^\sigma$ of tetrahedra that cross $\sigma$, and at most two of their facets cross $\sigma$. Put $N_{\mathcal{D}^\sigma} := |\mathcal{D}^\sigma|$, $N_{\mathcal{P}^\sigma} := |\mathcal{P}^\sigma|$, $N_{\mathcal{T}^\sigma} := |\mathcal{T}^\sigma|$, and $N_{\mathcal{W}^\sigma} := |\mathcal{W}^\sigma|$. As just argued, $\sum_\sigma N_{\mathcal{T}^\sigma} = O(N_T \cdot r \log^2 r)$.

The goal is to bound the number of inner vertices $v$, within $\sigma$, of the union $\mathcal{D}^\sigma \cup \mathcal{P}^\sigma \cup \mathcal{T}^\sigma \cup \mathcal{W}^\sigma$, such that $v$ is incident to the boundaries of two objects in $\mathcal{D}^\sigma \cup \mathcal{P}^\sigma$, and of one object in $\mathcal{T}^\sigma \cup \mathcal{W}^\sigma$. We classify each inner vertex $v$ in $\sigma$ of this kind as either *DDW*, if $v$ is incident to the boundaries of two objects in $\mathcal{D}^\sigma$ and one object in $\mathcal{W}^\sigma$, *DPW*, if the three objects whose boundaries are incident to $v$ are in $\mathcal{D}^\sigma$, $\mathcal{P}^\sigma$, and $\mathcal{W}^\sigma$, respectively, *PPW*, if two of these objects are in $\mathcal{P}^\sigma$ and one in $\mathcal{W}^\sigma$, *DDT*, if two of these objects are in $\mathcal{D}^\sigma$ and one in $\mathcal{T}^\sigma$, *DPT*, if the objects are in $\mathcal{D}^\sigma$, $\mathcal{P}^\sigma$, and $\mathcal{T}^\sigma$, respectively, or *PPT*, if two of the objects are in $\mathcal{P}^\sigma$ and one in $\mathcal{T}^\sigma$.

Since each vertex of type DDW, DPW, or PPW lies on the boundary of the union of $\alpha'$-fat dihedral wedges (or halfspaces), it follows, by applying the results of [112], that the number of such vertices is $O((N_{\mathcal{D}^\sigma} + N_{\mathcal{P}^\sigma} + N_{\mathcal{W}^\sigma})^{2+\varepsilon})$, for any $\varepsilon > 0$. Summing over all prisms $\sigma$, and using the facts that $r$ is constant and that $N_T \le N_D$, we obtain the overall bound $O(N_D^{2+\varepsilon})$, for any $\varepsilon > 0$. The general bound, using the partitioning trick described above, is $O((N_D + N_T)N_D^{1+\varepsilon})$, for any $\varepsilon > 0$. We next show how to bound the number of the remaining types of vertices.

Assume for the moment that we have managed to establish a nearly-quadratic bound, of the form $O((N_D + N_T)N_D^{1+\varepsilon})$, for any $\varepsilon > 0$, on the number of PPT-vertices and DPT-vertices (which will be accomplished in the next two steps of the analysis). Then we are only left with the task of bounding the number of DDT-vertices within $\sigma$, which we do recursively. To recap, there are $O(r^2 \log^2 r)$ such recursive subproblems, over all triangular prisms $\sigma$, in each of which we apply the nearly-quadratic bound on the number of vertices of all the remaining types, and continue to bound the number of DDT-vertices in a recursive manner.

The recursion bottoms out when either $N_T \le c$ or $N_D \le c$, for some absolute constant $c \ge 3$. We then bound the number of the remaining vertices of the union under consideration (that is, inner vertices whose type is still DDT) in a brute-force manner, and thus obtain an overall bound of $O({N_D}^2 N_T) = O(N_D^2 + N_T)$ on the number of these vertices.

Let $U_1(N_T, N_D)$ denote the maximum number of DDT-vertices that appear on the boundary of the union at a recursive step involving $N_D$ dihedral wedges and $N_T$ tetrahedra. Then $U_1$ satisfies the recurrence:

$$U_1(N_T, N_D) \le \begin{cases} O\left((N_D + N_T)N_D^{1+\varepsilon}\right) + \\ \sum_{i=0}^{\log\left(\frac{M}{r\log^2 r}\right)} O(2^i r \log^2 r) U_1\left(\frac{2N_T}{2^i}, \frac{N_D}{r}\right), & \text{if } \min\{N_T, N_D\} > c, \\ \\ O(N_D^2 + N_T), & \text{if } \min\{N_T, N_D\} \le c, \end{cases} \tag{6.1}$$

where $\varepsilon > 0$ is arbitrary, $c \ge 3$ is an appropriate constant, $M = O(r^2 \log^2 r)$ is the overall number of prisms in the decomposition, and where the constants of proportionality in the non-recursive terms depend on $r$ (and on $\varepsilon$, $\alpha$). This follows from the fact that the number of prisms $\sigma$ with $\frac{N_T}{2^i} \le N_{\mathcal{T}^\sigma} < \frac{2N_T}{2^i}$ is at most $O(2^i r \log^2 r)$. As above, it is easy to verify that the solution of this recurrence is $U_1(N_T, N_D) = O((N_D + N_T)N_D^{1+\varepsilon})$, for any $\varepsilon > 0$, with a constant of proportionality that depends on $\varepsilon$ and on $\alpha$.

**The number of PPT-vertices.** To bound the number of PPT vertices, we launch a new recursive analysis, which, as the analysis in Section 6.1.3, is based on cuttings in arrangements of planes in 3-space. Recycling for the moment some of the previous notations, we have, at each step, a subproblem within some simplex $\Delta_0$, involving a set $\mathcal{P} = \mathcal{P}^{\Delta_0}$ of pairs of planes, at least one of which crosses $\Delta_0$ (but their intersection line does not cross $\Delta_0$), and a set $\mathcal{T} = \mathcal{T}^{\Delta_0}$ of tetrahedra, so that, for each $T \in \mathcal{T}$, at least three of its facets cross $\Delta_0$. Put $N_P = |\mathcal{P}^{\Delta_0}|$, $N_T = |\mathcal{T}^{\Delta_0}|$. Initially, $\Delta_0$ is the (clipped) vertical triangular prism $\sigma$ of some specific recursive instance of the above recursion that handles DDT vertices.

We first draw a random sample $R \subset \mathcal{P}^{\Delta_0}$ of $O(r \log r)$ pairs of planes, for some sufficiently large constant parameter $r$, and construct the sampled arrangement $\mathcal{A}(R)$ within $\Delta_0$. We then collect only the cells in the complement of the union of the wedges enclosed between each sampled pair of planes (within $\Delta_0$). Since the wedges are all $\alpha'$-fat, the analysis of [112] implies that the overall number of these cells is $O(r^{2+\varepsilon})$, for any $\varepsilon > 0$. Furthermore, since $R$ is a collection of planes within $\Delta_0$, each cell that we consider is a convex (possibly unbounded) polyhedron. We next triangulate each of these cells $C$ using the Dobkin-Kirkpatrick hierarchical decomposition of convex polytopes (see the beginning of this section and [58]), and obtain an overall number of $O(r^{2+\varepsilon})$ simplicial subcells. Using similar considerations as in the original problem, we may assume that each simplicial cell $\Delta$ of the resulting decomposition is crossed by at most $N_P/r$ wedge boundaries (pairs of planes) in $\mathcal{P}^{\Delta_0}$, and each edge of any tetrahedron in $\mathcal{T}^{\Delta_0}$ crosses at most $O(r \log^2 r)$ cells. In addition, Theorem 6.1.5 implies that the overall number of simplicial cells, each of which meets at least three facets of any fixed tetrahedron $T \in \mathcal{T}^{\Delta_0}$, is $O(r \log^2 r)$.

Note that, in this problem, the decomposition generates only two types of vertices, PPW and PPT. We thus apply the above decomposition recursively, where we dispose immediately (i.e., derive a nearly-quadratic bound on the number) of all PPW-vertices within each cell $\Delta$ of the decomposition, and bound the number of the (remaining) PPT-vertices recursively. At the bottom of the recurrence we bound the number of PPT-vertices by brute force, as above. An appropriate variant of the preceding analysis leads to a recurrence relationship similar to (6.1), with the difference that (i) $N_P$ replaces $N_D$, and (ii) the upper bound on $M$ is $O(r^{2+\varepsilon})$, rather than $O(r^2 \log^2 r)$; this, however, has no effect on the asymptotic solution of the recurrence. That is, we obtain that the maximum number of PPT-vertices that appear on the boundary of the union at a recursive step, involving $N_P$ $\alpha'$-fat dihedral wedges (which behave like pairs of planes) and $N_T$ $\alpha$-fat tetrahedra, is $O((N_P + N_T)N_P^{1+\varepsilon})$, for any $\varepsilon > 0$.

**The number of DPT-vertices.** Here too we bound the number of DPT-vertices using a separate recursive analysis, where, at each step, we have a subproblem within some simplex $\Delta_0$, involving a set $\mathcal{D} = \mathcal{D}^{\Delta_0}$ of dihedral wedges whose boundary edges cross $\Delta_0$, a set $\mathcal{P} = \mathcal{P}^{\Delta_0}$ of pairs of planes, at least one of which crosses $\Delta_0$, and a set $\mathcal{T} = \mathcal{T}^{\Delta_0}$ of tetrahedra, so that, for each $T \in \mathcal{T}$, at least three of its facets cross $\Delta_0$. Put $N_D = |\mathcal{D}^{\Delta_0}|$, $N_P = |\mathcal{P}^{\Delta_0}|$, and $N_T = |\mathcal{T}^{\Delta_0}|$. Initially, $\Delta_0$ is a (clipped) vertical triangular prism, as above.

We choose some sufficiently large constant parameter $r$, and draw three random samples, each of which consists of $O(r \log r)$ planes, which contain the facets of the wedges of $\mathcal{D}^{\Delta_0}$, the facets of the wedges of $\mathcal{P}^{\Delta_0}$, and the facets of the tetrahedra of $\mathcal{T}^{\Delta_0}$, respectively. Let $R$ denote the union of the three samples. We form the arrangement $\mathcal{A}(R)$, and triangulate each of its cells, using, as usual, the Dobkin-Kirkpatrick hierarchical decomposition.

We obtain $O(r^3 \log^3 r)$ simplicial cells in the decomposition. Assuming that the drawn samples are good, we may assume consequently that each of these cells $\Delta$ is crossed by at most $N_D/r$ dihedral wedges of $\mathcal{D}^{\Delta_0}$, at most $N_P/r$ dihedral wedges (bounded by the pairs of planes) of $\mathcal{P}^{\Delta_0}$, and at most $N_T/r$ tetrahedra of $\mathcal{T}^{\Delta_0}$. Each edge of any tetrahedron in $\mathcal{T}^{\Delta_0}$ crosses at most $O(r \log^2 r)$ cells, and the overall number of simplicial cells, each of which meets at least three facets of a fixed tetrahedron $T \in \mathcal{T}^{\Delta_0}$, is $O(r \log^2 r)$. Each edge $\ell$ of a dihedral wedge of $\mathcal{D}^{\Delta_0}$ crosses only $O(r \log^2 r)$ cells, so the overall number of wedge-cell crossings, for which the edge of the wedge appears in the cell, is $O(N_D \cdot r \log^2 r)$. In any other crossing of a cell $\Delta$ by a dihedral wedge, the wedge behaves like a pair of planes, at least one of which crosses $\Delta$.

The decomposition therefore generates, within each simplicial cell $\Delta$, vertices of type PPW, DPW, PPT, and DPT. The number of vertices of the first three types is nearly-quadratic, by the bound in [112] and by the preceding analysis, and the number of DPT-vertices, within each cell $\Delta$, is bounded recursively. As in the original recursive scheme, presented in Section 6.1.3, at each step of the recursion, we only need to bound the number of new PPW, DPW, and PPT vertices within $\Delta$. Each such vertex is incident to the boundary of a dihedral wedge in $\mathcal{D}^{\Delta_0}$, a plane in (a pair in) $\mathcal{P}^{\Delta_0}$, and a tetrahedron in $\mathcal{T}^{\Delta_0}$. We can refine the quadratic bound on the number of these vertices using the following variant of the partition trick. That is, suppose, without loss of generality, that $N_D \le N_P \le N_T$. Partition $\mathcal{T}^{\Delta_0}$ into $\lceil N_T/N_P \rceil$ sets, each of size at most $N_P$. Bound separately the number of DPT vertices of the above kind for $\mathcal{D}^{\Delta_0}$, $\mathcal{P}^{\Delta_0}$, and each subset of $\mathcal{T}^{\Delta_0}$. The bound is $O(N_P^{2+\varepsilon})$, for any $\varepsilon > 0$, within each of these subproblems, for a total of

$$O\left(N_P^{2+\varepsilon} \cdot \frac{N_T}{N_P}\right) = O(N_T N_P^{1+\varepsilon}),$$

for any $\varepsilon > 0$. Considering also the other symmetric cases, we have thus shown that the number of new PPW, DPW, and PPT vertices is

$$O\left((N_D N_P + N_D N_T + N_P N_T) \cdot \max\{N_D, N_P, N_T\}^{\varepsilon}\right),$$

for any $\varepsilon > 0$.

