Guarantees Concerning Geometric Objects with Imprecise Points

by

Jeff Sember

B.Sc., School of Computing Science, Simon Fraser University, 2004

M.Sc., School of Computing Science, Simon Fraser University, 2006

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE STUDIES

(Computer Science)

The University Of British Columbia

(Vancouver)

October 2011

Abstract

Traditional geometric algorithms are often presented as if input imprecision does not exist, even though it is often unavoidable. We argue that in some cases, it may be desirable for geometric algorithms to treat this imprecision as an explicit component of the input, and to reflect this imprecision in the output. Starting with three problems from computational geometry whose inputs are planar point sets (Voronoi diagrams, convex hulls, and smallest bounding discs), we recast these as problems where each input point's location is imprecise, but known to lie within a particular *region of uncertainty*. Where algorithms to solve each of the original problems produce a single geometric object as output, the algorithms that we present typically produce either *guaranteed* or *possible* output objects. A guaranteed object represents qualities that can be guaranteed for every point set that is consistent with the uncertain regions, and a possible object represents qualities that exist for at least one such point set. By dealing with input imprecision explicitly, these guaranteed and possible objects can represent a more accurate view of what can be reliably inferred from the input data.

Preface

Portions of Chapter 2 have been published: Sember, J. and Evans, W. Guaranteed Voronoi diagrams of uncertain sites. *Proceedings of the 20th Canadian Conference on Computational Geometry*, pages 207–210, 2008. The research and writing is mine, except for the proof of Lemma 3, which was contributed by Will Evans. Will also assisted in proofreading the paper.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I am very grateful to the members of my supervisory committee. I consider myself very fortunate to have had Will Evans as my senior supervisor; his knowledge and guidance has been invaluable during my time at the University of British Columbia. I am also indebted to David Kirkpatrick, whose help during the final stages of my thesis writing has significantly improved my dissertation.

Some other members of the computational geometry community that I would be remiss in not giving thanks to include Maarten Löffler, whom I had the pleasure of having a number of fruitful discussions with during his visits to Vancouver; and Binay Bhattacharya, my supervisor at Simon Fraser University, whose ongoing friendship and encouragement I value.

I am grateful for the recognition and considerable financial support that I have received from the Natural Sciences and Engineering Research Council of Canada. The funding they have provided me in my graduate studies has been a key motivation for me continuing to pursue my academic goals, and I feel fortunate to live in a country that has such an institution.

My PhD studies have been made more enjoyable by my friends and colleagues in the Beta lab, as well as other computer science students at UBC. Inevitably, some of these people have moved on to bigger and better things, but I hope to keep them as friends for a long time to come.

Finally, I would like to thank my friends and family, for their love, encouragment, and cooking.

Dedication

For dad

Chapter 1

Introduction

Imprecision is usually unavoidable in geometric algorithms. Despite this fact, such algorithms are typically presented as if imprecision does not exist: they implicitly assume that the input values exactly represent some true value, when they actually include some degree of imprecision. This thesis argues that geometric algorithms should treat this imprecision as an explicit component of the input, and reflect this imprecision in the output. We address some fundamental problems from computational geometry involving (planar) point inputs, and recast them into problems where each point's precise location is unknown but is guaranteed to lie within a known region of uncertainty. Typically, these problems will come in one of two flavours: a *guaranteed* problem, whose solution expresses qualities that are shared by the solutions for every possible choice of precise inputs; and a *possible* problem, whose solution expresses qualities that are shared by the solution for at least one possible choice of precise inputs. By structuring the problems in this way, we ensure that the solutions reflect the strongest possible guarantees that can be made given the imprecision in the input.

1.1 Uncertainty, Precision, and Accuracy

Geometric algorithms rarely produce answers that can be viewed with complete certainty. Three sources of imprecision that contribute to this uncertainty have been identified by Löffler [45]. In this section, we examine each of these three

types of imprecision, and discuss how they will be addressed in this thesis.

We can label the first of the three types as measurement imprecision. When the input to a geometric algorithm is derived from objects in the physical world, there is inevitably some device or instrument (which may incorporate one or more human senses) that is used to take a measurement. Since these are real-world objects, they necessarily produce readings that are accompanied by some degree of imprecision. For example, the distance between two entrances to a termite hill may be measured with a ruler that is calibrated in sixteenths of an inch, and the location of the termite hill may be read from a GPS (Global Positioning Satellite) device that reports locations using some number of significant digits. Both instruments incorporate some degree of imprecision.

The second type of imprecision, modeling imprecision, concerns the way a real-world object is represented within a geometric algorithm. The rim of a canyon cannot be described exactly with limited resources, since this would entail measuring the location of the infinitely many points defining its boundary. Instead, some finite number of samples are usually taken, and the canyon rim is then represented as a geometric curve. Such a curve is clearly only an approximation of the true canyon rim. As another example, consider the task of recording the locations of every tree on a particular farmer's field. It is natural to construct a two-dimensional map of the trees. Such a map can only be an approximation to what is actually a portion of a three-dimensional object, the surface of the Earth. While it is true that the error introduced in this way is probably minimal (unless the field is the size of Belgium), the fact remains that in this example, some modeling imprecision cannot be avoided.

The third type of imprecision concerns the way computers store and manipulate numbers to calculate results, and can be labelled computational imprecision. It is usually implicitly assumed that the inputs to an algorithm are real numbers, yet such numbers cannot be exactly represented by computers (except for some subsets of real numbers, such as integers or rational numbers). Geometric algorithms are often presented as if they are to be run on a theoretical computer that can perform exact calculations on real numbers (one such machine is the *Real RAM*, or Random Access Machine). In practice, real numbers must be represented and manipulated using finite resources. This is usually done by approximating a real number with a

floating point value, which uses (typically 32 or 64) bits to represent a finite subset of the real numbers. This can result in programs that are *nonrobust*: for some valid input, they produce an incorrect or inaccurate result, or fail to return a result at all (e.g., by entering an infinite loop, or crashing).

An aspect of uncertainty that is closely related to precision is accuracy. Whereas the precision associated with a value determines how the value is stored (e.g., the number of digits to the right of the decimal point, or the resolution of a bitmap), accuracy represents how close the stored value is to the true value. Accuracy and precision are independent notions. For example, the number 1.9998 can be accurately, yet imprecisely, represented by the integer 2, and it can be inaccurately, yet precisely, represented by the number 3.1416. Clearly, precision without accuracy is not very useful; hence, when possible, the precision of a value (e.g. a reading, a measurement, or a calculation) is often reduced so that the value's inaccuracy matches its imprecision.

1.2 Robustness Issues

It is easy to find examples of geometric algorithms that lead to nonrobust programs. Consider the following problem: given nonparallel lines L_1 and L_2 in the plane, first calculate their point of intersection $p = L_1 \cap L_2$, then determine whether p lies on L_1. Clearly, for any nonparallel input lines, the algorithm should return true; yet when implemented using standard floating point libraries, the resulting program returns false nearly half the time [34].

Many techniques have been developed to address the robustness issues that arise when implementing geometric algorithms. Perhaps the most common method is 'epsilon tweaking': when deciding whether the result x of a computation is zero, the program returns true iff x is within some small constant ϵ of zero (i.e, iff $|x| < \epsilon$). This is usually implemented in an ad-hoc fashion, often with many different values of epsilon spread throughout a program. While this technique has the advantage of being simple, it does not solve the problem: the probability of false negatives is reduced only at the expense of increased probability of false positives, and the possibility of endless loops and program crashes still exists.

A more systematic approach for addressing robustness issues is the *exact com-*

putation paradigm [77], in which a program represents real numbers exactly. This technique is not without its drawbacks: first, the numbers that are represented are restricted to a small subset of real numbers (e.g., rational numbers); and second, the use of such methods often results in much longer execution times.

Another technique is *epsilon geometry* [30], in which standard geometric objects (e.g., points) are effectively replaced by 'fat' geometric objects (e.g., discs). Epsilon geometry employs interval arithmetic [52] to replace yes/no predicates with partial yes/no/maybe predicates.

While robustness issues often result from rounding errors and other shortcomings inherent in standard floating point libraries, they can also be a consequence of degeneracies in the inputs. For example, geometric algorithms often assume that the intersection between a pair of line segments is either the empty set, or a single point interior to both segments. When given a pair of segments that are collinear, or whose intersection consists of one of the segment's endpoints, the algorithm can misbehave in unforseen ways. For this reason, many geometric algorithms (including most of the ones presented in this thesis) make various *general position* assumptions: no two lines are collinear, no three lines intersect at a common point, and so on. Modifying an algorithm to handle such degeneracies can be complicated, and often requires significant work. A common method for dealing with degenerate inputs is to modify the inputs very slightly and at random, so that with high probability the new inputs are degeneracy-free. An implicit assumption of such *perturbation schemes* is that the output for the modified inputs is sufficiently close, in some sense, to that of the original inputs.

In some situations, it may be possible to mitigate computational imprecision at the expense of an increase in measurement imprecision. Consider an algorithm that manipulates axis-aligned rectangles. Perhaps the calculations involved will be much simpler, or at least more robust, if the coordinates of the sides of the rectangles are integers. If this is the case, we can expand each input rectangle so that it contains the original rectangle, and has integer coordinates. If the resolution of the integer grid is fine enough, the increase in measurement imprecision will be a small price to pay for the reduction in computational imprecision.

1.3 Geometric and Topological Concepts

In this section, we introduce some terminology concerning geometric and topological concepts that we will be using throughout the thesis.

All geometric objects that we will discuss are assumed to be two-dimensional: that is, subsets of $\mathbb{R} \times \mathbb{R}$. All distances will be assumed to be Euclidean, and we will denote the distance between points a and b as $d(a, b)$.

We may use the wildcard character '$*$' to denote a location within an expression where any appropriate index or variable can be substituted. If two or more wildcards appear in one expression, different values can be substituted for each occurrence.

The *polar angle* of a directed line is the angle that the line makes with the positive x-axis. The polar angle of a ray (or directed segment) is the polar angle of the directed line that contains, and is in the same direction as, the ray (or segment). When we refer to the left or right side of a particular segment that is denoted by $\overline{ab}$, we will assume the segment is directed from a to b. The polar angle of a directed line L is denoted $\theta(L)$.

We will make a distinction between circles and discs, by considering a circle to be the boundary of a disc. In other words, a disc includes the points it surrounds, while a circle does not.

In general, a region need not contain its boundary. If it does, then it is *closed*. For example, the set of points $\{(x, y) \mid 0 \leq x \leq 2, 1 \leq y \leq 3\}$, which represents a rectangle, is a closed region. By contrast, the set of points $\{(x, y) \mid x^2 + y^2 < 9\}$, which represents the interior of a disc, is not closed. The *closure* of a region is the union of a region and its boundary. A region is *bounded* if it is a subset of some disc of finite radius. In other words, there exists some constant k such that no two points of the region are separated by a distance greater than k. A region that is both closed and bounded is *compact*.

Let A and B be two regions. A and B are *disjoint* if they do not intersect, i.e., if $A \cap B = \emptyset$. If A contains B, and no part of B intersects the boundary of A, then we say that B is *interior to*, or *lies in the interior of*, A. If neither of two regions is interior to the other, then we say that the regions are *partially disjoint* (regions that are disjoint are also considered to be partially disjoint). We will say that B

penetrates A if B intersects the interior of A. To summarize, in Figure 1.1, the regions on the left are both disjoint and partially disjoint; the regions in the center are only partially disjoint; and to the right, D_j is interior to D_i. Regions D_i and D_j penetrate each other in all but the leftmost sets of both Figure 1.1 and Figure 1.2.

B is *inside-tangent* to A (or A has inside-tangent B) if $B \subseteq A$ and the boundary of A intersects B. We will say that A is tangent to B at the points where the boundaries intersect. B is *outside-tangent* to A (or A has outside-tangent B) if $B \cap A$ is a non-empty subset of the boundary of A. Again, A is tangent to B at the points in this intersection. Note that in contrast to inside-tangency, outside-tangency is reflexive: B is outside-tangent to A iff A is outside-tangent to B. We will say that directed line L is *right-tangent* to A if A lies in the right half-plane of L, and A intersects L (we may also say that A is right-tangent to L). We define *left-tangency* analogously.

We will use cw and ccw as abbreviations for clockwise and counterclockwise respectively. A *ccw range* $[\theta_1 \circlearrowleft \theta_2]$ is the set of angles from θ_1 ccw through θ_2. We will denote open or half-open sets as $(\theta_1 \circlearrowleft \theta_2)$, $[\theta_1 \circlearrowleft \theta_2)$, etc.

A *polygonal chain* is a sequence of directed line segments $\{\overline{p_1p_2}, \overline{p_2p_3}, \ldots, \overline{p_{n-1}p_n}\}$. For succinctness, we usually describe such a chain as a sequence of vertices $\{p_1, \ldots, p_n\}$. A polygonal chain is *simple* if it has no self-intersections.A polygon is a region whose boundary is a polygonal chain, and it is simple if its boundary is simple. We will assume that simple polygons have a ccw winding: the interior of the polygon lies to the left of each of its boundary's directed segments. A nonsimple polygon can be problematic, since it may not have a winding direction or interior that satisfies this definition. See, for example, polygon D_j of Figure 1.3.

Davenport-Schinzel sequences arise often in complexity proofs of geometric structures ([17],[67]). These are sequences of elements in which the number of times a particular pair of elements can appear in alternating sequence is limited. Formally, a sequence S is a *Davenport-Schinzel sequence of order* k if the following conditions hold:

(i) there are at most n distinct elements in S;

(ii) no two adjacent elements of S are the same; and

(iii) S contains no subsequence $(a, b, a, b, \ldots)$ of length $k + 2$.

It can be shown that the maximum size of a Davenport-Schinzel sequence of order 2 is $2n - 1$. For $k = 3$, the bound increases to $n\alpha(n)$, where $\alpha(\cdot)$ is the extremely slowly-growing inverse Ackermann function (for all practical values of n, $\alpha(n) < 5$). If, in addition to satisfying the three conditions above, S has the property that its first and last elements are distinct (or S has fewer than two elements), then S is a *Davenport-Schinzel cyclic sequence of order* k.

1.4 Regions of Uncertainty

A region of uncertainty is a subset of the plane that is associated with a point site whose exact location is unknown, but is guaranteed to lie within the region of uncertainty. We may refer to such a region as an uncertain region, which is perhaps misleading, since the region itself is known precisely; rather, it is the location of the point within the region that is uncertain.

Discs and squares are perhaps the two most common shapes for uncertain regions to take. As observed in [47], an uncertain disc is likely to arise when the coordinates of a point have been measured from some physical device (e.g., a scanner, or a GPS reading), while an uncertain square usually results when the point's coordinates have been stored as a pair of floating point values, with each value having independent amounts of imprecision. In addition to discs and squares, in this thesis we will also be considering other shapes, such as axis-aligned rectangles and polygons.

We will denote a set of uncertain regions $\{D_1, \ldots, D_n\}$ by $\mathcal{D}$. We will assume that each $D_i \in \mathcal{D}$ is compact. In addition, we will require the uncertain regions to satisfy (i) the general position assumption that no line contains more than two vertices of $\mathcal{D}$, and (ii) a *valid intersection* property: intersections between the boundaries of distinct regions of $\mathcal{D}$ must be proper intersections. This second condition can be stated formally as follows: if p is a point where the boundaries of distinct regions D_i and D_j of $\mathcal{D}$ intersect, then p must lie on the boundary separating two nonzero-area subsets of D_i, where one subset is strictly inside D_j, and the other is strictly outside D_j. Figure 1.1 displays examples of pairs of regions that satisfy this condition, while Figure 1.2 displays examples of pairs of regions that do not.

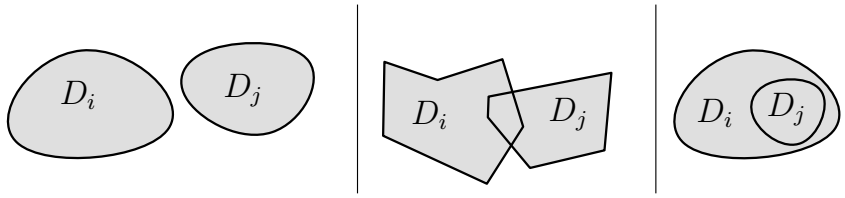

Figure 1.1: Valid pairs $D_i, D_j \in \mathcal{D}$.

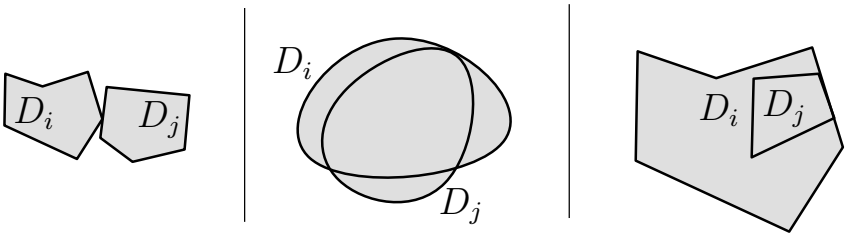

Figure 1.2: Invalid pairs $D_i, D_j \in \mathcal{D}$.

We will refer to a set of precise, pairwise distinct points drawn from uncertain regions as a *feasible set*.

If an uncertain region D_i is a disc, we will denote its origin by o_i, and its radius by r_i. If an uncertain region D_i is a polygon, we will assume that it is simple, has a ccw winding, and has at least three vertices. Figure 1.3 illustrates some valid and invalid uncertain polygons.

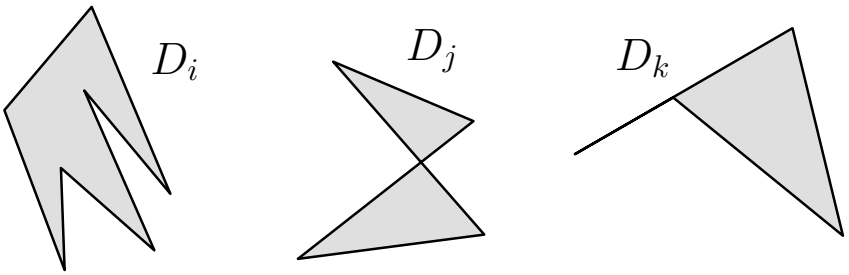

Figure 1.3: Valid (D_i) and invalid (D_j, D_k) uncertain polygons.

1.5 Contributions of this Thesis

This thesis is concerned with problems involving sets of points whose locations are uncertain, but known to lie within particular regions. We will present algorithms that when given such regions as input, produce either guaranteed or possible output objects. Together, these two objects allow us to make the strongest possible guarantees about arbitrary points in the plane, with respect to the input points. For example, consider the problem of determining the smallest axis-aligned rectangle that contains a set of input points (i.e., their 'smallest bounding rectangle'). We can solve this simple problem in linear time, and we can be confident that for any set of input points, the rectangle produced is the correct answer. Now consider the situation when the location of each input point is imprecise, and may lie anywhere within a particular disc. We cannot hope to produce the actual smallest bounding rectangle of the points, since we do not know exactly where the points lie. The strongest guarantee we can make about any particular point in the plane is to assign it to one of three sets: points that lie within the smallest bounding rectangle of every feasible set; points that lie within the smallest bounding rectangle of at least one feasible set; and a third set, which contains any points not in either of the previous two. The guaranteed object produced by our algorithm would be the first of these sets, and the possible object would be the union of the first and second sets.

Throughout this thesis, we will assume that the uncertain regions themselves are known exactly, and can be represented precisely (i.e., as a disc with known origin and radius, or as a polygon with known vertices). Since these regions are clearly mathematical entities, we will avoid introducing any modeling imprecision. We will also largely avoid addressing issues of computational imprecision, and will usually assume the use of the aforementioned Real RAM as the theoretical computing device. In addition, we will ignore any issues concerning the accuracy of the values manipulated by the algorithms.

In each of the next three chapters, we address some fundamental problems from computational geometry involving planar point sites, where each site lies within a region of uncertainty. In Chapter 2, we focus on Voronoi diagrams, and two related structures: Delaunay triangulations, and Gabriel graphs. We provide complexity

bounds and algorithms to construct guaranteed Voronoi diagrams, and to identify guaranteed Delaunay and Gabriel edges. We introduce a new geometric structure, the possible Voronoi diagram, and give a complexity result and algorithm to construct this figure. In Chapter 3, we consider the problem of convex hulls of imprecise points. We give new results on the complexity of guaranteed hulls, and new algorithms for their construction. We also investigate the possible hull of imprecise points. This is a novel problem for uncertain regions that are nonconvex (otherwise, the possible hull is just the convex hull of the regions), and we present an algorithm that constructs the possible hull of nonconvex polygons. In Chapter 4, we address the smallest bounding discs of point sets. We describe a new geometric figure, the possible 1-centers of uncertain regions, and for uncertain discs and uncertain axis-aligned rectangles, we provide two algorithms related to this figure. We also highlight a connection between this figure and guaranteed hulls, one that can be exploited to optimize algorithms involving possible 1-centers. We conclude with a brief summary of our results, and some directions for future research.

At the end of each chapter, we will provide a link to a website containing a program that illustrates some of the material presented in the chapter. Each of these programs is a Java applet which enables users to create and manipulate files of geometric objects (e.g., discs and polygons), and demonstrates some of the algorithms presented in this thesis.

Chapter 2

Voronoi Diagram of Imprecise Points

2.1 Introduction

Voronoi diagrams are a fundamental data structure in computational geometry. Given a set of points $\{p_1, \ldots, p_n\}$, the Voronoi diagram (Figure 2.1) partitions the plane into n regions, or cells, such that each point in cell i is closer to p_i than to any $p_{\neq i}$. One application of Voronoi diagrams is the *post-office problem* [42], in which the sites correspond to post office locations, and the Voronoi diagram indicates which post office is closest to a particular query point. Additional data structures can be used in conjunction with Voronoi diagrams to answer such queries efficiently (typically in logarithmic time).

There are many algorithms for constructing Voronoi diagrams. Techniques include the divide and conquer approach employed by Shamos and Hoey [65], Fortune's plane sweep method [26], and the higher dimensional embedding of Brown [12], which shows that the Voronoi diagram of a set of (planar) points can be recovered from a 3-dimensional hull induced by the points. Lee and Schachter [43] generate Voronoi diagrams by first constructing their dual, the Delaunay triangulation. Each of these algorithms runs in $O(n \log n)$ time, and (since Voronoi di-

agrams can be used to sort a set of numbers) is therefore worst-case optimal.[1] Voronoi diagrams can also be defined for other types of sites, such as circles [66] and line segments [38]. A survey of Vornoi diagrams can be found in [7].

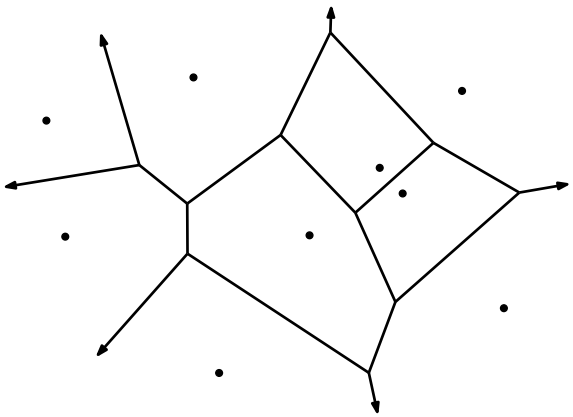

Figure 2.1: Voronoi diagram of a set of points.

If the location of each point is not known precisely, but lies within a region of uncertainty, then we cannot construct the Voronoi diagram of the points. Instead, we define the *guaranteed Voronoi diagram* for the uncertain regions to be a set of cells, where the cell for site i consists of those points that are guaranteed, for each feasible set, to lie within the i^{th} cell of the standard Voronoi diagram of the set. We denote the guaranteed Voronoi diagram of uncertain regions $\mathcal{D}$ by $\widetilde{V}(\mathcal{D})$. Figure 2.2 shows the guaranteed Voronoi diagram for a set of uncertain discs. One characteristic of guaranteed Voronoi diagrams that is immediately clear from Figure 2.2 is that for some points, we cannot guarantee a closest site. These points form a subset of the plane that we call the 'neutral zone'.

The Delaunay triangulation (or Delaunay graph) of a set of points is the dual of the Voronoi diagram of the points: an edge exists between two sites if and only if the Voronoi cells of the sites share an edge. The Delaunay triangulation has the property that for any triangle, the circumcircle of the triangle's points contains no other points from the set in its interior (Figure 2.3). It has been shown that the Delaunay triangulation is the triangulation that maximizes the minimum angle of

[1] Throughout this thesis, when discussing optimality, we will assume the standard algebraic decision tree model of computation.

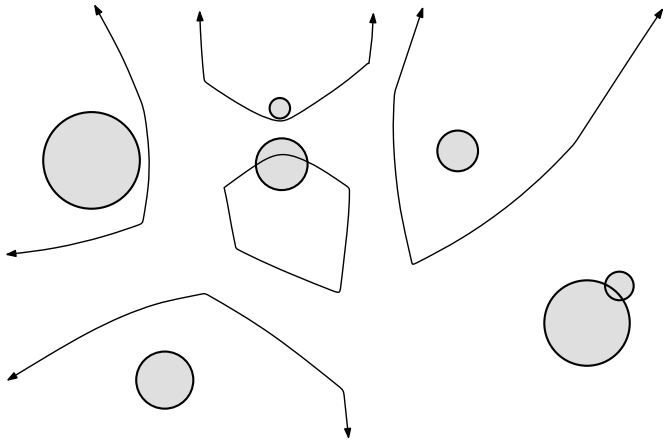

Figure 2.2: Guaranteed Voronoi diagram for uncertain discs.

its triangles. For this reason, they are usually the preferred triangulation of a set of points in graphics applications.

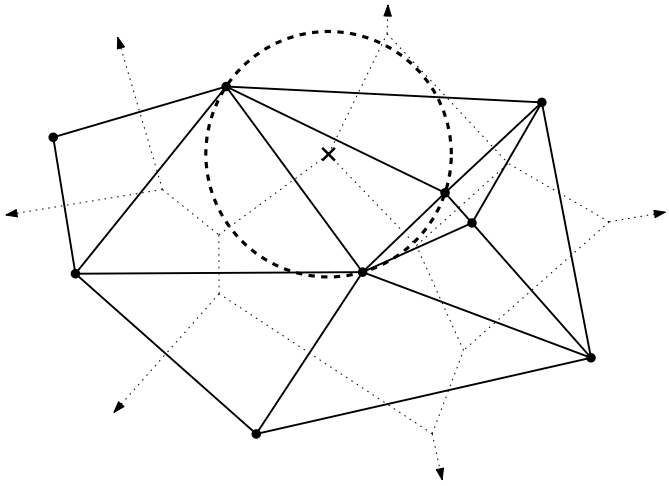

Figure 2.3: Delaunay triangulation of a set of points, the dual of the Voronoi diagram (circumcircle for one triangle is shown).

A set of imprecise points is associated with many feasible sets, and an edge that appears in the Delaunay triangulation of one feasible set may be absent in that of another feasible set. We define a *guaranteed Delaunay edge* to exist for a pair of uncertain regions drawn from a set $\mathcal{D}$ iff an edge associated with this pair exists

in the Delaunay triangulation of every feasible set of $\mathcal{D}$.

A Gabriel graph [27] is a subgraph of the Delaunay triangulation. A Gabriel edge exists between two points if the smallest disc that contains the points contains no other points in the set. We define a *guaranteed Gabriel edge* to exist for a pair of uncertain regions from $\mathcal{D}$ iff the corresponding edge exists in the Gabriel graph of every feasible set of $\mathcal{D}$.

2.1.1 Related Work

The guaranteed Voronoi diagram of a set of uncertain regions is closely related to the standard Voronoi diagram of those regions. Thus our results rely heavily on properties of standard Voronoi diagrams such as diagrams for discs ([66],[62]) and diagrams for segments [38].

There are two other generalizations of Voronoi diagrams that are closely related to our research. In an *order-k* Voronoi diagram, each cell is associated with a subset of the sites, and points within the cell have the property of being closer to each of the subset's sites than to any of the sites not in the subset. These diagrams were first introduced by Miles [51], and were investigated from an algorithmic perspective by Shamos and Hoey [65].

In an *additively weighted* Voronoi diagram, each cell is associated with a single site, but the distance from a site to a point includes an additive weighting factor. The first algorithm to construct such diagrams was given by Drysdale and Lee [20], and improved algorithms were later given by Fortune [26] and Rosenberger [62]. If the weights are negative, then these diagrams are equivalent to the standard Voronoi diagram of discs, where each site is the origin of a disc with radius equal to the negative of the site's weight.[2]

Voronoi diagrams involving uncertain sites that have a uniform probability distribution within discs were investigated in [8]. The authors defined the *expected closest disc* to a query point q to be the disc whose site has the smallest expected distance to q, and the *probably closest disc* to q to be the disc whose site is probably closest to q. These two definitions are not equivalent, and the authors have shown that while the Voronoi diagram for expected closest discs can be constructed us-

[2]The equivalence holds as long as we allow distances to be negative, e.g. when a point is in a disc's interior.

ing standard Voronoi algorithms, that of probably closest discs is more complex and cannot. Voronoi diagrams of probably closest discs have also been investigated in [4], and are shown to be useful for answering nearest neighbor queries in uncertain databases.

Khanban investigates guaranteed Voronoi diagrams of uncertain convex polygons (with a small number of edges) in [36]. He points out that the cells of these diagrams are disjoint, and that the diagram contains a neutral zone. He proves that the diagrams are computable, but does not provide any complexity bounds or algorithms.

Zone diagrams [5], like guaranteed Voronoi diagrams, have a neutral zone. In zone diagrams, for a point to be in a site's region, it must be closer to the site than to any point in any other site's region. The recursive nature of this definition raises the question of the uniqueness and existence of zone diagrams; a question that Asano *et al* [5] answered (positively).

The fundamental operation in many algorithms for calculating Delaunay triangulations is an *in-circle test*, which determines whether a (query) point is inside or outside the circumcircle associated with a triangle's points. Ely and Leclerc [24] investigated the application of interval arithmetic to this in-circle test for the case when the points are imprecise. They found that as the uncertainty in the input increases, the likelihood that the in-circle test will give reliable results decreases dramatically. To remedy this, they associated with each point a disc-shaped region of uncertainty, and showed that the set of all possible circumcircles consistent with the triangle points can be described by eight circles tangent to the triangle points' discs, and that if the query disc lies outside (resp., inside) these circles, then the in-circle test can reliably report false (resp., true). They found that an in-circle test that uses this tangent method is about sixteen times less likely to fail to report an answer than one that only uses interval arithmetic.

Khanban and Edalat [37] have argued that in practice, imprecision in input points are best represented by rectangles rather than discs, since the points are most often reported as a pair of x and y coordinates. They show how the uncertain rectangles corresponding to the three vertices of each Delaunay triangle give rise to six tangent circles, which can be used for an in-circle test in a similar manner to that of Ely and Leclerc. Khanban gives an algorithm for constructing the guaranteed

Delaunay edges of uncertain convex polygons (with a small number of edges) that that runs in expected $O(n \log n)$ time [36].

In a different take on imprecision, Löffler and Snoeyink [46] investigated the problem of preprocessing a set of disjoint uncertain unit discs to later allow efficient construction of the Delaunay triangulation of a feasible set of the discs. They show that once $O(n \log n)$ time has been spent preprocessing the uncertain discs, the Delaunay triangulation of any feasible set from the discs can be generated in $O(n)$ additional time. Randomized algorithms that extend and simplify these results are given by Buchin *et al* in [13]. For the case of uncertain discs with the property that no point lies in more than k discs, they provide an algorithm that (after an $O(n \log n)$ preprocessing step) can generate the Delaunay triangulation of any feasible set in expected $O(n \log k)$ time. The algorithms in both of these works are largely of theoretical interest, since the constants hidden in their postprocessing step running times are large enough that it is often faster just to spend $O(n \log n)$ time constructing the Delaunay triangulation of each feasible set using a standard algorithm. By contrast, Devillers [19] has shown that for disjoint uncertain discs of unit radius, there exists a much simpler randomized algorithm that produces better (expected) postprocessing running times than the standard Delaunay triangulation algorithms.

The *tolerance* of a geometric or combinatoric object is a concept that was introduced by Abellanas *et al* [1]. It refers to the object's ability to withstand perturbation while still maintaining some property. For example, the tolerance of the Delaunay triangulation of a set of points S is the maximum amount ϵ by which the points can move and still have the same Delaunay triangulation. It was shown in [2] that determining ϵ for this problem can be done in $O(n)$ time (if the Delaunay triangulation is given), or in $\Theta(n \log n)$ time (otherwise). This problem is related to our work in the following way: the tolerance of the Delaunay triangulation of S is the maximum radius that each of a set of uncertain discs $\mathcal{D}$ centered at S can have and still satisfy the property that every Delaunay edge of S is a guaranteed Delaunay edge of $\mathcal{D}$.

2.1.2 Contributions

This chapter is concerned with Voronoi diagrams, Delaunay triangulations, and Gabriel graphs of imprecise points.[3] We prove tight bounds on the complexities of the guaranteed Voronoi diagrams of uncertain discs and uncertain polygons, and present worst-case optimal algorithms for their construction.