We do not have to process in further recursive steps dihedral wedges of $\mathcal{D}^{\Delta_0}$ whose edge does not meet the current cell $\Delta$, as well as tetrahedra of $\mathcal{T}^{\Delta_0}$ with at most two facets meeting $\Delta$, since the DPT-vertices that they induce have already been counted.

At the bottom of the recursion (when $\min\{N_D, N_P, N_T\} \le c$, for some absolute constant $c \ge 3$), we bound the number of the remaining inner DPT-vertices of the union in a brute-force manner, and thus obtain an overall bound of $O(N_D N_P N_T) = O(N_D N_P + N_D N_T + N_P N_T)$ on this number.

Let $U_2(N_D, N_P, N_T)$ denote the maximum number of DPT-vertices that appear on the boundary of the union at a recursive step within some simplex $\Delta_0$, involving a set $\mathcal{D}^{\Delta_0}$ of $N_D$ dihedral wedges, a set $\mathcal{P}^{\Delta_0}$ of $N_P$ pairs of planes (dihedral wedges whose edge does not cross $\Delta_0$), and a set $\mathcal{T}^{\Delta_0}$ of $N_T$ tetrahedra. For each $i \ge 1$, consider those cells $\Delta$ for which

$$\frac{1}{2^i r} < \max\left\{\frac{N_{\mathcal{D}^\Delta}}{N_D}, \frac{N_{\mathcal{T}^\Delta}}{N_T}\right\} \le \frac{1}{2^{i-1} r}.$$

Since, as argued above, $\sum_\Delta N_{\mathcal{D}^\Delta} = O(N_D \cdot r \log^2 r)$, and $\sum_\Delta N_{\mathcal{T}^\Delta} = O(N_T \cdot r \log^2 r)$, it follows that the number of cells $\Delta$, satisfying the above inequalities, is $O(2^i r^2 \log^2 r)$. Moreover, all cells appear in these counts, because, by construction, we have $N_{\mathcal{D}^\Delta} \le N_D/r$ and $N_{\mathcal{T}^\Delta} \le N_T/r$, for each cell $\Delta$. Hence, putting $N = \min\{N_D, N_P, N_T\}$, $U_2$ satisfies the following recurrence:

$$U_2(N_D, N_P, N_T) \le \begin{cases} O\left((N_D N_P + N_D N_T + N_P N_T) \cdot \max\{N_D, N_P, N_T\}^{\varepsilon}\right) + \\ \sum_{i=0}^{\log\left(\frac{M}{r^2 \log^2 r}\right)} O(2^i r^2 \log^2 r) U_2\left(\frac{2N_D}{2^i r}, \frac{N_P}{r}, \frac{2N_T}{2^i r}\right), & \text{if } N > c, \\ \\ O(N_D N_P + N_D N_T + N_P N_T), & \text{if } N \le c, \end{cases}$$

where $\varepsilon > 0$ is arbitrary, $c \geq 3$ is an appropriate constant, $M = O(r^3 \log^3 r)$ is the overall number of cells in the decomposition (at the current recursive step), and the non-recursive terms also depend on $r$ (and on $\varepsilon$, $\alpha$). Again, it is easy to verify that the solution of this recurrence is $U_2(N_D, N_P, N_T) = O\left((N_D N_P + N_D N_T + N_P N_T) \cdot \max\{N_D, N_P, N_T\}^{\varepsilon}\right)$, for any $\varepsilon > 0$, with a constant of proportionality that depends on $\varepsilon$ and on $\alpha$ (see also Chapter 3 and [72] for similar considerations).

This finally completes the analysis, and establishes Theorem 6.1.1. $\square$

## 6.2 The Union of $\alpha$-Fat Triangles in the Plane

Let $\mathcal{T}$ be a collection of $n$ $\alpha$-fat triangles in the plane (i.e., each angle of any triangle is at least $\alpha$). In this section we follow a simple variant of our approach to the three-dimensional problem, and derive a nearly-linear bound on the combinatorial complexity of the union of the triangles in $\mathcal{T}$.

We first draw a random sample $R$ of $O(r \log r)$ of the lines containing the edges of the triangles in $\mathcal{T}$, for some sufficiently large constant parameter $r$, and form the arrangement $\mathcal{A}(R)$. We then triangulate each cell of the arrangement, using, e.g., bottom-vertex triangulation. The number of the resulting triangles (we will call them simplices, to avoid confusion with the triangles of $\mathcal{T}$) is $M = O(r^2 \log^2 r)$, and, with high probability, each of these simplices is crossed by at most $n/r$ edges of the triangles in $\mathcal{T}$. We can therefore assume that our sample has this property. Thus the resulting decomposition is a $(1/r)$-cutting for the edges of $\mathcal{T}$, and the simplices are the cells of this cutting.

Similarly to the original problem, we fix a triangle $T \in \mathcal{T}$, and a cell $\Delta$ of the cutting that $T$ meets, and classify $T$ as being either a *W-triangle* in $\Delta$, if $\Delta$ meets only one or two edges of $T$, or a *T-triangle* in $\Delta$, if $\Delta$ meets all the three edges of $T$. As a consequence, each intersection vertex $v$ of the union boundary that appears in $\Delta$ is classified as being either *WW*, if the two edges that are incident to $v$ belong to two respective W-triangles in $\Delta$, *WT*, if one of these edges belongs to a W-triangle and the other belongs to a T-triangle, or *TT*, if both of these edges belong to two respective T-triangles. (In all three cases, the relevant triangles are distinct.)

We next observe the easy fact that, for any triangle $T \in \mathcal{T}$, there is only a *single* cell of the cutting that meets all three edges of $T$; see Lemma 6.1.4 and Figure 6.6.

We now apply a recursive scheme, similar to that used in the three-dimensional setup. Let $\Delta$ be a simplex of the cutting and let $\mathcal{W}^{\Delta}$ (resp., $\mathcal{T}^{\Delta}$) denote the set of W-triangles (resp., T-triangles) within $\Delta$. Put $N_W^{\Delta} := |\mathcal{W}^{\Delta}|$, and $N_T^{\Delta} := |\mathcal{T}^{\Delta}|$. The preceding observation implies that $\sum_{\Delta} N_T^{\Delta} \leq N_T$, where $N_T$ is the overall number of triangles.

During each step of the recursion, we immediately dispose of any new WW- and WT-vertices within each subcell $\Delta$, and continue to bound the number of TT-vertices recursively. The recursion bottoms out when $N_T \leq c$, for some absolute constant $c \geq 2$. In this case the number of the remaining intersection vertices of the union (within the current $\Delta$) is $O(1)$.

To bound the number of WW-vertices in $\Delta$, we replace each triangle in $\mathcal{W}^\Delta$ by the equivalent halfplane or wedge, and face the problem of bounding the overall number of vertices appearing on the boundary of the union of $N_W^\Delta$ halfplanes and *$\alpha$-fat* wedges (that is, wedges whose angle is at least $\alpha$). As shown in [64], the number of such vertices is $O(N_W^\Delta)$. As we will shortly show, the number of WT-vertices in $\Delta$ is $O\left((N_T^\Delta + N_W^\Delta) \cdot (N_W^\Delta)^\varepsilon\right)$, for any $\varepsilon > 0$. Summing over all $\Delta$, and using the fact that $r$ is a constant, we get the overall bound $O(N_T^{1+\varepsilon})$, for any $\varepsilon > 0$.

Let $U_1(N_T)$ denote the maximum number of intersection vertices that appear on the boundary of the union at a recursive step involving $N_T$ triangles. For each $1 \le i \le \log(M/r)$, where $M = O(r^2 \log^2 r)$ is the overall number of cells in the cutting, the number of cells $\Delta$ with $\frac{N_T}{r2^i} < N_T^\Delta \le \frac{2N_T}{r2^i}$ is at most $2^i r$, and this also holds for the set of all the remaining cells (with $N_T^\Delta \le \frac{2N_T}{r2^i}$, where $i = \log(M/r) + 1$). Recall also that $N_T^\Delta, N_W^\Delta \le \frac{N_T}{r}$ always holds, by construction. Hence $U_1$ satisfies the recurrence:

$$U_1(N_T) \le \begin{cases} O\left(N_T^{1+\varepsilon}\right) + \sum_{i=1}^{1+\log\left(\frac{M}{r}\right)} 2^i r \cdot U_1\left(\frac{N_T}{r2^{i-1}}\right), & \text{if } N_T > c, \\ \\ O(1), & \text{if } N_T \le c, \end{cases}$$

where $\varepsilon > 0$ is arbitrary, $c \ge 2$ is an appropriate constant, and the constants of proportionality in the non-recursive terms depend on $r$ (and on $\varepsilon$, $\alpha$). Note that we process recursively only the T-triangles, since vertices incident to W-triangles are estimated and discarded immediately before processing the new recursive step.

It is easy to verify that the solution of this recurrence is $U_1(N_T) = O(N_T^{1+\varepsilon})$, for any $\varepsilon > 0$ (slightly larger than the $\varepsilon$ in the non-recursive term, but still arbitrarily close to 0), with a constant of proportionality that depends on $\varepsilon$ and on $\alpha$.

**The number of WT-vertices.** To complete the analysis, we next establish a near-linear bound on the number of WT-vertices. The analysis somewhat resembles the derivation of the upper bound on the number of PPT-vertices in Section 6.1.4. We have, at each step, a subproblem within some simplex $\Delta_0$, involving a set $\mathcal{W} = \mathcal{W}^{\Delta_0}$ of wedges, whose boundaries cross $\Delta_0$, and a set $\mathcal{T} = \mathcal{T}^{\Delta_0}$ of triangles, so that, for each $T \in \mathcal{T}$, all three of its edges cross $\Delta_0$. Put $N_W = |\mathcal{W}^{\Delta_0}|$, $N_T = |\mathcal{T}^{\Delta_0}|$.

We first draw a random sample $R \subset \mathcal{W}^{\Delta_0}$ of $O(r \log r)$ wedges, for some sufficiently large constant parameter $r$, form and triangulate the arrangement $\mathcal{A}(R)$, and collect only the cells in the complement of the union of these wedges. Since the wedges are all $\alpha$-fat, it follows by the analysis of [64] that the overall number of these cells is $O(r \log r)$ (where the constant of proportionality depends on $\alpha$). Arguing as above, we may assume that each subcell $\Delta$ of the resulting decomposition is crossed by at most $N_W/r$ wedge boundaries in $\mathcal{W}^{\Delta_0}$, and there is at most one cell that meets all three edges of any fixed triangle $T \in \mathcal{T}^{\Delta_0}$.

In this problem, the decomposition generates only two types of vertices that we need to bound: *new* WW-vertices (those that are incident to the boundary of at least one triangle that had been a T-triangle in the parent cell), which we dispose of immediately, and WT-vertices, which we process recursively. Using [64] and arguing as above, the number of new WW-vertices is $O(N_W + N_T)$. The recurrence bottoms out when $\min\{N_T, N_W\} \le c$, for some constant $c \ge 2$; we then bound the number of the (remaining) WT-vertices, using brute force, by $O(N_W + N_T)$.

Let $U_2(N_T, N_W)$ denote the maximum number of WT-vertices that appear on the boundary of the union at a recursive step within some simplex $\Delta_0$, involving a set $\mathcal{W}^{\Delta_0}$ of $N_W$ wedges and a set $\mathcal{T}^{\Delta_0}$ of $N_T$ triangles. Arguing as above, one can show that $U_2$ satisfies the following recurrence:

$$U_2(N_T, N_W) \le \begin{cases} O(N_W + N_T) + \sum_{i=1}^{1+\log M} 2^i \cdot U_2\left(\frac{2N_T}{2^i}, \frac{N_W}{r}\right), & \text{if } \min\{N_T, N_W\} > c, \\ \\ O(N_W + N_T), & \text{if } \min\{N_T, N_W\} \le c, \end{cases}$$

where $M = O(r \log r)$ is the overall number of cells in the decomposition (at the current recursive step), $c \ge 2$ is an appropriate constant, and the non-recursive terms depend also on $r$ (and on $\alpha$). Again, it is easy to verify that the solution of this recurrence is $U_2(N_T, N_W) = O\left((N_T + N_W) \cdot {N_W}^{\varepsilon}\right)$, for any $\varepsilon > 0$, with a constant of proportionality that depends on $\varepsilon$ and on $\alpha$. This completes the analysis, and establishes the following result:

**Theorem 6.2.1** *The complexity of the union of $n$ $\alpha$-fat triangles in the plane is $O(n^{1+\varepsilon})$, for any $\varepsilon > 0$, where the constant of proportionality depends on $\varepsilon$ and $\alpha$. The bound is almost tight in the worst case.*

**Remark:** The bound is not as sharp as the one obtained in [107, 116]. We note that the dependence on $\alpha$ is the same as that for the union of $\alpha$-fat wedges. Since the latter dependence is only proportional to $\frac{1}{\alpha} \log \frac{1}{\alpha}$ [116], the same holds for the union of fat triangles.

## 6.3 Conclusions

In this chapter we have solved a major open problem in the study of the union of objects in three dimensions, which has resisted a solution for over a decade. Yet, there is still a small remaining gap between our upper bound on the complexity of the union of $n$ $\alpha$-fat tetrahedra and the corresponding lower bound $\Omega(n^2\alpha(n))$ (which we conjecture to be tight). Closing this gap remains a challenging open problem.

A natural open problem is to extend the new machinery presented in this study to the problem of bounding the union of other families of geometric objects in 3-space. One such problem concerns the union of cylinders (with arbitrary radii); a nearly-quadratic bound is known only when all the cylinders have equal radii [13]. Another related problem is to obtain a nearly-quadratic bound on the complexity of the union of $n$ arbitrary $\alpha$-*fat* convex objects of constant description complexity (that is, convex objects $c$, for which there exist two concentric balls, $B \subseteq c \subseteq B'$, such that the ratio between the radii of $B'$ and $B$ is at most $\alpha$, for some fixed $\alpha > 1$). Weaker goals would be to prove this only for nearly equal objects of this kind, or obtaining a subcubic bound for the union of such objects of arbitrary sizes.

# Chapter 7

# Regular Vertices of the Union of Planar Convex Objects

In this chapter we show a nearly-tight upper bound on the number of regular vertices that appear on the boundary of the union $\partial U$ of $n$ compact convex sets in the plane, each pair of whose boundaries intersect in at most $s$ points, for some constant $s$. We present preliminaries and an overview in Section 7.1, and the full analysis is given in Section 7.2. We then study an extension to the non-convex case in Section 7.3. We present open problems and give concluding remarks in Section 7.4.

## 7.1 Preliminaries and Overview

Let $\mathcal{C}$ denote a collection of $n$ convex sets as above. For each $C \in \mathcal{C}$, the segment connecting the leftmost and rightmost points of $C$ is called the *spine* of $C$; we assume (without loss of generality, rotating the coordinate frame if necessary) that it is unique, and denote it by $\sigma_C$ (note that $\sigma_C$ is contained in $C$, due to its convexity).

As already defined, a pair $C, C'$ of sets in $\mathcal{C}$ are said to intersect *regularly* if $|\partial C \cap \partial C'| = 2$. Each of these two intersection points is called a *regular* vertex of the arrangement $\mathcal{A}(\mathcal{C})$ of (the boundary curves of the sets in) $\mathcal{C}$. All other intersection vertices are *irregular*.

We establish an upper bound on the maximum number of regular vertices on the *boundary* of the union $U$ of $\mathcal{C}$, which improves the earlier bound $O(n^{3/2+\varepsilon})$, for any $\varepsilon > 0$, due to Aronov *et al.* [20]. Specifically, we show:

**Theorem 7.1.1** *Let $\mathcal{C}$ be a set of $n$ compact convex sets as above. Then the number of regular vertices on the boundary of the union of $\mathcal{C}$ is at most $O(n^{4/3+\varepsilon})$, for any $\varepsilon > 0$, where the constant of proportionality depends on $\varepsilon$ and on $s$. This bound is nearly worst-case tight, that is, there are constructions that yield $\Omega\left(n^{4/3}\right)$ regular vertices that appear on the boundary of the union (already for $s = 4$).*

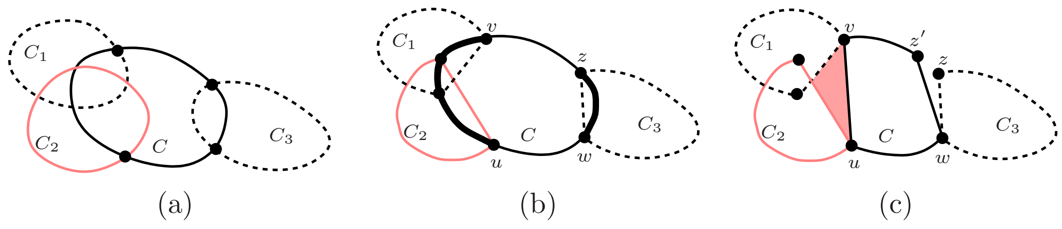


Figure 7.1: Demonstration of the transformation rule. (a) $\partial C$ creates with each of the three sets $C_1$, $C_2$, $C_3$ regular vertices on the boundary of the union. (b) Each of $C_1$, $C_2$, $C_3$ is shrunk by the chords connecting its intersections with $\partial C$. (c) Shrinking $C$ by similar shortcuts; $wz$ is replaced by a nearby chord $wz'$ to make $C$ and $C_3$ touch at a single point. The shaded region is a new connected component of the complement of the union (a new "hole").

**Overview of the proof.** We first assume that the given sets in $\mathcal{C}$ are in general position. The proof proceeds through the following stages. We apply the shrinking transformation of [20] to the input collection $\mathcal{C}$. The transformed sets satisfy the following properties (see [20, Lemma 1] and Figure 7.1 for further details): (i) They are convex. (ii) Any two boundaries intersect at most $s$ times. (iii) Any two sets $C, C' \in \mathcal{C}$ that intersected regularly (before the transformation) either become disjoint or *touch* at a single point. More precisely, if $C$, $C'$ intersected regularly with at least one point of intersection of their boundaries on $\partial U$, the transformed sets are openly disjoint and touch each other at a point on the new union boundary. If they intersected regularly without creating vertices on $\partial U$, they are now disjoint. To simplify the notation, we let $\mathcal{C}$ denote from now on the set of the transformed regions.

Note that the spines of the transformed sets may be different from those of the original sets. Note also that, after this transformation, any regular vertex on the boundary of the union must be formed by a pair of sets whose spines are disjoint.

We then apply a decomposition scheme that consists of two phases. The first phase represents all pairs of sets of $\mathcal{C}$ with disjoint spines, so that one of these spines lies below the other (see below for a precise definition), as the disjoint union of complete bipartite graphs, whose overall complexity is sufficiently small, in a sense to be made precise below.

We then fix one such complete bipartite subgraph $A \times B$, where the spines of the sets in $A$ all lie below those of the sets of $B$, and analyze the number of regular vertices that it contributes to the union boundary. A crucial property of such a graph is that each of these regular vertices must lie either on the upper envelope of the top boundaries of sets in $A$, or on the lower envelope of the bottom boundaries of sets in $B$. We then form, say, the upper envelope $E_A^+$ of the top boundaries of the sets in $A$, and decompose it into maximal connected arcs, each contained in the boundary of a single set, and having disjoint $x$-spans.

The fact that regular vertices are formed by touching pairs, suggests a second decomposition phase, in which we transform $A \times B$ into a union of complete bipartite subgraphs, such that each such subgraph $A' \times B'$ is associated with some vertical strip $\Sigma$, and each spine $\sigma$ of a set in $B'$ lies, within the strip $\Sigma$, above every arc $\delta$ whose incident set belongs to $A'$. It is then easy to show that the number of regular vertices of the union, induced by pairs of sets in $A' \times B'$, is only nearly linear in $|A'| + |B'|$.

The second decomposition phase is somewhat involved. It consists of decomposition steps that alternate between the primal and dual planes, where each step is based on a cutting of a certain line arrangement. While the dual decomposition is more “conventional”, the primal one is trickier, and requires careful analysis of the way in which the arcs $\delta$ interact with the spines from the other set.

Finally, we collect all these bounds, put them together, and obtain the bound asserted in the theorem.

## 7.2 The Number of Regular Vertices on the Boundary of the Union

**Transforming the sets.** We begin by applying to $\mathcal{C}$ the transformation of Aronov *et al.* [20], as explained in the overview. We continue to denote by $\mathcal{C}$ the collection of the shrunk sets.

Let $C$ and $C'$ be two members of $\mathcal{C}$ that touch each other at a point that lies on $\partial U$. Clearly, as already noted, $\sigma_C$ and $\sigma_{C'}$ are disjoint, and one of them, say, $\sigma_C$, *lies below* the other, which means that (i) their $x$-spans have nonempty intersection $J$; (ii) $\sigma_C$ lies below $\sigma_{C'}$ at each $x \in J$.

**The first bi-clique decomposition.** We collect all pairs of spines so that one of them lies below the other, as the disjoint union of complete bipartite graphs (*bi-cliques*), so that the overall size of their vertex sets is $O(n^{4/3+\varepsilon})$, for any $\varepsilon > 0$. More precisely, the following stronger property holds.

**Lemma 7.2.1** *Given a collection $\mathcal{C}$ as above, let $G$ be the graph whose vertices are the regions in $\mathcal{C}$, and whose edges connect pairs of regions $(C, C')$, such that $\sigma_C$ lies below $\sigma_{C'}$. Then there exists a decomposition $G = \bigcup_i A_i \times B_i$ into pairwise edge-disjoint bi-cliques, such that*

$$\sum_i \left(|A_i|^{2/3}|B_i|^{2/3} + |A_i| + |B_i|\right) = O(n^{4/3+\varepsilon}), \tag{7.1}$$

*for any $\varepsilon > 0$.*

**Proof:** This is a standard result in “batched” range searching, and can be found, e.g., in [3, 5, 17, 104]; see also Chapter 3 for a variant of this decomposition. The proof is given below for the sake of completeness.

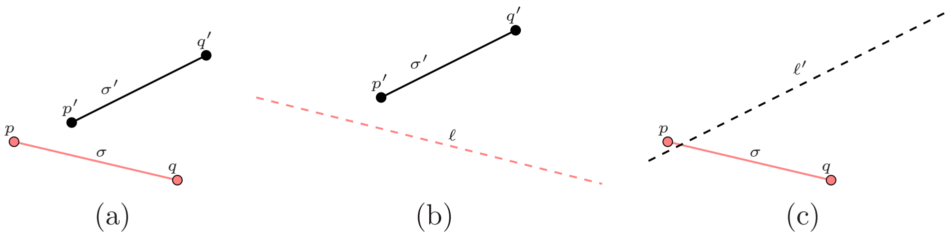


Figure 7.2: (a) The two spines $\sigma$, $\sigma'$ are disjoint and $\sigma$ lies below $\sigma'$. (b) The left endpoint $p'$ of $\sigma'$ lies above the line $\ell$ containing $\sigma$, and (c) the right endpoint $q$ of $\sigma$ lies below the line $\ell'$ containing $\sigma'$.

Let $\sigma = pq$, $\sigma' = p'q'$ be a pair of spines such that $\sigma$ lies below $\sigma'$. This relationship can be expressed as the disjunction of a constant number of conjunctions of above/below relationships, over the possible $x$-orders of $p, q, p'$ and $q'$, where each atomic relationship asserts that an endpoint of one spine lies above or below the line containing the other spine. For example, if the $x$-order of the endpoints is $p, p', q, q'$, then we require that $p'$ lie above the line $\ell$ containing $\sigma$ and that $q$ lie below the line $\ell'$ containing $\sigma'$; see Figure 7.2(a)–(c). For simplicity of exposition, we describe the construction only for the subgraph of $G$ that consists of pairs of sets with this specific order of the endpoints of their spines; all other subcases are handled in a fully symmetric manner.

We apply a multi-level decomposition scheme, where each level produces a decomposition into bi-cliques that satisfy some of the constraints, and each of them is passed to the next level to enforce additional constraints. At the two top levels, we produce a collection of pairwise edge-disjoint bi-cliques, such that, for each of these graphs $A_1 \times B_1$, for each spine $\sigma = pq \in A_1$ and for each spine $\sigma' = p'q' \in B_1$, the $x$-order of the endpoints is $p, p', q, q'$, and such that the union of these graphs gives all such pairs of spines. This is easily done using a 2-dimensional range tree construction [3, 72] (see also Chapter 3). The sum of the vertex sets of the resulting subgraphs is $O(n \log^2 n)$. Moreover, (a slightly sharper variant of) (7.1) is easily seen to hold for the decomposition thus far.

The next level enforces, for each resulting subgraph $A_1 \times B_1$, the condition that $p'$ lie above the line $\ell$ containing $\sigma$, for $\sigma \in A_1$ and $p'$ the left endpoint of a spine $\sigma' \in B_1$. Put $m_1 = |A_1|$ and $n_1 = |B_1|$. To accomplish the task at hand, we choose a sufficiently large constant parameter $r$, and construct a $(1/r)$-cutting of the arrangement of the lines that contain the spines of $A_1$. We obtain $O(r^2)$ cells, each of which is crossed by at most $m_1/r$ lines, and contains at most $n_1/r^2$ left endpoints of spines of $B_1$. (The latter property can be enforced by further splitting some cells of the cutting; also, assuming general position, we can construct the cutting so that no endpoint of any spine lies on the boundary of any of the cutting cells.) For each cell $\Delta$, we form the bi-clique $A_2'(\Delta) \times B_2(\Delta)$, where $B_2(\Delta)$ consists of all spines whose left endpoints are in $\Delta$, and where $A_2'(\Delta)$ consists of all spines whose supporting lines pass completely below $\Delta$. These graphs are passed to the next level of the structure. We then consider, for each cell $\Delta$, the set $A_2(\Delta)$ of spines of $A_1$ that cross $\Delta$, and the set $B_2(\Delta)$ as defined above. We pass to the dual plane, where the lines of the spines in $A_2(\Delta)$ are mapped to points and the left endpoints of spines in $B_2(\Delta)$ are mapped to lines. We construct a $(1/r)$-cutting of the arrangement of these dual lines, obtaining $O(r^2)$ cells, each of which is crossed by at most $|B_2(\Delta)|/r \le n_1/r^3$ lines and contains at most $|A_2(\Delta)|/r^2 \le m_1/r^3$ points. As above, we construct, for each cell of the cutting, a bi-clique from the dual points in the cell and the lines that pass fully above the cell, and pass all these graphs to the next level. We are left with $O(r^4)$ subproblems, each involving at most $m_1/r^3$ spines of $A_1$ and at most $n_1/r^3$ spines of $B_1$, which we process recursively. We continue to process each subproblem as above, going back to the primal plane, and keep alternating in this manner, until we reach subproblems in which either $m_1^2 < n_1$, or $n_1^2 < m_1$. In the former (resp., latter) case, we continue the recursive construction *only* in the primal (resp., dual) plane, and stop as soon as one of $m_1$, $n_1$ becomes smaller than $r$, in which case we produce a collection of singleton bi-cliques.