We define a new diagram, the possible Voronoi diagram. It reports the smallest subset of regions that are guaranteed to contain the closest site to a query point. In addition to proving a tight bound on its complexity, we provide a construction algorithm that is worst-case optimal up to logarithmic terms.

Finally, we provide worst-case optimal algorithms for determining both the guaranteed Delaunay edges and the guaranteed Gabriel edges associated with a set of uncertain discs.

2.2 Guaranteed Voronoi Diagram

In this section, we will investigate some properties of guaranteed Voronoi diagrams. We will then define a useful mapping function between boundary points of standard and guaranteed Voronoi cells. We will then describe algorithms for constructing these diagrams for uncertain discs and (disjoint) uncertain polygons.

2.2.1 Properties

We denote the standard Voronoi diagram of a set of regions $\mathcal{D}$ by $V(\mathcal{D})$, and the cell corresponding to region i by R_i. We denote the order-k Voronoi diagram of a set of regions $\mathcal{D}$ by $V^k(\mathcal{D})$, and a cell of this diagram corresponding to the $|S| = k$ regions $\{D_{i \in S \subseteq \{1 \ldots n\}}\}$ by R_S. Hence $V(\mathcal{D}) = V^1(\mathcal{D})$, and $R_i = R_{\{i\}}$.

Suppose we have a set of uncertain regions $\mathcal{D}$. The *bisector* of a pair of regions D_i and D_j is the set of points that are equidistant to both regions. We denote this bisector by $\langle\langle i, j \rangle\rangle$. Formally,

$$\begin{aligned} \langle\langle i, j \rangle\rangle &= \{p \mid \min_{x \in D_i} d(p, x) = \min_{y \in D_j} d(p, y)\} \\ &= \{p \mid d(p, D_i) = d(p, D_j)\} \ . \end{aligned}$$

[3] Portions of this chapter appeared in [63].

Let $H(i,j)$ be the set of points that are guaranteed to be at least as close to site i as site j. That is,

$$H(i,j) = \{p \mid \forall x \in D_i \; \forall y \in D_j \; d(p,x) \leq d(p,y)\} \; .$$

We denote the boundary of $H(i,j)$ by $\langle i,j \rangle$; formally,

$$\langle i,j \rangle = \{p \mid \max_{x \in D_i} d(p,x) = \min_{y \in D_j} d(p,y)\} \; .$$

The *guaranteed cell* for site i, denoted $\widetilde{R}_i$, is:

$$\widetilde{R}_i = \bigcap_{j \neq i} H(i,j) \; . \tag{2.1}$$

The boundaries of all such cells $\widetilde{R}_i$ form the *guaranteed Voronoi diagram* for the set $\mathcal{D}$, and we denote it by $\widetilde{V}(\mathcal{D})$. An *edge of* $\langle i,j \rangle$ *in* $\widetilde{V}(\mathcal{D})$ is a maximal connected set of points $p \in \langle i,j \rangle$ that lie on the boundary of cell $\widetilde{R}_i$.

In order to further investigate the properties of $\widetilde{V}(\mathcal{D})$, we will require some additional terminology. Let A be a disc. *Shrinking* A refers to the process of reducing the radius of A while keeping its origin fixed. If p is a point on A's boundary, then *shrinking* A *with respect to* p refers to the process of reducing the radius of A while simultaneously moving its origin towards p, so that p remains on the boundary of (the shrinking) A. If p is a tangency point of A and some region B, then shrinking A with respect to p ensures that (the shrinking) A and B will remain tangent at p. When clear from the context, we refer to such an operation as *shrinking* A *with respect to* B.

Lemma 2.1 *Point* p *is in* $\widetilde{R}_i$ *in* $\widetilde{V}(\mathcal{D})$ *iff there exists a disc* C_p *centered at* p *such that (i)* C_p *has inside-tangent* D_i*, and (ii)* $\mathcal{D} \setminus \{D_i\}$ *does not penetrate* C_p*. In addition,* p *is on the boundary of* $\widetilde{R}_i$ *iff* C_p *has outside-tangent some* $D_{j \neq i} \in \mathcal{D}$*.*

Proof. This follows immediately from the definitions of $\widetilde{R}_i$ and $H(i,j)$. □

Figure 2.4 illustrates Lemma 2.1.

Lemma 2.2 *If region* D_i *of* $\mathcal{D}$ *is a single point, then* $\widetilde{R}_i = R_i$*.*

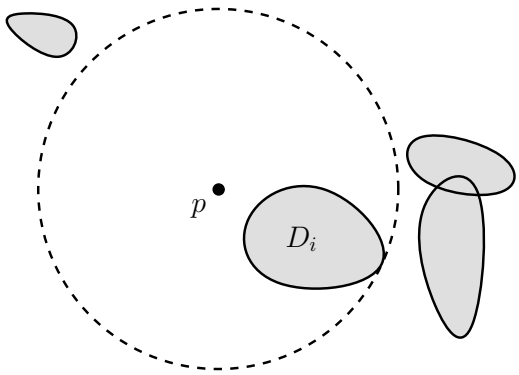

Figure 2.4: Lemma 2.1: $p \in \widetilde{R}_i$.

Proof. Suppose D_i is a single point. We can apply an analog of Lemma 2.1 (one concerned with standard Voronoi diagram cells) to show that for every point $p \in R_i$, there exists a disc C_p that has outside-tangent D_i, and is not penetrated by $\mathcal{D} \setminus \{D_i\}$. Since D_i is a point, each such C_p also has inside-tangent D_i, satisfying Lemma 2.1. □

Lemma 2.3 *Every cell $\widetilde{R}_i$ of $\widetilde{V}(\mathcal{D})$ is a subset of the corresponding cell R_i in the standard Voronoi diagram $V(\mathcal{D})$.*

Proof. If p is a point within $\widetilde{R}_i$, then the distance from p to the farthest point of D_i is not greater than the distance from p to the nearest point of any region $D_{\neq i}$. Since the former distance is an upper bound on the distance from p to the *nearest* point of D_i, p must lie within R_i as well. □

Lemma 2.4 *If D_i and D_j are intersecting regions of $\mathcal{D}$, then both $\widetilde{R}_i$ and $\widetilde{R}_j$ will be empty.*

Proof. Let D_i and D_j be intersecting regions of $\mathcal{D}$, and p be any point in the plane (Figure 2.5). Observe that there must exist a feasible set of $\mathcal{D}$ in which s_i is closer to p than is s_j, which implies that $\widetilde{R}_j$ is empty. A symmetric argument implies that $\widetilde{R}_i$ is empty as well. □

A region whose cell is empty can still influence the cell of another region. For example, the two uncertain discs in the bottom right of Figure 2.2 have empty cells, yet still induce edges in some nearby cells.

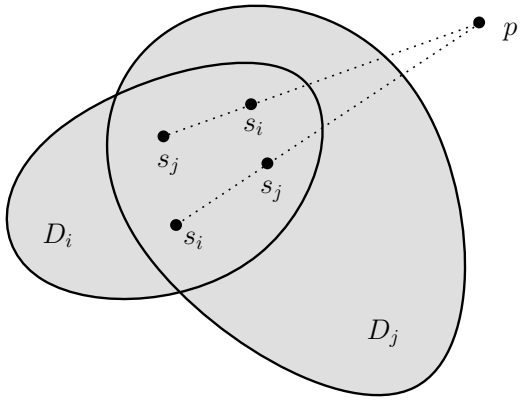

Figure 2.5: Lemma 2.4.

Lemma 2.5 *Each cell $\widetilde{R}_i$ of $\widetilde{V}(\mathcal{D})$ is convex.*

Proof. Let p and q be arbitrary points in $\widetilde{R}_i$. We show that segment $\overline{pq}$ lies within $\widetilde{R}_i$. Lemma 2.1 implies that there exist discs C_p centered at p and C_q centered at q that contain D_i, and are penetrated by no disc $D_{\neq i}$. This implies that $D_i \subseteq C_p \cap C_q$. Now, each point $p' \in \overline{pq}$ is the center of a disc $C_{p'}$ that contains $C_p \cap C_q$, and lies within $C_p \cup C_q$; hence $C_{p'}$ satisfies condition (ii) of Lemma 2.1. We can shrink $C_{p'}$ until it has inside-tangent D_i, at which point it will satisfy condition (i) as well. Thus $p' \in \widetilde{R}_i$. □

2.2.2 A Mapping between Boundaries of Voronoi Cells

We will make extensive use of a mapping from boundaries of cells of $\widetilde{V}(\mathcal{D})$ (i.e., the guaranteed diagram) to boundaries of cells of $V(\mathcal{D})$ (i.e., the standard diagram). We will construct the mapping, $\delta(\cdot)$, in the following way. Consider a point p on the boundary of $\widetilde{R}_i$ in $\widetilde{V}(\mathcal{D})$, and its disc C_p from Lemma 2.1. Since p is on the boundary of the region, C_p has outside-tangent some $D_{m\neq i}$ at some point a;[4] see Figure 2.6.

We now shrink C_p with respect to a, until it reaches $C_{p'}$ that has outside-tangent

[4] If more than one region D_m is outside-tangent to C_p (or if D_m is tangent to C_p at more than one point), then we make $\delta(p)$ well-defined by selecting point a according to some total order on possible a's.

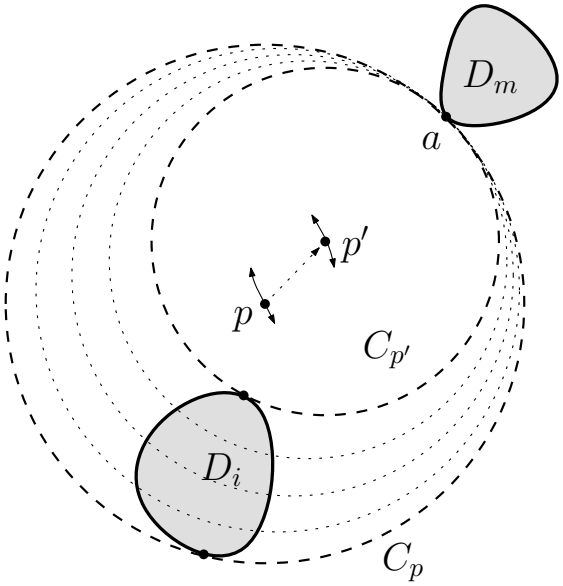

Figure 2.6: Definition of $\delta(p)$.

D_i. Now, p' is on an edge of $\langle\langle i, m\rangle\rangle$ on the boundary of $R_i \subseteq V(\mathcal{D})$, and we define $\delta(p)$ to be this point p'.

If we start at a point s on the boundary of a connected region, and traverse the boundary in a ccw direction, then we denote a point p being encountered before q during this traversal by $p \prec_s q$.

Lemma 2.6 *If p, q, and s are points on the boundary of cell $\widetilde{R}_i$ within $\widetilde{V}(\mathcal{D})$, and $p \prec_s q$, then $\delta(p) \preceq_{\delta(s)} \delta(q)$.*

Proof. We will first show that if the $\delta(\cdot)$ mapping does not preserve this ordering, then some pair of the line segments $\overline{p\delta(p)}$, $\overline{q\delta(q)}$, $\overline{s\delta(s)}$ must intersect; we will then derive a contradiction from this fact.

Let p be a boundary point of $\widetilde{R}_i$, and L be segment $\overline{p\delta(p)}$ (Figure 2.7). Now, L does not penetrate $\widetilde{R}_i$; for if it did, some point p' interior to L must lie in the interior of $\widetilde{R}_i$, but no disc centered at p' can contain D_i without also being penetrated by some $D_{\neq i}$. Note also that L is within R_i, since for each $p' \in L$, (i) there exists a disc centered at p' that intersects D_i and that has outside-tangent some $D_{\neq i}$, and (ii) no smaller disc at p' intersects any $D_{\neq i}$.

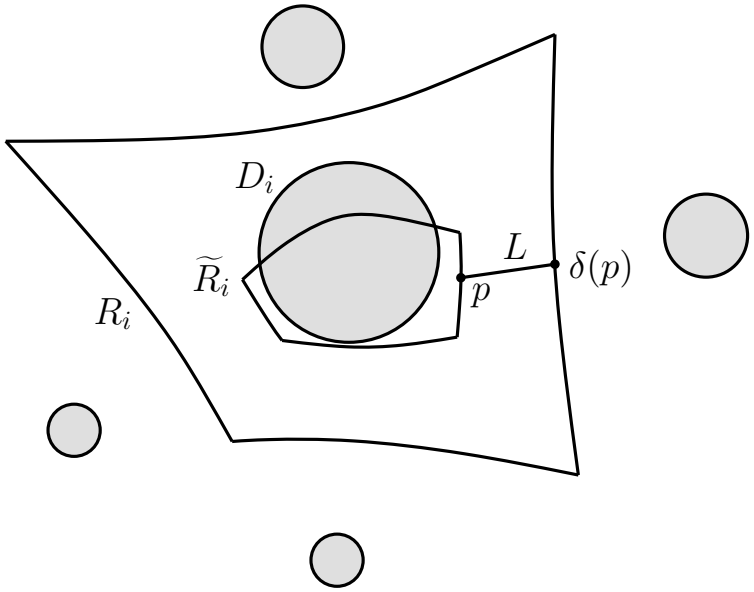

Figure 2.7: Lemma 2.6.

Thus if $p \prec_s q$ and $\delta(p) \succ_{\delta(s)} \delta(q)$, then $p \neq q$, $\delta(p) \neq \delta(q)$, and some two of the segments $\overline{s\delta(s)}$, $\overline{p\delta(p)}$, $\overline{q\delta(q)}$ intersect. Without loss of generality, assume $\overline{p\delta(p)}$ intersects $\overline{q\delta(q)}$ at a point v.

By Lemma 2.1, there exists disc C_p which has outside-tangent some $D_{\neq i}$ at a, such that $\delta(p) \in \overline{pa}$. Similarly, disc C_q exists which has outside-tangent some $D_{\neq i}$ at $b(\neq a)$ where $\delta(q) \in \overline{qb}$. The mapping $\delta(\cdot)$ implies that every point on $\overline{p\delta(p)}$ is the center of a disc that is penetrated by no $D_{\neq i}$, and has outside-tangent some $D_{\neq i}$ at a. Since a symmetric situation holds for q, $\delta(q)$, and b, v must be the center of a disc C_v that has a and b on its boundary. Now observe that every point of C_v except a must lie in the interior of C_p, including b (Figure 2.8); hence C_p is penetrated by some $D_{\neq i}$, a contradiction. □

2.2.3 Uncertain Discs

We now consider the case where the uncertain regions are discs (e.g., Figure 2.2). Each disc has a nonnegative radius r_i, and a center o_i. Every point $p \in \langle i, j \rangle$

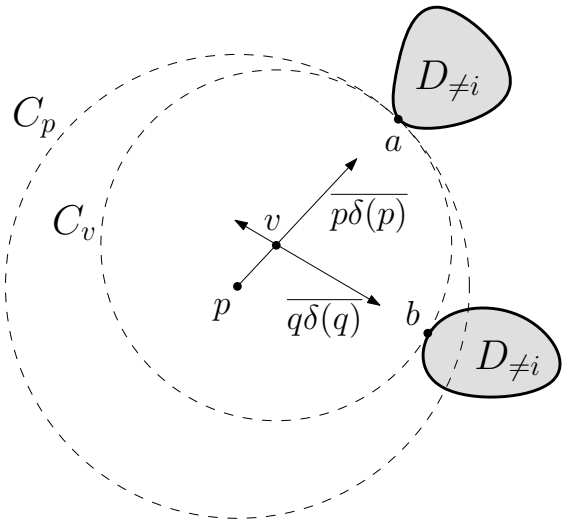

Figure 2.8: Lemma 2.6: $\overline{p\delta(p)}$ and $\overline{q\delta(q)}$ cannot intersect.

satisfies

$$d(p, o_i) + r_i = d(p, o_j) - r_j \; . \tag{2.2}$$

Since r_i and r_j are constants, the points p which satisfy (2.2) lie on the arm of the hyperbola with foci at o_i and o_j that is closest to o_i (if D_i and D_j intersect, then $\langle i, j \rangle = \emptyset$).

The boundary of a cell $\widetilde{R}_i$ within $\widetilde{V}(\mathcal{D})$ may contain two or more distinct edges of $\langle i, j \rangle$; see Figure 2.9.

We now show that the number of edges in the guaranteed Voronoi diagram of a set of uncertain discs is at most twice that of the standard Voronoi diagram of the discs. We do this by showing that each edge of each cell of the guaranteed diagram maps to a distinct edge in the corresponding cell of the standard diagram. The doubling factor is explained by the fact that each edge of the standard diagram is shared by two cells, whereas edges of the guaranteed diagram typically are not.

Theorem 2.7 *The guaranteed Voronoi diagram of n uncertain discs has $O(n)$ edges.*

Proof. We will show that the number of edges in $\widetilde{V}(\mathcal{D})$ is at most twice the number of edges in $V(\mathcal{D})$. The theorem then follows from the fact that $V(\mathcal{D})$ has $O(n)$

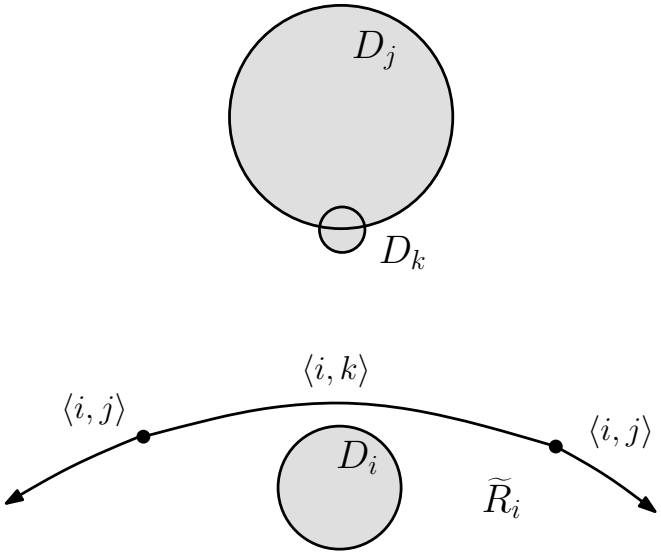

Figure 2.9: Multiple edges of $\langle i,j\rangle$ on the boundary of $\widetilde{R}_i$.

edges [66].

Consider the edges around $\widetilde{R}_i$ in ccw order. Lemma 2.5 ensures that these edges are adjacent. We charge each edge E of $\langle i,j\rangle \in \widetilde{V}(\mathcal{D})$ to the edge F of $\langle\langle i,j\rangle\rangle$ on which $\delta(p)$ lies, for p the ccw-first point of E (or any interior point p if E is ccw-infinite). Suppose two distinct edges E_1 and E_2 of $\langle i,j\rangle$ map to the same edge F of $\langle\langle i,j\rangle\rangle$ in $R_i \subseteq V(\mathcal{D})$. Since E_1 and E_2 are distinct but both of $\langle i,j\rangle$, there must exist an edge E' of $\langle i,k\rangle$ (with $k \neq j$) between them in the ccw traversal of $\widetilde{R}_i$ that maps to some other edge F' of $\langle\langle i,k\rangle\rangle$ in R_i. This contradicts Lemma 2.6 since all points of F either precede or follow the points of F' in ccw order.

Thus each edge in $V(\mathcal{D})$ of $\langle\langle i,j\rangle\rangle$ is charged at most twice: once by an edge of $\langle i,j\rangle$, and once by an edge of $\langle j,i\rangle$. Hence $\widetilde{V}(\mathcal{D})$, like $V(\mathcal{D})$, has $O(n)$ edges. □

We now show how $\widetilde{V}(\mathcal{D})$ for a set of discs can be constructed by first constructing $V(\mathcal{D})$ for the discs, then performing a linear-time transformation from $V(\mathcal{D})$ to $\widetilde{V}(\mathcal{D})$.

Let $\widetilde{I}_i$ be the sequence of pairs (i,j) corresponding to the edges $\langle i,j\rangle$ encoun-

tered in a ccw traversal of the boundary of $\widetilde{R}_i$, when starting from an edge containing some point s on the boundary. We similary define I_i to be the sequence of pairs (i, j) corresponding to the edges $\langle\langle i, j \rangle\rangle$ traversed on the boundary of R_i, when starting from the edge containing $\delta(s)$.

Lemma 2.8 *For every cell $\widetilde{R}_i \in \widetilde{V}(\mathcal{D})$, $\widetilde{I}_i$ is a subsequence of I_i.*

Proof. If (i, j) is in $\widetilde{I}_i$, then since $\delta(\cdot)$ maps points on edges of $\langle i, j \rangle$ to points on edges of $\langle\langle i, j \rangle\rangle$, (i, j) must be in I_i. Furthermore, the order of pairs in $\widetilde{I}_i$ is preserved in I_i since $\delta(\cdot)$ preserves this order by Lemma 2.6. □

One complication arises if the discs are not partially disjoint. In order to discuss this situation, we require some additional terminology. A disc $D_i \in \mathcal{D}$ is an *outer* disc if it is not contained by any other disc of $\mathcal{D}$. In addition, if D_i is the largest (outer) disc containing D_j, then we will say that D_j is the *child* of D_i, and that D_i is its *parent*. Note that while a child may be contained by several discs, it has a single parent, and is a child of only that parent.[5]

Each child disc will have an empty region within $V(\mathcal{D})$, since every point in the plane will be closer (allowing for negative distances) to the child's parent than it is to the child. The standard Voronoi diagram thus offers no evidence that one disc contains another, which may lead to problems since (by Lemma 2.4) the parent's guaranteed cell is actually empty. With these facts in mind, we are ready to describe our algorithm for constructing the guaranteed Voronoi diagram of uncertain discs.

Theorem 2.9 *$\widetilde{V}(\mathcal{D})$ for n uncertain discs can be constructed in $O(n \log n)$ time.*

Proof. We construct $V(\mathcal{D})$ for the disc sites in $O(n \log n)$ time, using Fortune's algorithm [26]. That algorithm can determine, as a side effect, (i) the set of outer discs, (ii) which outer discs are parents, and (iii) the set of children of each parent. By Lemma 2.4, every parent will have an empty cell within $\widetilde{V}(\mathcal{D})$, as will each of its children. Thus, we need construct guaranteed cells only for outer discs that are not parents.

[5] In the case where two discs of equal size contain a particular disc, we will arbitrarily choose one of the containing discs to be its parent.

For each such disc D_i, we construct its guaranteed cell from its standard cell R_i as follows. We generate the sequence I_i of sites comprising the boundary of R_i in linear time by a simple traversal. We then construct the boundary of $\widetilde{R}_i$ by generating and intersecting the sequence of hyperbolic arcs $\langle i, j \rangle$ for each pair $(i, j) \in I_i$. If any of these hyperbolic arcs do not exist, then D_i and D_j must intersect, which implies (by Lemma 2.4) that $\widetilde{R}_i$ is empty. Lemma 2.8 ensures that we consider a correctly ordered super-sequence of the arcs bounding $\widetilde{R}_i$; hence, by performing a straightforward clipping procedure, cell $\widetilde{R}_i$ can be constructed from this super-sequence in time proportional to the length of I_i.

Observe that if D_j is the parent of D_k, and D_i contains D_k as well, then our algorithm will still attempt to construct a guaranteed cell for D_i. It will correctly produce an empty guaranteed cell, since D_j must intersect D_i (and hence $\langle i, j \rangle$ will not exist), and D_j will be a neighbor of D_i within the standard Voronoi diagram.[6]

Since each of the $O(n)$ edges of $V(\mathcal{D})$ appears in two cell boundaries, the running time for the construction of the edges of all cells of $\widetilde{V}(\mathcal{D})$ is $O(n)$. The time to construct $\widetilde{V}(\mathcal{D})$ is thus dominated by the time to construct $V(\mathcal{D})$. □

The algorithm of Theorem 2.9 is worst-case optimal. To see this, observe that when the disc radii are all zero, $\widetilde{V}(\mathcal{D})$ is the standard Voronoi diagram of n points, which requires $\Omega(n \log n)$ time to construct [65].

2.2.4 Uncertain Polygons

In this section, we present an efficient algorithm for constructing the guaranteed Voronoi diagram of a set of disjoint[7] uncertain polygons (Figure 2.10). As noted earlier, this is an important class of uncertain region, since it includes axis-aligned squares, a type of region likely to be encountered in practice. The algorithm will take the same approach as that for the case of uncertain discs: namely, it will start by constructing $V(\mathcal{D})$, the standard Voronoi diagram of the polygons, then will transform each cell R_i of this diagram to its guaranteed counterpart, $\widetilde{R}_i$. Our analysis of the algorithm's running time will also prove that these diagrams have linear complexity.

[6] Actually, D_j might not be a neighbor of D_i, but only if some other (outer) disc that intersects

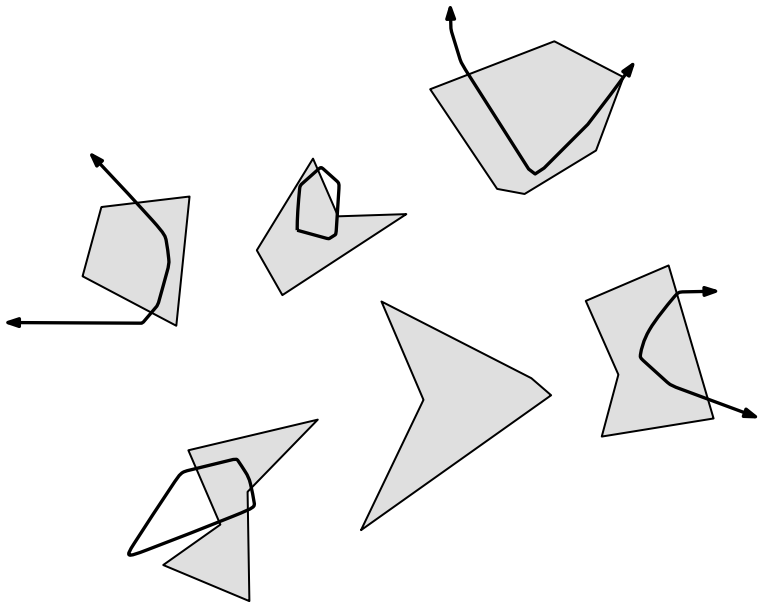

Figure 2.10: Guaranteed Voronoi diagram of uncertain polygons.

Throughout this section, we assume that $\mathcal{D}$ is a set of disjoint uncertain polygons that satisfies the following general position conditions: no vertex is contained in another polygon's edge, no three edges intersect at a common point, and no disc exists that contains four points that are either polygon vertices, or points where the disc is tangent to a polygon edge.

To further simplify the following discussion, we will also assume that the cells R_i and $\widetilde{R}_i$ are both bounded. The algorithm can be easily modified to deal with unbounded regions, without affecting its running time. Alternatively, we could view $\mathcal{D}$ as being a cluster of polygons centered within a large rectangle, and add a small triangle to $\mathcal{D}$ at each corner of this rectangle. This would ensure that the Voronoi cells of the original polygons are bounded, but would not change the structure of these cells except near the box boundaries.

The boundary of each cell R_i is composed of subsets of bisectors $\langle\!\langle e_i, e_j \rangle\!\rangle$, where e_i is an edge of polygon D_i, and e_j is an edge of a polygon $D_{j \neq i} \in \mathcal{D}$.

D_i is.

[7] Section 2.6 explains why we require the polygons to be disjoint.

To aid in our subsequent analysis, we note that each boundary point p on such a bisector is the center of a disc C_p that is not penetrated by any polygon of $\mathcal{D}$, and has outside-tangent[8] edges e_i and e_j of polygons D_i and D_j respectively.

The boundary of a guaranteed Voronoi cell is similarly composed of subsets of (guaranteed) bisectors $\langle v, e\rangle$, where v is a vertex of D_i, and e is an edge of some $D_{j\neq i} \in \mathcal{D}$. Each point p on such a bisector is the center of a disc C_p which has inside-tangent D_i at v, has outside-tangent D_j at e, and is not penetrated by any polygon $D_{\neq i} \in \mathcal{D}$. We will refer to C_p as a *guaranteed disc for* $\widetilde{R}_i$.

As a consequence of our general position assumption, each vertex p of $\widetilde{R}_i$ can be identified as a triple $\langle v_1, v_2, e\rangle$ or a triple $\langle v, e_1, e_2\rangle$. The former corresponds to a guaranteed disc C_p that has inside-tangent D_i at vertices v_1 and v_2, and has outside-tangent edge e, while the latter corresponds to a guaranteed disc C_p that has inside-tangent D_i at v, and has outside-tangent distinct edges e_1 and e_2. In both cases, we order the triples to agree with the ccw order of the tangent points of these vertices and edges with C_p.

Lemma 2.10 *Each edge of $\widetilde{R}_i$ that lies on bisector $\langle v, e\rangle$ will terminate at a vertex of the form $\langle v, v', e\rangle$ or $\langle v, e, e'\rangle$.*

Proof. Each point $p \in \langle v, e\rangle$ that lies on the boundary of $\widetilde{R}_i$ is the center of a guaranteed disc C_p in a family of such discs tangent to v and e. Observe that the arc on the boundary of each C_p from v ccw to (the point of tangency with) e is receding, in the sense that the interior points of this arc will not appear in subsequent discs in the family. Similarly, the boundary arc from e ccw to v is advancing, in the sense that points interior to the arc will be in the interior of the subsequent discs in the family (unless they appear on the receding arc of a later disc, which may occur as the point of tangency with e changes). Now, note that for p to be a vertex of $\widetilde{R}_i$, some vertex of D_i must appear on the receding arc of C_p, or some edge of a polygon $D_{\neq i} \in \mathcal{D}$ must appear on the advancing arc of C_p. □

Let $\mathbb{V}_i$ represent the ccw-ordered sequence of vertices on the convex hull of polygon D_i, and $\mathbb{E}_i$ represent the ccw-ordered sequence of edges e that appear in

[8]We can consider tangency with an edge endpoint as a special case of tangency with the edge itself, and to eliminate ambiguity, we can define each (directed) edge as closed at its source endpoint, and open at its destination endpoint.

the bisectors $\langle\langle \cdot, e \rangle\rangle$ that form the boundary of cell R_i. We will let k denote the total complexity of $\mathbb{V}_i$ and $\mathbb{E}_i$.

Lemma 2.11 *The sequence of vertices of uncertain polygon D_i that induce vertices of $\widetilde{R}_i$ are a subsequence of $\mathbb{V}_i$.*

Proof. We will denote point u preceding point v on a (ccw oriented) region boundary by $u \prec v$. Let p, q, and r be distinct boundary points of $\widetilde{R}_i$, with $p \prec q \prec r$. Since $\widetilde{R}_i$ is convex, r must lie to the left of $\overline{pq}$.

For each point s on the boundary of $\widetilde{R}_i$, there exists a guaranteed disc C_s that has inside-tangent D_i at a vertex t_s. Clearly, t_s must be on the hull of D_i; hence, we need only show that the ordering of these vertices agrees with that of the vertices of $\widetilde{R}_i$.

Now, D_i must lie in the common intersection $I = C_p \cap C_q \cap C_r$. We can show (by our general position assumption, and the fact that each disc intersects, but is not penetrated by, some $D_{\neq i}$) that none of the three discs is contained by the union of the other two. This implies that the boundary of I is comprised of a single circular arc from each of C_p, C_q, and C_r, and that these arcs have a ccw ordering consistent with that of p, q, and r (Figure 2.11). Since each disc's tangent point must lie on its corresponding arc, this implies that $t_p \preceq t_q \preceq t_r$; and as this holds for any choice of distinct points $p \prec q \prec r$ on the boundary of $\widetilde{R}_i$, the proof is complete. □

Lemma 2.12 *If disc C_p contains polygon D_i and is penetrated by any edge of $\mathcal{D} \setminus \{D_i\}$, then it will be penetrated by some edge of $\mathbb{E}_i$.*

Proof. Suppose $D_i \subset C_p$, and C_p is penetrated by edge e of $U = \mathcal{D} \setminus \{D_i\}$. Starting with $C = C_p$, we can shrink it until it has outside-tangent either D_i or U, then shrink it further with respect to that tangency point until it has outside-tangent both D_i and some edge e of U, and is not penetrated by $\mathcal{D}$. Now, the final C is a guaranteed disc, hence $e \in \mathbb{E}_i$; and since it lies within C_p, e penetrates C_p. □

We will construct $\widetilde{R}_i$ in two stages. The first stage finds a guaranteed disc for $\widetilde{R}_i$. If none exists, then $\widetilde{R}_i$ must be empty. Otherwise, its origin is a point on the boundary of $\widetilde{R}_i$, and starting from this point, the second stage generates the ordered sequence of vertices of $\widetilde{R}_i$.

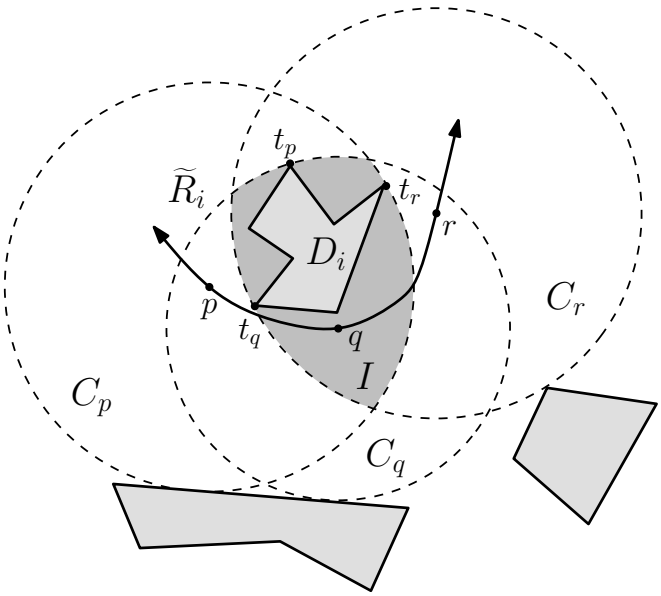

Figure 2.11: Lemma 2.11.

We begin the first stage by constructing the smallest disc containing polygon D_i. If this disc is not penetrated by any other polygons of $\mathcal{D}$, then we perform the following *grow operation*: we expand the disc, maintaining its inside-tangency with D_i, until it has outside-tangent some $D_{\neq i}$, at which time it will be a guaranteed disc. If instead, the disc is penetrated by some edge of a polygon $D_{\neq i}$, then we perform a *move operation*: we transform the disc, maintaining its tangency with D_i, until no edges penetrate the disc (or we determine that no guaranteed disc exists).

We now describe these steps with more detail. We first construct the farthest vertex Voronoi diagram of $\mathbb{V}_i$, and the nearest edge Voronoi diagram of $\mathbb{E}_i$, in $O(k \log k)$ time. Using the farthest vertex Voronoi diagram, we can find C_s, the smallest bounding disc of $\mathbb{V}_i$, in $O(k)$ time [65]. We query the nearest edge Voronoi diagram to determine the nearest edge $e \in \mathbb{E}_i$ to s. If e does not penetrate C_s, then by Lemma 2.12, no edge of $\mathcal{D} \setminus \{D_i\}$ will either, which implies that $s \in \widetilde{R}_i$. To perform the grow operation, we start with any vertex $v \in \mathbb{V}_i$ that lies on the boundary of C_s. We then determine (in linear time, by iterating over $\mathbb{E}_i$) the small-

est disc C_p that is (i) centered at a point on ray $\overrightarrow{vs}$, (ii) tangent to v, and (iii) tangent to some edge of $\mathbb{E}_i$. Now, C_p must be a guaranteed disc for $\widetilde{R}_i$, by Lemma 2.12.

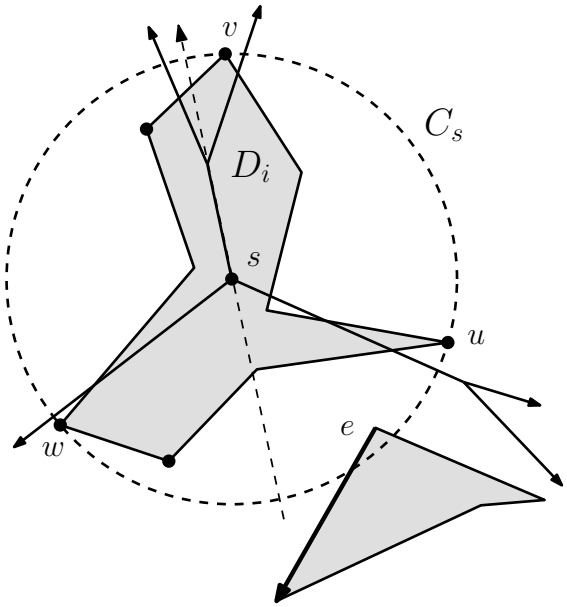

Figure 2.12: Finding point on boundary of $\widetilde{R}_i$. Also shown are farthest point Voronoi diagram of $\mathbb{V}_i$ (solid lines) and bisector of u and w (dashed line).

If the bounding disc C_s is penetrated by edge e of some other polygon, we perform the move operation instead. Its steps are a little more involved, and will require the following lemma.

Lemma 2.13 ([18], Lemma 9.4) *If disc C with boundary points α and β intersects points p and q lying on opposite sides of directed line $\overrightarrow{\alpha\beta}$, then every other disc intersecting α and β will be penetrated by either p or q (Figure 2.13).*

Let u and v be two of the three vertices of $\mathbb{V}_i$ that lie on the boundary of C_s, such that some portion of e lies in the interior of C_s to the left of $\overrightarrow{uv}$ (Figure 2.12). We transform C_s by moving its center towards the right of $\overrightarrow{uv}$, maintaining its tangency with u and v. In a manner analogous to that of the proof of Lemma 2.10, the boundary of C_s can be partitioned into advancing and receding arcs, separated by the ray $\overrightarrow{uv}$. We transform C_s until one of three events occurs (Figure 2.14):

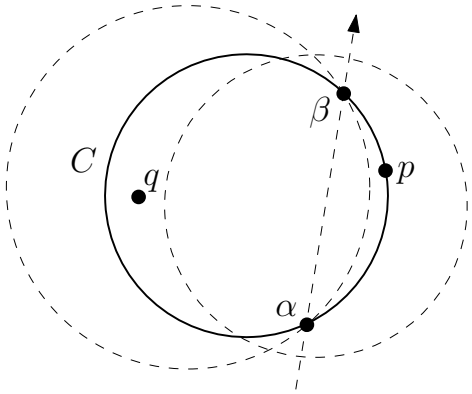

Figure 2.13: Lemma 2.13.

Event 1: e does not penetrate C_s.
Just prior to this point, C_s was penetrated by some edge of $\mathbb{E}_i$; hence, it must now have outside-tangent one of these edges on its receding arc. Since it also has inside-tangent D_i, we can return C_s as a guaranteed disc.

Event 2: Some other vertex $w \in \mathbb{V}_i$ appears on the receding arc of C_s.
We must replace either u or v with w before continuing to transform C_s, otherwise w will not be contained by C_s. If e penetrates the portion of C_s lying to the left of $\overline{uw}$ (as in Figure 2.14), then we replace v with w; otherwise, we replace u with w. Afterwards, we continue transforming C_s.

Event 3: e penetrates C_s to the right of $\overrightarrow{uv}$.
Observe that just prior to this point, C_s was penetrated to the left of $\overrightarrow{uv}$ by the previous edge e and to the right of $\overrightarrow{uv}$ by the new edge e (which may be the same as the previous e, if the last event was of type 2). It follows (by Lemma 2.13 and Lemma 2.1) that $\widetilde{R}_i$ is empty, so we return null.

Clearly, we will want to grow D_s in a finite number of discrete steps. At each iteration, it will suffice to construct two candidates for the next C_s, both tangent to u and v: a *type 1* candidate, which has outside-tangent edge e; and a *type 2* candidate, which is tangent an additional vertex w of D_i. We replace C_s with the candidate whose origin is closest to the current s. Next, we update e to be the nearest edge of $\mathbb{E}_i$ to the new location of s. We then determine which of the three events applies to the new C_s, and take the appropriate action.

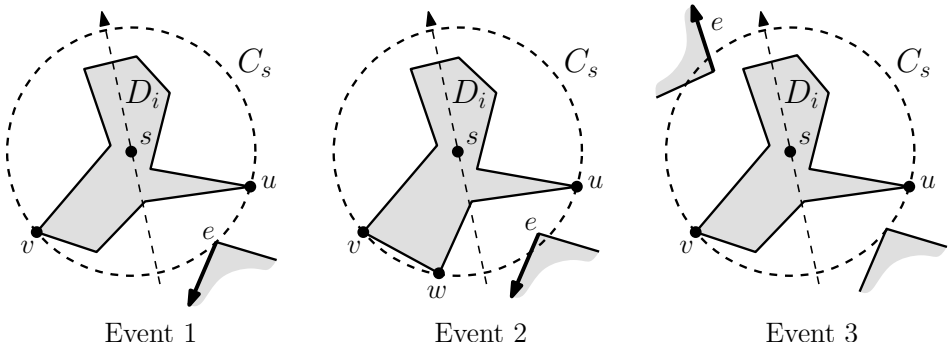

Figure 2.14: Events that can occur while growing C_s.

Let us examine the running time of an iteration of the above procedure. At the start of each iteration, we can determine e, the closest edge of $\mathbb{E}_i$ to s, by querying the nearest edge Voronoi diagram. This takes $O(\log k)$ time. We can determine which of the three events has occurred, and take the appropriate action, in constant time (note that an event of type 2 has occurred iff a type 2 candidate was chosen to replace C_s in the previous iteration). Constructing type 1 candidates can be done in constant time.

To construct a type 2 candidate, observe that s is moving along an edge f of the farthest point Voronoi diagram bordering the cells of u and v, and that the next (Voronoi) vertex encountered will be the center of the type 2 candidate. If we maintain a pointer to f, then the candidate and its associated vertex w can be determined in constant time.

The running time of each iteration is thus dominated by the time spent updating e, which can be done in $O(\log k)$ time. To bound the number of iterations, let us consider the sequence of discs C_s generated by this procedure. Observe that (i) each iteration sees a vertex of $\mathbb{V}_i$ or an edge of $\mathbb{E}_i$ becoming outside-tangent to the receding arc of some C_s, and (ii) the receding arc for a disc is a strict subset of that of the previous disc in the sequence. We can thus charge each iteration to a unique element of $\mathbb{V}_i$ or $\mathbb{E}_i$. It follows that there are at most k iterations, and we can make the following claim.

Lemma 2.14 *Given $\mathbb{E}_i$ and $\mathbb{V}_i$ of total complexity k, a boundary point of $\widetilde{R}_i$ can be found (or it can be determined that $\widetilde{R}_i$ is empty) in $O(k \log k)$ time.*

If we apply the above procedure and determine that no guaranteed disc for $\widetilde{R}_i$ exists, then $\widetilde{R}_i$ must be empty. Otherwise, we are ready to perform the second stage, which actually constructs the guaranteed cell for D_i. We will make use of a circular array data structure, one that provides the constant time operations `peek()` to return the current item, and `advance()`, to advance the cursor to the next item (or to the first item, if the current item was the last item). We perform these steps:

1. Determine the bisector $\langle v, e\rangle$ that contains p, the origin of the guaranteed disc C_p found earlier. Also, determine which edge of $\mathbb{E}_i$ contains $\delta(p)$ (the corresponding boundary point of R_i; see Section 2.2.2), by finding the disc $C_{\delta(p)}$ which is outside-tangent (at the same point) the same edge of $\mathbb{E}_i$ as C_p.

2. Construct (i) circular array $A_{\mathbb{V}}$, consisting of the ordered vertices $\mathbb{V}_i$, with its cursor initially pointing to the vertex following v; and (ii) circular array $A_{\mathbb{E}}$, consisting of the ordered edges $\mathbb{E}_i$,[9] with its cursor initially pointing to the edge following that which contains $\delta(p)$.

3. Construct candidate vertices $\langle v, v', e\rangle$ and $\langle v, e, e'\rangle$, as follows. Let $v' = A_{\mathbb{V}}.\texttt{peek}()$ and $e' = A_{\mathbb{E}}.\texttt{peek}()$. If $v = v'$ (resp., $e = e'$), advance $A_{\mathbb{V}}$ (resp., $A_{\mathbb{E}}$), and repeat from step 3. Otherwise, construct the discs $\langle v, v', e\rangle$ and $\langle v, e, e'\rangle$. Let $C_{p'}$ be the disc of these two whose origin represents the shorter distance forward from p along $\langle v, e\rangle$.

4. If $C_{p'}$ does not contain D_i, then advance $A_{\mathbb{V}}$, and repeat from step 3.

5. If $C_{p'}$ is penetrated by any polygon $D_{\neq i} \in \mathcal{D}$, then advance $A_{\mathbb{E}}$ and repeat from step 3.

6. If p' was already output as the first vertex, then stop. Otherwise, output p', replace p with p', replace v or e with the new vertex or edge that produced $C_{p'}$, and repeat from step 3.

Let us now examine the above steps in detail, to prove that the procedure generates $\widetilde{R}_i$. Point p lies on bisector $\langle v, e\rangle$, and serves as our starting point for generating the vertices of $\widetilde{R}_i$.

[9] Consecutive occurrences of the same edge can be omitted.

By Lemma 2.10, the next vertex of $\widetilde{R}_i$ is either of the form $\langle v, v', e\rangle$ or $\langle v, e, e'\rangle$. In the former case, Lemma 2.11 implies that v' is either the next element of $\mathbb{V}_i$, or appears after some number of elements of $\mathbb{V}_i$ that can be discarded. In the latter case, we can use an argument analogous to that of the proof of Lemma 2.8 to show that if e' is not the next element of $\mathbb{E}_i$, then it must appear after one or more elements of $\mathbb{E}_i$ (which can similarly be discarded).

Once we have chosen a candidate next vertex p' in step 3, we verify that its associated guaranteed disc $C_{p'}$ contains D_i. If not, then some vertex w on the hull of D_i lies outside of $C_{p'}$. Observe that (i) w must have appeared on the receding arc of some disc C_q that is tangent to v, w, and e; and (ii) v' must lie within C_q and (since v' is the next vertex on the convex hull of D_i) to the right of $\overline{vw}$ (Figure 2.15). Note also that every point in $\widetilde{R}_i$ must be the center of a disc that is not penetrated by e, but contains both v and w. This implies that v' cannot lie on the boundary of such a disc. Hence, we can discard v' (by advancing $A_{\mathbb{V}}$), and repeat from step 3.

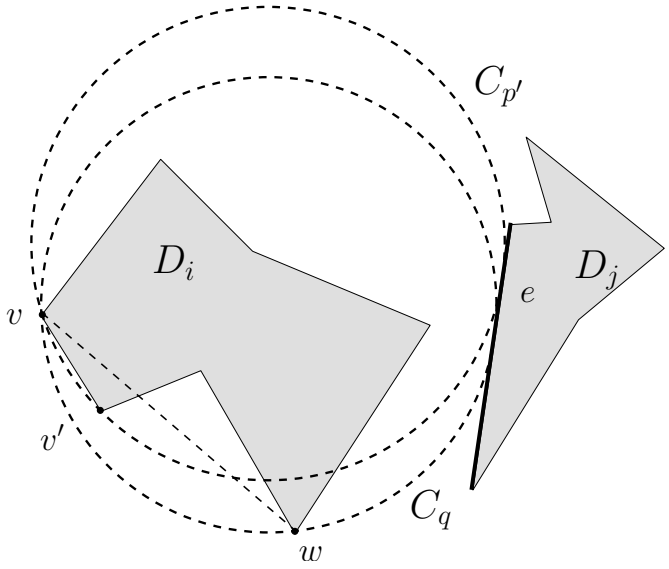

Figure 2.15: v' cannot contribute to $\widetilde{R}_i$ ($C_{p'}$ equals $\langle v, v', e\rangle$ in this example).

We next verify that no other polygon penetrates $C_{p'}$ in step 5. If some penetrating edge f exists, then p' cannot lie in $\widetilde{R}_i$. It follows that there exists a point q

between p and p' on $\langle v, e\rangle$ such that f lies on the advancing arc of C_q. This implies that $\langle v, e, f\rangle$ is the next vertex of $\widetilde{R}_i$. Hence, by Lemma 2.6, we can discard e' (by advancing $A_{\mathbb{E}}$) and repeat from step 3 (actually, we can safely advance $A_{\mathbb{E}}$ until f appears).

By the time we reach step 6, we have established that $C_{p'}$ is a guaranteed disc for $\widetilde{R}_i$, which implies that p' is a vertex of $\widetilde{R}_i$. Observe that the next vertex of $\widetilde{R}_i$ cannot be $\langle v, w, e\rangle$ where w is a vertex of $\mathbb{V}_i$ other than v', since each such w must appear after v but before v' on the convex hull of D_i (by Lemma 2.11), and will thus have been eliminated as a possibility by some execution of step 4. Similar reasoning can be applied to show that the next vertex of $\widetilde{R}_i$ cannot be $\langle v, e, f\rangle$ where f is an edge of $\mathbb{E}_i$ other than e', for each such edge f will have been eliminated in step 5. Hence, p' must in fact be the next vertex of $\widetilde{R}_i$. If p' was already output as the first vertex, then we have generated the entire boundary of $\widetilde{R}_i$, and we are done.

Now that we have shown that the algorithm correctly generates $\widetilde{V}(\mathcal{D})$, we will determine its running time. In order to convert standard Voronoi cells to their guaranteed counterparts, we must construct the standard Voronoi diagram of the polygons. This can be done in $O(n \log n)$ time (e.g., [26, 38, 76]). The first stage, finding an initial guaranteed disc for $\widetilde{R}_i$, can be done in $O(k \log k)$ time, by Lemma 2.14. Note that the algorithm we gave as proof of this lemma constructs two Voronoi diagrams: the farthest vertex of $\mathbb{V}_i$, and the nearest edge of $\mathbb{E}_i$. We will make use of these diagrams below.

Steps 1 and 2 run in $O(k)$ time, and step 3 runs in constant time. We can test if $C_{p'}$ contains D_i by querying the farthest vertex Voronoi diagram to verify that v is the farthest point of D_i from p'. Hence, step 4 runs in $O(\log k)$ time.

In step 5, we need to determine if any $D_{\neq i}$ penetrates $C_{p'}$. We can do this by querying the nearest edge Voronoi diagram to see if the nearest edge of $\mathbb{E}_i$ penetrates $C_{p'}$. If not, then by Lemma 2.12, no polygon of $\mathcal{D} \setminus \{D_i\}$ penetrates $C_{\cdot}$ Step 5 thus runs in $O(\log k)$ time.

Since step 6 runs in constant time, a complete iteration of steps 3 through 6 runs in $O(\log k)$ time. Note that each such iteration either (i) eliminates a vertex of $A_{\mathbb{V}}$, (ii) eliminates an edge of $A_{\mathbb{E}}$, or (iii) outputs a unique vertex of $\widetilde{R}_i$. Hence, at most $O(k)$ iterations are performed, for a total running time of $O(k \log k)$ per cell.

Since the total of k for all the guaranteed cells is $O(n)$, we can claim that:

Theorem 2.15 *The guaranteed Voronoi diagram of disjoint uncertain polygons with a total of n vertices has $O(n)$ complexity, and can be constructed in $O(n \log n)$ time.*

2.3 Possible Voronoi Diagram

One of the significant differences between guaranteed Voronoi diagrams and their standard counterparts is the presence of a neutral zone: areas of the plane for which no guaranteed closest site can be determined. In this section, we introduce a Voronoi diagram of imprecise points which has no neutral zone, and thus can make a guarantee about every point in the plane. We achieve this by modifying the guarantee. In particular, this variant determines, for each point p, the set of *possibly closest sites*: the smallest set of sites such that one is guaranteed to be a closest site to p (throughout this section, we will assume that the uncertain regions are discs). We call the resulting partition of the plane, which we denote by $\widetilde{V}^{\{\}}(\mathcal{D})$, a *possible Voronoi diagram* (Figure 2.16). In contrast to guaranteed Voronoi diagrams, these diagrams have cells that are not necessarily connected. In Figure 2.16, for instance, the two shaded regions belong to the same cell.

The possible Voronoi diagram can also be defined by generalizing equation (2.1):

$$\widetilde{R}_S = \bigcup_{i \in S} \Big[\bigcap_{j \notin S} H(i,j) \Big] - \bigcup_{S' \subset S} \widetilde{R}_{S'}$$

where $\widetilde{R}_\emptyset = \emptyset$. Note that only the closure of this definition of $\widetilde{R}_S$ is strictly equivalent to the first definition.

The following construct will prove useful in generating $\widetilde{V}^{\{\}}(\mathcal{D})$. The *possible cell* for a site i, denoted $\widetilde{P}_i$, is the set of all points for which it is possible that site i is the closest site. Formally,

$$\widetilde{P}_i = \{p \mid \exists q_1 \in D_1, q_2 \in D_2, \ldots, q_n \in D_n \; \forall j \; d(p,q_i) \leq d(p,q_j)\} \; .$$

The possible cell for one particular site is shown in Figure 2.17.

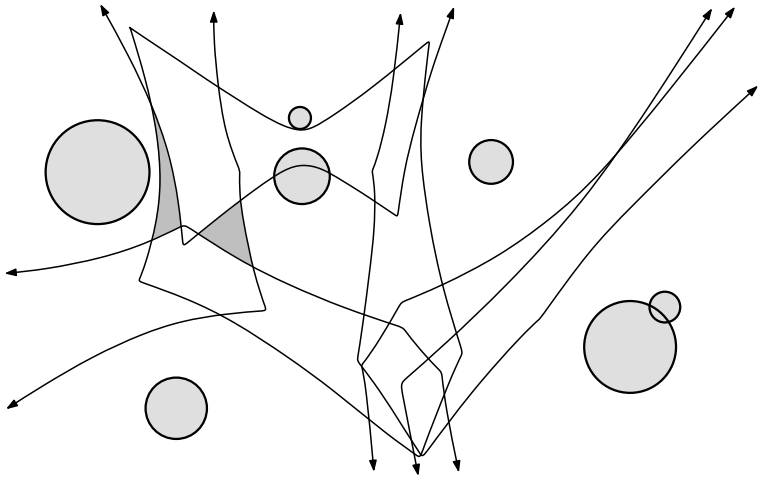

Figure 2.16: Possible Voronoi diagram.

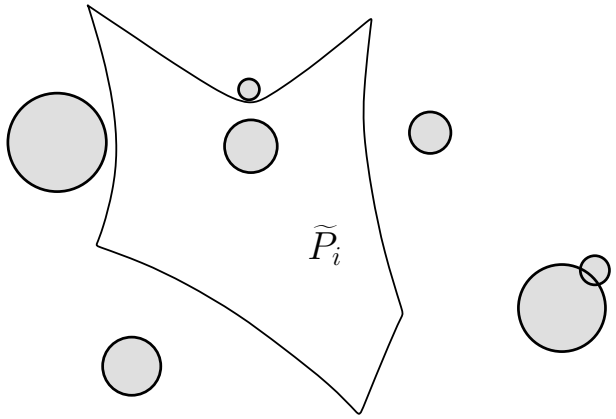

Figure 2.17: The possible cell for a site.

The possible cells for a set of sites do not generally have disjoint interiors, unless each uncertain region is a single point, in which case the possible cells are just standard Voronoi cells for point sites (and each site's location is no longer uncertain).

Lemma 2.16 *Point p is within $\widetilde{P}_i$ iff there exists a disc C_p centered at p that intersects D_i, and no $D_{\neq i} \in \mathcal{D}$ is strictly interior to the disc. In addition, p is on the boundary of $\widetilde{P}_i$ iff C_p has outside-tangent D_i and inside-tangent some $D_{\neq i}$.*

Observe that the possible Voronoi diagram associates each point in the plane with the subset of sites whose possible cells contain the point. Hence, to construct the possible Voronoi diagram, it suffices to construct the arrangement of the possible cell boundaries.

To do this efficiently, we will make use of the additively weighted Voronoi diagram of a set of point sites derived from the set of discs. We associate with each disc D_i a weighted site $(o_i, -r_i)$. We denote the resulting Voronoi diagram by $V^m(\mathcal{D})$, and the cell for $(o_i, -r_i)$ by R_i^m. The bisector between sites i and j is

$$\langle\langle i, j \rangle\rangle = \{p \mid d(p, o_i) + r_i = d(p, o_j) + r_j\}$$

To gain some intuition concerning this type of Voronoi diagram, note that it is essentially a standard Voronoi diagram of discs, except that the distance from a point to a disc is defined to be the distance from the point to the disc's *most distant* boundary point.

Let us replace a particular site i's weight with $+r_i$. Any point p on edge $\langle\langle i, j \rangle\rangle$ bordering cell i now satisfies

$$d(p, o_i) - r_i = d(p, o_j) + r_j \ .$$

By Lemma 2.16, this implies that the set of all edges bordering the new cell i forms the boundary of $\widetilde{P}_i$. In addition, since the new diagram is still an additively weighted Voronoi diagram, each possible cell has linear complexity.

Observe that the possible cell is produced by modifying a single site's additive weight in such a way that the possible cell is a superset of the corresponding cell of $V^m(\mathcal{D})$. It follows that the possible cell's vertices must lie on existing edges of $V^m(\mathcal{D})$; see Figure 2.18.

This suggests an approach to generating the possible cells: we construct the additively weighted Voronoi diagram of sites $\{(o_i, -r_i) \mid i = 1 \ldots n\}$, then for each disc D_i in turn, we:

1. insert a new site $(o_i, +r_i)$,
2. extract $\widetilde{P}_i$ from this new site's cell boundary, and

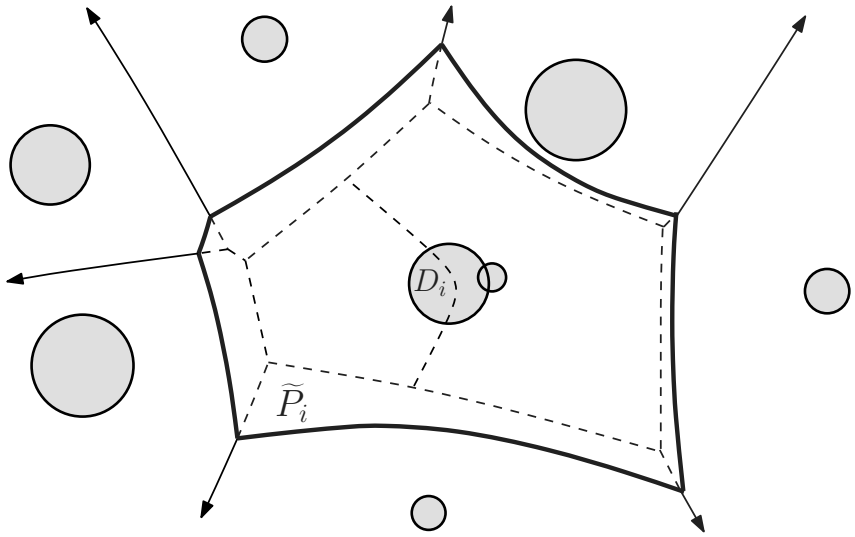

Figure 2.18: Possible cell vertices lie on edges of $V^m(\mathcal{D})$.

3. delete $(o_i, +r_i)$.

As we will see, we will not actually perform this insertion and deletion of sites, and can instead extract each $\widetilde{P}_i$ by performing a search along edges of $V^m(\mathcal{D})$.

Before describing the algorithm in more detail, we need to investigate further the relationship between intersecting discs, cells of $V^m(\mathcal{D})$, and possible regions.

Lemma 2.17 *If D_i intersects D_j, then $R_j^m \subseteq \widetilde{P}_i$.*

Proof. If D_i intersects D_j, then for any point $p \in R_j^m$, we have, for all D_k,

$$\begin{aligned} d(p, o_k) + r_k &\geq d(p, o_j) + r_j \\ &\geq d(p, o_i) - d(o_i, o_j) + r_j \ \text{(triangle inequality)} \\ &\geq d(p, o_i) - (r_i + r_j) + r_j \\ &= d(p, o_i) - r_i \ , \end{aligned}$$

which implies that $p \in \widetilde{P}_i$. □

Observe that Lemma 2.17 implies that for every $D_i \in \mathcal{D}$, $R_i^m \subseteq \widetilde{P}_i$.

Lemma 2.18 *If D_i contains D_j, then R_i^m is empty.*

Proof. If D_i contains D_j, then the most distant boundary point of D_i from any point p will be farther from p than the most distant boundary point of D_j. □

Lemma 2.19 *If D_i and D_j are nonintersecting discs, and R_j^m is nonempty, then $R_j^m \not\subseteq \widetilde{P}_i$.*

Proof. Suppose R_j^m is nonempty. Lemma 2.18 implies that o_j is closer to the most distant point of D_j than it is to the most distant point of any $D_{\neq j}$; hence $o_j \in R_j^m$. Since D_i does not intersect D_j, o_j is closer to the most distant point of D_j than it is to any point of D_i; hence $o_j \notin \widetilde{P}_i$. □

Our first step is to generate $V^m(\mathcal{D})$. For convenience, we embed it within a large circle, to eliminate unbounded edges. For ease of exposition, we will assume that every vertex has degree three[10]. We will also assume that the cell containing each disc's origin is known.

Some terminology will be helpful. Let $V_i^- = V^m(\mathcal{D}) \cap \widetilde{P}_i$. It has been shown ([41], Lemma 1) that this skeleton (e.g., the dotted lines of Figure 2.18) is connected. The *core discs* of D_i are the set of discs that include D_i and any discs intersecting D_i. The *core* of D_i is the union of cells of D_i's core discs.

For each edge e of $V^m(\mathcal{D})$, we determine if the possible cell for the site D_i to its left needs constructing. If so, by Lemma 2.17, e is within V_i^-, and every vertex of $\widetilde{P}_i$ can be found by performing, in a manner similar to that of ([35], Section 3.2), a depth first search (DFS) of V_i^- starting from e. If the DFS is done in a systematic fashion, the vertices of $\widetilde{P}_i$ can be found in sorted order.

As was the case in Section 2.2.3, a complication arises if the discs are not partially disjoint. If D_i is a parent of D_j, then by Lemma 2.18, R_i^m will be empty; yet by Lemma 2.17, $\widetilde{P}_i$ has nonzero area. To deal with this problem, we define $\mathcal{D}^m$ to be the set of discs $\mathcal{D}$ where each disc's radius r_i has been replaced by $r^m - r_i$, r^m being the radius of the largest disc of $\mathcal{D}$. We will denote the modified version of each D_i by D_i^m. It is easy to show that $V^m(\mathcal{D})$ is equivalent to the standard Voronoi diagram of the discs $\mathcal{D}^m$, and that the discs of $\mathcal{D}$ that do not contain other

[10] This assumption can be relaxed if a perturbation scheme (e.g., [35]) is employed.

discs (i.e., the 'innermost' discs) are exactly the outer discs of $\mathcal{D}^m$ (and the discs that contain these discs correspond to children within $\mathcal{D}^m$).

Let us return to the situation where D_i is a parent of D_j, and D_j itself does not contain any other discs of $\mathcal{D}$. It follows that D_j^m is an outer disc of $\mathcal{D}^m$, and D_i^m is one of its children. For each such child, we can (by Lemma 2.17) construct $\widetilde{P}_i$ by starting the DFS on any edge of the boundary of R_j^m. As noted in Section 2.2.3, by using the algorithm of [26] to construct $V^m(\mathcal{D})$, we can determine the outer discs, and their corresponding sets of children, at no additional cost.

Theorem 2.20 *The possible Voronoi diagram of n uncertain discs can be constructed in time $O((n+k)\log n + \sum_{i=1}^{n} g(i))$, where k is the complexity of the diagram, and $g(i)$ is the number of discs intersecting disc D_i.*

Proof. We first use the procedure described above to construct $V^m(\mathcal{D})$ and to generate the possible regions. Next, we perform a plane sweep to construct the arrangement of these regions. We need only prove the running time.

$V^m(\mathcal{D})$ can be constructed in $O(n \log n)$ time. For each D_i, the DFS covers every edge in V_i^-. We partition these edges into three sets: non-core edges, core boundary edges, and interior core edges. Note that non-core edges do not contain a cycle, else $\widetilde{P}_i$ would contain a non-core cell, contradicting Lemma 2.19. These edges thus form a set of 3-ary trees, each rooted at core boundary vertex, with vertices of $\widetilde{P}_i$ as leaves; hence the number of non-core edges is at most $|\widetilde{P}_i|$, the complexity of $\widetilde{P}_i$. For each core boundary vertex, there is a core boundary edge, so these number at most $|\widetilde{P}_i|$. Since the interior core edges are a subset of a Voronoi diagram of the core discs, there are $O(g(i))$ of them. The total edges examined in the DFS is thus $O(|\widetilde{P}_i| + g(i))$.

Since $\sum_{i=1}^{n} |\widetilde{P}_i| = O(k)$, generating all n possible cells can be done in time $O(n \log n + k + \sum_{i=1}^{n} g(i))$. The arrangement of the cells can then be generated, using a plane sweep, in time $O(k \log n)$, for the stated total running time. □

Theorem 2.21 *The number of edges in a possible Voronoi diagram of n uncertain discs is $O(n^3)$, and this bound is tight.*

Proof. As noted earlier, $\widetilde{V}^{\{\}}(\mathcal{D})$ is the arrangement of boundaries of possible cells $\widetilde{P}_{i \in 1 \ldots n}$. Its complexity is thus (i) the sum of the complexities of the individual

possible cells, plus (ii) the number of proper intersection points of edges from pairs of cells $(\widetilde{P}_i, \widetilde{P}_j)$. Since each possible cell is a cell of an additively weighted Voronoi diagram of n sites (which has $O(n)$ complexity [66]), (i) is $n \cdot O(n)$. We can bound (ii) by noting that each point p that lies on the boundaries of $\widetilde{P}_i$ and $\widetilde{P}_j$ is the center of two discs: C_i, which has outside-tangent D_i and inside-tangent some D_k, and C_j, which has outside-tangent D_j and inside-tangent D_m. Since no discs of $\mathcal{D}$ can lie in the interior of C_i or C_j, it follows that $C_i = C_j$. We can assume, without loss of generality, that $r_j > 0$; for if $r_i = r_j = 0$, $\widetilde{P}_i$ cannot properly intersect $\widetilde{P}_j$. This implies that $j \neq k$, since D_j cannot be both inside- and outside-tangent C_i. Each point p is therefore determined by an ordered triple of distinct discs (D_i, D_j, D_k). The total complexity of $\widetilde{V}^{\{\}}(\mathcal{D})$ is thus $O(n \cdot O(n) + \binom{n}{3}) = O(n^3)$.