Suppose first that $\sqrt{m_1} \le n_1 \le m_1^2$. We show below that, for any fixed initial subgraph $A_1 \times B_1$, the resulting bi-clique decomposition $\{A_2'(\Delta) \times B_2(\Delta)\}_\Delta$, over all cells $\Delta$ of all the cuttings, satisfies

$$\sum_{\Delta} \left(|A_2'(\Delta)|^{2/3}|B_2(\Delta)|^{2/3} + |A_2'(\Delta)| + |B_2(\Delta)|\right) = \qquad (7.2)$$
$$O\left(|A_1|^{2/3+\varepsilon}|B_1|^{2/3+\varepsilon} + |A_1|^{1+\varepsilon} + |B_1|^{1+\varepsilon}\right) \quad ,$$

for any $\varepsilon > 0$, and the same holds for the corresponding decompositions in the dual spaces.

Indeed, let us consider only the primal decompositions, since the dual ones are handled in exactly the same manner. Since $r$ is taken to be a constant, the sum in (7.2), over the graphs produced at the top level of the recursion, is at most $C(r)\left(|A_1|^{2/3}|B_1|^{2/3} + |A_1| + |B_1|\right)$, where $C(r)$ is a constant that depends on $r$. In the next level, we have at most $C'r^4$ subproblems, for some absolute constant $C' > 0$, each involving at most $|A_1|/r^3$ spines of $A_1$ and at most $|B_1|/r^3$ spines of $B_1$. The overall contribution to the sum in (7.2) by the bi-cliques produced at this level is at most

$$C'r^4 \cdot C(r)\left(\left(\frac{|A_1|}{r^3}\right)^{2/3}\left(\frac{|B_1|}{r^3}\right)^{2/3} + \frac{|A_1|}{r^3} + \frac{|B_1|}{r^3}\right) =$$
$$C'C(r)\left(|A_1|^{2/3}|B_1|^{2/3} + |A_1|r + |B_1|r\right).$$

Continuing in this manner, the contribution to the sum in (7.2) at the $j$-th level of the recursion is at most

$$(C')^j r^{4j} \cdot C(r)\left(\frac{|A_1|^{2/3}}{r^{2j}}\frac{|B_1|^{2/3}}{r^{2j}} + \frac{|A_1|}{r^{3j}} + \frac{|B_1|}{r^{3j}}\right) =$$
$$(C')^j C(r)\left(|A_1|^{2/3}|B_1|^{2/3} + |A_1|r^j + |B_1|r^j\right),$$

where, at the last level, $r^j = \min\left\{\frac{|A_1|^{2/3}}{|B_1|^{1/3}}, \frac{|B_1|^{2/3}}{|A_1|^{1/3}}\right\}$. Summing over the logarithmically many levels of the recursion, we obtain the overall bound $O(|A_1|^{2/3+\varepsilon}|B_1|^{2/3+\varepsilon})$, for any $\varepsilon > 0$, assuming $r$ is sufficiently large.

It remains to consider the cases $|A_1|^2 < |B_1|$, $|B_1|^2 < |A_1|$. It suffices to consider only the first case. Here, after $j$ levels of recursion (only in the primal plane), the contribution to (7.2) is at most

$$(C'')^j r^{2j} \cdot C(r)\left(\left(\frac{|A_1|}{r^j}\right)^{2/3}\left(\frac{|B_1|}{r^{2j}}\right)^{2/3} + \frac{|A_1|}{r^j} + \frac{|B_1|}{r^{2j}}\right) =$$

$$(C'')^j C(r)\left(|A_1|^{2/3}|B_1|^{2/3} + |A_1|r^j + |B_1|\right),$$

where $C''$ is another absolute constant, and where the last $j$ satisfies $r^j = O(|A_1|)$. Substituting this value, summing over all $j$, and using the inequality $|A_1|^2 \le |B_1|$, we get the overall bound $O(|B_1|^{1+\varepsilon})$, for any $\varepsilon > 0$. Similarly, when $|B_1|^2 < |A_1|$, we get the overall bound $O(|A_1|^{1+\varepsilon})$, for any $\varepsilon > 0$.

Note that, in the preceding case $\sqrt{|A_1|} \le |B_1| \le |A_1|^2$, when the recursion bottoms out, we have sets $A'$, $B'$ that satisfy $|A_1'|^2 \le |B_1'|$ or $|B_1'|^2 \le |A_1'|$, so the same analysis adds to (7.2) the terms $O(|A_1|^{1+\varepsilon} + |B_1|^{1+\varepsilon})$, for any $\varepsilon > 0$, which thus completes the proof of the claim.

The final level of the structure enforces, for each resulting subgraph $A_2 \times B_2$, the condition that $q$ lie below the line $\ell'$ containing $\sigma'$, for $\sigma' \in B_2$ and $q$ the right endpoint of a spine $\sigma \in A_2$. This is done in a fully analogous manner to the preceding step. It is easily checked that, in complete analogy to the preceding analysis, the bi-clique decomposition $\{A_\alpha \times B_\alpha\}_\alpha$, that results from the fixed bi-clique $A_2 \times B_2$, over all cells of all the cuttings, satisfies

$$\sum_\alpha \left(|A_\alpha|^{2/3}|B_\alpha|^{2/3} + |A_\alpha| + |B_\alpha|\right) = O\left(|A_2|^{2/3+\varepsilon}|B_2|^{2/3+\varepsilon} + |A_2|^{1+\varepsilon} + |B_2|^{1+\varepsilon}\right),$$

for any $\varepsilon > 0$. Combining this with (7.2), summing over the entire collection of these last-stage decompositions, and using the fact that (7.1) holds for the initial-level decomposition, we conclude that (7.1) holds for the overall final decomposition, thus completing the proof of the lemma. □

**Handling a single bi-clique.** Fix one of the resulting graphs $A \times B$. All the spines of the sets in $A$ lie below all the spines of the sets in $B$. Put $n_A = |A|$ and $n_B = |B|$.

Let $v$ be a regular vertex of the union lying on the top boundary $\partial^+ C$ and on the bottom boundary $\partial^- C'$, for two sets $C \in A$, $C' \in B$; clearly, this is the only possible situation. We claim that $v$ lies either on the upper envelope $E_A^+$ of the top boundaries of the sets in $A$, or on the lower envelope $E_B^-$ of the bottom boundaries of the sets in $B$. Indeed, if this

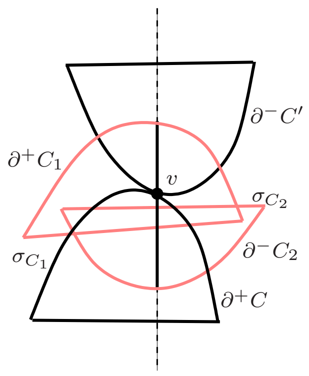


Figure 7.3: If the vertex $v$ does not lie on either $E_A^+$ or $E_B^-$, then it is "hidden" from $E_A^+$ by $\partial^+ C_1$, and from $E_B^-$ by $\partial^- C_2$, for some $C_1 \in A$, $C_2 \in B$. But then $v$ is contained in (the interior of) $C_1 \cup C_2$, contrary to the construction.

were not the case, then $v$ must lie below some top boundary $\partial^+ C_1$, for $C_1 \in A$, and above some bottom boundary $\partial^- C_2$, for $C_2 \in B$; see Figure 7.3. By construction, $\sigma_{C_1}$ lies below $\sigma_{C_2}$ at the $x$-coordinate $x_v$ of $v$, which implies that the entire vertical segment connecting $\partial^+ C_1$ and $\partial^- C_2$ at $x_v$ is fully contained in $C_1 \cup C_2$, so $v$ cannot lie on the boundary of the union, a contradiction that establishes the claim.

Without loss of generality, we consider only the case where $v$ lies on the upper envelope $E_A^+$ of the top boundaries of the sets in $A$. Since any pair of these boundaries intersect in at most $s$ points, the number $m = m_A$ of connected portions of top boundaries that constitute $E_A^+$ satisfies $m \le \lambda_{s+2}(n_A)$. Enumerate these arcs from left to right as $\delta_1, \ldots, \delta_m$, and let $A^*$ denote the set of these arcs.

Let $H_0$ denote the subgraph of $A^* \times B$ consisting of all the pairs $(\delta, C)$, such that $C$ forms with (the set of $A$ containing) $\delta$ a regular vertex on $\partial U$ (where the two sets touch each other), and so that (a) the touching point lies on $\delta$ and on $\partial^- C$, and (b) $\delta$ lies fully below $\sigma_C$ (i.e., the $x$-span of $\sigma_C$ contains that of $\delta$). If (b) does not hold, then an endpoint of $\sigma_C$ lies above $\delta$, and there can be at most two such arcs $\delta$ (for any fixed $C$), so the number of excluded pairs is at most $2n_B$. Hence, the number of regular "bichromatic" vertices formed by $A \cup B$, lying on $E_A^+$, and not counted in $H_0$, is only $O(n_B)$.

Our next step is to construct a collection of complete bipartite graphs $\{A_i^* \times B_i\}_i$, such that, for each $i$, $A_i^* \subset A^*$, $B_i \subset B$, and the union of these graphs is edge-disjoint and covers $H_0$. In addition: (a) The sum $\sum_i (|A_i^*| + |B_i|)$ will be small, in a sense to be made precise below. (b) For each $i$, there is an $x$-interval $I_i$ such that, for each $\delta \in A_i^*$ and $C \in B_i$, the line $\ell_C$ containing the spine $\sigma_C$ of $C$ passes fully above $\delta$ over $I_i$ (although $\sigma_C$ may end within $I_i$). (c) For each pair $(\delta, C) \in H_0$, there exists $i$ such that $\delta \in \mathcal{A}_i^*$, $C \in B_i$, and the $x$-coordinate of the touching point $\delta \cap \partial^- C$ lies in $I_i$.

Suppose we have such a collection at hand. Fix one of the graphs $A_i^* \times B_i$. We claim that, for any $\delta \in A_i^*$, $C \in B_i$, such that $(\delta, C) \in H_0$, the relevant touching vertex $v$ of $\delta \cap \partial^- C$ lies on the lower envelope $E_{B_i}^-$ of the bottom boundaries of the sets in $B_i$. This follows using the same arguments as in the preceding step (see also [20]). That is, suppose to the contrary that $v$ lies above the bottom boundary $\partial^- C'$ of another set $C' \in B_i$. By assumption, $\sigma_{C'}$ lies above $v$ (because $v \in \delta$) and thus $v$ lies in the interior of $C'$, contradicting the assumption that $v$ is a vertex of the union. Note that it is crucial that the $x$-coordinate of $v$ lies in the $x$-interval $I_i$ as above; see Figure 7.4(a).

In other words, each vertex of this kind is an intersection point of $E_{B_i}^-$ and the concatenation of the arcs in $A_i^*$. Hence, by merging, in the $x$-order, the breakpoints of $E_{B_i}^-$ and the endpoints of the arcs in $A_i^*$, it easily follows that the number of such vertices is $O(\lambda_{s+2}(|B_i|) + |A_i^*|)$. Summing this bound over all subgraphs $A_i^* \times B_i$ yields an overall bound for the number of pairs $(\delta, C) \in H_0$ (to which we add the linear number of pairs that are not counted in $H_0$, as above). See below for the precise bound.

To obtain the desired cover of $H_0$, we proceed as follows. Let $L$ denote the set of the lines supporting the spines of the sets in $B$. Fix a sufficiently large constant parameter $r$, and construct a $(1/r)$-cutting $\Xi$ of the arrangement $\mathcal{A}(L)$, as in [47]. It consists of $O(r^2)$ vertical trapezoids, each crossed by at most $n_B/r$ lines of $L$ (and thus by at most $n_B/r$ spines of the sets in $B$).

Consider a pair $(\delta, C) \in H_0$, where the touching between $\delta$ and $\partial^- C$ occurs at some cell $\tau$ of $\Xi$. In this case $\delta$ crosses $\tau$ (or has an endpoint inside $\tau$), and $\sigma_C$ either intersects $\tau$ or lies above $\tau$ (i.e., within the common $x$-span of $C$ and $\tau$, $\sigma_C$ lies fully above $\tau$). For technical reasons, we classify the arcs $\delta$ that cross $\tau$ as being either *short*, if either $\delta$ has an endpoint inside $\tau$, or $\delta$ does not intersect the top edge of $\tau$, or *tall*, if $\delta$ intersects the top edge and has no endpoint in $\tau$. Let $A_\tau^s$ be the set of short arcs in $\tau$, and $A_\tau^t$ the set of tall arcs in $\tau$.