We can show that this bound is tight; see Figure 2.19. We place two sets of $n/3$ discs of large radii r clustered tightly around the points $(-r - \epsilon, 0)$ and $(r + \epsilon, 0)$, for small positive ϵ. We place an additional $n/3$ discs with zero radii in a stack near the origin. For each large disc D_i, this stack induces $\Omega(n)$ arcs in the boundary of $\widetilde{P}_i$; and for each pair of large discs (D_i, D_j) on opposite sides of the x-axis, the boundaries of $\widetilde{P}_i$ and $\widetilde{P}_j$ will intersect in $\Omega(n)$ points. The total number of intersections in $\widetilde{V}^{\{\}}(\mathcal{D})$ is thus $(n/3)^2 \cdot \Omega(n) = \Omega(n^3)$. □

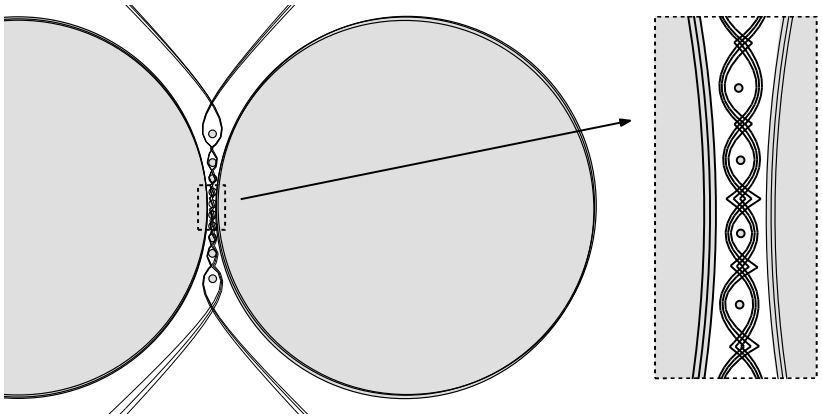

Figure 2.19: $\widetilde{V}^{\{\}}(\mathcal{D})$ complexity for discs is $\Omega(n^3)$ (some edges omitted for clarity).

2.4 Guaranteed Delaunay Edges

In this section, we consider the Delaunay triangulation of a set of sites, where each site lies within a region of uncertainty. Our goal will be to find an algorithm to determine the guaranteed Delaunay edges for these uncertain regions. Such an algorithm would be useful, since it would allow us to preprocess a set of these regions to determine a set of Delaunay edges that is a subgraph of every feasible Delaunay triangulation of the regions. Having this subgraph pre-computed could save a significant amount of the work required to (later) construct a Delaunay triangulation of a feasible set of the regions.

Earlier, we defined a guaranteed Delaunay edge (a, b) to exist iff (a, b) is an edge in the Delaunay triangulation of every feasible set $S = \{s_1 \in D_1, \ldots, s_n \in D_n\}$ of $\mathcal{D}$. This definition is not, however, as precise as we would like. If four sites of a set are cocircular, then their respective Voronoi cells will meet at a vertex of degree four. It is not immediately clear whether the cells of two sites that have only this vertex in common are considered to share an edge; what is more, the Delaunay triangulation of such a set is not unique (Figure 2.20).

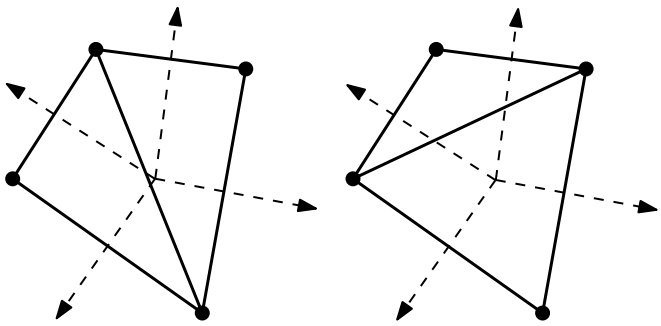

Figure 2.20: Cocircular points with two valid Delaunay triangulations (Voronoi diagram drawn with dotted lines).

To deal with this problem, we will require guaranteed Delaunay edges to correspond to Voronoi edges of nonzero length. Formally, we will say that (a, b) is a guaranteed Delaunay edge of $\mathcal{D}$ iff for every feasible set S of $\mathcal{D}$, the Voronoi cells for s_a and s_b share an edge of nonzero length.

Our approach in determining the guaranteed Delaunay edges for a set of uncertain regions $\mathcal{D}$ will be to start with a set of pairs of these regions, then examine each

candidate pair (D_a, D_b) to see if it corresponds to a guaranteed Delaunay edge. We will let $\overline{\mathcal{D}}$ denote the set $\bigcup\big(D_i \in (\mathcal{D} \setminus \{D_a, D_b\})\big)$.

Lemma 2.22 *If $\overline{\mathcal{D}}$ intersects the convex hull of D_a and D_b, then (a, b) is not a guaranteed Delaunay edge.*

Proof. Suppose point $q \in \overline{\mathcal{D}} \cap \text{CH}(D_a \cup D_b)$ exists. First observe that as a consequence of the regions' valid intersection property, there must exist some such q that does not lie on the boundary of either D_a or D_b. Now note that there exist points $s_a \in D_a$ and $s_b \in D_b$ such that q lies in the interior of $\overline{s_a s_b}$ (see Figure 2.21). This implies that there exists a feasible set of $\mathcal{D}$ in which q prevents s_a and s_b from

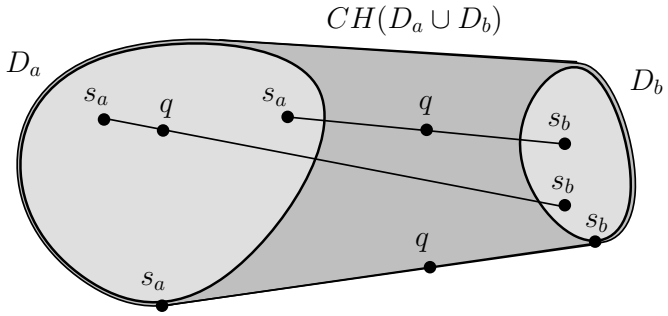

Figure 2.21: Lemma 2.22.

being neighbors in the Voronoi diagram of the set. Hence, (a, b) is not a guaranteed Delaunay edge. □

If D_a and D_b are regions of $\mathcal{D}$, then we define a *guaranteed Delaunay disc* for (a, b) to be a disc that contains D_a and D_b, and does not intersect $\overline{\mathcal{D}}$.

Lemma 2.23 *If each region of $\mathcal{D}$ is convex, then (a, b) is a guaranteed Delaunay edge if and only if a guaranteed Delaunay disc exists for (a, b).*

Proof. If C_p is a guaranteed Delaunay disc for (a, b), then for every feasible set S of $\mathcal{D}$, s_a and s_b lie within C_p, and C_p does not contain any other points of the set. We can shrink C_p until (without loss of generality) s_a is on its boundary, then shrink it again, with respect to s_a, to reach a disc $C_{p'}$ that also has s_b on its

boundary. Hence, p' lies on the Voronoi edge bordering cells of s_a and s_b; and since $C_{p'} \subseteq C_p$, no other sites of S intersect $C_{p'}$, which implies that this Voronoi edge has nonzero length. As this holds for every feasible set S, (a, b) is by definition a guaranteed Delaunay edge.

Now suppose (a, b) is a guaranteed Delaunay edge. For ease of exposition, we will assume that the uncertainty regions are discs, though the following proof can easily be adapted for any convex uncertainty regions.

If D_a and D_b are not partially disjoint, then without loss of generality, D_a contains D_b. By Lemma 2.22, $\overline{\mathcal{D}}$ does not intersect D_a; hence D_a is a guaranteed Delaunay disc for (a, b).

Suppose instead that D_a and D_b are partially disjoint. Consider the sequence of discs F that have inside-tangent both D_a and D_b, ordered by their origin's position along $\langle\langle a, b\rangle\rangle$. We now prove that some element of F does not intersect $\overline{\mathcal{D}}$, and hence is a guaranteed Delaunay disc for (a, b).

We start with an arbitrary disc $C_p \in F$. Let α (resp., β) be the tangent point of C_p with D_a (resp., D_b), and let R (resp., L) be the portion of C_p that lies strictly to the right (resp., left) of directed segment $\overline{\alpha\beta}$ (Figure 2.22).

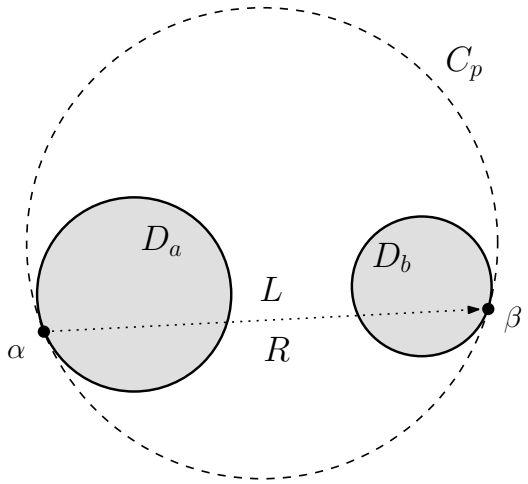

Figure 2.22: Lemma 2.23.

If $\overline{\mathcal{D}}$ intersects neither R nor L, then we are done (by Lemma 2.22, $\overline{\mathcal{D}}$ cannot intersect $\overline{\alpha\beta}$). Otherwise, assume by way of contradiction that $\overline{\mathcal{D}}$ intersects both R

and L. Since every element of $\overline{\mathcal{D}}$ is convex, the same $D_k \in \overline{\mathcal{D}}$ cannot intersect both R and L (without intersecting $\overline{\alpha\beta}$); hence, distinct discs D_i and D_j of $\overline{\mathcal{D}}$ intersect R and L respectively. It follows that there exists a feasible set S where $s_a = \alpha$, $s_b = \beta$, $s_i \in R$, and $s_j \in L$. We can apply Lemma 2.13 to this set to show that the Voronoi edge for (a, b) either consists of a single point (if $\overline{\mathcal{D}}$ does not penetrate C_p), or does not exist (otherwise). In either case, (a, b) cannot be a guaranteed Delaunay edge, a contradiction.

We can then assume, without loss of generality, that only R intersects $\overline{\mathcal{D}}$. Let us now replace C_p with its neighbor within F whose origin lies to the left of $\overline{\alpha\beta}$, and repeat the above procedure. This process must terminate, since at some point before C_p becomes infinitely large (and $R = \emptyset$), neither R nor L will intersect $\overline{\mathcal{D}}$. □

Interestingly, Lemma 2.23 does not hold for non-convex uncertainty regions. For example, for the regions shown in Figure 2.23, no guaranteed Delaunay disc exists for (a, b); yet for every feasible set of the regions, the Voronoi cells of D_a and D_b will share a nonzero-length edge, which implies that (a, b) is a guaranteed Delaunay edge.

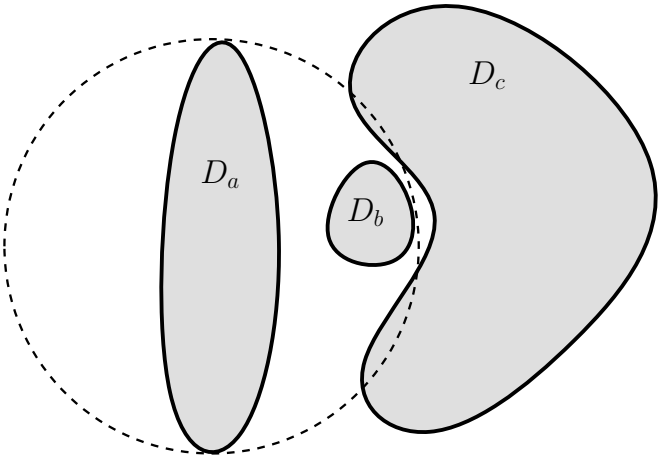

Figure 2.23: Lemma 2.23 does not apply for nonconvex regions.

From this point on, we will assume that $\mathcal{D}$ is a set of uncertain discs. We will make use of $V^2(\mathcal{D})$, the standard order-2 Voronoi diagram of the discs. Recall that

we denote the cell of this diagram corresponding to a disc pair (D_i, D_j) by $R_{\{i,j\}}$.

The *spine* of a pair of uncertain discs (D_a, D_b) (denoted $sp(a, b)$) is the set of origins of the guaranteed Delaunay discs for (a, b) that have inside-tangent both D_a and D_b. Spines will be fundamental to our algorithm. We can identify some of their properties:

Lemma 2.24 *If (D_a, D_b) is a partially disjoint pair of uncertain discs, then (a, b) is a guaranteed Delaunay edge if and only if $sp(a, b)$ is nonempty.*

Proof. Assume (D_a, D_b) is a partially disjoint pair as stated. If there exists a point $p \in sp(a, b)$, then p is the center of a guaranteed Delunay disc for (a, b); hence, by Lemma 2.23, (a, b) is a guaranteed Delaunay edge.

If (a, b) is a guaranteed Delaunay edge, then there exists some guaranteed Delaunay disc C_p for (a, b). We can shrink C_p until it has inside-tangent one of D_a or D_b, then shrink it further with respect to that tangent point until it has inside-tangent both D_a and D_b. Since the final disc is within C_p, it must be a guaranteed Delaunay disc; hence, its origin lies on $sp(a, b)$. □

Observe that if the spine of a pair of uncertain discs is nonempty, the pair must be partially disjoint.

Lemma 2.25 *If (D_a, D_b) are partially disjoint discs from $\mathcal{D}$, and $sp(a, b)$ is nonempty, then $R_{\{a,b\}}$ is a nonempty cell within $V^2(\mathcal{D})$.*

Proof. Let p be a point on $sp(a, b)$. The disc centered at p that has inside-tangent both D_a and D_b does not intersect $\overline{\mathcal{D}}$; hence, p is strictly closer to both D_a and D_b than to any point within $\overline{\mathcal{D}}$, which implies that p is in the interior of $R_{\{a,b\}}$. □

Lemma 2.26 *$sp(a, b)$ is a connected subset of a hyperbolic arc.*

Proof. Every point $p \in sp(a, b)$ satisfies

$$d(p, o_a) + r_a = d(p, o_b) + r_b \,,$$

so p lies on a hyperbolic arc. We now prove that the spine is connected. Place the axes so o_a and o_b are on the x-axis, with o_a to the left of o_b. Each point on the

spine now has a unique y-coordinate. Assume by way of contradiction that (i) u, v, and w are points on the hyperbolic arc containing $sp(a, b)$, (ii) $u_y < v_y < w_y$, and (iii) $u, w \in sp(a, b)$, and $v \notin sp(a, b)$.

Let discs C_u,C_v,C_w be the discs centered at u, v, w that have inside-tangent D_a and D_b. Since $v \notin sp(a, b)$, some point $q \in D_{k \notin \{a,b\}} \in \mathcal{D}$ exists that penetrates C_v. Let L be the line through the points of tangency of C_v with D_a and D_b. Observe that the two intersection points of the boundaries of C_u and C_v lie on or above L, while those of C_v and C_w lie on or below L; so if q lies on or above L, it penetrates C_w, and if it lies below L, it penetrates C_u. Thus either u or w is not in $sp(a, b)$; a contradiction. Hence $sp(a, b)$ is connected. □

We are ready to describe a procedure for constructing the guaranteed Delaunay edges of a set of n partially disjoint uncertain discs. We start by constructing $V^2(\mathcal{D})$, the order-2 Voronoi diagram of the discs. By Lemma 2.25, only those discs (D_a, D_b) which have a nonempty cell $R_{\{a,b\}}$ within this diagram can be guaranteed Delaunay edges. Let us examine how the spine corresponding to each region can be efficiently constructed.

If p is an endpoint (not at infinity) of $sp(a, b)$, there must exist a disc C_p that has inside-tangent both D_a and D_b, and has outside-tangent some disc D_k contributing to $\overline{\mathcal{D}}$. By the proof of Lemma 2.25, $p \in R_{\{a,b\}}$. We can shrink C_p with respect to D_k until it reaches disc C_q that intersects both D_a and D_b, and (without loss of generality) has outside-tangent D_a. Note that q lies on the boundaries of both $R_{\{a,b\}}$ and $R_{\{a,k\}}$. It follows that every such endpoint p of $sp(a, b)$ is induced by a disc associated with a neighboring cell of $R_{\{a,b\}}$, and we can construct the spine as follows.

We initialize the spine to be the complete hyperbolic arc $\langle\langle a, b\rangle\rangle$. The we clip the spine to the hyperbolic arcs containing $\langle\langle a, k\rangle\rangle$ (resp., $\langle\langle b, k\rangle\rangle$) for each neighboring cell $R_{\{a,k\}}$ (resp., $R_{\{b,k\}}$) of $R_{\{a,b\}}$. If the spine is nonempty after this process, then by Lemma 2.24, (a, b) is a guaranteed Delaunay edge.

$V^2(\mathcal{D})$ contains $O(n)$ edges, and can be generated in $O(n \log n)$ time [62]. The fact that the spine is connected (Lemma 2.26) implies that each clipping operation can be performed in constant time, and there are at most two such operations for each edge in $V^2(\mathcal{D})$; thus the running time is dominated by the $O(n \log n)$ time

spent constructing $V^2(\mathcal{D})$.

This procedure can be modified to handle discs that may not be partially disjoint. As described in Section 2.2.3, we can determine (in $O(n \log n)$ time) which discs of $\mathcal{D}$ are outer discs, and which are the parents of those that are not. By Lemma 2.23, if D_a is an outer disc with a child D_b, then the only guaranteed Delaunay edge involving either D_a or D_b is (a, b), and only if no third disc intersects D_a (note that this implies that D_b is the only child of D_a).

Lemma 2.27 *If $\mathcal{D}$ is a set of partially disjoint discs, and $D_a, D_b \in \mathcal{D}$ intersect, then D_a must be Voronoi neighbors with at least one disc of $\mathcal{D}$ that intersects D_a.*

Proof. If R_a and R_b are neighboring cells in the Voronoi diagram of $\mathcal{D}$, then we are done. Otherwise, let p be the intersection of $\overline{o_a o_b}$ with bisector $\langle\!\langle a, b \rangle\!\rangle$. We have

$$d(p, o_a) - r_a = d(p, o_b) - r_b.$$

Since D_a and D_b intersect,

$$d(p, o_a) + d(p, o_b) \leq r_a + r_b \, ,$$

and the two equations can be combined to show that $d(p, o_a) \leq r_a$, or in other words, $p \in D_a$. Now, some point q in the interior of $\overline{o_a p}$ must lie on bisector $\langle\!\langle a, c \rangle\!\rangle$, where D_c is a neighbor of D_a. Hence

$$\begin{aligned} d(q, o_c) - r_c = \; & d(q, o_a) - r_a \\ \leq \; & d(p, o_a) - r_a \\ \leq \; & r_a - r_a \, , \end{aligned}$$

which implies that $q \in D_c$. □

Our modified algorithm consists of these steps:

1. Construct $V(\mathcal{D})$, the (order-1) Voronoi diagram of $\mathcal{D}$. As a side effect, construct S, the subset of outer discs of $\mathcal{D}$. Observe that since only discs of S will have cells within $V(\mathcal{D})$, $V(\mathcal{D})$ equals $V(S)$.

2. Construct $V^2(S)$, the order-2 Voronoi diagram of S.

3. For each $D_a \in S$ with exactly one child D_b, examine the neighboring discs of D_a within $V(\mathcal{D})$ to see if any other discs intersect D_a. If none are found, then (by Lemma 2.27) D_a is a guaranteed Delaunay disc for (i, j); output (a, b) as a guaranteed Delaunay edge.

4. For each cell $R_{\{a,b\}}$ of $V^2(S)$, if either D_a or D_b has children, do nothing. Otherwise, construct $sp(a, b)$ as in the original algorithm, and output (a, b) as a guaranteed Delaunay edge if $sp(a, b) \neq \emptyset$.

Steps 1 and 2 can be performed in $O(n \log n)$ time ([26],[62]). Steps 3 and 4 involve linear traversals of the cells of the two Voronoi diagrams. Each cell is traversed at most once, hence the steps take $O(n)$ time. Since the total running time is the same as the original algorithm, we can claim:

Theorem 2.28 *The guaranteed Delaunay edges of n uncertain discs can be found in $O(n \log n)$ time.*

2.5 Guaranteed Gabriel Edges of Uncertain Discs

In this section, we present an algorithm to find the guaranteed Gabriel edges of a set of uncertain discs. As was the case for Delaunay triangulations, it is reasonable to suppose that constructing such a set as a preprocessing step could be useful for any application that required the Gabriel graph of a feasible set to be computed later.

2.5.1 Gabriel Discs and Zones

From this point on, we will assume that $\mathcal{D}$ is a set of uncertain discs. We define a *Gabriel disc* of a pair of discs $D_a, D_b \in \mathcal{D}$ to be a disc with diameter $\overline{\alpha\beta}$ where $\alpha \in D_a$ and $\beta \in D_b$. We denote this Gabriel disc by $G_{\alpha,\beta}$. The *Gabriel zone* of the pair is the union of all Gabriel discs of the pair, and is denoted $Z_{a,b}$. Formally, $Z_{a,b} = \bigcup_{\alpha \in D_a, \beta \in D_b} G_{\alpha,\beta}$. An example of a Gabriel zone is shown in Figure 2.24.

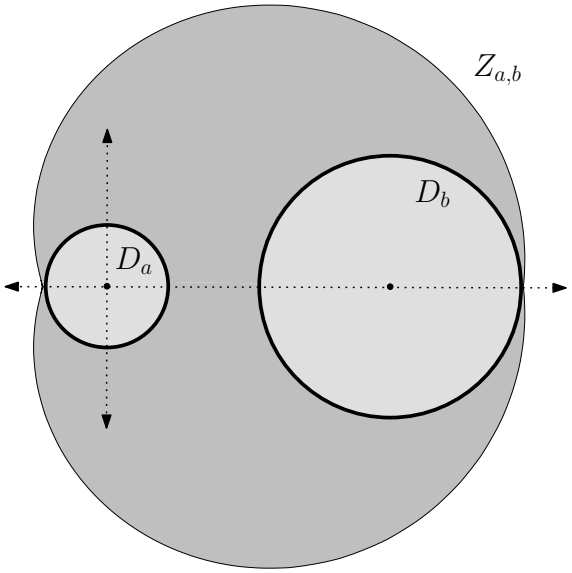

Figure 2.24: The Gabriel zone for a pair of uncertain discs.

Lemma 2.29 *(a, b) is a guaranteed Gabriel edge of $\mathcal{D}$ iff $Z_{a,b} \cap \overline{\mathcal{D}} = \emptyset$.*

Proof. If $Z_{a,b} \cap \overline{\mathcal{D}} = \emptyset$, then for every feasible set S of $\mathcal{D}$, Gabriel disc G_{s_a,s_b} will not intersect $\overline{\mathcal{D}}$; hence (a, b) is a guaranteed Gabriel edge.

If there exists a point $q \in Z_{a,b} \cap \overline{\mathcal{D}}$, then there exists some Gabriel disc $G_{\alpha,\beta}$ containing q. It follows that (a, b) is not a guaranteed Gabriel edge. □

Lemma 2.30 *The Gabriel zone of discs D_a and D_b contains the convex hull of the discs.*

Proof. Every point in $\text{CH}(D_a \cup D_b)$ lies on a segment $\overline{\alpha\beta}$ for some $\alpha \in D_a$, $\beta \in D_b$; and each such segment is the diameter of Gabriel disc $G_{\alpha,\beta}$. □

Lemma 2.31 *Every boundary point of $Z_{a,b}$ is the source for a pair of perpendicular rays defining a quarter plane, where D_a is tangent one ray, D_b is tangent the other, and the disc centers lie within the quarter plane.*

Proof. Let p be a boundary point of $Z_{a,b}$. Since $Z_{a,b}$ is a union of Gabriel discs, p must lie on the boundary of some Gabriel disc $G_{\alpha,\beta}$ where $\theta(\angle\alpha p\beta) = \frac{\pi}{2}$. Con-

sider the quarter plane bounded by the rays $\overrightarrow{p\alpha}$ and $\overrightarrow{p\beta}$. Suppose, by way of contradiction, that p does not satisfy the stated properties (i.e., one of the discs is not tangent to one of the rays, or its center is outside the quarter plane). Without loss of generality, we can then assume that the quarter plane lies to the left of $\overrightarrow{p\alpha}$, and D_a penetrates the right half-plane of $\overrightarrow{p\alpha}$ (Figure 2.25). Now, consider the line L

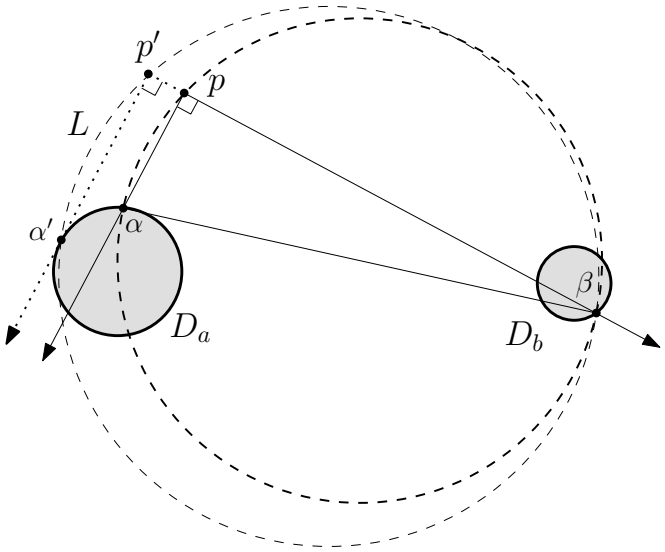

Figure 2.25: Lemma 2.31.

that is parallel to and to the right of $\overrightarrow{p\alpha}$, and tangent D_a at point α'. Let p' be the intersection of L with the line containing $\overrightarrow{p\beta}$. Observe that p' lies on the boundary of Gabriel disc $G_{\alpha',\beta}$, and p is strictly interior to this disc. Hence p cannot lie on the boundary of $Z_{a,b}$. □

We will now investigate the structure of Gabriel zones in more detail. It will simplify our analysis if we place the coordinate axes so that D_a is centered at the origin, and D_b is centered on the positive x-axis.

Let p be a point on the boundary of $Z_{a,b}$, and α, β be the points of tangency of the associated rays with the two discs. We will say that p is *left-oriented* (resp., *right-oriented*) if β lies to the left of (resp., right of) $\overrightarrow{p\alpha}$.

Lemma 2.32 *Segment $\overline{o_a o_b}$ lies in the kernel of $Z_{a,b}$.*

Proof. Let K be the kernel of $Z_{a,b}$. We need only show that $o_a \in K$, since a symmetric argument can be used to show that $o_b \in K$, and the convexity of K then implies that $\overline{o_a o_b} \subset K$.

Let p be any point on the boundary of $Z_{a,b}$. By Lemma 2.31, p lies on the boundary of a Gabriel disc with diameter $\overline{\alpha\beta}$, where $\overline{p\alpha}$ is tangent to D_a, and $\overline{p\beta}$ is tangent to D_b (Figure 2.26). Consider a point s moving from α toward o_a. For each s, we construct disc C^s, the circumcircle of points s, s', and β, where s' is the projection of s onto ray $\overrightarrow{p\beta}$. Since $\angle ss'\beta$ is a right angle, $\overline{s\beta}$ is a diameter of C^s; hence, C^s is a Gabriel disc of D_a and D_b. Now observe that $\overline{ss'}$ sweeps out a rectangle, one of whose diagonals is $\overline{po_a}$; hence, $\overline{po_a} \subset Z_{a,b}$. As this holds for any boundary point p of $Z_{a,b}$, $o_a \in K$. □

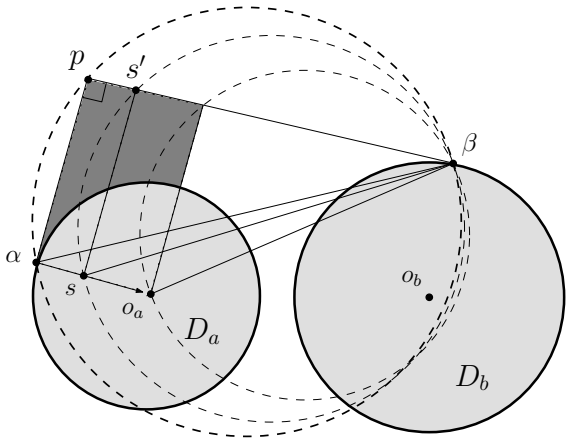

Figure 2.26: Lemma 2.32.

Lemma 2.33 *Every boundary point of $Z_{a,b}$ that lies to the left (resp., right) of $\overline{o_a o_b}$ is left-oriented (resp., right-oriented).*

Proof. We will first show that any points where the orientation of the boundary points change must lie on the x-axis (the line containing $\overline{o_a o_b}$). We will then show that this implies that there are exactly two such points. Finally, we will show that at least one point above (resp., below) the x-axis is left- (resp., right-) oriented. The proof then follows from the fact that the boundary is connected.

Let p be any *transition* point: a boundary point of $Z_{a,b}$ that marks a transition from left- to right-orientation. Now, there must exist rays $\overrightarrow{p\alpha}$ and $\overrightarrow{p\alpha'}$ tangent to D_a, and corresponding rays $\overrightarrow{p\beta}$ and $\overrightarrow{p\beta'}$ tangent to D_b, as shown in Figure 2.27. Note that $\theta(\angle\alpha p\beta')$ must equal $\theta(\angle\beta p\alpha')$. This implies that p is collinear with the

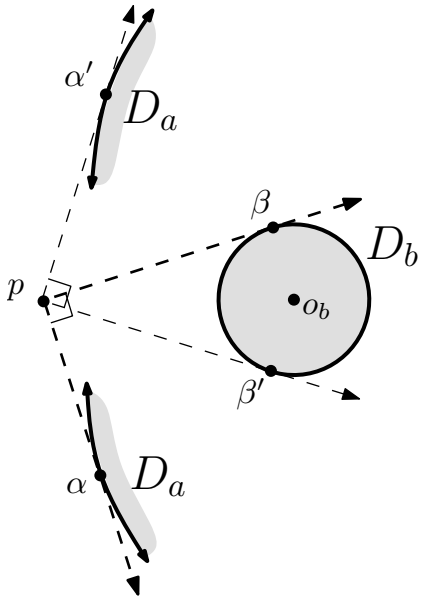

Figure 2.27: Lemma 2.33: there are exactly two transition points.

disc centers. Lemma 2.32 then implies that there are exactly two transition points, both lying on the x-axis.

Now consider the vertical line that is tangent to D_a at α and the horizontal line that is tangent to D_b at β depicted in Figure 2.28. Observe that the intersection of these lines, p, lies within $Z_{a,b}$. Suppose the vertical line contains a right-oriented boundary point p' of $Z_{a,b}$ that lies above α. Clearly, its tangency point with D_a must equal α. By Lemma 2.31, the disc centers must lie to the right of $\overrightarrow{p'\alpha}$; but this means p' lies below α, a contradiction. Hence, every boundary point of $Z_{a,b}$ on or above p (in fact, p is the only such point) must be left-oriented. A symmetric argument can be used to show that boundary points below the x-axis are right-oriented. □

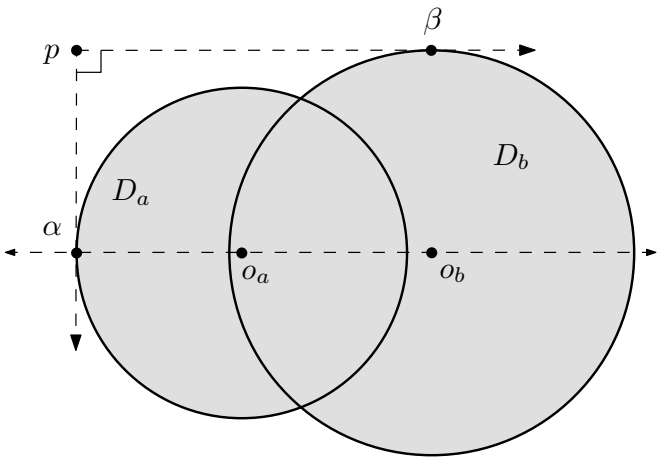

Figure 2.28: Lemma 2.33: boundary points above the x-axis are left-oriented.

We will now derive an expression for points on the boundary of $Z_{a,b}$. Since $Z_{a,b}$ is symmetric with respect to the x-axis, it will suffice to consider only *upper* boundary points: points above this axis.

Let q be the common endpoint of perpendicular segments whose other endpoints are o_a and o_b respectively, and let $\theta = \theta(\angle o_b o_a q)$. Observe that q is one corner of a rectangle with sides of length r_a and r_b. Lemma 2.33 implies that p, the opposite corner of this rectangle, is a boundary point of $Z_{a,b}$; see Figure 2.29.

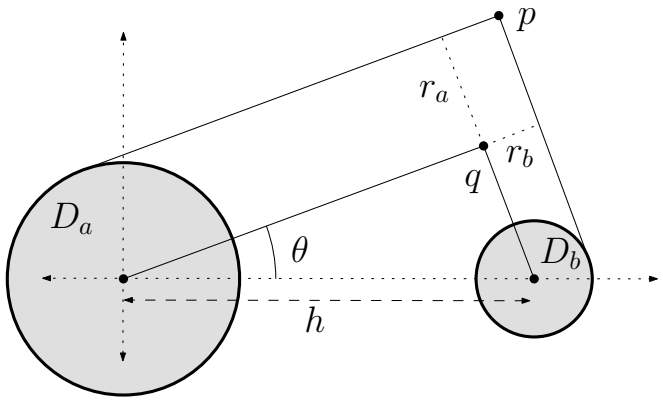

Figure 2.29: Deriving an expression for boundary points of $Z_{a,b}$.

Let h be the distance between o_a and o_b. Then

$$q_x = h\cos^2\theta$$
$$q_y = h\cos\theta\sin\theta\ ,$$

and

$$p_x = q_x + (r_b\cos\theta - r_a\sin\theta)$$
$$p_y = q_y + (r_b\sin\theta + r_a\cos\theta)\ ;$$

hence,

$$p_x = h\cos^2\theta - r_a\sin\theta + r_b\cos\theta$$
$$p_y = h\cos\theta\sin\theta + r_a\cos\theta + r_b\sin\theta\ .$$

Gabriel zone boundaries are essentially generalizations of a class of curves known as *pedal curves* [44]. The pedal of a curve C with respect to a *pedal point* s is the set of points that are the intersections of two perpendicular lines, where one line is tangent to C, and the other contains s. A *circle pedal curve* results if C is a disc boundary. Observe that if the radius of D_b is zero, then the (upper) boundary of the Gabriel zone for D_a and D_b is the circle pedal curve of the boundary of D_a with pedal point $s = D_b$. In the general case, $r_b > 0$, and each point on the Gabriel zone boundary is generated by a line tangent to D_a, and a second line that is tangent to D_b instead of intersecting a fixed point. We call this generalization of pedal curves a *bipedal curve*, and when the curves are circles, as in our case, we call it a *circle* bipedal curve.

To determine if some D_k contributing to $\overline{\mathcal{D}}$ intersects the Gabriel zone $Z_{a,b}$, we proceed as follows. First, we transform D_k to the coordinate system of Figure 2.29. If D_k's origin is below the x-axis, then we replace it with its reflection through this axis (as noted earlier, Gabriel zones are symmetric about this axis, so D_k will intersect the zone if and only if its reflection does as well). Next, we find p, the nearest upper boundary point to o_k, and calculate the directed line T tangent to the boundary at p (Figure 2.30). Finally, we return true if D_k intersects the left

half-plane of T.

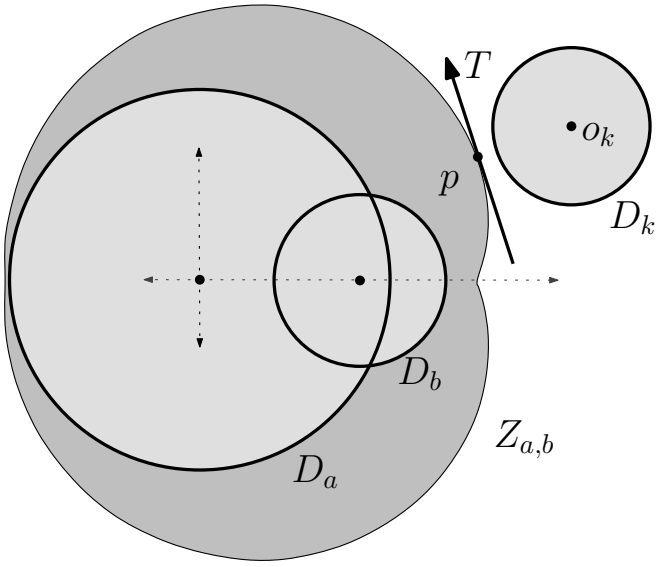

Figure 2.30: Determining if D_k intersects $Z_{a,b}$.

We will denote the coordinates of o_k by k_x and k_y respectively. To find p, we will find the parameter θ that minimizes the square of the distance of boundary point p from o_k, by setting the derivative of this distance to zero:

$$\frac{d}{d\theta}\big((p_x - k_x)^2 + (p_y - k_y)^2\big) = 0$$

After algebraic manipulation, this reduces to finding the zeros of

$$c_1 \cos\theta \sin\theta + c_2 \sin^2\theta + c_3 \cos\theta + c_4 \sin\theta + c_5$$

where the coefficients are the constants

$$\begin{aligned} c_1 &= h(2k_x - h) \\ c_2 &= 2hk_y \\ c_3 &= r_a k_x - r_b k_y \\ c_4 &= r_b(k_x - h) + r_a k_y \\ c_5 &= -hk_y \ . \end{aligned}$$

The direction of the line T tangent to a boundary point p is:

$$\begin{aligned} \frac{\partial p_x}{\partial \theta} &= -2h\cos\theta\sin\theta - r_a\cos\theta - r_b\sin\theta \\ \frac{\partial p_y}{\partial \theta} &= h(2\cos^2\theta - 1) - r_a\sin\theta + r_b\cos\theta \ , \end{aligned}$$

which simplifies to

$$\begin{aligned} \frac{\partial p_x}{\partial \theta} &= -h\cos\theta\sin\theta - p_y \\ \frac{\partial p_y}{\partial \theta} &= -h\sin^2\theta + p_x \ . \end{aligned}$$

Clearly, D_k will intersect the left half-plane of T iff the signed distance of o_k from T is not more than r_k.

2.5.2 Finding Guaranteed Gabriel Edges

Our algorithm for generating the guaranteed Gabriel edges of a set of discs will consist of two steps. In the first, we generate a set of candidate edges; and in the second, we construct the Gabriel zone for each candidate and use it to determine whether the candidate is a valid guaranteed Gabriel edge. For the time being, we will assume that the uncertain discs are partially disjoint. Later, we will modify our algorithm to relax this restriction.

Since the Gabriel graph of a set of points is a subset of the Delaunay triangulation of the points, the set of guaranteed Gabriel edges for $\mathcal{D}$ must be a subset of the guaranteed Delaunay edges of $\mathcal{D}$. Hence, for the first step, we can use the same

set of candidates that we used for generating the guaranteed Delaunay edges in Section 2.4. Specifically, we can apply Lemma 2.24 and Lemma 2.25 to show that we need consider only candidates (a, b) that correspond to nonempty cells within $V^2(\mathcal{D})$.

Let us now consider the second step of our algorithm. By Lemma 2.29, if (a, b) is not a guaranteed Gabriel edge, then some disc contributing to $\overline{\mathcal{D}}$ must intersect $Z_{a,b}$. We will say that D_k is a *certificate* that (a, b) is not a guaranteed Gabriel edge. Determining whether (a, b) is a guaranteed Gabriel edge is thus equivalent to determining if no certificate for (a, b) exists.

A simple approach for this problem would be to test each disc contributing to $\overline{\mathcal{D}}$ for intersection with $Z_{a,b}$. This would require $\Omega(n)$ time for each candidate edge (a, b). In the following discussion, we show that we can improve this running time by reducing the number of uncertain discs that must be examined for each candidate edge.

We will show that if a certificate for (a, b) exists, then some certificate exists that intersects a disc associated with a neighbor of Voronoi cell $R_{\{a,b\}}$. Note that every cell adjacent to $R_{\{a,b\}}$ must be of the form $R_{\{a,x\}}$ or $R_{\{x,b\}}$, where D_x is a disc contributing to $\overline{\mathcal{D}}$. We will denote this set of neighboring discs, $\{D_x\}$, by N.

It is interesting to note that not every disc $D_k \in \mathcal{D}$ intersecting a Gabriel disc C_p is necessarily a member of N. In Figure 2.31, D_k is the only disc of $\overline{\mathcal{D}}$ intersecting Gabriel disc C_p, yet N consists of the single element $D_{k'}$ ($R_{\{a,b\}}$ lies far below the discs shown in the figure).

We will prove the following claim:

Lemma 2.34 *If $\mathcal{D}$ is a set of partially disjoint uncertain discs, cell $R_{\{a,b\}}$ is nonempty, and a certificate for (a, b) exists, then some member of N is a certificate for (a, b).*

Our proof will show that if no disc of N is a certificate for (a, b), then no certificate exists. We will do this by demonstrating that any Gabriel disc of D_a and D_b that intersects a certificate D_k can be manipulated to show that either $D_k \in N$, or that some surrogate certificate $D_{k'} \in N$ exists. We will require the following two lemmas.

Lemma 2.35 *If $\mathcal{D}$ is a set of partially disjoint uncertain discs, and $R_{\{a,b\}} \neq \emptyset$, then $R_{\{a,b\}}$ contains a point on $\langle\langle a, b \rangle\rangle$.*

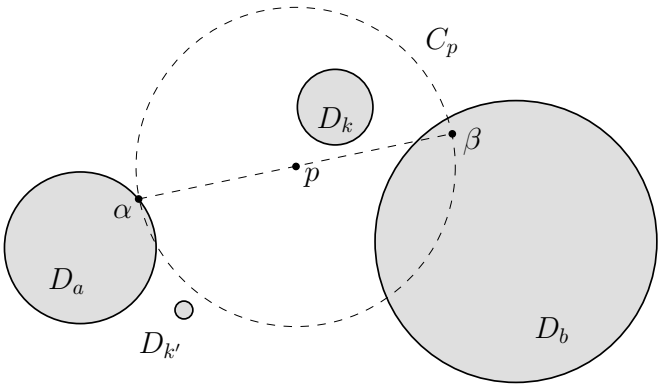

Figure 2.31: D_k intersects Gabriel disc C_p, yet is not a member of N.

Proof. Let p be a boundary point of $R_{\{a,b\}}$ (not at infinity). There exists a disc C_p and a disc D_k contributing to $\overline{\mathcal{D}}$ where, without loss of generality (see Figure 2.32): (i) C_p is outside-tangent to D_a and D_k, intersects D_b, and is not penetrated by $\overline{\mathcal{D}}$; or (ii) C_p is inside-tangent both D_a and D_k, is within D_b, and is not interior to any disc contributing to $\overline{\mathcal{D}}$.

If (i), then let t_a be the point of tangency of D_a with C_p. If $t_a \notin D_b$, then we can shrink C_p with respect to t_a until it reaches a disc $C_{p'}$ that is outside-tangent to D_b as well. Now, $p' \in \langle\langle a, b \rangle\rangle$; and since $\overline{\mathcal{D}}$ does not penetrate $C_{p'}$, $p' \in R_{\{a,b\}}$. If instead, $t_a \in D_b$, then we can grow the zero-radius disc t_a, maintaining its inside-tangency with D_a, until it reaches a disc $C_{p'}$ inside-tangent D_b as well. This must occur before it reaches D_a, since the discs are partially disjoint. Observe that $p' \in \langle\langle a, b \rangle\rangle$, and since t_a is not in $\overline{\mathcal{D}}$'s interior, $C_{p'}$ is not interior any disc contributing to $\overline{\mathcal{D}}$. Hence, $p' \in R_{\{a,b\}}$.

If (ii), we can grow C_p, maintaining its inside-tangency with D_a, until it reaches disc $C_{p'}$ that is inside-tangent D_b as well, so that $p' \in \langle\langle a, b \rangle\rangle$. As with the previous case, $C_{p'}$ is not interior any disc contributing to $\overline{\mathcal{D}}$; hence, $p' \in R_{\{a,b\}}$. □

Lemma 2.36 *If $\mathcal{D}$ is a set of disjoint uncertain discs, cell $R_{\{a,b\}}$ is nonempty, and $D_a \cap D_b \cap \overline{\mathcal{D}} \neq \emptyset$, then some member of N is a certificate for (a, b).*

Proof. Suppose $R_{\{a,b\}}$ is nonempty, and $I = D_a \cap D_b \cap \overline{\mathcal{D}}$ is nonempty as well. Observe that there must be a point p on the boundary of I that lies on the boundaries

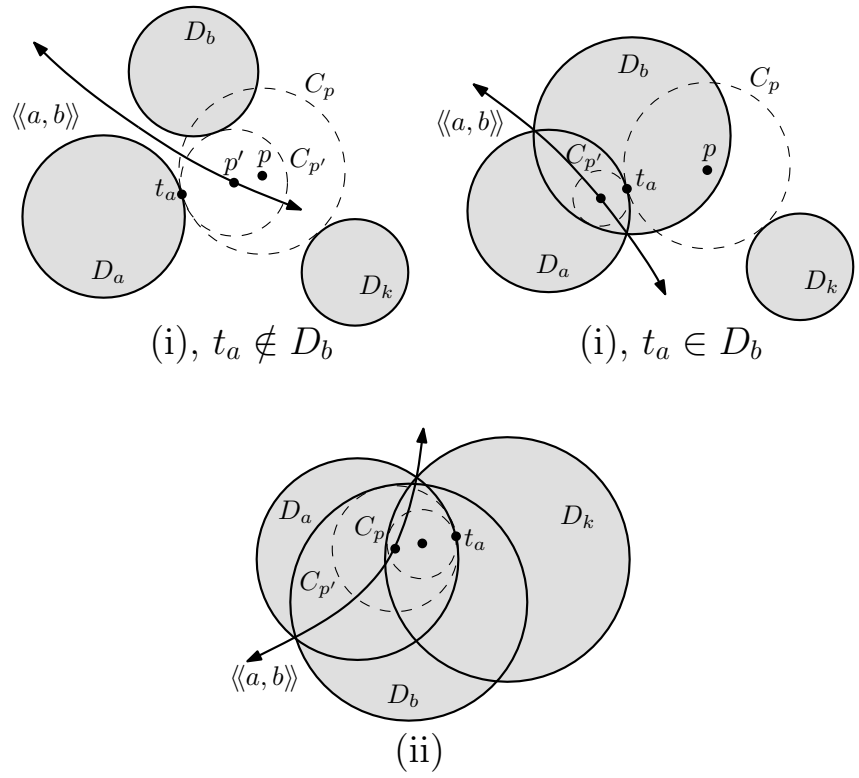

Figure 2.32: Lemma 2.35.

of (i) both D_a and $\overline{\mathcal{D}}$, (ii) both D_b and $\overline{\mathcal{D}}$, or (iii) both D_a and D_b. In the first two cases, p is a point on the boundary of $R_{\{a,b\}}$, and intersects some disc D_k contributing to $\overline{\mathcal{D}}$; hence, D_k is within N, and (since p is a Gabriel disc of D_a and D_b) D_k is a certificate for (a, b).

In case (iii), p lies on $\langle\langle a, b\rangle\rangle$, at one of the two vertices of the lune $D_a \cap D_b$. By Lemma 2.35, $\langle\langle a, b\rangle\rangle \cap R_{\{a,b\}} \neq \emptyset$; so starting with disc $C_p = p$, let us move C_p along $\langle\langle a, b\rangle\rangle$ towards $R_{\{a,b\}}$, maintaining its tangency with both D_a and D_b, until we reach C_q, where q is on the boundary of $R_{\{a,b\}}$.

If $q \in I$, then C_q is inside-tangent D_a, D_b, and some D_k contributing to $\overline{\mathcal{D}}$. Since $q \in R_{\{a,b\}}$, $D_k \in N$; and since $C_q \subset I$, Lemma 2.30 implies that D_k is a certificate for (a, b).

If $q \notin I$, then C_q is outside-tangent D_a, D_b, and some D_k contributing to

$\overline{\mathcal{D}}$. Since $q \in R_{\{a,b\}}$, $D_k \in N$. Note also that t, the point of tangency of C_q and D_k, must have appeared on the receding arc of C_p as it reached C_q, and that this receding arc lies within CH$(D_a \cup D_b)$. Hence, by Lemma 2.30, D_k (which contains t) is a certificate for (a, b). □

We are now ready to prove Lemma 2.34. Suppose $R_{\{a,b\}} \neq \emptyset$, and that C_{p_0} is a Gabriel disc (of D_a and D_b) that intersects $\overline{\mathcal{D}}$. If D_a, D_b, and $\overline{\mathcal{D}}$ have a common point of intersection, then we can apply Lemma 2.36, and we are done. Otherwise, since no common intersection point exists, we can shrink C_{p_0} until it first becomes outside-tangent to either D_a, D_b, or $\overline{\mathcal{D}}$. We can then shrink it further, with respect to this tangency point, until it reaches a disc C_{p_1} that intersects all three objects, and is outside-tangent two of them. Now, if one of the tangent objects is $\overline{\mathcal{D}}$, then p_1 is on the boundary of $R_{\{a,b\}}$, and the tangency point belongs to some D_k contributing to $\overline{\mathcal{D}}$, a member of N(Figure 2.33). In addition, since $C_{p_1} \subset C_{p_0}$, D_k is a certificate for (a, b).

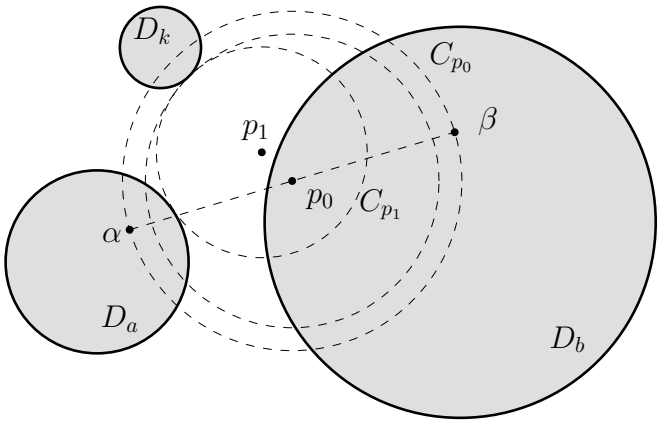

Figure 2.33: p_1 is on the boundary of $R_{\{a,b\}}$.

If the two tangent objects are D_a and D_b, then p_1 lies on $\langle\langle a, b\rangle\rangle$ (Figure 2.34). We can now use a similar approach to that of the proof of Lemma 2.36 to find some C_q that has outside-tangent D_a, D_b, and some $D_{k'}$ contributing to $\overline{\mathcal{D}}$. Clearly, $D_{k'} \in N$. We need only show that $D_{k'}$ is a certificate for (a, b). There is one complication that did not occur in Lemma 2.36: t (the point of tangency of $D_{k'}$ with C_q) still lies on the receding arc of C_q (shown in bold in Figure 2.34), but

this arc may not be contained by CH($D_a \cup D_b$). Note however that the arc will lie within the union of CH($D_a \cup D_b$) and the original Gabriel disc C_{p_0}. Hence, t lies within some Gabriel disc of D_a and D_b, and $D_{k'}$ (since it contains t) is a certificate for (a, b).

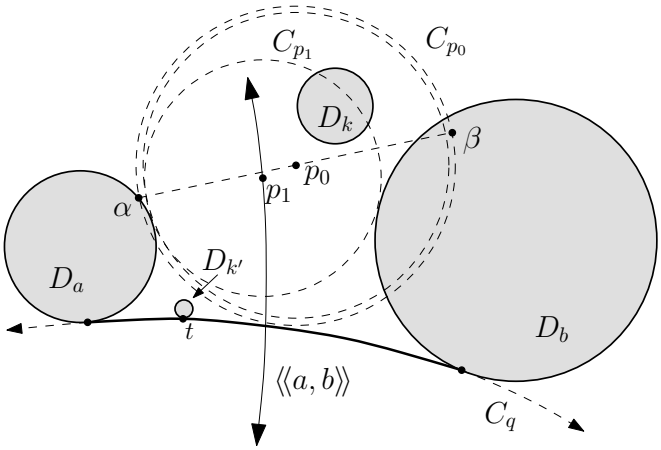

Figure 2.34: p_1 lies on $\langle\langle a, b \rangle\rangle$.

Now that we have proved Lemma 2.34, we are ready to describe an efficient algorithm for finding guaranteed Gabriel edges. Assume we are given $\mathcal{D}$, a set of n partially disjoint uncertain discs. We first construct $V^2(\mathcal{D})$, in $O(n \log n)$ time, to generate the set of candidate edges. For each candidate (a, b), we examine each neighbor D_x of $R_{\{a,b\}}$, to see if D_x intersects Gabriel zone $Z_{a,b}$. By Lemma 2.34, (a, b) is a guaranteed Gabriel edge iff no such intersecting neighboring disc is found. Since $V^2(\mathcal{D})$ has $O(n)$ complexity, we will examine $O(n)$ neighbors in total. Hence, the guaranteed Gabriel edges of $\mathcal{D}$ can be found in $O(n \cdot (C + \log n))$ time, where C is the time required to determine if a disc intersects a Gabriel zone.

We can modify this algorithm to handle discs that are not partially disjoint. The approach is very similar to that employed in Section 2.4, and will make use of the following lemma.

Lemma 2.37 *Let S be the outer discs of uncertain discs $\mathcal{D}$, and suppose $\mathcal{D}$ includes D_a and D_b, where R_a is nonempty, and D_b is the only child of D_a. If a certificate for (a, b) exists, then one of the neighbors of cell $R_a \subset V(S)$ is a*

certificate for (a, b).

Proof. If D_a intersects some $D_{k \neq b} \in \overline{\mathcal{D}}$, then we can apply Lemmas 2.27 and 2.30 to show that some neighbor of R_a is a certificate for (a, b).

Otherwise, if a certificate for (a, b) exists, there exists some Gabriel disc C_{p_0} (of D_a and D_b) that intersects $\overline{\mathcal{D}}$. Since the diameter $\overline{\alpha\beta}$ defining C_{p_0} lies within D_a, and $D_a \cap \overline{\mathcal{D}} = \emptyset$, we can shrink C_{p_0} until it is outside-tangent some D_k contributing to $\overline{\mathcal{D}}$ (and not penetrated by $\overline{\mathcal{D}}$), at which time it will still intersect D_a. We can then shrink the disc again, with respect to D_k, until it reaches some C_{p_1} that is outside-tangent D_a and D_k. Now, D_k is a neighbor to D_a in $V(S)$; and since $C_{p_1} \subset C_{p_0}$, D_k is also a certificate for (a, b). □

Our modified algorithm consists of these steps:

1. We construct $V(\mathcal{D})$, and S, the outer discs of $\mathcal{D}$.
2. We construct $V^2(S)$.
3. For each $D_a \in S$ with exactly one child D_b, we determine if any neighbors of D_a within $V(\mathcal{D})$ intersect $Z_{a,b}$. If none are found, then by Lemma 2.37, we output (a, b) as a guaranteed Gabriel edge.
4. For each cell $R_{\{a,b\}}$ of $V^2(S)$, if either D_a or D_b has children, we do nothing. Otherwise, as in the original algorithm, we examine the neighbors of $R_{\{a,b\}}$ to determine whether (a, b) is a guaranteed Gabriel edge.

Since the modified algorithm has the same asymptotic running time as the original, we can claim:

Theorem 2.38 *The guaranteed Gabriel edges of* n *uncertain discs can be found in* $O(n \cdot (C + \log n))$ *time, where* C *is the time required to determine if a disc intersects a Gabriel zone.*

In Section 2.5.1, we showed that determining if a disc intersects $Z_{a,b}$ can be performed efficiently using analytical methods (and a simple local coordinates transformation); hence, in practice, $C = O(1)$, and the guaranteed Gabriel edges can be found in $O(n \log n)$ time.

2.6 Closing Remarks

In this chapter, we provided complexity bounds and worst-case optimal algorithms for guaranteed Voronoi diagrams of uncertain discs and disjoint uncertain polygons. We introduced a new diagram, the possible Voronoi diagram of uncertain discs, and provided corresponding complexity bounds and an algorithm that is worst-case optimal (up to logarithmic terms) for its construction. We also demonstrated how to efficiently construct the guaranteed Delaunay and Gabriel edges of uncertain discs.

One possible avenue for extending the results of this chapter would be to develop an algorithm for the guaranteed order-k Voronoi diagram of discs. Much of the theory we have presented concerning guaranteed (order-1) Voronoi diagrams can be applied to the order-k case, and an algorithm for constructing order-k diagrams might be very similar to that of Theorem 2.9. Perhaps the only complication with the order-k version is that no direct analog to Lemma 2.8 exists: the sequence of bisectors $(\langle i, j\rangle, \ldots)$ forming the boundary of a guaranteed cell is not necessarily derived from a subset of the bisectors $(\langle\langle i, j\rangle\rangle, \ldots)$ forming the boundary of the standard cell.

To elaborate, let S be (the indices of) a subset of k discs, and p be a point on the boundary of the guaranteed cell, $\widetilde{R}_S$. As we did in for the order-1 case, we can apply a mapping similar to that of Section 2.2.2 to find an analogous point q on the boundary of the standard cell, R_S. Now, p will be the center of a disc C_p that has inside-tangent some $D_{u\in S}$, and outside-tangent some $D_{v\notin S}$; and q will be the center of a disc C_q that has outside-tangent some $u' \in S$, and outside-tangent some $D_{v'\notin S}$. As with the order-1 case, v' will equal v; but unlike that case, u' may be different than u. Basically, the disc of S that induces a point on the boundary of the guaranteed cell may be different than that which induces the point on the boundary of the standard cell.

We can show that the inside-tangent discs u are closely related to the farthest disc Voronoi diagram of S, and hence can be efficiently determined by querying that diagram. What is more, we believe that an efficient algorithm for generating the order-k guaranteed Voronoi diagram can be found by extending the algorithm for order-1 diagrams and incorporating some of the techniques used in our

algorithm for constructing the guaranteed Voronoi diagram of uncertain polygons (Section 2.2.4), since it uses farthest point Voronoi diagrams to solve a very similar problem.

Another possible extension to the results presented here would be to extend the results of Section 2.2.4 to allow the uncertain polygons to intersect. The (standard) Voronoi diagram of polygons is derived from that of line segments, and if the line segments are allowed to properly intersect, this has implications not only for the algorithm, but also for the complexity of the resulting Voronoi diagrams. Despite these concerns, we believe that the algorithm of Theorem 2.15 can be adapted with little or no changes to the case where the uncertain polygons are allowed to intersect, provided an efficient algorithm for constructing the standard Voronoi diagram of the polygons is available.

Some other structures related to Delaunay triangulations that we did not consider, but that could be investigated within the framework of regions of uncertainty, are (Euclidean) minimum spanning trees and relative neighborhood graphs.

In the relative neighborhood graph [71] of a point set, an edge exists between points a and b if no third point c exists that is closer to both a and b than they are to each other. These graphs were introduced by Toussaint [71], and he showed that they are subsets of Delaunay triangulations, and supersets of minimum spanning trees (in fact, they are subsets of Gabriel graphs as well). A natural extension of these graphs to the framework of uncertain regions would be to label two uncertain regions as *guaranteed neighbors* if they share an edge in the relative neighborhood graph of every feasible set of the regions.

In its simplest formulation, the minimum spanning tree of a point set is the set of edges that (i) causes the points to be connected, and (ii) minimizes the sum of the lengths of the edges. One might define (a, b) to belong to the set of *guaranteed spanning edges* of a set of uncertain regions if (a, b) is an edge in the minimum spanning tree of every feasible set of the regions. We suspect that an efficient algorithm for generating such edges may be difficult to find, since unlike Delaunay, Gabriel, and relative neighborhood graphs, uncertainty involving sites in one part of the plane can affect whether two sites in a distant part of the plane are joined by a guaranteed edge. For example, in Figure 2.35, whether or not (a, b) is a guaranteed spanning edge depends upon the distance between c and d.

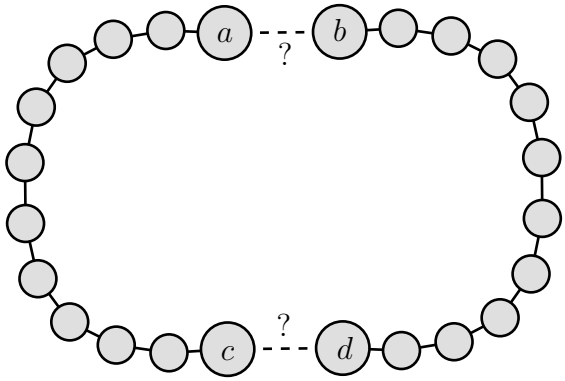

Figure 2.35: Guaranteed spanning edges.

There is some existing research on minimum spanning trees of uncertain regions. Yang *et al* [75] showed that the problem of finding the feasible set of a set of uncertain discs or rectangles that minimizes the minimum spanning tree total edge length is NP-hard. Chambers *et al* [14] examined a related problem, that of finding the feasible set that minimizes the length of the minimum spanning tree's longest edge. They showed that this problem is NP-hard for two classes of uncertain regions: unit length squares, and vertical line segments.

A *β-skeleton* of a set of points is a generalization of a Gabriel graph, and was introduced by Kirkpatrick and Radke [39]. If p and q are two points in a set, then (p, q) is an edge in the β-skeleton iff for every third point r from the set, the angle $\angle prq$ (or $\angle qrp$, whichever is smaller) is smaller than an angle derived from β. When $\beta = 1$, this angle is $\frac{\pi}{2}$, and the β-skeleton equals the Gabriel graph. For values $\beta > 1$, the analog to a Gabriel disc with diameter $\overline{pq}$ is a lune with vertices p and q. A natural extension of the results of this chapter to β-skeletons would start by stating that a *guaranteed β-skeleton edge* (p, q) exists iff for every feasible set S of a set of uncertain regions, the lune induced by p and q does not intersect $S \setminus \{p, q\}$.

An applet exploring some of the geometric objects described in this chapter can be found at http://www.cs.ubc.ca/~jpsember/gv.html. The applet (Figure 2.36) includes the following sample files:

1. `discs.dat`: Constructs the guaranteed Voronoi diagram of uncertain discs,

using an algorithm similar to the one described in the proof of Theorem 2.9. Additional objects that can be displayed include the possible Voronoi diagram of the discs, the possible cell of a particular disc, and the guaranteed Delaunay edges of the discs.

2. `polygons.dat`: Constructs a guaranteed Voronoi cell for one of a set of uncertain polygons. The program uses a simple, 'brute-force' algorithm, instead of that of Section 2.2.4.

3. `gabriel.dat`: Constructs the Gabriel zone for a pair of uncertain discs, and determines whether it is intersected by a third disc.

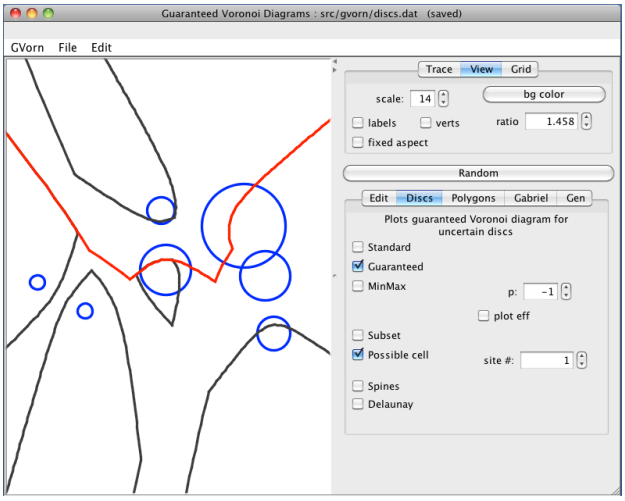

Figure 2.36: Applet for Chapter 2.

Chapter 3

Convex Hull of Imprecise Points

3.1 Introduction

Computing convex hulls is a fundamental problem in computational geometry. The concept of a convex hull is easy to grasp; imagine that the geometric plane is represented by a board, and a set of points in the plane is represented by a set of nails sticking out of the board. If a rubber band is stretched to enclose all of the nails and released, then the rubber band will come to rest in the shape of a convex polygon (Figure 3.1). A nail will be located at each vertex of the polygon, preventing the rubber band from contracting further. This polygon is the convex hull of the points.

A set of points is *convex* if, for every pair of points a, b in the set, the segment $\overline{ab}$ is also within the set. The *convex hull* of a set of points S is the smallest convex set that contains S, and is denoted CH(S). Equivalent definitions for CH(S) include: the intersection of every convex set that contains S; and, the set of all points that can be expressed as convex combinations of points in S.