The next lemma shows that the overall number of short arcs, over all cells $\tau$, is small.

**Lemma 7.2.2** $\sum_\tau |A_\tau^s| = O(r^2 \log r + |A^*| \log r)$ .

**Proof:** Note first that there are at most $2|A^*|$ pairs $(\delta, \tau)$, such that $\delta$ ends inside $\tau$. We may therefore ignore these short arcs. Construct a segment tree $\mathcal{T}$ on the $x$-projections of the cells of $\Xi$. Consider a node $v$ of the tree, let $\Xi_v$ denote the set of cells stored at $v$, and let $I_v$ denote the $x$-span of $v$. The cells in $\Xi_v$ are linearly ordered in the $y$-direction, in the sense that for each $x_0 \in I_v$ the vertical line $x = x_0$ crosses all of them in a fixed order; see Figure 7.4(b).

In each cell $\tau$, there are at most two (either tall or short) arcs, whose $x$-spans *overlap*, but not contained in, $I_v$ (the first intersects the vertical line through the left endpoint of $I_v$, and the second intersects the vertical line through its right endpoint), for a total of $O(r^2 \log r)$ such arcs, over all $O(r^2 \log r)$ cells $\tau$ and all nodes $v$ of $\mathcal{T}$.

We thus continue the analysis for those (short) arcs $\delta$ of $A^*$, whose $x$-span is *contained* in $I_v$. There is at most one cell $\tau \in \Xi_v$ such that $\delta \in A_\tau^s$; see Figure 7.4(c). The number

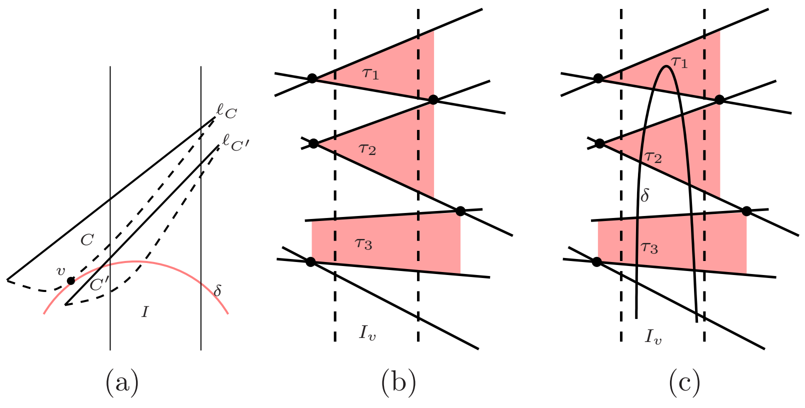


Figure 7.4: (a) The boundary touching $v$, formed by the lower boundary of $C$ and $\delta$, and lying outside the interval $I$, can be "hidden" from $E_B^-$ by the lower boundary of another $C' \in B$. (b)–(c) The cells $\tau_1, \tau_2, \tau_3$, cross the $x$-span $I_v$ of $v$ from left to right. (b) Any vertical line in $I_v$ crosses all these cells in the same order. (c) The arc $\delta$ crosses $\tau_1, \tau_2, \tau_3$, where $\tau_1$ is the unique cell whose top boundary edge is not crossed by $\delta$; $\delta$ is short there and tall at $\tau_2, \tau_3$.

of nodes $v$ at which $\delta$ has this property is $O(\log r)$, because $I_v$ contains the $x$-coordinate of an endpoint (actually, both endpoints) of $\delta$. Hence, the contribution of arcs $\delta$ as above to $\sum_\tau |A_\tau^s|$ is $O(|A^*| \log r)$. Combining this with the previous bound completes the proof of the lemma. □

**Remarks:** 1) The fact that the arcs $\delta$ have pairwise openly disjoint $x$-projections is crucial for the bound that we obtain in Lemma 7.2.2. The decomposition of (a cover of) $H_0$ that we construct is a variant of the decomposition obtained in [20]; however, the analysis in [20] does not exploit the special structure of the arcs $\delta$, and results in a suboptimal bound.
2) An individual arc $\delta$ may cross $\Omega(r)$ cells $\tau$, each of whose top boundary is disjoint from $\delta$. However, Lemma 7.2.2 shows that the overall number of these crossings, summed over all arcs $\delta$, is relatively small.

Each cell $\tau$ for which $|A_\tau^s| > \frac{|A^*| \log r}{r^2}$ is next split, by vertical lines, into subcells, such that each subcell $\tau'$ satisfies $|A_{\tau'}^s| \le \frac{|A^*| \log r}{r^2}$; the number of cells is still $O(r^2)$.

Fix a (new) cell $\tau$, and form the complete bipartite graph $A_\tau^s \times \mathcal{C}_\tau^*$, where $\mathcal{C}_\tau^*$ consists of all sets $C \in B$ such that $\ell_C$ passes above $\tau$. We associate the interval $I_\tau$ (the $x$-span of $\tau$) with this graph. Since $r$ is a constant, we have

$$\sum_\tau (|A_\tau^s| + |\mathcal{C}_\tau^*|) = O(m_A + n_B)$$

(where the constant of proportionality depends on $r$), and the overall number of boundary touchings that they involve, over all cells $\tau$, is $O(m_A + \lambda_{s+2}(n_B))$.

We next claim that the overall number of boundary touchings on the boundary of the union, occurring within a cell $\tau$ and involving a tall arc in $\tau$, summed over all cells $\tau$, is

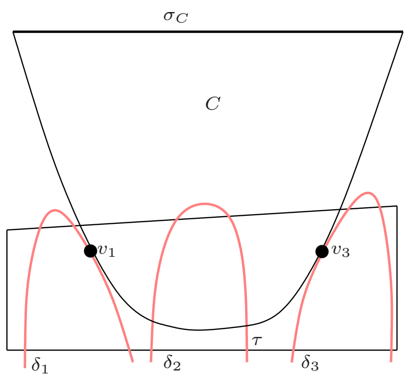


Figure 7.5: $\partial^- C$ touches $\delta_1$ and $\delta_3$ inside $\tau$ at $v_1$ and $v_3$, respectively, and cannot touch the intermediate tall arc $\delta_2$.

only linear in $n_B$ (and thus we need not provide a compact representation for these pairs). Indeed, let $\tau$ be a cell of $\Xi$, and let $\ell_C$ be the line containing the spine $\sigma_C$ of a set $C$, so that $\ell_C$ intersects $\tau$ or passes fully above $\tau$. We claim that there are at most two tall arcs in $\tau$ that touch $\partial^- C$ at a point that lies on $\partial U$. Indeed, suppose, to the contrary, that there are three such arcs $\delta_1, \delta_2, \delta_3$, which appear on $E_A^+$ in that order (from left to right). Consider the two respective boundary touchings that $\partial^- C$ forms with $\delta_1$, $\delta_3$, at two respective points $v_1$, $v_3$ inside $\tau$. Then, due to the convexity of $C$, its portion between $v_1$ and $v_3$ lies below the top edge of $\tau$, and $\delta_2$ lies fully below (and touches) that portion, so it cannot be tall in $\tau$, a contradiction that establishes the claim; see Figure 7.5. Thus the overall number of regular vertices of the above kind is $O(r^2 n_B) = O(n_B)$, as asserted.

We thus conclude that the overall number of boundary touchings involving both short and tall arcs, in all subcases considered so far, is $O(m_A + \lambda_{s+2}(n_B))$.

We continue the construction recursively, within each cell $\tau$, with $A_\tau^s$ and the subset $\mathcal{C}_\tau$ of those $C \in B$ whose line $\ell_C$ crosses $\tau$. We have $|A_\tau^s| \leq \frac{|A^*| \log r}{r^2} = \frac{m_A \log r}{r^2}$, $|\mathcal{C}_\tau| \leq \frac{n_B}{r}$. However, the next stage of the recursion is performed in the *dual* plane, and proceeds as follows. For each resulting cell $\tau$, map $A_\tau^s$ and $\mathcal{C}_\tau$ to the dual plane. For each $C \in \mathcal{C}_\tau$, we map $\ell_C$ to a dual point $\ell_C^*$, and each arc $\delta$ in $A_\tau^s$ is mapped to a convex $x$-monotone curve $\delta^*$, which is the locus of all points dual to lines that are tangent to $\delta$ (possibly at one of its endpoints) and pass above $\delta$ (see [20] and [59] for further details). Thus a line $\ell_C$ lies above an arc $\delta$ if and only if the dual point $\ell_C^*$ lies above $\delta^*$. Each pair of dual arcs $\delta_1^*$, $\delta_2^*$ intersect each other exactly once, since any such intersection point is the dual of a common tangent to $\delta_1$, $\delta_2$ that passes above both of them, and since $\delta_1$, $\delta_2$ are two convex curves that have disjoint $x$-spans, there is exactly one such common tangent. We now construct (for each cell $\tau$ obtained at the preceding step) a $(1/r)$-cutting of the arrangement of the dual arcs $\delta^*$, obtaining $O(r^2)$ subcells, each of which is crossed by at most $\frac{|A_\tau^s|}{r} \leq \frac{m_A \log r}{r^3}$

dual arcs $\delta^*$, and contains at most $\frac{|\mathcal{C}_\tau|}{r^2} \le \frac{n_B}{r^3}$ dual points $\ell^*_C$; see [14] for details.

As above, we construct, for each subcell $\tau'$ of this cutting, a complete bipartite graph that connects the dual points in $\tau'$ to the dual arcs that pass fully below that subcell. Again, since $r$ is a constant, the sum of the sizes of the vertex sets of these graphs is $O(m_A + n_B)$. We are thus left with $O(r^4)$ subproblems, each involving at most $\frac{m_A \log r}{r^3}$ arcs $\delta$ of $A^*$, and at most $\frac{n_B}{r^3}$ sets in $B$. We now process each subproblem recursively, going back to the primal plane, and keep alternating in this manner, until we reach subproblems in which either $m_A^2 < n_B$, or $n_B^2 < m_A$. In the former (resp., latter) case, we continue the recursive construction *only* in the dual (resp., primal) plane, and stop as soon as one of $m_A$, $n_B$ becomes smaller than $r$, in which case we output the complete bipartite graph $A^s_\tau \times \mathcal{C}_\tau$ involving the input sets to the subproblem. Note that in the bottom of the recurrence the boundary touchings are not necessarily obtained on the lower envelope of the boundaries of the sets in $\mathcal{C}_\tau$, and thus the bound on their number in this particular case is $|A^s_\tau| \cdot |\mathcal{C}_\tau| = O(|A^s_\tau| + |\mathcal{C}_\tau|)$, where the constant of proportionality depends on $r$.

The preceding arguments imply that the union of all the bi-cliques constructed by this procedure, including the interactions with tall arcs and other "leftover" pairs detected by the decomposition, covers $H_0$. Indeed, for each such pair $(\delta, C) \in H_0$, the line $\ell_C$ containing the spine $\sigma_C$ of $C$ lies fully above $\delta$. Our procedure detects all such pairs $(\delta, C)$ either (i) at the bottom of the recurrence, in which case all these pairs are reported in a brute force manner, or (ii) at a recursive step, performed in the primal plane and involving a cell $\tau$ in which the boundary touching appears, such that $\ell_C$ lies above $\tau$ and $\delta$ is short in $\tau$, or (iii) at a recursive step, performed in the dual plane and involving a cell $\tau'$, such that $\ell^*_C$ lies inside $\tau'$ and $\delta^*$ passes fully below it.

Let $R(m_A, n_B)$ denote the maximum number of boundary touchings on the boundary of the union, that arise at a recursive step involving $m_A$ arcs $\delta$ and $n_B$ sets $C$, as above, and which are formed between one of the arcs $\delta$ and the bottom boundary of one of the sets $C$. As argued above, the number of such bichromatic touchings, that arise for any of the complete bipartite graphs generated at this stage, is nearly-linear in the sizes of the vertex sets of that graph. Hence $R$ satisfies the following recurrence (where in the first three cases, $\min\{m_A, n_B\} \ge r$):

$$R(m_A, n_B) \le \begin{cases} O\left(m_A + \lambda_{s+2}(n_B)\right) + O(r^4) R\left(\frac{m_A \log r}{r^3}, \frac{n_B}{r^3}\right), & \text{if } m_A^2 \ge n_B \ge \sqrt{m_A}, \\ O\left(m_A + \lambda_{s+2}(n_B)\right) + O(r^2) R\left(\frac{m_A}{r}, \frac{n_B}{r^2}\right), & \text{if } n_B > m_A^2, \\ O\left(m_A + \lambda_{s+2}(n_B)\right) + O(r^2) R\left(\frac{m_A \log r}{r^2}, \frac{n_B}{r}\right), & \text{if } m_A > n_B^2, \\ O\left(m_A + n_B\right), & \text{if } \min\{m_A, n_B\} < r. \end{cases}$$

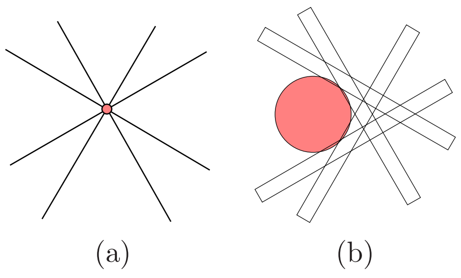


Figure 7.6: From incidences to regular vertices.