There are many applications for convex hulls. In computer graphics, for instance, convex hulls can be used as an approximation of the boundary of an object. If no part of the convex hull of a shape is visible on the screen, it follows that no part of the shape is visible either. If the complexity of the shape's convex hull is significantly less than that of the shape itself, then ensuring that the convex hull is at least partially visible before plotting the shape can lead to increased perfor-

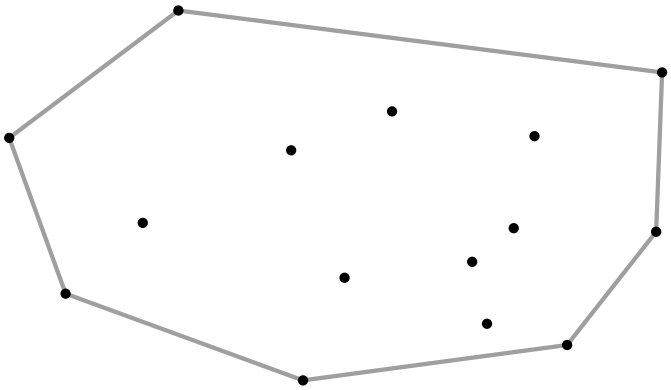

Figure 3.1: The convex hull of a set of points

mance. Convex hulls can also be used to calculate extrema associated with point sets. For example, consider the *diameter* of a set of points, which is the maximum distance separating any two points of the set. It is easy to show that the two points in question must lie on the convex hull of the points. Once the convex hull has been calculated, the technique of *rotating calipers* [72] can be applied to find these two points. Closely related to the diameter of a point set is its *width*, defined as the width of the smallest strip (or region bounded by parallel lines) that contains the set. This too can be found by applying rotating calipers to the set's convex hull.

Perhaps the simplest algorithm for constructing the convex hull of a set of points is the Jarvis march (also known as the gift wrapping algorithm) [33]. The algorithm constructs the ccw sequence of hull points, starting with the lowest point in the set (which is clearly guaranteed to lie on the hull). At each iteration, it calculates the next vertex by constructing a ray from the previous vertex through each of the original points. The point associated with the ray that makes the smallest ccw angle with the previous hull edge is output as the next hull vertex. It has an $O(nh)$ running time, where n is the number of points, and h is the number of points on the convex hull. If every point lies on the convex hull (i.e., so that $h = n$), its running time is $\Theta(n^2)$, which is suboptimal.

$O(n \log n)$ algorithms for constructing convex hulls of points include Graham's scan [29], and a divide-and-conquer algorithm from Preparata and Hong [58]. The Quickhull algorithm [9] is a randomized algorithm with expected $O(n \log n)$

performance, and is so named due to its similarity to the Quicksort algorithm for sorting numbers. An output-sensitive algorithm is one whose running time depends not only on the size of the input, but also on the size of the output. An output-sensitive, $O(n \log h)$ convex hull algorithm was first proposed by Kirkpatrick and Seidel [40], and a simpler algorithm with the same running time was later given by Chan [15].

If the location of each point is unknown, but known to lie within a particular region of uncertainty, then the convex hull of the points (i.e., the 'true' convex hull) cannot be determined, and may be one of a continuum of *feasible hulls*: convex hulls of feasible sets. In this case, in order to construct a boundary approximation of the set, or to calculate extrema associated with the set, it may be useful to know which points in the plane are guaranteed to be within the true convex hull. To this end, we define the *guaranteed hull* of the set of uncertain regions as the set of points that are within the convex hull of every feasible set of the regions. If, instead, we wish to know which points in the plane *may* lie within the true convex hull, then we are interested in knowing the *possible hull* of the regions: the set of points that are within the convex hull of at least one feasible set.

Figure 3.2 depicts a set of uncertain regions and its corresponding convex hull variants. The dotted boundary represents the convex hull of the regions, the dashed boundary represents the guaranteed hull of the regions, and the solid boundary represents the possible hull of the regions.

As a motivating example for the possible hull problem, suppose each of the regions of Figure 3.2 is an island, and each island contains a sensor whose exact location is unknown. If a line-of-sight signal is transmitted between each pair of sensors, and the signal can detect any object passing between the pair, then a boat that wishes to approach the islands surreptitiously must remain outside of the possible hull of the islands.

Possible hulls are also useful as a means of preprocessing a set of uncertain regions. Later, we will see how they can be used to identify those regions that contribute vertices to every feasible hull of the regions.

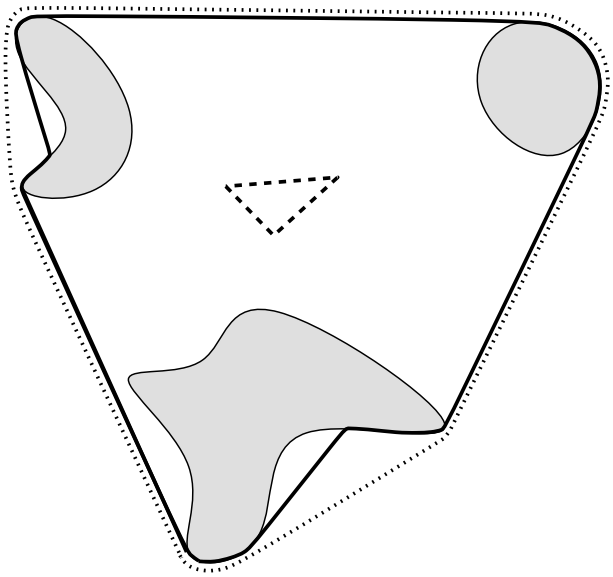

Figure 3.2: Convex hull, guaranteed hull, and possible hull of uncertain regions

3.1.1 Related Work

An $O(n \log n)$ algorithm for constructing the guaranteed hull of a set of uncertain discs is given by Rappaport [61]. He shows that by modifying the discs' radii in a particular way, the guaranteed hull of the original discs can be derived from the boundary segments of the modified discs' convex hull.

Using a domain-theoretic approach to computational geometry, Edalat *et al* ([21],[22]) define a *partial convex hull* as being a pair of open sets (E, I), where E is a portion of the plane guaranteed to be outside the hull, I is a portion guaranteed to be in the hull's interior, and the hull boundary is exterior to both E and I. The closure of I is what we refer to as the guaranteed hull, while the closure of the complement of E is what we refer to as the possible hull. The authors prove that E and I can be calculated in $O(n \log n)$ time, when the uncertain input points are represented as *partial points*, which are essentially axis-aligned rectangles representing the uncertainty in the x and y coordinates. They extend this work to uncertain points represented by convex polygons in [23], and give an $O(m \cdot N \log N)$ algorithm for guaranteed hulls of such points, where N is the number of polygons, and

m is the number of distinct slopes of the polygons' sides.

Guaranteed and possible hulls of uncertain regions have also been investigated by Nagai, Seigo, and Tokura [55]. They provide $O(n \log n)$ algorithms for constructing the guaranteed hull of uncertain discs, and a particular class of polygon that includes axis-aligned rectangles. Nagai and Tokura expand this work in [54]; they represent a convex region as a function representing the distance of the object from a directed line with a particular polar angle, and show that the the convex hull of a set of regions represented in this way corresponds to the upper envelope of a set of such functions. Using this result, they achieve an $O(n \log n)$ algorithm for guaranteed hulls of uncertain convex polygons with n total vertices (an improvement over the algorithm of [23]).

The same tools used to construct guaranteed hulls can be used to determine extremal feasible values for the width and diameter of imprecise points ([61],[55],[54]). For example, the minimum feasible diameter for a set of uncertain polygons with n total vertices can be found in $O(n \log n)$ time, while the maximum feasible diameter for the set can be found in expected $O(n\alpha(n) \log n)$ time [54].

Löffler and van Kreveld have also studied convex hulls of uncertain regions [47]. They investigate the problem of optimizing a function among feasible hulls, such as finding convex hulls with maximal or minimal areas or boundary lengths, where the hulls are associated with regions of uncertainty that are line segments or axis-aligned squares. They provide polynomial-time algorithms ranging from $O(n \log n)$ to $O(n^{13})$ for some variants of these problems, and show that others are NP-hard. Variants involving uncertain discs are also discussed.

Ezra and Mulzer [25] have investigated the problem of preprocessing uncertain regions to later allow efficient construction of convex hulls (when the uncertainty in the input has presumably been resolved). They show that with $O(n^2)$ preprocessing time, the convex hull of any feasible set of points drawn from a set of uncertain lines (or line segments) can be constructed in $o(n \log n)$ time.

3.1.2 Contributions

This chapter looks at guaranteed and possible hulls of uncertain regions. We prove that the complexity of the guaranteed hull of a set of uncertain regions is strictly

linear in the number of the regions, regardless of the complexities of the individual regions. This improves the $O(n\alpha(n))$ bound for uncertain polygons implicit in the paper by Nagai and Tokura [54].

We present two optimal algorithms for constructing guaranteed hulls of an important subclass of uncertain regions. This subclass includes discs and squares, the most common uncertain regions likely to be encountered in practice. The second of these algorithms, an adaptation of Chan's convex hull algorithm [15], is output-sensitive (subject to some assumptions that we will give later). While Nagai and Tokura's guaranteed hull algorithm [54] has a running time that matches that of our first algorithm, and can be modified so that its running time matches that of our second algorithm as well, it relies upon a dual-space approach that somewhat obscures the geometry of the uncertain regions. In our algorithms, this geometry remains explicit.

In addition to these two new guaranteed hull algorithms, we show how three existing planar guaranteed hull algorithms can be modified to produce algorithms for 3-dimensional guaranteed hulls.

We investigate possible hulls of nonconvex uncertain regions, and present an optimal algorithm for constructing such hulls for sets of uncertain polygons. We also show how possible hulls can be used within a preprocessing step to determine which regions contribute vertices to every feasible hull.

3.2 Guaranteed Hull

The guaranteed hull of a set of uncertain regions $\mathcal{D}$ is defined formally as the set of points p such that for every feasible set S of $\mathcal{D}$, $p \in \text{CH}(S)$. We will denote the guaranteed hull of $\mathcal{D}$ by $GH(\mathcal{D})$ (or, when $\mathcal{D}$ is clear from the context, simply by GH).

One way in which guaranteed hulls can be useful is in preprocessing a set of imprecise points to optimize some later operation on (precise) feasible sets. For example, consider the problem of filtering out the regions (of a set of regions) that *never* contribute feasible hull vertices. If these regions can be identified, then their corresponding sites within any feasible set can be ignored when constructing feasible hulls. It is easy to show that a region has this property if and only if it lies

within the guaranteed hull of the other regions.

The guaranteed hull of a set of uncertain regions is closely related to the set of directed lines that are tangent to pairs of the regions. This was shown in [55], though with a more complicated analysis that obscured this relationship to some extent. In this section, we will provide a geometrically intuitive definition of these tangent lines. For simplicity, we will assume that each set of uncertain regions $\mathcal{D}$ that we discuss has at least three regions. In addition, we assume that no line exists that is tangent to three regions of $\mathcal{D}$, and also that no vertical line exists that is tangent to two regions of $\mathcal{D}$. If a directed line L is right-tangent[1] to a region of $\mathcal{D}$, then we say that the region *supports* L. By contrast, if a region lies strictly to the right of L, then we say that the region *invalidates* L (Figure 3.3). A directed

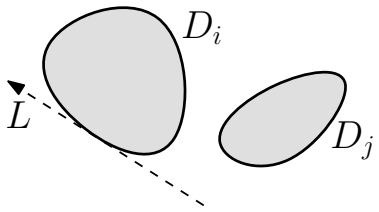

Figure 3.3: L is supported by D_i, and invalidated by D_j.

line is a *hull tangent* of $\mathcal{D}$ if some region of $\mathcal{D}$ supports the line, and the line is not invalidated by any other region of $\mathcal{D}$. Observe that each angle in $[0 \circlearrowleft 2\pi)$ is the polar angle of a unique hull tangent of $\mathcal{D}$.

A *bitangent* of uncertain regions D_i and D_j is a directed line L that is right-tangent to both D_i and D_j, with $L \cap D_i$ preceding $L \cap D_j$ along L (by our general position assumption, these two intersections must each consist of a single point, and must be disjoint). Figure 3.4 shows some of the bitangents associated with some sets of uncertain regions. Note that our definition of bitangent differs from the standard definition, in which L may be *left*-tangent to either D_i or D_j instead. We denote the bitangent of D_i and D_j by B_{ij}. For convenience, we will abbreviate $\theta(B_{ij})$ (the polar angle of B_{ij}) as θ_{ij}.

No two bitangents induced by a set of uncertain regions can lie on the same line, as a consequence of our general position assumption. It is possible, however,

[1]Recall that L is right-tangent to A iff (i) A intersects L, and (ii) A lies in L's right half-plane.

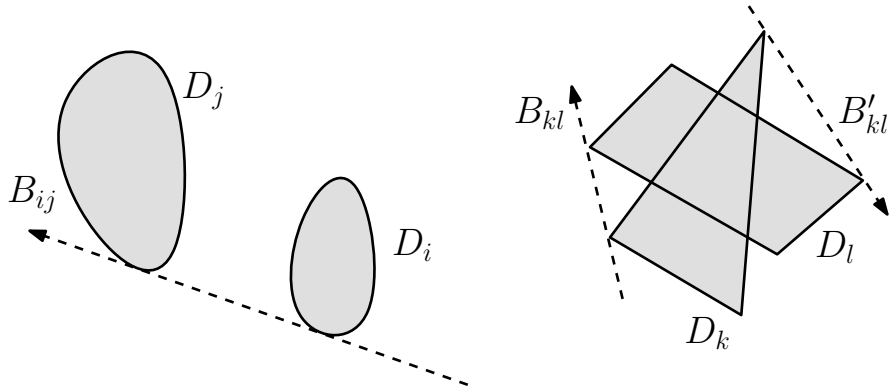

Figure 3.4: Bitangents

for an ordered pair of regions to be associated with two (or more) bitangents (e.g., B_{kl} and B'_{kl} of Figure 3.4).

We will say that B_{ij} is a hull bitangent if it is also a hull tangent. By contrast, a hull tangent L is a *pure hull tangent* if it is not also a bitangent; e.g., exactly one region is right-tangent to L.

There is a close connection between hull bitangents and guaranteed hulls:

Theorem 3.1 *If GH has nonzero area, then GH equals the intersection of the left half-planes of $\mathcal{B}$, the hull bitangents of $\mathcal{D}$; otherwise, either $\mathcal{B} = \emptyset$, or the intersection of their left half-planes does not have finite, nonzero area.*

Proof. If $\mathcal{X}$ is a set of bitangents, we will denote the intersection of their left half-planes by $I(\mathcal{X})$. We first consider the case where GH has nonzero area. We start by proving that $GH = I(\mathcal{B}')$, where $\mathcal{B}'$ is the set of directed lines whose left half-planes intersect every region of $\mathcal{D}$. If some point $p \in GH$ is not within $I(\mathcal{B}')$, then p lies strictly to the right of some line $L \in \mathcal{B}'$. Since every region of $\mathcal{D}$ intersects L's left half-plane, there exists a feasible set S of $\mathcal{D}$ that lies to the left of L. This is a contradiction, since it implies that $p \notin \text{CH}(S)$. Hence, $p \in I(\mathcal{B}')$, and $GH \subseteq I(\mathcal{B}')$. To show that $I(\mathcal{B}') \subseteq GH$, let p be any point in $I(\mathcal{B}')$, and suppose $p \notin GH$. Then for some feasible set S, $p \notin \text{CH}(S)$, which implies that S lies strictly left of some line Q through p. Let L be the hull tangent of $\mathcal{D}$ that is parallel to Q, and $D_i \in \mathcal{D}$ be its right-tangent region. Since $p \in I(\mathcal{B}')$, L must lie on or

to the right of Q; but then S lies strictly left of L, a contradiction (since no point of D_i lies to the left of P). We now show that $I(\mathcal{B}') = I(\mathcal{B})$ by demonstrating that every boundary point s of $I(\mathcal{B}')$ lies on a line of $\mathcal{B}$. Since $I(\mathcal{B}')$ and $I(\mathcal{B})$ are convex, and $I(\mathcal{B}') \subseteq I(\mathcal{B})$, it then follows that $I(\mathcal{B}) = I(\mathcal{B}') = GH$.

Let s be any boundary point of GH, and q be any interior point of GH. There must exist a line $L' \in \mathcal{B}'$ through s; let L' be the closest such line to q. If L' is right-tangent to two regions of $\mathcal{D}$, then L' is a hull bitangent, and we are done. Otherwise, we can reach a contradiction as follows. If L' is not right-tangent to any regions of $\mathcal{D}$, we can translate L' to the left to yield another line within $\mathcal{B}'$ that has s strictly to its right, a contradiction since this implies that $s \notin GH$. If L' is right-tangent to a single region of $\mathcal{D}$, then we can reach a contradiction by rotating L' around the single tangency point to yield an element of $\mathcal{B}'$ that either has s strictly to its right, or (if s is the tangency point) is closer to q.

Now let us consider the case where GH has zero area (which includes the case where $GH = \emptyset$). Suppose, for the sake of contradiction, that $\mathcal{B} \neq \emptyset$, and that $I(\mathcal{B})$ has bounded, nonzero area. There must exist a point s that is interior to $I(\mathcal{B})$ and exterior to GH. In addition, s must lie strictly to the right of L, some hull tangent of $\mathcal{D}$. Let D_i be the region right-tangent to L, t be its tangency point, and N be the line through s that is perpendicular to L (Figure 3.5). If we now now rotate L cw (or ccw, if t lies to N's right) around D_i, it must either reach a bitangent of $\mathcal{D}$, or become parallel to and strictly left of some bitangent containing an edge of $\mathcal{B}$, before it reaches s. We now have a contradiction: the former implies that $s \notin I(\mathcal{B})$, and the latter implies that the original L was not a hull tangent. □

Observe that since the set of lines right-tangent to a region is equal to the set of lines right-tangent to a region's convex hull, Theorem 3.1 implies that each uncertain region can be replaced with its convex hull without affecting the guaranteed hull of the regions.

When ordered by their polar angles, the hull tangents for a set of uncertain regions have a particular structure:

Lemma 3.2 *For every hull tangent L of $\mathcal{D}$ supported by a region D_i, there exist adjacent bitangents $(\ldots B_{*i}, B_{i*} \ldots)$ in the polar angle-ordered sequence of hull bitangents of $\mathcal{D}$ where $\theta(L) \in [\theta_{*i} \circlearrowleft \theta_{i*}]$.*

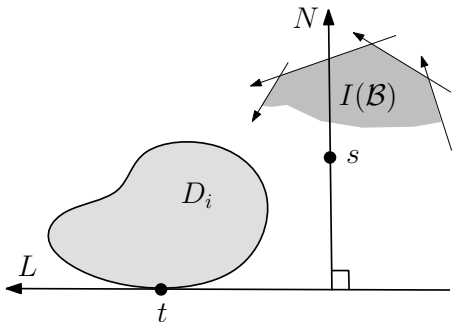

Figure 3.5: Theorem 3.1.

Proof. We prove the claim by showing that if L is a pure hull tangent, and is rotated ccw (resp., cw) around D_i, the first hull bitangent it reaches will be of the form B_{i*} (resp., B_{*i}). We will prove only the ccw case, as the cw case is very similar. First, observe that D_i cannot contain any other region of $\mathcal{D}$, since that region would invalidate L. Note also that since L is a pure hull tangent, every other region of $\mathcal{D}$ penetrates its left half-plane. Let M be the first hull bitangent reached when L is rotated ccw around D_i, and $p_i \in D_i$ and $p_j \in D_j$ be its associated tangent points. Suppose p_j precedes p_i on M. We can now rotate M slightly cw around D_i to reach line N that is right-tangent to D_i and has D_j strictly to its right (Figure 3.6). Now, D_j penetrated the left half-plane of (the original) L; hence

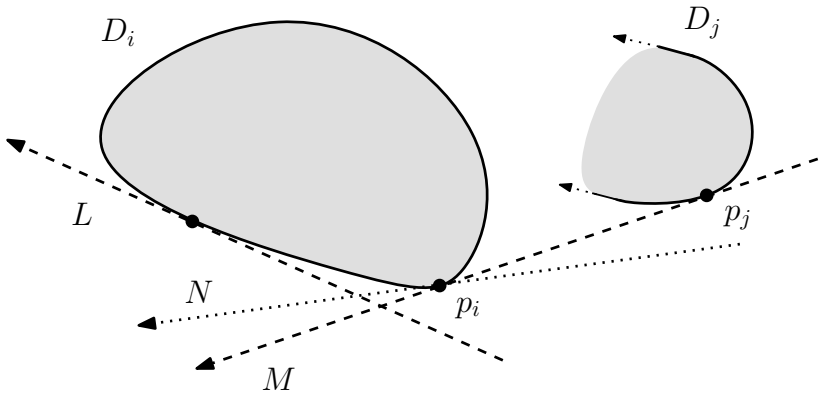

Figure 3.6: Lemma 3.2.

L must have become right-tangent to D_j before M. This is a contradiction, since

we would have stopped rotating L at that time. Thus, p_j follows p_i along M, and $M = B_{ij}$.

Note that a similar argument will show that if L is invalidated by some other region of $\mathcal{D}$ (and hence is not a hull tangent), then if it is rotated ccw (resp., cw) around D_i, the first hull bitangent it reaches will be B_{*i} (resp., B_{i*}). □

Besides their role in constructing guaranteed hulls, we can identify another use for hull bitangents:

Theorem 3.3 *If L is a directed line, B is the angle-ordered set of hull bitangents of uncertain regions $\mathcal{D}$, and c is the time required to construct tangent lines for regions of $\mathcal{D}$, then it can be determined in $O(\log |B| + c)$ time whether the left half-plane of L contains a feasible set of $\mathcal{D}$.*

Proof. By Lemma 3.2, B will contain adjacent bitangents $(\ldots, B_{ij}, B_{jk}, \ldots)$ where D_j supports a hull tangent M that is parallel to L, and $\theta(L) \in [\theta_{ij} \circlearrowleft \theta_{jk}]$. Note that the left half-plane of L will contain a feasible set of $\mathcal{D}$ iff M lies in the left half-plane of L. We can perform a binary search, in $O(\log |B|)$ time, to identify D_j; and we can then construct M, in time c. □

We define an *aligned similar* set of uncertain regions to be a set with the property that each region in the set can be derived from a common *base shape* by applying only translation and scaling transformations. We say that each such region is an *instance* of this base shape. We further require such sets to have the following properties:

1. Each region is compact (i.e., closed and bounded).

2. It can be determined in constant time which of two instances of the base shape is larger.

3. In constant time, the line with a particular polar angle that is right-tangent to an instance of the base shape can be constructed.

4. In constant time, the bitangent B_{ij} of instances D_i and D_j can be constructed, or it can be determined that no such bitangent exists.

Aligned similar uncertain regions include sets of discs with varying radii and origins, and sets of axis-aligned squares. We will say that two points are *similar* if they belong to distinct instances of a common base shape, and correspond to the same point in the base shape. For example, the lower left corners of a pair of axis-aligned squares are similar points.

We conclude this section by proving two properties of aligned similar uncertain regions that we will make use of later:

Lemma 3.4 *If D_i and D_j are aligned similar uncertain regions of $\mathcal{D}$, and D_i and D_j are partially disjoint, then B_{ij} and B_{ji} are unique and distinct, and the polar angle of every hull tangent supported by D_i is within $[\theta_{ji} \circlearrowleft \theta_{ij}]$.*

Proof. Let D_i and D_j be as stated, H be their convex hull, and S be the set of directed lines right-tangent to H (ordered by polar angle). Observe that we can partition S into a sequence of four subsets, where each subset contains, in order: lines right-tangent only to D_i, a bitangent of the form B_{ji}, lines right-tangent only to D_j, and a bitangent of the form B_{ij}. It follows that B_{ij} and B_{ji} are unique and distinct, and that every line right-tangent to D_i with polar angle within $(\theta_{ij} \circlearrowleft \theta_{ji})$ is a member of the first subset (and is thus invalidated by D_j). □

Lemma 3.5 *If D_i and D_j are aligned similar uncertain regions, D_i and D_j are partially disjoint, and D_i is no smaller than D_j, then $[\theta_{ji} \circlearrowleft \theta_{ij}]$ spans at most π radians.*

Proof. Let D_i and D_j be two such regions. Place the axes so that B_{ij} is aligned with the negative x-axis. Let $p_i \in D_i$, $p_j \in D_j$ be the tangent points of B_{ij}, and $q_i \in D_i$, $q_j \in D_j$ be the tangent points of B_{ji}; see Figure 3.7. Since the function that transforms points of D_i to the similar points of D_j preserves angles, p_i and q_i are similar, as are p_j and q_j. This implies that $\theta(\overrightarrow{p_i q_i}) = \theta(\overrightarrow{p_j q_j})$. Note also that since D_i is no smaller than D_j, $d(p_i, q_i) \geq d(p_j, q_j) > 0$. Hence the angle that B_{ji} makes with B_{ij} (the negative x-axis) is in the range $(0 \circlearrowleft \pi]$. □

3.3 Guaranteed Hull Complexity

In this section, we show that the guaranteed hull of a set of n uncertain regions has $O(n)$ complexity, regardless of the complexities of the regions themselves.

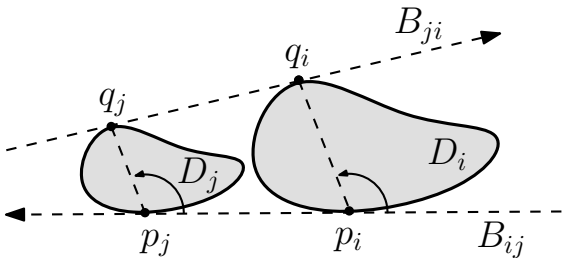

Figure 3.7: Lemma 3.5

This improves the $O(n\alpha(n))$ complexity bound for guaranteed hulls of uncertain polygons implied in [54]. Since hull bitangents are closely tied to guaranteed hulls, we also provide tight bounds for the number of hull bitangents induced by sets of aligned similar uncertain regions. We also discuss an open problem, the number of hull bitangents induced by uncertain polygons.

Theorem 3.6 *The guaranteed hull of n uncertain regions has $O(n)$ complexity, and for some sets of uncertain regions, this bound is tight.*

Proof. Let $\mathcal{D}$ be a set of n uncertain regions. By Theorem 3.1, GH is the intersection of the left half-planes of the hull bitangents of $\mathcal{D}$. We will show that at most n bitangents contribute to GH by showing that for a particular region $D_i \in \mathcal{D}$, at most one bitangent of the form B_{i*} contributes to GH. Suppose, by way of contradiction, that bitangents B_{ij} and B_{ik} contribute to GH. Without loss of generality, we can assume that $\theta_{ij} < \theta_{ik}$ (Figure 3.8). Let $p_i \in D_i$ be the tangent point of D_i associated with B_{ij}. Observe that if we rotate B_{ij} slightly cw around p_i, every region of $\mathcal{D}$ will intersect the left half-plane of the (rotated) bitangent. Hence, some feasible hull of $\mathcal{D}$ lies in this half-plane, which implies that no point on the original bitangent before p_i lies within GH. Since B_{ij} contributes to GH, there must exist a point $x \in B_{ij}$ that is a non-vertex boundary point of GH. Now, to be within GH, x must lie to the left of B_{ik}; but then x lies before p_i, a contradiction.

For some sets $\mathcal{D}$, the bound is tight; for example, if $\mathcal{D}$ consists of n small discs each centered at the vertices of a large regular n-gon, $GH(\mathcal{D})$ is a (slightly smaller) regular n-gon. □

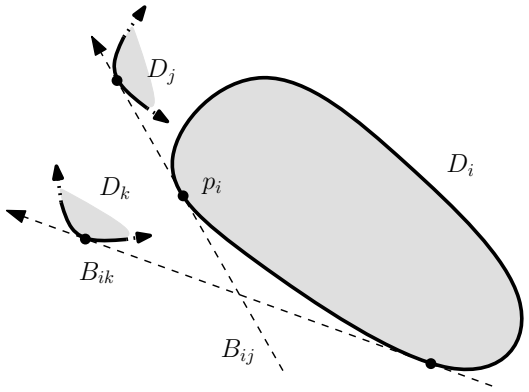

Figure 3.8: Theorem 3.6

Observe that the complexity of a guaranteed hull is linear in the number of uncertain regions, and is independent of the complexity of the individual regions. This improves the $O(n\alpha(n))$ bound for guaranteed hulls of uncertain polygons given in [54] (p. 259, Corollary 11).

While the complexity of the guaranteed hull is linear at worst, no linear bound has been proven for the number of hull bitangents induced by a set of uncertain regions. For discs at least, the number of hull bitangents is strictly linear ([55]). We can generalize this result:

Theorem 3.7 *If $\mathcal{D}$ is a set of n uncertain regions, and each pair of regions induces at most two bitangents, then (i) $\mathcal{D}$ induces at most $2n-2$ hull bitangents; and (ii) there exist sets $\mathcal{D}$ for which this bound is tight.*

Proof. Let $\mathcal{D}$ be one such set of n regions. The hull bitangents of $\mathcal{D}$, ordered by polar angle, have the form $(B_{i_1*}, B_{i_2*}, \ldots, B_{i_k*})$, which can be expressed compactly as the (cyclic) sequence $(i_1, i_2, \ldots, i_k, i_1)$. We will prove the upper bound by showing that this is a Davenport-Schinzel cyclic sequence [67] of order 2.

Suppose, by way of contradiction, that (i) $D_a \in \mathcal{D}$ induces hull bitangents B_{au} and B_{av}, (ii) $D_b \in \mathcal{D}$ induces hull bitangents B_{bw} and B_{bx}, and (iii) $\theta_{au} < \theta_{bw} < \theta_{av} < \theta_{bx}$. Observe that by Lemma 3.4, θ_{au} and θ_{av} are both within $(\theta_{ba} \circlearrowleft \theta_{ab}]$, whereas θ_{bw} and θ_{bx} are outside of this range. Since this contradicts the ccw ordering of the bitangents, no such subsequence of hull bitangents exists.

The ordered hull bitangents thus form a Davenport-Schinzel cyclic sequence of order 2, the maximum length of which is $2n-2$ ([67], Corollary 1.10). This bound is tight, as Figure 3.9 illustrates.[2] □

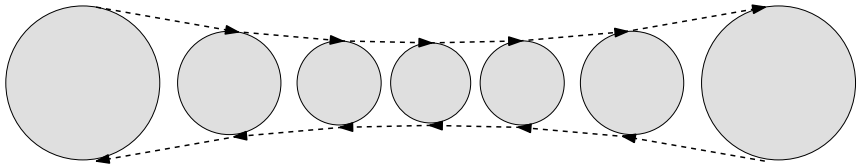

Figure 3.9: Lemma 3.7: $2n-2$ hull bitangents

The bound of Theorem 3.7 applies to aligned similar uncertain regions, and also to any set of uncertain regions that are pairwise disjoint (since each pair of such regions will induce exactly two bitangents). If every region in a set of aligned similar uncertain regions is the same size, then we can tighten the bound:

Lemma 3.8 *If $\mathcal{D}$ is a set of n aligned similar uncertain regions of uniform size, then $\mathcal{D}$ induces at most n hull bitangents.*

Proof. Let $S = \{s_i \in D_i\}$ be any set of pairwise-similar points drawn from regions of $\mathcal{D}$, and let L be the directed line containing $\overline{s_i s_j}$, where s_i and s_j are any two points of S. Suppose we move L to its left until either (i) it reaches B_{ij}, or (ii) some region of $\mathcal{D}$ lies strictly to its right. Since the regions are all translates of a common base shape, (i) will occur first if and only if $\overline{s_i s_j}$ is an edge of $\mathrm{CH}(S)$. It follows that the hull bitangents of $\mathcal{D}$ correspond exactly to the edges of $\mathrm{CH}(S)$. □

The proof of Lemma 3.8 suggests the following approach for constructing guaranteed hulls of aligned similar uncertain regions of uniform size. We first select a set of similar points from the regions, e.g., by choosing each region's leftmost point (assuming it is unique). We then construct the convex hull of these points. Any convex hull algorithm will suffice, as long as it produces the h hull vertices in ccw order and identifies which region contributed each vertex. We then construct the

[2] Interestingly, while the regions of Figure 3.9 realize the upper bound, their guaranteed hull is actually empty.

h hull bitangents corresponding to these vertices, and calculate the intersection of their left half-planes (in $O(h)$ additional time).

The complexity of the guaranteed hull of a set of uncertain regions can be substantially smaller than the number of bitangents induced by the regions. In fact, it is easy to construct sets of uncertain regions that induce a linear number of hull bitangents yet have an empty guaranteed hull (e.g., Figure 3.9).

For uncertain polygons with total complexity n, it was shown in [55] (using techniques from [6]) that the number of hull bitangents is $O(n\alpha(n))$. It is not known whether this bound is tight. The polygons in Figure 3.10 suggest that it is unlikely that an argument using Davenport-Schinzel theory similar to that employed in the proof of Theorem 3.7 will work to improve this bound. In the figure, vertex a supports hull bitangents B_{ac} and B_{ad}, vertex b supports hull bitangents B_{be} and B_{bf}, and $\theta_{ac} < \theta_{be} < \theta_{ad} < \theta_{bf}$.

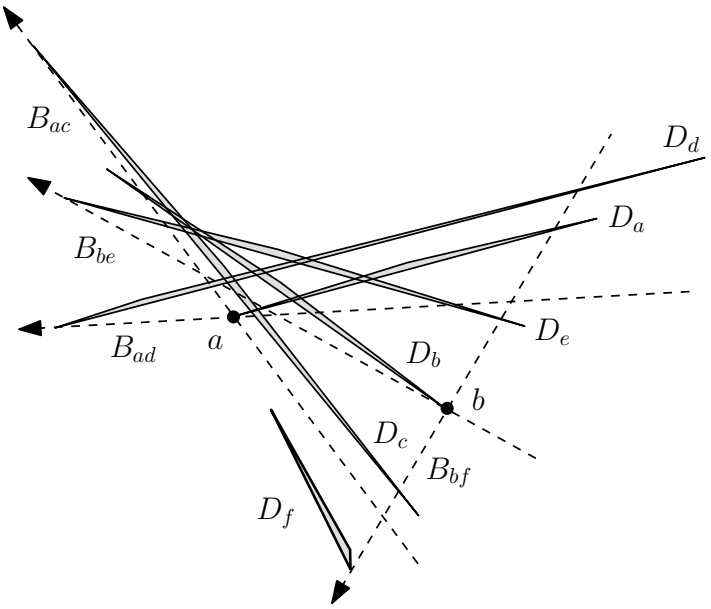

Figure 3.10: Some hull bitangents of a set of uncertain triangles

3.4 Guaranteed Hull of Aligned Similar Uncertain Regions

In this section, we present two optimal algorithms for constructing guaranteed hulls of aligned similar uncertain regions. The second of these, an adaptation of Chan's convex hull algorithm [15], is output-sensitive, if we consider its output to be the hull bitangents of the regions (and not the guaranteed hull, which is the intersection of their left half-planes). The first algorithm has the advantage of being simpler (and often faster in practice) than the second. The geometry of the uncertain regions remains explicit in both algorithms. This is in contrast to the dual-space approach used in Nagai and Tokura's algorithm [54], which can be adapted to match the running time of our second algorithm as well.

It will be convenient to focus our attention on the upper guaranteed hull of $\mathcal{D}$, which is the intersection of the left half-planes of the hull bitangents whose polar angles lie in the range $[\frac{\pi}{2}, \frac{3\pi}{2}]$. Once the upper and (by symmetric procedures) lower hulls are found, the complete guaranteed hull is easily constructed as their intersection. For simplicity, throughout this section, we will assume that uncertain regions are disc-shaped (the algorithm can be easily adapted for general aligned similar uncertain regions).

We define the *east* disc, D_e, to be the disc whose leftmost point is farthest to the right, and the *west* disc, D_w, to be the disc whose rightmost point is farthest to the left. Let $B_{\bullet e}$ be the line right-tangent to D_e with polar angle $\frac{\pi}{2}$, and $B_{w\bullet}$ be the line right-tangent to D_w with polar angle $\frac{3\pi}{2}$. Observe that both $B_{\bullet e}$ and $B_{w\bullet}$ are hull tangents of $\mathcal{D}$. Note also that if $B_{w\bullet}$ is not strictly left of $B_{\bullet e}$, then GH is empty. We will assume from this point on that this is not the case (which implies that $D_e \neq D_w$).

3.4.1 Algorithm 1

Our first algorithm has an $O(n \log n)$ running time. Starting with the east and west discs of a set of aligned similar uncertain discs, it maintains a polar angle-ordered sequence of upper hull bitangents for the set, and updates these bitangents as new discs are added to the set.

This approach for generating the guaranteed hull is similar to the one used by

the standard convex hull algorithms of Shamos [64] and Preparata[57]. Those two algorithms are *on-line* algorithms: they process their input in a serial fashion, one item (in this case, a point) at a time, and must be able to maintain the solution for the inputs seen so far without delaying any processing until all inputs have been seen. The algorithm for uncertain discs that we will present is not on-line, since it will require a step that preprocesses the entire input; but once this preprocessing is complete, the algorithm behaves essentially as an on-line algorithm, since it maintains the list of upper hull bitangents corresponding to the set of discs for the subsequence of discs seen up to the current point.

One surprising characteristic of this approach is that if the remaining discs (other than D_e and D_w) are processed in an arbitrary order, a suboptimal algorithm must result. Consider the discs $\{A_1, B_1, B_2, B_3, D_e, D_w\}$ of Figure 3.11. The

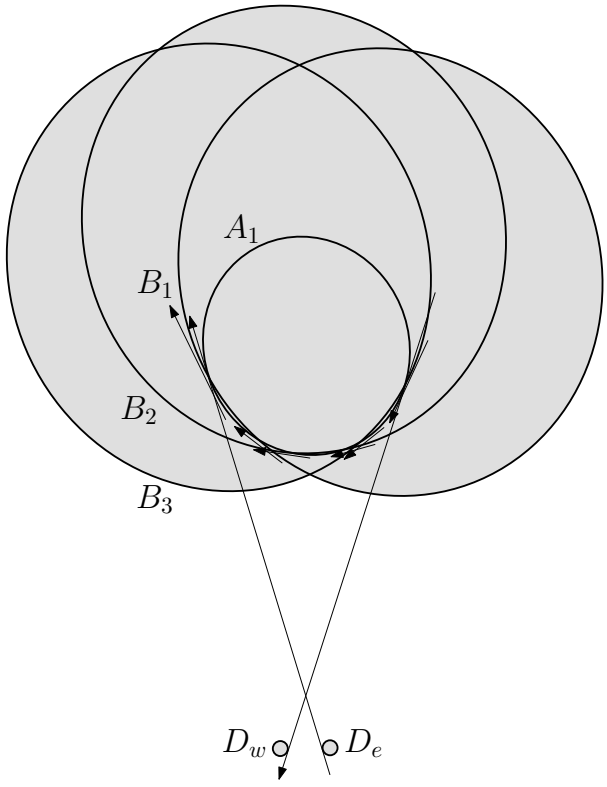

Figure 3.11: Hull bitangents of discs

(upper) hull bitangents for these six discs are shown as directed lines. Each B_i induces a pair of hull bitangents with A_1. In fact, we could construct an example with discs $\{B_1, \ldots, B_{(n-2)/2}\}$, inducing $\Theta(n)$ hull bitangents (the first two discs are D_e and D_w). Now, consider a disc A_2, the same size as A_1, but whose origin is slightly above that of A_1, so that A_2 lies strictly to the right of the existing hull bitangents. A_2 would thus invalidate these bitangents, and induce a new set of (slightly different) hull bitangents. We can construct $\Theta(n)$ such discs, $\{A_1, \ldots, A_{(n-2)/2}\}$, where each new disc $A_{i>1}$ invalidates $\Theta(n)$ hull bitangents of the previous set.

Hence, any algorithm that incrementally maintains a list of hull bitangents as new discs are added, where the discs are processed in an arbitrary order, will require $\Omega(n^2)$ steps in the worst case. In the remainder of this section, we will show how sorting the discs by size leads to an optimal algorithm.

We will assume that $\mathcal{D}$ is the ordered set $\{D_e, D_w, D_3, \ldots, D_n\}$, where the discs $\{D_{\geq 3}\}$ are sorted by increasing size. We will denote by $\mathcal{B}$ the ordered sequence of upper hull bitangents maintained by the algorithm. To simplify our algorithm, we will include the vertical hull tangents $B_{\bullet e}$ and $B_{w\bullet}$ in this sequence as well.

At the heart of our algorithm is a binary search performed for each disc in the sorted sequence. We will show that if the disc contributes new hull bitangents (i.e., so that $\mathcal{B}$ changes as a result of the new disc), then a binary search can find the location where the new bitangents are to be added. An interesting point about the search is that if the new disc does *not* contribute new hull bitangents, then this fact can be deduced from the binary search's output, whatever it ends up being.

1. Let $\mathcal{D} = \{D_e, D_w, \ldots, D_n\}$, where $\{D_{\geq 3}\}$ are sorted by increasing size.
2. Initialize q to 2, and $\mathcal{B}$ to $(B_{\bullet e}, B_{ew}, B_{w\bullet})$.
3. Increment q; if $q > n$, stop.
4. Find B_{ij}, the last bitangent in $\mathcal{B}$ for which $\theta_{iq} \in (\theta_{ij} \circlearrowleft \frac{3\pi}{2})$.
5. If no such B_{ij} is found, or B_{ij} is not followed by B_{jk} where $\theta_{jq} \in (\theta_{ij} \circlearrowleft \theta_{jk})$, then go to step 3.

6. Find B_{mn}, the first bitangent of $\mathcal{B}$ following B_{ij} that is not invalidated by B_q.

7. Replace bitangents of $\mathcal{B}$ strictly between B_{ij} and B_{mn} with the pair (B_{jq}, B_{qm}), and go to step 3.

Let us prove the correctness of the algorithm. We will denote the contents of $\mathcal{B}$ at the start of each step 3 by $\mathcal{B}_q$.

Lemma 3.9 *If aligned similar uncertain discs D_a and D_b support hull tangents A and B respectively, where $\theta(A) < \theta(B)$, then B_{ab} exists and has a polar angle within the range $[\theta(A) \circlearrowleft \theta(B)]$.*

Proof. If D_a and D_b support hull tangents A and B, then they must be partially disjoint, which implies that B_{ab} exists. Now, Lemma 3.4 implies (i) $\theta(A) \in [\theta_{ba} \circlearrowleft \theta_{ab}]$, and (ii) $\theta(B) \in [\theta_{ab} \circlearrowleft \theta_{ba}]$. If $\theta_{ab} < \theta(A)$, then (i) implies that $\theta(A) \geq \theta_{ba} > \theta_{ab}$, violating (ii); and if $\theta_{ab} > \theta(B)$, then (ii) implies that $\theta_{ab} > \theta_{ba} \geq \theta(B)$, violating (i). Hence, $\theta(B) \geq \theta_{ab} \geq \theta(A)$. □

By Lemma 3.9, $\frac{3\pi}{2} = \theta(B_{w\bullet}) > \theta_{ew} > \theta(B_{\bullet e}) = \frac{\pi}{2}$, while $\theta_{we} \in (\frac{3\pi}{2} \circlearrowleft \frac{\pi}{2})$; hence B_{ew} is the only upper hull bitangent of $\mathcal{B}_2$, and step 2 is correct. The remaining steps of the algorithm are repeated for each additional disc D_q. The following lemma is a key to the algorithm's efficiency:

Lemma 3.10 *Let $q \in \{3, \ldots, |\mathcal{D}|\}$. Then $\mathcal{B}_q$ is either $\mathcal{B}_{q-1}$, or can be constructed by replacing a contiguous subsequence of $\mathcal{B}_{q-1}$ with a pair of bitangents (B_{xq}, B_{qy}).*

Proof. If D_q contributes no bitangents to $\mathcal{B}_q$, then $\mathcal{B}_q \subseteq \mathcal{B}_{q-1}$; and since D_q supports no hull tangents of $\{D_e, D_w, \ldots, D_q\}$, no bitangent of $\mathcal{B}_{q-1}$ is invalidated by D_q. Hence, $\mathcal{B}_q = \mathcal{B}_{q-1}$, and we are done. Let us proceed assuming that D_q contributes bitangents to $\mathcal{B}_q$.

Suppose, by way of contradiction, that it contributes more than two. Then, by Lemma 3.2, it must contribute (at least) distinct hull bitangents of the form B_{qi} and B_{qj}, where $\theta_{qi} < \theta_{qj}$. Consider the case where D_i is no larger than D_q. By

Lemma 3.4, $\theta_{qj} \in [\theta_{iq} \circlearrowleft \theta_{qi}]$; and by Lemma 3.5, $\theta_{qi} - \pi$ is a lower bound for θ_{iq}, and $\theta_{iq} + \pi$ is an upper bound for θ_{qi}. Hence, $\theta_{qj} \in [\theta_{qi} - \pi \circlearrowleft \theta_{qi}] \cap [\theta_{iq} \circlearrowleft \theta_{iq} + \pi]$, which (since B_{qi} is an upper hull bitangent) implies that $\theta_{qj} \leq \theta_{qi}$, a contradiction.

If instead, D_i is larger than D_q, then D_i is either the east or west disc. If D_i is the east disc, then it supports $B_{\bullet e}$; but since D_i is larger than D_q, the above argument will show that $\frac{\pi}{2} \in [\theta_{qi} \circlearrowleft \theta_{qi} + \pi]$, a contradiction. If D_i is the west disc, then Lemma 3.4 implies that $\theta_{qj} \in (\theta_{iq} \circlearrowleft \theta_{qi})$. Since $\theta_{qj} > \theta_{qi}$, this implies $\frac{3\pi}{2} > \theta_{qj} > \theta_{iq} > \theta_{qi} > \frac{\pi}{2}$. But since D_i is the west disc, the same lemma implies that $\frac{3\pi}{2} \in (\theta_{qi} \circlearrowleft \theta_{iq})$, a contradiction. Thus, if D_q contributes any hull bitangents to $\mathcal{B}_q$, it contributes exactly two: B_{xq} and B_{qy}. By Lemma 3.2, every hull tangent of $\mathcal{D}_q$ with polar angle $\theta_{qy} > \theta > \theta_{xq}$ must be a pure hull tangent supported by D_q. Hence, every bitangent of $\mathcal{B}_{q-1}$ with polar angle in this range will not appear in $\mathcal{B}_q$.

All that remains is for us to show that every bitangent of $\mathcal{B}_{q-1}$ with a polar angle outside of this range appears in $\mathcal{B}_q$ as well. Suppose B_{uv} appears in $\mathcal{B}_{q-1}$ but not in $\mathcal{B}_q$, and that $\theta_{uv} > \theta_{qy}$. Now, B_{uv} must have been invalidated by D_q, which implies that there exists a directed line L supported by D_q that is parallel to and strictly to the right of B_{uv}. But then L is a hull tangent of $\mathcal{D}_q$, so by Lemma 3.2, there exists in $\mathcal{B}_q$ some hull bitangent B_{q*} where $\theta_{q*} > \theta_{qy}$, a contradiction. A symmetric argument handles the case where $\theta_{uv} < \theta_{xq}$. □

Suppose we are starting step 4 of a particular iteration of the algorithm, so that D_q is the next disc to be processed. For the moment, assume D_q contributes upper hull bitangents to $\mathcal{B}_q$. By Lemma 3.10, it must induce a new pair of hull bitangents (B_{xq}, B_{qy}), and there may exist a subsequence of bitangents of $\mathcal{B}_{q-1}$ that are invalidated by D_q. $\mathcal{B}_{q-1}$ is the concatenation of three contiguous subsequences:

1. bitangents not invalidated by D_q whose polar angles are less than θ_{xq};
2. bitangents invalidated by D_q; and
3. bitangents not invalidated by D_q whose polar angles are greater than θ_{qy}.

Note that the first and third subsequences are nonempty, since they must include $B_{\bullet e}$ and $B_{w\bullet}$ respectively. This implies that there must exist a last bitangent of the first subsequence. Let us now prove that this bitangent must be B_{ij} of step 4.

Suppose $D_{q\geq 3}$ contributes upper hull bitangents to $\mathcal{B}_q$, and that B_{ij} is a bitangent in $\mathcal{B}_{q-1}$. We need to show that B_{ij} belongs to the first subsequence iff $\theta_{iq} \in (\theta_{ij} \circlearrowleft \frac{3\pi}{2})$. Observe that D_q supports some pure upper hull tangent L of $\mathcal{D}_q$. If B_{ij} belongs to the first subsequence, then by Lemma 3.9, $\theta(L) > \theta_{iq} > \theta_{ij}$, which implies $\theta_{iq} \in (\theta_{ij} \circlearrowleft \frac{3\pi}{2})$, and we are done. If B_{ij} does not belong to the first subsequence, then we can assume $\theta_{iq} \in (\theta_{ij} \circlearrowleft \frac{3\pi}{2})$ (otherwise we are done). Hence, $\frac{3\pi}{2} > \theta_{iq} > \theta_{ij} > \frac{\pi}{2}$.

If B_{ij} belongs to the second subsequence, then since D_q invalidates B_{ij}, D_q must support a hull tangent that is parallel to, and to the right of, B_{ij}. It follows (by Lemma 3.4) that $\theta_{ij} \in (\theta_{iq} \circlearrowleft \theta_{qi})$, which in turn implies that $\frac{3\pi}{2} > \theta_{iq} > \theta_{qi} > \frac{\pi}{2}$. Now, if D_q is at least as large as D_i, then Lemma 3.5 implies that the range $[\theta_{iq} \circlearrowleft \theta_{qi}]$ spans at most π radians, a contradiction. If instead, D_q is smaller than D_i, then D_i must be either the east or west disc and support a hull tangent of $\mathcal{D}_q$ with polar angle $\frac{\pi}{2}$ or $\frac{3\pi}{2}$. Lemma 3.4 then implies that either $\frac{\pi}{2}$ or $\frac{3\pi}{2}$ is within $[\theta_{qi} \circlearrowleft \theta_{iq}]$, a contradiction.

Finally, if B_{ij} belongs to the third subsequence, then since B_{ij} is a hull bitangent of $\mathcal{D}_q$, Lemma 3.4 implies that $\theta_{ij} \in (\theta_{qi} \circlearrowleft \theta_{iq})$. Thus $\frac{3\pi}{2} > \theta_{iq} > \theta_{qi} > \frac{\pi}{2}$, and we can continue as in the previous case to reach a contradiction.

Observe that D_q contributes upper hull bitangents to $\mathcal{B}_q$ if and only if consecutive bitangents (B_{ij}, B_{jk}) exist in $\mathcal{B}_{q-1}$, where $\theta_{jq} \in (\theta_{ij} \circlearrowleft \theta_{jk})$ (i and k need not be distinct). As we have shown, if such a pair exists, then a binary search will return B_{ij} in agreement with step 5.

Now consider the situation where D_q does not contribute upper hull bitangents to $\mathcal{B}_q$. As we mentioned earlier, regardless of what the binary search of step 4 returns, we can recognize that $\mathcal{B}_q$ remains unchanged. Since in this situation, no pair of consecutive bitangents (B_{ij}, B_{jk}) exists in $\mathcal{B}_{q-1}$ that satisfy $\theta_{jq} \in (\theta_{ij} \circlearrowleft \theta_{jk})$, one of three *failure events* must occur: the search will (i) fail to return a bitangent, (ii) return a bitangent with no successor (i.e., the last bitangent in the list), or (iii) return a bitangent B_{ij} with successor B_{jk} where $\theta_{jq} \notin (\theta_{ij} \circlearrowleft \theta_{jk})$. Since these are exactly the tests performed in step 5, if a failure event occurs, the algorithm continues with $\mathcal{B}_q = \mathcal{B}_{q-1}$. Otherwise, steps 6 and 7 convert the bitangent sequence, from $\mathcal{B}_{q-1}$ to $\mathcal{B}_q$, in accordance with Lemma 3.10.

Let us now determine the algorithm's running time. Step 1 requires $O(n \log n)$

time to determine the east and west discs, and sort the remaining discs by size. Step 2 can be done in constant time. To allow $O(\log n)$ insertions, deletions, searches, and traversals, we store the hull bitangents $\mathcal{B}$ in a balanced binary tree. As a consequence, each iteration of the remaining steps of the algorithm can be performed in $O((1+k)\log q)$ time, where k is the number of bitangents removed[3] from $\mathcal{B}_q$. Since a bitangent can be removed only once, and at most two bitangents are added in each of the $O(n)$ iterations, the total k over all iterations is $O(n)$. Hence, the running time of the algorithm is $O(n \log n)$.

One complication arises if the discs are not partially disjoint (i.e., some discs are contained by others). The key observation to make in this setting is that no disc that contains another can contribute to the discs' guaranteed hull:

Lemma 3.11 *If $D_i, D_j \in \mathcal{D}$, and CH(D_i) contains CH(D_j), then D_i cannot contribute to $GH(\mathcal{D})$.*

Proof. If CH(D_i) contains CH(D_j), then by our general position assumption, every line right-tangent to D_i is invalidated by D_j, and hence cannot be a hull tangent. The rest of the proof follows from Theorem 3.1. □

Suppose that in some iteration of the algorithm, D_q contains some other disc D_k of $\mathcal{D}$. Since the discs are processed in order of increasing size, and the east and west discs never contain other discs, k must be less than q; it follows that D_q cannot contribute upper hull bitangents to $\mathcal{B}_q$. When processing D_q, we thus expect our algorithm to encounter one of the three failure events described earlier. In fact, since no bitangent B_{kq} or B_{qk} exists, the algorithm may encounter a fourth type of failure event: an attempt to construct a bitangent that doesn't exist. This additional test can be included in step 5.

Theorem 3.12 *The above algorithm can be used to generate the guaranteed hull of n aligned similar uncertain discs in $O(n \log n)$ time.*

Proof. As we have shown, the upper and lower guaranteed hulls can be found in $O(n \log n)$ time (note that the sorted disc sequence $(D_e, D_w, \ldots, D_n)$ need be

[3] If a suitable tree variant is used (e.g., [73]), each step can be done in $O(\log q + k)$ time. This will not change the asymptotic running time of the algorithm, however.

constructed only once). By Theorem 3.1, GH is equal to the intersection of the left half-planes of the upper and lower bitangents. Finding such an intersection is a basic problem of computational geometry, and an optimal $O(n \log n)$ algorithm was given by Preparata and Shamos [59]. In our case, since the half-planes are presorted by their polar angles, it can be done in linear time. □

3.4.2 Algorithm 2

In this section, we show how Chan's output-sensitive convex hull algorithm ([15], Section 2.2) can be adapted to generate guaranteed hulls of aligned similar uncertain regions. Its running time is $O(n \log h)$, where n is the number of aligned similar uncertain regions, and h is the number of hull bitangents they induce. We are perhaps cheating a little, by considering the algorithm's output to be the hull bitangents, and not the guaranteed hull that results from the intersection of their left half-planes. The difference between the complexities of these two types of output can be dramatic, as Figure 3.9 showed. In any case, the running time is an improvement over that of our first algorithm.

Interestingly, Chan's algorithm can also be used to generate the envelopes of functions in an output-sensitive manner ([15], Section 2.3). Since the hull bitangents of aligned similar uncertain regions can be represented as the vertices of an envelope of functions (by using the techniques of [55], Section 5.2, and [54], Section 4.2), this use of Chan's algorithm will also construct guaranteed hulls in $O(n \log h)$ running time.[4] In this section, we show how Chan's algorithm can generate guaranteed hulls directly, in a more explicitly geometric fashion.

Chan's algorithm incorporates the Jarvis march algorithm with a novel grouping technique to construct the convex hull of a set of n points. In the first, or *grouping* step, the points are divided into $\lceil n/m \rceil$ groups of at most m points each. The convex hull of each group is calculated, using any $O(m \log m)$ convex hull algorithm (e.g., Graham's scan [29]). The second, or *wrapping* step[5], is a variant of a Jarvis march. The hull vertices are generated in ccw order, by querying each group

[4] Another output-sensitive envelope algorithm that can be used for this purpose is presented by Nielsen and Yvinec [56].

[5] The term 'wrapping' is chosen due to the step's similarity to the *gift wrapping* algorithm, which in the planar case, is a Jarvis march.

for a candidate next vertex, then determining the actual next vertex from these candidates. These queries can be performed in constant time for each group (plus a term linear in the total size of the groups that is amortized over all the queries);[6] hence the running time of the wrapping step is $O(h \cdot \frac{n}{m} + n)$.

If h were known in advance, then setting m to h would result in an $O(n \log h)$ algorithm. Since this is not the case, the algorithm is repeated with the following sequence of 'guesses' at h: $(2^{2^1}, 2^{2^2}, \ldots, n)$. If the number of hull vertices generated for a particular guess does not exceed the guess itself, then clearly every hull vertex has been found, and the iterations can stop. It can be shown that the total running time of all of these iterations, and hence the total algorithm, is $O(n \log h)$.

To describe how the algorithm can be modified for our purposes, we require some additional terminology. As before, we will assume that the regions are discs. Let $\mathcal{H}_\theta$ represent the ordered sequence of upper hull bitangents of $\mathcal{D}$ whose polar angles do not exceed θ. Let $\{G_1, \ldots, G_m\}$ represent the partitioning of $\mathcal{D}$ into groups of $\lceil n/m \rceil$ discs. Each of these groups has a sequence of upper hull bitangents $(B_{\bullet e}, \ldots, B_{w\bullet})$, ordered by polar angle. Note that each group has its own east and west discs, and each sequence starts and ends with their corresponding vertical hull tangents. It will be useful to think of a group's sequence of bitangents as a *stack*, and define popping this stack as equivalent to removing its first (i.e., topmost) bitangent.

The major work of the grouping step is constructing the guaranteed hulls of each group. Since we can use our first algorithm (Section 3.4.1) for this task, we will focus on adapting Chan's wrapping step to our setting. We will start with $\mathcal{H}_\theta = \mathcal{H}_{\frac{\pi}{2}}$, and extend it by one bitangent at a time, until it is equal to $\mathcal{H}_{\frac{3\pi}{2}}$. We can construct $\mathcal{H}_{\frac{\pi}{2}}$ by examining each group's stack and retaining the rightmost bitangent $B_{\bullet e}$. Let us consider how to perform the remaining iterations efficiently.

If, at the start of a wrapping step, the previous bitangent has the form B_{*q}, then the next bitangent in the sequence will (by Lemma 3.2) be of the form B_{qv}. We will show that not only will some group's stack contain a bitangent of the form B_{v*}, but that if B_{v*} is not the topmost element of its stack, then we can recognize this fact and permanently remove those elements above B_{v*}. What is more, we can

[6] The version of Chan's algorithm that we discuss here incorporates one of his suggested improvements ('Idea 5'; [15], p. 29).

do this without knowing the actual value of v.

To elaborate, at the start of each wrapping step, we process each group's stack. If it is nonempty, we pop bitangents from it while they satisfy at least one of the three conditions of the following lemma:

Lemma 3.13 *Let B_{*q} be the last bitangent added to $\mathcal{H}_\theta$, G_i be any group, and B_{xy} be the topmost bitangent on G_i's stack (with $x \neq q$). If (i) D_x contains D_q, (ii) $\theta_{qx} \leq \theta_{*q}$, or (iii) $\theta_{qx} \geq \theta_{xy}$, then either D_x does not contribute any upper hull bitangents of $\mathcal{D}$ with polar angle greater than θ_{*q}, or some bitangent B_{xz} exists lower in G_i's stack.*

Proof. Let B_{*q}, G_i, and B_{xy} be as described. If D_x contains D_q, then (by Lemma 3.11) D_x does not contribute any hull bitangents, and we are done.

If $\theta_{qx} \leq \theta_{*q}$, then $\theta_{qx} < \theta_{*q}$, and (by Lemma 3.4) $\theta_{*q} \in [\theta_{xq} \circlearrowleft \theta_{qx})$; hence $\theta_{*q} \geq \theta_{xq} > \theta_{qx}$. Now observe that by Lemma 3.4, no hull bitangent B_{x*} of $\mathcal{D}$ can exist with polar angle greater than θ_{*q}, and we are done.

Suppose $\theta_{qx} \geq \theta_{xy}$. If no hull bitangent B_{x*} of $\mathcal{D}$ exists where $\theta_{x*} > \theta_{*q}$, we are done. Otherwise, B_{x*} must be a hull tangent of G_i (since $D_x \in G_i$). Lemma 3.2 then implies that there exists some hull bitangent B_{xz} of G_i, where $\theta_{xz} \geq \theta_{x*}$. Lemma 3.9 now implies that $\theta_{xz} > \theta_{qx} > \theta_{*q}$, and since $\theta_{xy} \leq \theta_{qx}$, B_{xz} must be a distinct bitangent appearing below B_{xy} in G_i's stack, completing the proof. □

We now show that after each group's stack has been processed in this way, the next hull bitangent of $\mathcal{D}$ can be deduced by examining the topmost elements of the remaining nonempty stacks. Suppose we are at start of an iteration of the wrapping step, that B_{*q} was the last hull bitangent added to $\mathcal{H}_\theta$, and that the next hull bitangent to be added is B_{qv}. For the moment, assume D_q and D_v belong to different groups. We will first demonstrate that some group's stack contains a bitangent of the form B_{vw}, where $\theta_{vw} > \theta_{qv}$.

Observe that since D_v supports a hull bitangent of $\mathcal{D}$, it must support a hull tangent of the group G_i containing D_v. Hence, by Lemma 3.2, G_i's stack must have originally contained some B_{vw} where $\theta_{vw} > \theta_{qv}$. We need only show that B_{vw} is still in this stack. If this is not the case, then B_{vw} must have been popped

during some previous iteration of the wrapping step. Let $B_{*q'}$ be the hull bitangent added to $\mathcal{H}_\theta$ just prior to B_{vw} being popped. By Lemma 3.13, $q' \neq v$, and one of these three conditions must hold: (i) D_v contains $D_{q'}$, (ii) $\theta_{q'v} \leq \theta_{*q'}$, or (iii) $\theta_{q'v} \geq \theta_{vw}$. Now, (i) must be false, otherwise D_v could not support a hull tangent of $\mathcal{D}$. Also, hull bitangents $B_{*q'}$ and B_{qv} are supported by discs $D_{q'}$ and D_v respectively, so by Lemma 3.9, $\theta_{qv} \geq \theta_{q'v} > \theta_{*q'}$; and since $\theta_{vw} > \theta_{qv}$, both (ii) and (iii) must be false.

Now that we have shown that a bitangent of the form B_{vw} remains in G_i's stack, we will show that it is this stack's topmost element (or at least, the topmost element is of this form as well). Since B_{qv} is a hull bitangent of $\mathcal{D}$, it must also be a pure hull tangent of G_i. Lemma 3.2 implies that G_i's stack originally contained some bitangent B_{v*}, where $\theta_{v*} > \theta_{qv}$; and as we have shown, B_{v*} must still be in the stack.

Assume, by way of contradiction, that the stack's topmost element is B_{xy}, where $x \neq v$. Since B_{xy} was not popped in the previous iteration, $\theta_{xy} > \theta_{qx} > \theta_{*q}$. Note that D_q can support no bitangents with polar angle in the range $(\theta_{*q} \circlearrowleft \theta_{qv})$; hence $\theta_{qx} > \theta_{qv}$, which implies $\theta_{xy} > \theta_{qv}$. Let L be B_{qv}, and M the directed line parallel to L that has right-tangent D_x; see Figure 3.12. Observe that (i) D_v is the only disc of G_i right-tangent to L, (ii) M is strictly to the left of L, and (iii) $\theta(M) < \theta_{xy}$. If we now rotate L and M ccw in tandem around D_v and D_x respectively, then by the argument within the proof of Lemma 3.2, L must reach some hull bitangent B_{vz} of G_i before M reaches B_{xy}, since D_v lies strictly to the right of M (preventing it from being a hull bitangent of G_i) until L and M coincide at B_{vx}. Hence, $\theta_{xy} > \theta_{vz} > \theta_{qv} > \theta_{*q}$. As we showed above, this implies that B_{vz} must still be on G_i's stack, and B_{xy} is not its topmost element (a contradiction).

Now consider the case where D_q and D_v both belong to G_i. Since B_{*q} was the last hull bitangent generated, G_i's stack must have a topmost element of the form B_{qp}. Suppose, by way of contradiction, that $p \neq v$. Since D_q was chosen to support the last hull bitangent, B_{qp} must not have been popped per Lemma 3.13. It follows that $\theta_{qp} > \theta_{*q}$. Note also that since B_{*q} and B_{qv} are consecutive hull bitangents of $\mathcal{D}$, D_q can support no bitangent with polar angle within $(\theta_{*q} \circlearrowleft \theta_{qv})$. Hence, $\theta_{qp} > \theta_{qv}$. In addition, since $\{D_q, D_v\} \subseteq G_i$, D_p can support no hull bitangent of G_i with polar angle in $(\theta_{*q} \circlearrowleft \theta_{v*})$; thus $\theta_{qp} > \theta_{v*}$, and we have a

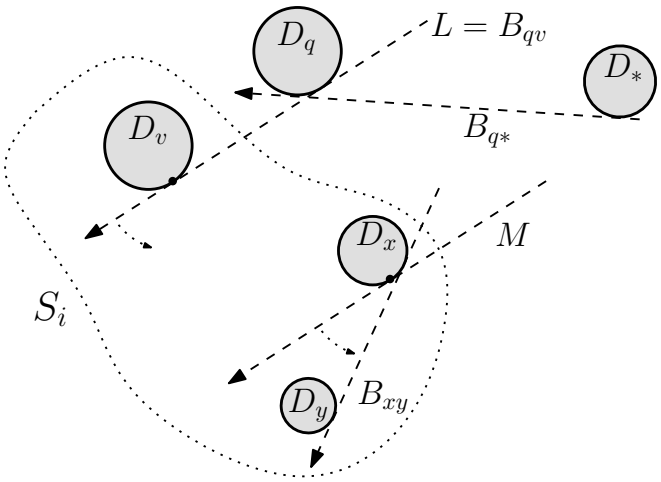

Figure 3.12: B_{xy} cannot be the stack's topmost element.

contradiction (since this implies that θ_{qp} is not the topmost element).