It is then easy to see, using induction on $m_A$ and $n_B$, that the solution of this recurrence is

$$R(m_A, n_B) = O(m_A^{2/3+\varepsilon} n_B^{2/3+\varepsilon} + m_A^{1+\varepsilon} + n_B^{1+\varepsilon}) = O(n_A^{2/3+\varepsilon} n_B^{2/3+\varepsilon} + n_A^{1+\varepsilon} + n_B^{1+\varepsilon}), \qquad (7.3)$$

for any $\varepsilon > 0$.

Summing these bounds over all bi-cliques $A \times B$ of the first decomposition phase, and using the bound in (7.1), the upper bound of Theorem 7.1.1 follows.

**Lower bounds.** We use a construction given in [114]. Construct a system of $n$ lines and $n$ points with $\Theta(n^{4/3})$ incidences between them (see, e.g., [115]). Map each line to a long and thin rectangle, and each point to a small disk, in such a way that, for each pair of a point $p$ incident to a line $\ell$, the disk into which $p$ is mapped slightly penetrates the rectangle into which $\ell$ is mapped, and all the intersections between the boundaries of the disks and the rectangles are regular, and lie on the boundary of their union. See Figure 7.6. Clearly, $s = 4$ in this construction. Hence, we obtain a collection of $2n$ convex regions, each pair of whose boundaries intersect in at most four points, which have $\Theta(n^{4/3})$ regular vertices on the boundary of their union. This completes the proof of Theorem 7.1.1. $\Box$

## 7.3 Extensions

In this section, we study the case where the input sets are not necessarily convex, but still satisfy certain properties, listed below, and obtain improved bounds on the number of regular vertices that appear on the boundary of their union. These bound are weaker than the one derived for the convex case, but are still significantly stronger than those derived in [20] for the general case. Specifically, let $\mathcal{C}$ be a collection of $n$ compact simply-connected sets in the plane with the following properties:

(i) For each $C \in \mathcal{C}$, the spine $\sigma_C$ of $C$ is contained in $C$.

(ii) The boundary of each $C \in \mathcal{C}$ is $x$-monotone.

(iii) For each pair $C, C' \in \mathcal{C}$, the top (resp., bottom) boundary of $C$ intersects in at most $s$ points each of the top and the bottom boundaries of $C'$.

The main result of this section is:

**Theorem 7.3.1** *Let $\mathcal{C}$ be a set of $n$ compact simply-connected sets satisfying properties (i)–(iii). Then the number of regular vertices on the boundary of the union of $\mathcal{C}$ is at most*

$$O\left(n^{(3s+1)/(2s+1)+\varepsilon}\right) ,$$

*for any $\varepsilon > 0$, where the constant of proportionality depends on $\varepsilon$ and on $s$.*

**Overview of the proof.** The proof somewhat resembles the proof for the convex case, where the main difference is that we do not apply the transformation of [20] to the input collection $\mathcal{C}$, since this may cause the boundary of each of the resulting sets to consist of an arbitrarily large number of different arcs, which the preceding analysis cannot handle. We therefore modify the preceding analysis to fit into the current scenario.

We first reduce the analysis to the case where the regular vertices are formed by pairs of sets whose spines are disjoint, arguing that there are only $O(n)$ other regular vertices on $\partial U$. (We did not use these considerations in the preceding analysis, since the transformation of [20] guarantees that each pair of spines of the resulting sets that create regular vertices on $\partial U$ are disjoint.)

We then apply a similar decomposition scheme to that of the preceding analysis, which represents all pairs of sets of $\mathcal{C}$ with disjoint spines, so that one of these spines lies below the other, as the disjoint union of complete bipartite graphs.

We then fix one such complete bipartite subgraph $A \times B$, where the spines of the sets in $A$ all lie below those of the sets of $B$, and analyze the number of regular vertices that it contributes to the union boundary. Applying similar arguments as in the preceding analysis, we conclude that each of these regular vertices must lie either on the upper envelope of the top boundaries of sets in $A$, or on the lower envelope of the bottom boundaries of sets in $B$. The main difference from the preceding analysis is the manner in which we continue processing each of these subgraphs $A \times B$: The above property allows us to distribute these regular vertices among a small number of chains with pairwise disjoint $x$-projections, so that each chain is an upper or lower envelope of some subcollection of $\mathcal{C}$.

This latter problem extends the problem of bounding the complexity of $m$ pairwise disjoint concave chains in an arrangement of $n$ lines, as studied in [82, 83]. We apply a different analysis, and derive a sharp bound on the complexity of our lower (or upper) envelope chains, a result that may be of independent interest.

Finally, we collect these bounds, put them together, and obtain the bound asserted in Theorem 7.3.1.

We now proceed to present the proof in detail.

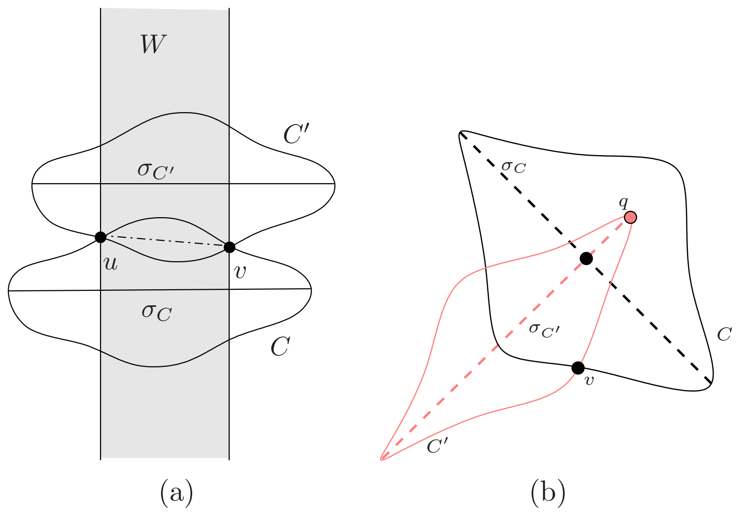


Figure 7.7: (a) If $\partial C$ and $\partial C'$ meet regularly, and neither endpoint of any of the spines $\sigma_C$, $\sigma_{C'}$ is contained in the other set, then the spines must be disjoint. (b) $\partial C$ and $\partial C'$ meet regularly, and $C$ contains the right endpoint $q$ of $\sigma_{C'}$; the regular vertex $v \in \partial U$ can be charged to $q$.

**Reduction to the case of disjoint spines.** Let $C$ and $C'$ be two members of $\mathcal{C}$ that intersect regularly, so that at least one of the two intersection points lies on $\partial U$. Suppose first that $\sigma_C$ and $\sigma_{C'}$ intersect. Then it must be the case that either some endpoint of $\sigma_C$ is contained in $C'$, or some endpoint of $\sigma_{C'}$ is contained in $C$. Indeed, suppose to the contrary that neither of these endpoints is contained in the other set. Since $\partial C$ and $\partial C'$ meet regularly, at two points $u, v$, and no endpoint of either spine is contained in the other set, $C' \cap \partial C$ is a connected arc, delimited by $u$ and $v$, which is fully contained in the top boundary or in the bottom boundary of $C$, and a symmetric property holds for $C \cap \partial C'$. It is easily checked that one of these arcs must be contained in the respective top boundary and the other in the respective bottom boundary. See Figure 7.7(a). Let $W$ be the vertical strip spanned by these two arcs. Outside $W$, $C$ and $C'$ are disjoint, and thus so are their spines. Inside $W$, the segment $uv$ separates the two spines, so they are disjoint there too, a contradiction that establishes the claim.

Hence either $C$ or $C'$ must contain one of the endpoints of the spine of the other set in its interior, and we can charge the regular vertex (or pair of vertices) to this endpoint, so that no endpoint is charged more than twice; see Figure 7.7(b). Hence there are only $O(n)$ such regular vertices on $\partial U$. It therefore suffices to consider only regular vertices whose corresponding spines are disjoint.

**The bi-clique decomposition.** We next collect all pairs of sets, so that the spine of one set lies below the the spine of the other set, as the disjoint union of bi-cliques $A_i \times B_i$, so that the overall size of their vertex sets is $O(n^{4/3+\varepsilon})$, for any $\varepsilon > 0$. We use the same

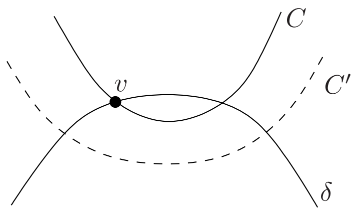


Figure 7.8: Illustrating the proof that the vertex $v \in \partial U$, where the bottom boundary $\partial^- C$ meets $\delta$ regularly, cannot lie above the bottom boundary of another $C' \in \mathcal{C}_B^{(i)}$.

decomposition as in Lemma 7.2.1. Adapting the analysis in the proof of that lemma, one can show that the following also holds:

$$\sum_i \left(|A_i|^{(s+1)/(2s+1)}|B_i|^{2s/(2s+1)} + |A_i| + |B_i|\right) = O\left(n^{(3s+1)/(2s+1)+\varepsilon}\right), \tag{7.4}$$

for any $\varepsilon > 0$.

We now fix one of the resulting graphs $A \times B$. All the spines of the sets in $A$ lie below all the spines of the sets in $B$. Put $n_A = |A|$ and $n_B = |B|$. Let $v$ be a regular vertex of the union lying on the top boundary $\partial^+ C$ and on the bottom boundary $\partial^- C'$, for two sets $C \in A$, $C' \in B$; as just discussed, this is the only possible situation if no endpoint of either spine is contained in the other set. Applying similar arguments to those in Section 7.2, we observe that $v$ lies either on the upper envelope $E_A^+$ of the top boundaries of the sets in $A$, or on the lower envelope $E_B^-$ of the bottom boundaries of the sets in $B$. We only consider, without loss of generality, the case where $v$ lies on the upper envelope $E_A^+$.

As in Section 7.2, $E_A^+$ is a concatenation of arcs $\delta_1, \ldots, \delta_m$, each contained in the boundary of a single set $C \in A$, so that $m \le \lambda_{s+2}(n_A)$. Let $\mathcal{C}_B^{(i)}$ denote the collection of sets $C \in B$ such that $\partial^- C$ meets $\delta_i$ regularly. (We may ignore cases where $\partial^- C$ meets the top boundary containing $\delta_i$ regularly, but crosses $\delta_i$ only once. In such cases, $C$ must contain an endpoint of $\delta_i$, and there can exist at most two vertices on $\delta_i$ with this property.) We claim that each of these regular vertices lies on the lower envelope of the boundaries $\{\partial^- C \mid C \in \mathcal{C}_B^{(i)}\}$. Indeed, if a set $C \in \mathcal{C}_B^{(i)}$ is such that its bottom boundary $\partial^- C$ meets $\delta_i$ at a vertex $v \in \partial U$ that does not lie on that envelope, then there exists another $C' \in \mathcal{C}_B^{(i)}$ whose bottom boundary passes below $v$. Since $\partial^- C'$ meets $\delta$ regularly, one of the intersection points must lie to the left of $v$ and one must lie to the right of $v$. But then the entire portion of $\delta$ between these intersection points, and in particular $v$ itself, must be contained in $C'$, contradicting the fact that $v \in \partial U$; see Figure 7.8.

We thus face the following problem. We are given a collection $\Gamma$ of $n_B$ $x$-monotone curves $\partial^- C$, for $C \in B$, and wish to bound the overall combinatorial complexity of $m \le \lambda_{s+2}(n_A)$ chains, such that the chains have pairwise openly disjoint $x$-projections, and such that each chain is a lower envelope of a subset of $B$; here the complexity of a chain is simply the

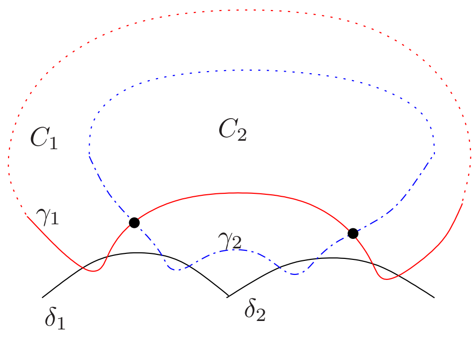


Figure 7.9: Illustrating the proof that $H$ does not contain $K_{s+1,2}$.

number of distinct curves that appear in the chain. This is a special case of a natural extension of the problem, studied in [82, 83], of bounding the complexity of $m$ pairwise disjoint *concave chains* in an arrangement of $n$ lines. (It is a special case since we assume that the chains have disjoint $x$-projections; the chains studied in [82, 83] are only assumed to be pairwise disjoint.)

Even though this abstract formulation yields nontrivial bounds on the combined complexity of the chains, our analysis will "squeeze" the bounds further, by exploiting the properties that (a) the curves of $\Gamma$ are the bottom boundaries of regions that satisfy assumptions (i)–(iii), and (b) each curve that appears in some chain has a point (within the chain) that lies on the boundary of the union of the regions participating in the chains.

Consider the bi-clique $H \subseteq \Delta \times \Gamma$, whose edges connect those pairs $(\delta, \gamma)$ for which $\gamma$ participates in the chain defined by $\delta$, and such that $\gamma$ appears in at least one chain to the left of $\delta$ and in at least one chain to the right of $\delta$. We claim that $H$ does not contain a subgraph isomorphic to $K_{s+1,2}$. Indeed, suppose to the contrary that there exist arcs $\delta_1, \ldots, \delta_{s+1}$, in this left-to-right order, and two regions $C_1, C_2$, whose respective bottom boundaries $\gamma_1, \gamma_2$ appear in all the chains induced by the $\delta_i$'s (and each of them appears in at least one chain to the left of $\delta_1$ and in at least one chain to the right of $\delta_{s+1}$). See Figure 7.9. It is easily checked that $\gamma_1$ and $\gamma_2$ must intersect at least once above each of the arcs $\delta_i$; this is trivial for "middle" arcs (with $1 < i < s+1$), and follows, say, for the leftmost arc $\delta_1$ by recalling that each of $\gamma_1, \gamma_2$ must appear in some chain to the left of $\delta_1$, and have a point there that lies on $\partial U$; a symmetric argument applies for the rightmost arc $\delta_{s+1}$. This, however, results in a contradiction, since $\gamma_1$, $\gamma_2$ intersect in at most $s$ points, and thus $H$ cannot contain $K_{s+1,2}$, as asserted.

It thus follows, by the Kövari-Sós-Turán theorem [111], that the number of edges of $H$, and thus the maximum possible complexity $K(m,n)$ of $m$ lower envelope chains in an arrangement of $n$ curves with the above properties, is $O(mn^{s/(s+1)} + n)$. That is, we have

$$K(m,n) = O\left(mn^{s/(s+1)} + n\right) . \tag{7.5}$$

Fix a parameter $r$, whose value will be determined later, and, to simplify the notation, put $n = n_B$. Let $\Lambda_k$ denote the $k$-th level in the arrangement $\mathcal{A}(\Gamma)$, for $k = 0, \ldots, n-1$. As in [101], we can find a shift $q \in \{0, 1, \ldots, \frac{n}{r} - 1\}$, such that the overall complexity of the levels $\Lambda_q, \Lambda_{q+n/r}, \ldots, \Lambda_{q+(r-1)n/r}$ is $O(nr)$. Let $L_j$ denote the portion of the plane between $\Lambda^*_{j-1} := \Lambda_{q+(j-1)n/r}$ and $\Lambda^*_j := \Lambda_{q+jn/r}$ (excluding the former and including the latter), for $j = 1, \ldots, r-1$; $L_0$ (resp., $L_r$) is defined as the portion of the plane below and including $\Lambda^*_0$ (resp., above $\Lambda^*_{r-1}$). We refer to $L_j$ as the *$j$-th belt* of $\mathcal{A}(\Gamma)$.

For each of the levels $\Lambda^*_j$, we consider the sequence of vertices of $\mathcal{A}(\Gamma)$ along $\Lambda^*_j$, and mark the vertices that appear at the places $n/r, 2n/r, 3n/r$, and so on. For each marked vertex $v$, we draw two vertical segments from $v$, up and down, and extend them till they hit $\Lambda^*_{j+1}$ (or $+\infty$ for the top level) and $\Lambda^*_{j-1}$ (or $-\infty$ for the bottom level), respectively. These vertical segments partition the belts into a collection $\Xi$ of a total of $O(r^2)$ regions, each of which is crossed by at most $2n/r$ curves of $\Gamma$; as a matter of fact, the total number of maximal connected portions of the curves within a single region is also at most $2n/r$. (Indeed, an endpoint of such a component lies either on the left or right vertical edges of the region, and there are $2n/r$ such points, or at a vertex of its top or bottom boundary, and again there are at most $2n/r$ such vertices.)

Let $W_i$ denote the vertical strip spanned by $\delta_i$, for $i = 1, \ldots, m$. The chain $\xi_i$ defined by $\delta_i$ is the lower envelope of a collection $\Gamma_i$ of some number, $k_i$, of curves. Let $t = t_i$ denote the largest index for which $\xi_i$ appears within $L_t$, at some point $x$. By definition, for each $\gamma \in \Gamma_i$ which is defined at $x$, $\gamma(x) \geq \xi_i(x)$, so all curves of $\Gamma_i$ which do not end inside $W_i$ must intersect $L_t$ (a curve that lies fully above $L_t$ will not appear along $\xi_i$ at all, and so will not be included in $\xi_i$, by construction). The number of curves that end within $W_i$, summed over all $W_i$, is only $2n$, and we ignore these curves in the following analysis. Let $k_{i,p}$ denote the number of curves of $\Gamma_i$, that appear along $\xi_i$ within $L_{t-p}$, but not within any lower belt, for $p = 0, 1, \ldots$ (recall that all these appearances are confined to $W_i$). We have $\sum_{p \geq 0} k_{i,p} = k_i$. See Figure 7.10.

Consider a curve $\gamma \in \Gamma_i$ that is counted in $k_{i,p}$, for some $p \geq 2$. Since $\gamma$ appears in $L_{t-p}$ and also in $L_t$, it must cross (within $W_i$) at least $(p-1)n/r$ distinct levels of $\mathcal{A}(\Gamma)$, and each such crossing defines a vertex of $\mathcal{A}(\Gamma)$ along $\gamma$ within $W_i$. Moreover (assuming general position), each such vertex can be encountered in this way at most twice. It follows that $W_i$ must contain at least

$$\frac{n}{2r} \cdot \sum_{p \geq 2} (p-1) k_{i,p} \geq \frac{n}{2r} \cdot \sum_{p \geq 2} k_{i,p}$$

vertices of $\mathcal{A}(\Gamma)$. If we denote by $X = O(n^2)$ the total number of vertices of $\mathcal{A}(\Gamma)$, we conclude that

$$\sum_{i=1}^{m} \sum_{p \geq 2} k_{i,p} \leq \frac{2rX}{n} = O(nr).$$

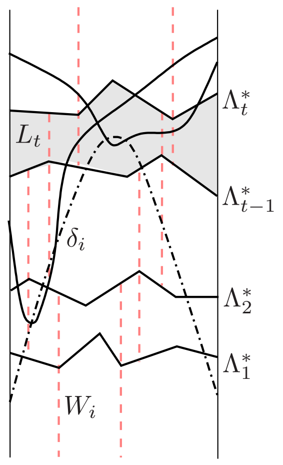


Figure 7.10: The belts within the strip $W_i$ spanned by $\delta_i$.

Recall that our objective is to bound

$$k^* := K(m,n) = \sum_{i=1}^{m} k_i = \sum_{i=1}^{m} \sum_{p \geq 0} k_{i,p}.$$

We thus obtain

$$k^* \leq \sum_{i=1}^{m} (k_{i,0} + k_{i,1}) + \frac{2rX}{n}. \tag{7.6}$$

In other words, we may assume that each $\xi_i$ lives within at most two consecutive belts of $W_i$. To be more precise, we remove from $\Gamma_i$ the arcs that extend downward below belt $L_{t_i-1}$, and redefine $\xi_i$ to be the lower envelope of the reduced collection of curves. The new envelope might now appear in belts that are higher than $L_{t_i}$ or are lower than $L_{t_i-1}$, but each of its arcs must still appear on the envelope within $L_{t_i} \cup L_{t_i-1}$. Thus, restricting the new envelope to only its portion within $L_{t_i} \cup L_{t_i-1}$, we will still account for all of its (undiscarded) arcs.

Let us return to the partition $\Xi$. For each cell $\tau$ of $\Xi$, let $m_\tau$ denote the number of (reduced and restricted) chains that appear within $\tau$. Let $m_\tau^{(s)}$ (resp., $m_\tau^{(l)}$) denote the number of *short* (resp., *long*) chains in $\tau$, where a chain is short if either it does not appear in any cell preceding $\tau$ within its belt, or it does not appear in any cell succeeding $\tau$ within its belt; a long chain appears within the belt of $\tau$ both to its left and to its right. Clearly,

$$m_\tau = m_\tau^{(s)} + m_\tau^{(l)} , \qquad \text{and} \qquad \sum_\tau m_\tau^{(s)} \leq 4m.$$

Long chains are easy to handle: Since the original chains have pairwise disjoint $x$-spans, it follows that a cell $\tau$ can have at most one long chain (and then it has no short chains).

Hence, the contribution of such a cell $\tau$ to $k^*$ is $O(n/r)$, for a total of $O(nr)$. Combining this bound with (7.6), we have

$$K(m,n) \leq \sum_{\tau} K(m_\tau^{(s)}, 2n/r) + O(nr).$$

Substituting the earlier bound (7.5), we obtain

$$K(m,n) = O\left( \sum_{\tau} \left( m_\tau^{(s)} (n/r)^{s/(s+1)} + n/r \right) + nr \right) = O\left( m(n/r)^{s/(s+1)} + nr \right).$$

We now choose $r = m^{(s+1)/(2s+1)}/n^{1/(2s+1)}$, where we assume that $n^{1/(s+1)} \leq m \leq n^2$; otherwise $K(m,n) = O(m+n)$, as is easily checked. This implies, substituting back $m = \lambda_{s+2}(n_A)$ and $n = n_B$, that the number of regular vertices on the boundary of the union, generated by a fixed bi-clique $A \times B$, is at most

$$O\left( (\lambda_{s+1}(|A|))^{(s+1)/(2s+1)} |B|^{2s/(2s+1)} + \lambda_{s+2}(|A)| + |B| \right).$$

Summing these bounds over all bi-cliques $A \times B$, and using the bound in (7.4), the upper bound of Theorem 7.3.1 is easily seen to follow.

## 7.4 Concluding Remarks and Open Problems

We have presented a nearly-optimal bound on the number of regular vertices that appear on the boundary of the union of $n$ compact convex sets in the plane, each pair of whose boundaries intersect in at most some constant number, $s$, of points. We have also established a weaker bound for sets that are non-convex but satisfy properties (i)–(iii) of Section 7.3.

A major open problem is to extend our bound to the case where the sets in $\mathcal{C}$ are not convex (and, still, each pair of whose boundaries intersect in up to a constant number of points). Weaker goals are to extend the bound $O\left(n^{4/3+\varepsilon}\right)$ to the case that we studied in Section 7.3, or even only to the further restricted setting, where each input set has a constant description complexity. We tend to conjecture that, in all these cases, the actual bound is close to $O\left(n^{4/3}\right)$.

# Bibliography

[1] P. K. Agarwal. Partitioning arrangements of lines, part II: applications. *Discrete Comput. Geom.*, 5(6):533–573, 1990.

[2] P. K. Agarwal. Geometric partitioning and its applications. In J.E. Goodman, R. Pollack, and W. Steiger, editors, *Computational Geometry: Papers from the DIMACS Special Year*, pages 1–37. American Mathematical Society, 1991.

[3] P. K. Agarwal. *Intersection and Decomposition Algorithms for Planar Arrangements.* Cambridge University Press, New York, USA, 1991.

[4] P. K. Agarwal, M. de Berg, S. Har-Peled, M. H. Overmars, M. Sharir, and J. Vahrenhold. Reporting intersecting pairs of polytopes in two and three dimensions. *Comput. Geom. Theory Appl.*, 23(2):195–207, 2002.

[5] P. K. Agarwal and J. Erickson, Geometric range searching and its relatives. *Contemporary Mathematics, Amer. Math. Soc.* 223:1–56, 1999.

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

[7] P. K. Agarwal, S. Har-Peled, M. Sharir, and Y. Wang. Hausdorff distance under translation for points and balls. In *Proc. 19th Annu. ACM Sympos. Comput. Geom.*, pages 282–291, ACM Press, 2003.

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

[9] P. K. Agarwal and J. Matoušek. Dynamic half-space range reporting and its applications. *Algorithmica*, 13:325–345, 1995.

[10] P. K. Agarwal, J. Pach and M. Sharir. State of the union, of geometric objects: A review, to appear in *Proc. Joint Summer Research Conference on Discrete and Computational Geometry—Twenty Years Later*, Contemp. Math, AMS, Providence, RI.

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

[12] P. K. Agarwal and S. Sen, Randomized algorithms for geometric optimization problems. Handbook of Randomization (P. Pardalos, S. Rajasekaran, J. Reif, and J. Rolim, eds.), Kluwer Academic, 2000.

[13] 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.

[14] P. K. Agarwal and M. Sharir, Pseudo-line arrangements: Duality, algorithms, and applications, *SIAM J. Comput.*, 34(3):526–552, 2005.

[15] P. K. Agarwal and M. Sharir. Red-blue intersection detection algorithms, with applications to motion planning and collision detection. *SIAM J. Comput.*, 19(2):297–321, 1990.

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

[17] P. K. Agarwal and K. R. Varadarajan. Efficient algorithms for approximating polygonal chains. *Discrete Comput. Geom.*, 23:273–291, 2000.

[18] H. Alt and L. Guibas. Discrete geometric shapes: matching, interpolation, and approximation. In *Handbook of Computational Geometry* (J.-R. Sack and J. Urrutia, eds.), Elsevier, Amsterdam, pages 121–153, 2000.

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

[20] B. Aronov, A. Efrat, D. Halperin and M. Sharir. On the number of regular vertices of the union of Jordan regions. *Discrete Comput. Geom.* 25:203–220, 2001.

[21] B. Aronov, A. Efrat, V. Koltun, and M. Sharir. On the union of $\kappa$-round objects in three and four dimensions. *Discrete Comput. Geom.*, 36:511–526, 2006.

[22] B. Aronov, M. Pellegrini, and M. Sharir. On the zone of a surface in a hyperplane arrangement. *Discrete Comput. Geom.*, 9(2):177–186, 1993.

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

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

[25] B. Aronov and M. Sharir. On translational motion planning of a convex polyhedron in 3-space. *SIAM J. Comput.*, 26(6):1785–1803, 1997.

[26] B. Aronov and M. Sharir. The common exterior of convex polygons in the plane. *Comput. Geom. Theory Appl.*, 6(3):139–149, 1997.

[27] B. Aronov, M. Sharir, and B. Tagansky. The union of convex polyhedra in three dimensions. *SIAM J. Comput.*, 26(6):1670–1688, 1997.

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

[29] D. Athur and S. Vassilvitskii. How slow is the $k$-means method? In *Proc. 22th Annu. ACM Sympos. Comput. Geom.*, pages 144–153, ACM Press, 2006.

[30] D. Athur and S. Vassilvitskii. Worst-case and smoothed analyses of the ICP algorithm, with an application to the $k$-means method. In *Proc. 47th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 153–164, 2006.

[31] D. Attali and A. Montanvert. Computing and simplifying 2d and 3d continuous skeletons. *Comput. Vision Imag. Understand.*, 67(3):261–273, 1997.

[32] F. Aurenhammer and R. Klein. Voronoi diagrams. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 201–290. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[33] S. Basu. On the combinatorial and topological complexity of a single cell. *Discrete Comput. Geom.*, 29(1):41–59, 2003.

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

[35] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992.

[36] J. D. Boissonnat, M. Sharir, B. Tagansky, and M. Yvinec. Voronoi diagrams in higher dimensions under certain polyhedral distance functions. *Discrete Comput. Geom.*, 19:485–519, 1998.

[37] J. D. Boissonnat and M. Yvinec. Algorithmic Geometry. Cambridge University Press, New York, 1998.

[38] H. Brönnimann and B. Chazelle. Optimal slope selection via cuttings. *Comput. Geom. Theory Appls.*, 10(1):23–29, 1998.

[39] H. Brönnimann and O. Devillers. The union of unit balls has quadratic complexity, even if they all contain the origin. Technical Report 3758, INRIA, 1999.

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

[41] D. E. Cardoze and L. J. Schulman. Pattern matching for spatial point sets. In *IEEE Sympos. Found. Comput. Sci.*, pages 156–165, 1998.

[42] B. Chazelle. Reporting and counting segment intersections. *J. Comput. System Science*, 32:156–182, 1986.

[43] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9(2):145–158, 1993.

[44] B. Chazelle, H. Edelsbrunner. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1(2):133–162, 1986.

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

[46] B. Chazelle, H. Edelsbrunner, L. J. Guibas, M. Sharir, and J. Snoeyink. Computing a face in an arrangement of line segments and related problems. *SIAM J. Comput.*, 22(6):1286–1302, 1993.

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

[48] L. Chew and K. Kedem. Improvements on approximate pattern matching problems. In O. Nurmi and E. Ukkonen, editors, In *Proc. Third Scandinavian Workshop on Algorithm Theory (SWAT'92)*, pages 318–325. Lecture Notes in Computer Science 621, Springer-Verlag, 1992.

[49] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2:195–222, 1987.

[50] K. L. Clarkson. Randomized Geometric Algorithms. *Comput. Euclid. Geom.*, (F. K. Hwang and D. Z. Hu eds), World Scientific, Singapore, 117–162, 1992.

[51] K. L. Clarkson. Algorithms for polytope covering and approximation. In *Proc. 3rd Workshop on Algorithms and Data Structures*, pages 246–252, 1993.

[52] K. L. Clarkson. Las Vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42(2):488–499, 1995.

[53] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4(5):387–421, 1989.

[54] R. Cole, J. Salowe, W. Steiger, and E. Szemerédi. An optimal time algorithm for slope selection. *SIAM J. Comput.*, 18:792–810, 1989.

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

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

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

[58] D. P. Dobkin and D. G. Kirkpatrick. Determining the separation of preprocessed polyhedra: a unified approach. *Proc. 17th Internat. Colloq. Automat. Lang. Prog.*, pages 400–413, 1990.

[59] H. Edelsbrunner, *Algorithms in Combinatorial Geometry.* Springer-Verlag, New York, 1987.

[60] H. Edelsbrunner, L. J. Guibas, J. Pach, R. Pollack, R. Seidel, and M. Sharir. Arrangements of curves in the plane: Topology, combinatorics, and algorithms. *Theoretical Comput. Sci.*, 92:319–336, 1992.

[61] H. Edelsbrunner and R. Seidel. Voronoi diagrams and arrangements. *Discrete Comput. Geom.*, 8(1):25–44, 1986.

[62] A. Efrat. The complexity of the union of $(\alpha, \beta)$-covered objects. *SIAM J. Comput.*, 34:775–787, 2005.

[63] A. Efrat and M. Katz. On the union of $\alpha$-curved objects. *Comput. Geom. Theory Appl.*, 14:241–254, 1999.

[64] A. Efrat, G. Rote, and M. Sharir. On the union of fat wedges and separating a collection of segments by a line. *Computat Geom: Theory Appl.*, 3:277–288, 1994.

[65] A. Efrat and M. Sharir. On the complexity of the union of fat objects in the plane. *Discrete Comput. Geom.*, 23:171–189, 2000.

[66] J. Erickson. New lower bounds for convex hull problems in odd dimensions. *SIAM J. Comput.*, 28:1198–1214, 1999.

[67] G. Even, M. Shahar, and D. Rawitz. Hitting sets when the VC-dimension is small. *Inform. Process. Lett.*, 95(2):358–362, 2005.

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

[69] E. Ezra, J. Pach and M. Sharir. On regular vertices on the union of planar objects. In *Proc. 23th Annu. ACM Sympos. Comput. Geom.*, pages 220–226, ACM Press, 2007.

[70] 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, 2004.

[71] E. Ezra and M. Sharir. Output-sensitive construction of the union of triangles. *SIAM J. Comput.*, 34(6):1331–1351, 2005.

[72] E. Ezra and M. Sharir. Counting and representing intersections among triangles in three dimensions. *Comput. Geom. Theory Appl.*, 32:196–215, 2005.

[73] E. Ezra. and M. Sharir. A single cell in an arrangement of convex polyhedra in $R^3$. *Discrete Comput. Geom.*, 37:21–41, 2007.

[74] E. Ezra and M. Sharir. Almost tight bound for the union of fat tetrahedra in three dimensions. In *Proc. 48nd Annu. IEEE Sympos. Found. Comput. Sci.*, 2007, to appear.

[75] E. Ezra, M. Sharir and A. Efrat On the ICP Algorithm In *Proc. 22th Annu. ACM Sympos. Comput. Geom.*, pages 95–104, ACM Press, 2006. Also accepted to *Comput. Geom. Theory Appl.*.

[76] S. Fortune. Voronoi diagrams and Delaunay triangulations. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 23, pages 513–528. CRC Press, 2004.

[77] 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.

[78] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy. Geometrically stable sampling for the ICP algorithm. In *Fourth International Conference on 3D Digital Imaging and Modeling (3DIM)*, pages 260–267, Oct. 2003.

[79] L. J. Guibas, M. Overmars, and M. Sharir. Counting and reporting intersections in arrangements of line segments. Technical Report 434, Dept. Computer Science, New York University, March 1989.

[80] L. J. Guibas, M. Sharir, and S. Sifrony. On the general motion planning problem with two degrees of freedom. *Discrete Comput. Geom.*, 4:491–521, 1989.

[81] D. Halperin. Arrangements. In J.E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 24, pages 529–562. CRC Press LLC, Boca Raton, FL, 2004.

[82] D. Halperin and M. Sharir. On disjoint concave chains in arrangements of (pseudo) lines. *Inform. Process. Lett.* 40:189–192, 1991.

[83] D. Halperin and M. Sharir. Corrigendum: On disjoint concave chains in arrangements of (pseudo) lines. *Inform. Process. Lett.* 51:53–56, 1994.

[84] D. Halperin and M. Sharir. New bounds for lower envelopes in three dimensions with applications to visibility of terrains. *Discrete Comput. Geom.*, 12:313–326, 1994.

[85] D. Halperin and M. Sharir. Almost tight upper bounds for the single cell and zone problems in three dimensions. *Discrete Comput. Geom.*, 14(4):385–410, 1995.

[86] D. Halperin and M. Sharir. Arrangements and their applications in robotics: Recent developments. *Algorithmic Foundations of Robotics*, 495–511, A. K. Peters, Wellesley, MA, 1995.

[87] I. Halperin, B. Ma, H. Wolfson, and R. Nussinov. Principles of docking: An overview of search algorithms and a guide to scoring functions. *Proteins: Structue, Function, and Genetics*, 47:409–443, 2002.

[88] S. Har-Peled and B. Sadri. How fast is the $k$-means method? *Algorithmica*, 41(3):185–202, 2005.

[89] A. Hatcher. *Algebraic Topology.* Cambridge University Press, 2002.

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

[91] D. P. Huttenlocher, K. Kedem, and J. M. Kleinberg. On dynamic Voronoi diagrams and the minimum Hausdorff distance for point sets under Euclidean motion in the plane. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 110–119, ACM Press, 1992.

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

[93] 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.

[94] V. Koltun. Almost tight upper bounds for vertical decompositions in four dimensions. *J. ACM*, 51(5):699–730, 2004.

[95] V. Koltun and M. Sharir. Curve-sensitive cuttings. *SIAM J. Comput.*, 34:863–878, 2005.

[96] V. Koltun, and M. Sharir. On overlays and minimization diagrams. In *Proc.* $22^{nd}$ *Annu. ACM Sympos. Comput. Geom.*, pages 395–401, 2006.

[97] E. Lacourse. *Handbook of Solid Modeling.* McGraw Hill, Ney York ,1995.

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

[99] D. Leven and M. Sharir. Intersection and Proximity Problems and Voronoi Diagrams. *Algorithmic and Geometric Aspacets of Robotics*, (1):187–228, 1987. (J. Schwartz and C. K. Yap Eds.).

[100] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Mach. Learn.*, 2(4):285–318, 1988.

[101] J. Matoušek. Construction of epsilon-nets. *Discrete Comput. Geom.* 5:427–448, 1990.

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

[103] J. Matoušek. Randomized optimal algorithm for slope selection. 39(4):183–187, 1991.

[104] J. Matoušek. Geometric Range Searching. *ACM Computing Surveys* 26(4):421–461, 1994.

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

[106] 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.

[107] 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.

[108] P. McMullen. The maximum number of faces of a convex polytope. *Mathematika*, 17:179–184, 1970.

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

[110] J. Pach. On the complexity of the union of geometric objects. *Lecture Notes in Computer Science*, 2098:292–307, 2001.

[111] J. Pach and P. K. Agwral. *Combinatorial Geometry*, Wiley Interscience, New York, 1995.

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

[113] J. Pach and M. Sharir. The upper envelope of piecewise linear functions and the boundary of a region enclosed by convex plates: Combinatorial analysis. *Discrete Comput. Geom.* 4(1):291–309, 1989.

[114] J. Pach and M. Sharir. On the boundary of the union of planar convex sets. *Discrete Comput. Geom.* 21:321–328, 1999.

[115] J. Pach and M. Sharir, Geometric incidences. in *Towards a Theory of Geometric Graphs* (J. Pach, ed.), *Contemporary Mathematics, Amer. Math. Soc.*, 342:185–223, 2004.

[116] J. Pach and G. Tardos. On the boundary complexity of the union of fat triangles. *SIAM J. Comput.*, 31(6):1745–1760, 2002.

[117] L. Palazzi and J. Snoeyink. Counting and reporting red/blue segment intersections. *CVGIP: Graph. Models Image Process.*, 56(4):304–310, 1994.

[118] M. Pellegrini. On counting pairs of intersecting segments and off-line triangle range searching. *Algorithmica*, 17:380–398, 1997.

[119] H. Pottmann, Q.-X. Huang, Y.-L. Yang, and S.-M. Hu. Geometry and convergence analysis of algorithms for registration of 3D shapes. Technical Report 117, Geometry Preprint Series, TU Wien, June 2004.

[120] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Third Internat. Conf. 3D Digital Imag. Model. (3DIM)*, pages 145–152, 2001.

[121] J. T. Schwartz, and M. Sharir. On the "Piano Movers" problem: II. General techniques for computing topological properties of real algebraic manifolds. *Adv. Appl. Math.*, 4:298-351, 1983.

[122] O. Schwarzkopf, and M. Sharir. Vertical decomposition of a single cell in a 3-dimensional arrangement of surfaces. *Discrete Comput. Geom.*, 18:269–288, 1997.

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

[124] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, New York, 1995.

[125] G. C. Sharp, S. W. Lee, and D. K. Wehe. ICP registration using invariant features. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(1):90–102, 2002.

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

[127] G. M. Voronoi. Nouvelles applications des parametres continus a la theorie des formes quadratiques. Deuxieme Memoire: Recherches sur les parallelloedres primitifs. *J. Reine Angew. Math.*, 134:198–287, 1908.

[128] E. Welzl. Partition trees for triangle counting and other range searching problems. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 23–33. ACM Press, 1988.