To conclude, we have shown that after popping each group's stack in accordance with Lemma 3.13, the topmost element of D_v's group's stack will be either B_{qv} or B_{v*}. We can thus examine the topmost element of each (nonempty) stack and generate a candidate bitangent B_{qx}. The candidate with the smallest polar angle must be B_{qv}, the next hull bitangent in $\mathcal{H}_\theta$. Each iteration of the wrapping step finds the next hull bitangent in time linear in the number of groups, plus the time spent popping each group's stack per Lemma 3.13. Since a bitangent can be popped only once, the total time spent in the wrapping step is linear in n. It follows that the running time of the complete algorithm is $O(n \log h)$.

3.4.3 Guaranteed Hulls in $\mathbb{R}^3$

While the focus of this thesis is on planar regions of imprecision, it is worth describing how some planar guaranteed hull algorithms can be adapted to construct guaranteed hulls in 3-dimensional space ($\mathbb{R}^3$). In particular, we will provide algorithms for constructing guaranteed hulls for uncertain regions that are (i) translations of a common base shape, (ii) spheres (i.e., the $\mathbb{R}^3$-analogs of discs), and (iii) axis-aligned boxes (the $\mathbb{R}^3$-analogs of rectangles).

To work with guaranteed hulls in $\mathbb{R}^3$, we need some terminology. A plane P

in $\mathbb{R}^3$ can be specified by a point and a normal vector. We will refer to the closed half-space bounded by P that does not contain the normal vector as the *lower* half-space of P, and denote it by P^-. If a lower half-space P^- intersects every region in a set $\mathcal{D}$, and three of these regions do not penetrate P^-, then we will say that P is a *hull plane* of $\mathcal{D}$, and we will say that each non-penetrating region is *upper-tangent* to P (we can also say that P is upper-tangent to these regions). Hull planes are thus $\mathbb{R}^3$-analogs of hull bitangents. They also have a fundamental relationship to guaranteed hulls:

Theorem 3.14 *If $GH(\mathcal{D})$ has nonzero volume, then $GH(\mathcal{D})$ is the intersection of the lower half-spaces of the hull planes of $\mathcal{D}$.*

Proof. We will assume GH has nonzero volume. Let $\mathcal{P}'$ be the set of planes whose lower half-spaces intersect every region of $\mathcal{D}$. We start by proving that $GH = I(\mathcal{P}')$ (where $I(\mathcal{P}')$ denotes the intersection of the lower half-spaces of $\mathcal{P}'$). If some point $p \in GH$ is not within $I(\mathcal{P}')$, then p lies strictly above some plane $P \in \mathcal{P}'$. Since every region of $\mathcal{D}$ intersects P^-, there exists a feasible set S of $\mathcal{D}$ that lies below P. This is a contradiction, since it implies that $p \notin \text{CH}(S)$. Hence, $p \in I(\mathcal{P}')$, and $GH \subseteq I(\mathcal{P}')$. To show that $I(\mathcal{P}') \subseteq GH$, let p be any point in $I(\mathcal{P}')$, and suppose $p \notin GH$. Then for some feasible set S, $p \notin \text{CH}(S)$, which implies that S lies strictly below some plane Q containing p. Let P be the plane of $\mathcal{P}'$ that is parallel to Q and whose lower half-space contains no other plane of $\mathcal{P}'$. Observe that some $D_i \in \mathcal{D}$ is upper-tangent to P. Since $p \in I(\mathcal{P}')$, P must lie on or above Q; but then S lies strictly below P, a contradiction (since no point of D_i lies below P).

We complete the proof by showing that $I(\mathcal{P}') = I(\mathcal{P})$. We will do this by demonstrating that every boundary point s of $I(\mathcal{P}')$ lies on a plane of $\mathcal{P}$. It then follows that $\mathcal{P}$ is nonempty, and (since $I(\mathcal{P}')$ and $I(\mathcal{P})$ are convex, and $I(\mathcal{P}') \subseteq I(\mathcal{P})$) $I(\mathcal{P}) = I(\mathcal{P}') = GH$.

Let s be any boundary point of GH, and q any point interior to GH. There must exist a plane of $\mathcal{P}'$ that contains s; let P be the closest such plane to q, and $t(P)$ be the number of regions of $\mathcal{D}$ upper-tangent to P. If $t(P) = 3$, then P is a hull plane, and we are done. Otherwise, P contains a line L through each of the (at most two) tangency points of P. Now observe that since $q \notin L$, we can pivot P

around L to reach a new plane of $\mathcal{P}'$ that is closer to q, or whose lower half-space excludes s. Since both cases imply a contradiction, we are done. □

As we observed in Section 3.3, for uncertain regions that consist of translated versions of a common shape, we can generate their hull bitangents by constructing the convex hull of a set of similar points drawn from the regions. The same approach applies in $\mathbb{R}^3$. The convex hull of a set of similar points is a polyhedron, and can be found in $O(n \log n)$ time. The hull planes of the original regions can be found by first constructing the planes containing the faces of this polyhedron, then lowering each of these planes by an appropriate distance. For regions such as spheres and polyhedra with a small number of faces, this poses no challenge. The intersection of the hull planes derived from the convex hull can be found in an additional $O(n \log n)$ time [59]. The running time of the complete algorithm is thus optimal.

The second $\mathbb{R}^3$ guaranteed hull algorithm works with uncertain spheres of varying sizes. We will first describe the algorithm of Rappaport ([61], Section 3) for constructing the guaranteed hull of uncertain discs, then show how it can be adapted to $\mathbb{R}^3$. The algorithm applies a simple transformation to each disc, then uses any standard algorithm to construct the convex hull of the transformed discs. Next, it transforms each convex hull edge to a hull bitangent of the original discs.

Let $r_{\max}$ be the radius of the largest disc in $\mathcal{D}$, a set of uncertain discs. A transformed set of discs is constructed, $\mathcal{D}' = \{D'_1, \ldots, D'_n\}$, by replacing each disc's radius, r_i, with the transformed radius $r'_i = r_{\max} - r_i$. It is easy to show that the hull bitangents of the original discs correspond to the boundary segments[7] of the convex hull of these transformed discs.

This algorithm extends directly to $\mathbb{R}^3$. We transform the original spheres, construct their convex hull, lower each plane that contains a face of this hull by distance $r_{\max}$, and output each lowered plane as a hull plane. By Theorem 3.14, the intersection of the half-spaces of these hull planes yields the guaranteed hull of the original spheres.

The convex hull of a set of spheres can be constructed in $O(n^2)$ time [11].

[7] The other components of the hull's boundary, arcs on the boundaries of the transformed discs, are ignored.

While the boundary of this hull is composed of planar facets, conical facets, and spherical facets [11], we are interested only in the planar facets. The running time of the complete algorithm is thus $O(n^2 + m \log m)$, where m, which never exceeds n, is the number of planar facets.

Our third $\mathbb{R}^3$ algorithm, which constructs the guaranteed hull of uncertain axis-aligned boxes (with varying sizes in each dimension), is an adaptation of Nagai *et al*'s algorithm for guaranteed hulls of uncertain axis-aligned rectangles [55]. Their algorithm is based on the observation that any hull bitangent of a set of axis-aligned rectangles will be tangent at points that correspond to the same corners of two of the rectangles. The algorithm calculates the convex hulls of four sets of points, where each set consists of a particular corner drawn from each rectangle (e.g., the upper left corners), then constructs the guaranteed hull as the intersection of these four convex hulls. Converting this algorithm to $\mathbb{R}^3$ is simply a matter of constructing the intersection of eight convex hulls, corresponding to the eight corners of the axis-aligned boxes. The eight convex hulls can be constructed in $O(n \log n)$ time, and the intersection of the lower half-spaces corresponding to these convex hulls' faces can be constructed in $O(n \log n)$ time as well, yielding an optimal algorithm.

3.5 Possible Hull

The possible hull of a set[8] of uncertain regions $\mathcal{D}$ is defined formally as the set of points p such that for some feasible set S of $\mathcal{D}$, $p \in \text{CH}(S)$. We will denote the possible hull of $\mathcal{D}$ by $PH(\mathcal{D})$ (or, when $\mathcal{D}$ is clear from the context, simply by PH).

One use for possible hulls is in identifying which regions of $\mathcal{D}$ are *guaranteed extreme regions*: regions that contribute a vertex to the convex hull of every feasible set of $\mathcal{D}$.

Theorem 3.15 *If $\mathcal{D}$ is a set of at least three uncertain regions, then $D_i \in \mathcal{D}$ is a guaranteed extreme region of $\mathcal{D}$ iff it does not intersect the possible hull of $\mathcal{D} \setminus \{D_i\}$.*

Proof. Let D_i be a region of $\mathcal{D}$, where $|\mathcal{D}| \geq 3$. If D_i does not intersect $PH(\mathcal{D} \setminus$

[8] Throughout this section, we will assume that sets are multisets.

$\{D_i\}$), then D_i intersects the convex hull of no feasible set of $\mathcal{D} \setminus \{D_i\}$. Since this implies that D_i is linearly separable from every such feasible set, D_i must be a guaranteed extreme region of $\mathcal{D}$.

Suppose D_i intersects $PH(\mathcal{D} \setminus \{D_i\})$. If there exists a feasible set S of $\mathcal{D} \setminus \{D_i\}$ such that D_i intersects CH(S) at a point p that is not a vertex of CH(S), then p is not a vertex of CH($\{p\} \cup S$), D_i is not a guaranteed extreme region of $\mathcal{D}$, and we are done. If no such S exists, then D_i must intersect a vertex p of the convex hull of some feasible set S' of $\mathcal{D} \setminus \{D_i\}$. Let D_j be the region of $\mathcal{D} \setminus \{D_i\}$ contributing vertex p to CH(S'). Now, D_i must penetrate D_j, otherwise (since $p \in D_i \cap D_j$) D_i and D_j violate the regions' valid intersection property. It follows that some point $q \in D_i$ lies in the interior of a line segment connecting D_j and some third region of $\mathcal{D}$. Since q is not a vertex of the convex hull of any feasible set which includes this segment's endpoints, D_i is not a guaranteed extreme region of $\mathcal{D}$. □

Nagai *et al* [55] have observed that for convex uncertain regions, the convex hull of the regions and their possible hull are the same. Hence, the guaranteed extreme regions of a set of convex uncertain regions are exactly those regions that lie outside the convex hull of the other regions in the set. In fact, it can be shown that in this case, a region is a guaranteed extreme region iff it lies outside the convex hull of its Voronoi neighbors. This implies that for many natural types of uncertain regions, such as discs and convex polygons with a small constant number of vertices, the guaranteed extreme regions can be identified in $O(n \log n)$ time.

In this section, we develop a sequence of possible hull algorithms. Each algorithm in the sequence will use its predecessor as a subroutine, and the final algorithm will construct the possible hull of a set of (possibly nonconvex) polygons. We begin by stating some properties of possible hulls.

Lemma 3.16 $PH(\{A\}) = A$.

Lemma 3.17 $PH(\{A, B\}) = \bigcup_{a \in A, b \in B} \overline{ab}$.

Lemma 3.18 $\bigcup_{D_i \in \mathcal{D}} D_i \subseteq PH(\mathcal{D})$.

Lemma 3.19 *If $\mathcal{A}$ and $\mathcal{B}$ are nonempty sets of uncertain regions, then* $PH(\mathcal{A} \cup \mathcal{B}) = PH(\{PH(\mathcal{A}), PH(\mathcal{B})\})$.

Proof. Let $Q = PH(\{PH(\mathcal{A}), PH(\mathcal{B})\})$. Suppose p is a point within $PH(\mathcal{A} \cup \mathcal{B})$. Then there exists a feasible set S of $\mathcal{A} \cup \mathcal{B}$ such that $p \in \text{CH}(S)$. Let S_a and S_b be the subsets of S corresponding to the subsets $\mathcal{A}$ and $\mathcal{B}$. By using the definition of convex hull, it is easy to show that (i) $\text{CH}(S) = \text{CH}(\text{CH}(S_a) \cup \text{CH}(S_b))$, and (ii) there exist points a and b within $\text{CH}(S_a) \cup \text{CH}(S_b)$ such that $p \in \overline{ab}$. Now, without loss of generality, either (i) $a, b \in \text{CH}(S_a)$, or (ii) $a \in \text{CH}(S_a)$ and $b \in \text{CH}(S_b)$. If (i), then $\overline{ab} \subseteq \text{CH}(S_a)$, and since (by definition) $\text{CH}(S_a) \subseteq PH(\mathcal{A})$, $\overline{ab} \subseteq PH(\mathcal{A})$, which implies (by Lemma 3.18) that $\overline{ab} \subseteq Q$. If (ii), then since $\text{CH}(S_a) \subseteq PH(\mathcal{A})$ and $\text{CH}(S_b) \subseteq PH(\mathcal{B})$, Lemma 3.17 implies $\overline{ab} \in Q$. Hence $PH(\mathcal{A} \cup \mathcal{B}) \subseteq Q$.

If p is a point in Q, then by Lemma 3.17, $p \in \overline{ab}$, where $a \in PH(\mathcal{A})$ and $b \in PH(\mathcal{B})$. There must then exist feasible sets S_a of $\mathcal{A}$ and S_b of $\mathcal{B}$ where $a \in \text{CH}(S_a)$ and $b \in \text{CH}(S_b)$. Note that $\overline{ab}$ is within $\text{CH}(S_a \cup S_b)$, and since $S_a \cup S_b$ is a feasible set of $\mathcal{A} \cup \mathcal{B}$, $\text{CH}(S_a \cup S_b)$ is within $PH(\mathcal{A} \cup \mathcal{B})$; hence $Q \subseteq PH(\mathcal{A} \cup \mathcal{B})$. □

Lemma 3.20 *The possible hull of any set of two or more connected uncertain regions is simply connected.*

Proof. Let $\mathcal{D} = \{A, B\}$ be a set of connected uncertain regions (Lemma 3.19 implies that the proof extends by induction for sets of more than two regions). By Lemma 3.17, every point of PH is connected within PH to both A and B; and by Lemma 3.18, both A and B lie within PH. Hence PH is connected, and we need only show that it has no holes. We will do this by showing that every exterior point of PH is the source of a ray exterior to PH.

Let q be any point exterior to PH. We first make the observation that if no lines through q that are tangent to A exist, then (i) every line through q will intersect A, and (ii) at least one line through q will intersect A to both sides of q. Since the same argument applies to B, if no such lines exist for A or B, then some segment $\overline{ab}$ exists (where $a \in A$ and $b \in B$) that contains q, implying that $q \in PH$, a contradiction. Hence, without loss of generality we can assume that there exist

directed lines L_1 and L_2 through q that are right-tangent to A. Let W_1 (resp., W_2) be the wedge lying on or to the right (resp., left) of both L_1 and L_2. Note that W_1 contains A. Note also that W_2 cannot intersect B, otherwise some segment $\overline{ab}$ exists that contains q. Let R be any ray from q lying in W_2. Observe that no point $r \in R$ can lie on a segment $\overline{ab}$, since for any choice of $a \in A$, the portion of ray $\overrightarrow{ar}$ lying at or beyond r lies within W_2 (Figure 3.13). Hence, by Lemma 3.17, R does not intersect PH. □

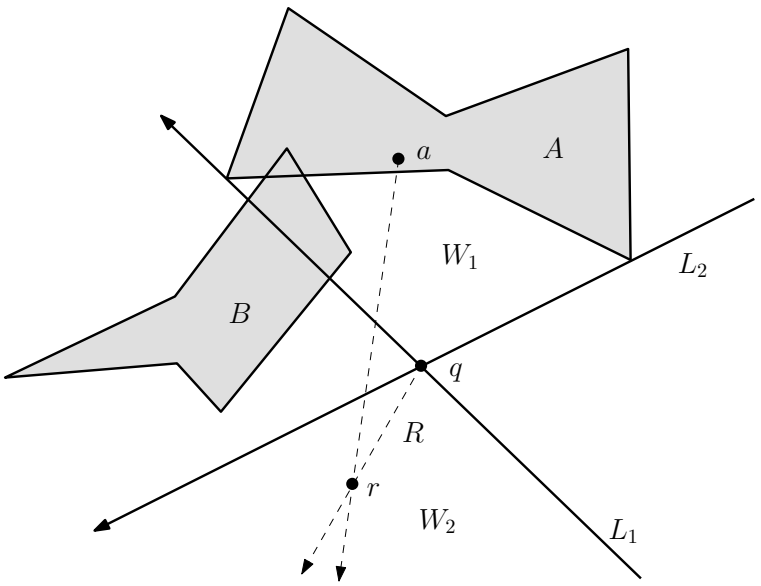

Figure 3.13: Lemma 3.20

Corollary 3.21 *Let $(p_1, \ldots, p_k, p_1)$ be a cyclic sequence of points. If each consecutive pair (p_i, p_{i+1}) are endpoints of a segment known to lie within PH, and P is a simple polygon whose edges lie on these segments, then the interior of P lies within PH.*

A region R is *star-shaped* if there exists a point $s \in R$ such that

$$\forall_{r \in R}\, \overline{sr} \subseteq R.$$

The *kernel* of a region R is the set of all such points s that have this property.

Hence, a region is star-shaped if it has a nonempty kernel.

Theorem 3.22 *The possible hull of any set of two or more connected uncertain regions is star-shaped.*

Proof. Let $\mathcal{D} = \{A, B\}$ be a set of two connected uncertain regions (Lemma 3.19 can be applied to prove the claim for sets of more than two regions). We will prove that PH is star-shaped by showing that it has a nonempty kernel.

Let $I = \text{CH}(A) \cap \text{CH}(B)$. If $I \neq \emptyset$, then let s be any point in I. Observe that there must exist points $a_1, a_2 \in A$, and $b_1, b_2 \in B$ where s lies on both $\overline{a_1 a_2}$ and $\overline{b_1 b_2}$. Let q be any point in PH. By Lemma 3.17, there exist points $a' \in A$, $b' \in B$ such that $q \in \overline{a'b'}$. Without loss of generality we can assume that a_1, b_1, and a' are on or to the left of $\overrightarrow{sq}$, and that a_2, b_2, and b' are on or to the right of $\overrightarrow{sq}$. There are now two cases (Figure 3.14): s, b_1, and a_2 are either to the left or to the right of $\overrightarrow{a_1 b_2}$. In both cases, we can construct a cyclic sequence of points defining a polygon (per Corollary 3.21) that lies within PH, and which contains $\overline{sq}$. (In the former case, the sequence is $(a_1, b_2, a_2, b', a', b_1)$; and in the latter case, it is (b_1, a_2, b', a').) Since this holds for any $q \in PH$, s is in the kernel of PH.

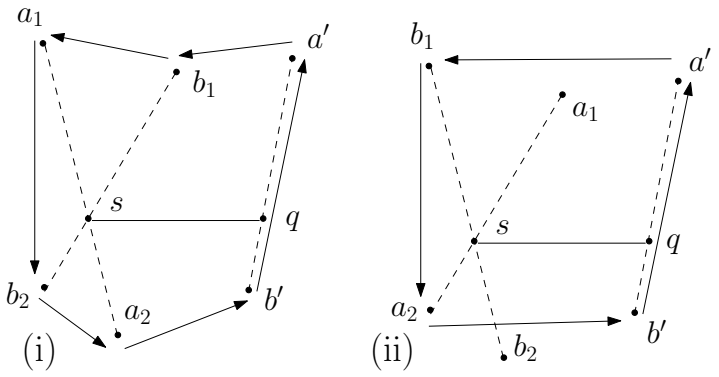

Figure 3.14: Theorem 3.22

If $I = \emptyset$, then there exists line L_1 right-tangent to A and left-tangent to B, and line L_2 left-tangent to A and right-tangent to B. Let s be the point where L_1 and L_2 cross, and q be any point in PH. By Lemma 3.17, $q \in \overline{ab}$, for some $a \in A$, $b \in B$. Note that there exist points $a' \in A$ and $b' \in B$ such that $s \in \overline{a'b}$ and $s \in \overline{ab'}$. We

can again construct a sequence of points that defines a polygon that lies within PH and contains $\overline{sq}$. If s lies to the right of $\overline{ab}$, this sequence is (b, a, b', a'); otherwise, it is (a, b, a', b'). □

From this point on, we will assume that each of the uncertain regions is a polygon (or a point, which can be viewed as a degenerate polygon). The possible hull of one such set is shown in Figure 3.15. Note that as a consequence of our general position assumption, no three vertices lie on the same line, which implies that any pair of polygon edges will intersect only at their endpoints (if the edges belong to the same polygon) or at a single interior point (if the edges belong to distinct polygons).

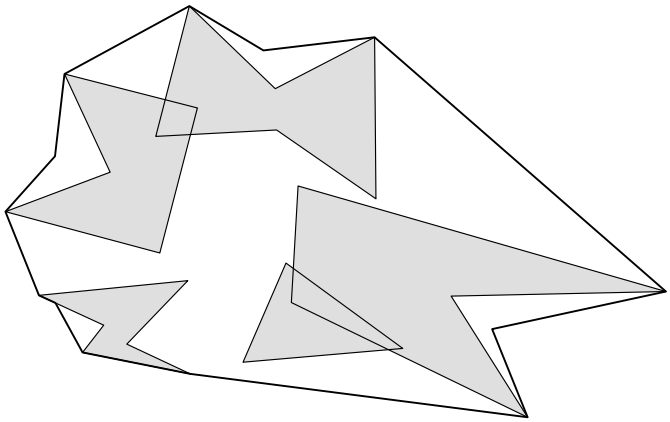

Figure 3.15: Possible hull of uncertain polygons

Before proving a bound on the complexity of possible hulls of polygons, we will require some terminology. We will refer to an edge of a polygon of $\mathcal{D}$ as a *native segment*, and to a segment connecting vertices of distinct polygons of $\mathcal{D}$ as a *bridge segment*.

Lemma 3.23 *Each point on the boundary of PH lies on either a native segment or a bridge segment of $\mathcal{D}$.*

Proof. Let q be a point on the boundary of PH. Note that q must lie on the boundary of some feasible hull C of $\mathcal{D}$, and cannot lie in the interior of any feasible

hull of $\mathcal{D}$. Suppose, by way of contradiction, that q does not lie on a native or bridge segment of $\mathcal{D}$.

If q is a vertex of C, then q is some feasible point within some polygon $A \in \mathcal{D}$. Since q does not lie on a native segment of $\mathcal{D}$, it must lie in the interior of A. Observe that we can replace q by some nearby $q' \in A$ to create a new feasible hull with q in its interior. Since PH contains this feasible hull, q cannot lie on the boundary of PH, a contradiction.

Point q must therefore lie in the interior of some segment $\overline{ab}$, where a and b are vertices of C, and a and b lie within distinct polygons A and B of $\mathcal{D}$. We can assume that a (resp., b) is the farthest point of A (resp., B) that intersects the line containing $\overline{ab}$. Note that this implies that a lies on some edge $\overline{a_1 a_2}$ of A, and that b lies on some edge $\overline{b_1 b_2}$ of B. Since q is not on a bridge segment, we can assume (without loss of generality) that a is not a vertex of A, and that neither a_1 nor a_2 are collinear with $\overline{ab}$. Now observe that the sequence (a_1, a_2, b) defines a polygon (per Corollary 3.21) within PH whose interior contains q, a contradiction. □

Theorem 3.22 and Lemma 3.23 imply that PH is a star-shaped polygon. As we now show, it has no more vertices than $\mathcal{D}$.

Theorem 3.24 *If $\mathcal{D}$ is a set of uncertain polygons with a total of n vertices, then the boundary of $PH(\mathcal{D})$ has at most n vertices.*

Proof. Let $\mathcal{D}$ be a set of uncertain polygons. We will show that each vertex v on the boundary of PH that is not already a vertex of $\mathcal{D}$ can be associated with a subset of the boundary of one of the polygons A of $\mathcal{D}$ that (i) is disjoint from the subset associated with any other vertex of PH, and (ii) contains a vertex of A (in the interior of PH) to which we can charge v.

If v is a convex vertex, then since the native and bridge segments of $\mathcal{D}$ lie within PH (Corollary 3.21) and contain its boundary points (Lemma 3.23), v must be a vertex of $\mathcal{D}$, and we are done. If v is a reflex vertex, we proceed as follows. Let u (resp., w) be the vertex of PH immediately preceding (resp., following) v on the boundary of PH. Let $\overline{s_1 s_2}$ (resp., $\overline{t_1 t_2}$) be the native or bridge segment containing $\overline{uv}$ (resp., $\overline{vw}$). If $\overline{s_1 s_2}$ and $\overline{t_1 t_2}$ are native segments of the same polygon of $\mathcal{D}$, then $v(= s_2 = t_1)$ must be a vertex of this polygon, and we are done. If $\overline{s_1 s_2}$ and $\overline{t_1 t_2}$

are native segments of distinct polygons of $\mathcal{D}$, then sequence (t_1, s_1, t_2, s_2) induces (by Corollary 3.21) a polygon within PH whose interior contains v, implying that v is not a vertex of PH, a contradiction. Without loss of generality, there are two remaining cases: $\overline{s_1 s_2}$ is a native segment of a polygon $A \in \mathcal{D}$, and $\overline{t_1 t_2}$ is a bridge segment; or $\overline{s_1 s_2}$ and $\overline{t_1 t_2}$ are both bridge segments.

Consider the former case. We will let $\overline{A}$ represent the set $\mathcal{D} \setminus \{A\}$. Note that no point $p \in \overline{A}$ can lie to the right of $\overline{s_1 s_2}$, otherwise (by Corollary 3.21) triangle $\triangle p s_2 s_1$ is within PH, and v is interior to PH. Hence, $t_2 \in A$, and $t_1 \in \overline{A}$. Assume, for the sake of contradiction, that $w \neq t_2$. Then the next boundary edge of PH is $\overline{wx}$, lying on some native or bridge segment $\overline{q_1 q_2}$, where q_1 is to the left of $\overline{t_1 t_2}$ and q_2 is to its right (Figure 3.16). If $q_2 \in A$, then $\triangle w t_1 q_2$ is within PH, and v is interior to PH. Thus $q_2 \in \overline{A}$, and lies to the left of $\overline{s_1 s_2}$. But now segments $\overline{wv}$, $\overline{vs_1}$, and $\overline{q_2 w}$ bound a triangle within PH whose interior contains v, and we have a contradiction. Hence, $w = t_2$. Now note that the portion of the boundary of A from v ccw to w exclusive lies in PH's interior, and includes vertex s_2 of A. We can thus charge v to s_2.

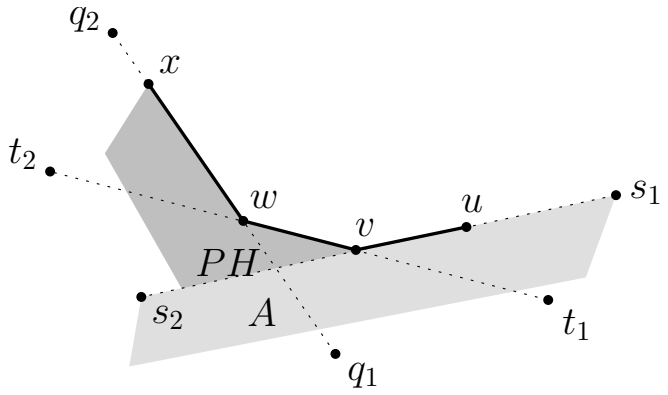

Figure 3.16: Theorem 3.24

Consider the latter case: $\overline{s_1 s_2}$ and $\overline{t_1 t_2}$ are both bridge segments, and s_1 is a vertex of a polygon $A \in \mathcal{D}$. We can use very similar arguments to that of the first case to show that $u = s_1$, and $w = t_2 \in A$. There is now a path on the boundary of A, ccw from u to w exclusive, that is interior to PH and contains some vertex of A to which we can charge v. □

It is easy to construct examples of polygons for which the bound of Theorem 3.24 is tight. See, for example, Figure 3.17.

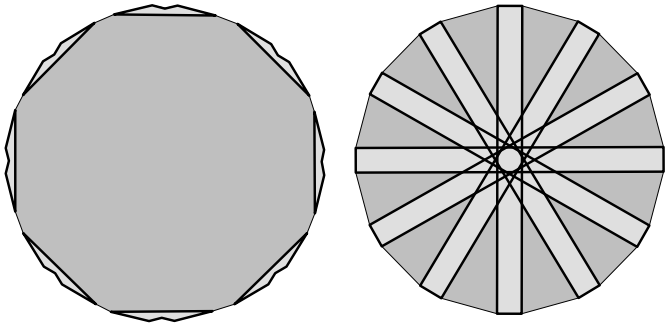

Figure 3.17: Uncertain polygons with n vertices generating PH with n vertices

3.5.1 Point and Polygon

The first of our possible hull algorithms constructs the hull of a point and a polygon. Suppose P is a (possibly nonconvex) simple polygon, and s is a point. We wish to construct $PH = PH(\{s, P\})$; see Figure 3.18. We will maintain our general position assumption that s and the vertices of P are distinct, and that no three of them are collinear.

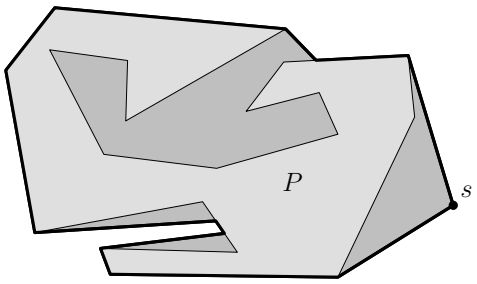

Figure 3.18: Possible hull of polygon P and point s

Our algorithm will actually construct the possible hull of a simple polygonal chain and a point, which is sufficient for two reasons: by Lemma 3.20, the possible hull of P and s is equal to the possible hull of the boundary of P and s; and the boundary of P is a simple polygonal chain (albeit one with no endpoints).

The algorithm may be useful in other contexts as well, since the possible hull of a polygon P and a point s is the smallest polygon containing P whose kernel contains s.

Our algorithm is reminiscent of Melkman's algorithm for finding the convex hull of a simple polygonal chain [50] for the following reasons. First, both are on-line algorithms: they require no preprocessing of the input, and maintain the respective hulls (i.e., convex vs. possible) of the portion of the input seen so far. Second, both algorithms look for points where the polygonal chain enters and emerges from the interior of the hull, and rely upon the simplicity of the polygonal chain to perform this efficiently.

We motivate our algorithm with the following observation, implied by Lemma 3.17: the possible hull of a point s and a polygonal chain P is equal to a union of triangles, where each triangle's vertices are s and the endpoints of an edge of P.

Let $(p_1, \ldots, p_n)$ be the ordered vertices of P. We will denote the connected subset of P from a to b by $(a \ldots b)$. Our algorithm starts with point u initialized to p_2, and H initialized to the possible hull of s and $(p_1 \ldots p_2)$, which (as noted above) is the triangle with vertices s, p_1, and p_2. The algorithm advances u along P, processing each new edge (or part of an edge) in one of two ways. If the edge is exterior to H, then we expand H by adding its associated triangle; and if the edge is interior to H, then we skip the edge and advance u until it emerges from H. In either case, at the start of each iteration, u is a point that is on both P and the boundary of H. When u reaches p_n, H will equal $PH(\{s, P\})$.

Since H is a simple polygon, we will assume its vertices have a ccw ordering. For added flexibility when manipulating H, we will associate with it a variable orientation whose values are *ccw* or *cw*. If a and b are adjacent vertices of H, with b ccw from a, then we will consider b to follow a when H has ccw orientation, and to precede a when H has cw orientation.

Our algorithm consists of these steps:

1. Initialize H to be the triangle with vertices s, p_1, and p_2, and set u to p_2.

2. If $u = p_n$, stop.

3. Set the orientation of H to ccw (resp., cw) if s lies to the left (resp., right) of

$\overline{uv}$, where v is the vertex of P following u.[9]

4. If the points of P immediately following u lie in the interior of H, go to step 6.

5. **Expansion step.** Let $T = \triangle suv$, $\overline{uv}$'s contribution to the hull. Starting with $x = u$, advance x along H, deleting those edges that lie within T, until the first of three events occurs:

   (i) H has no more edges (Figure 3.19). Replace H with T, advance u to v, and go to step 2.

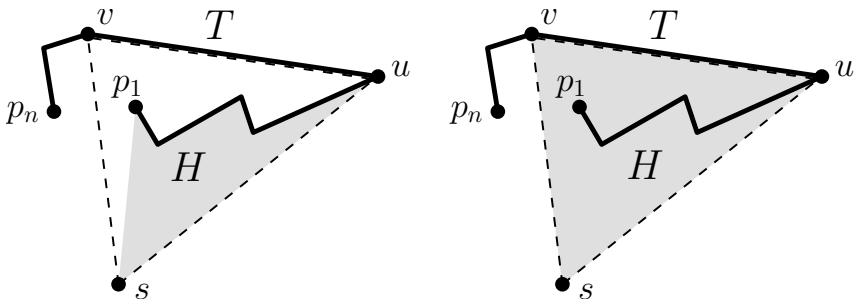

Figure 3.19: Expansion step, case (i), before and after changes to hull.

   (ii) x reaches the point where edge $\overline{ab}$ of H intersects $\overline{sv}$. Replace $\overline{ab}$ with edges $\overline{uv}$, $\overline{vx}$, and $\overline{xb}$ (Figure 3.20). Advance u to v, and go to step 2.

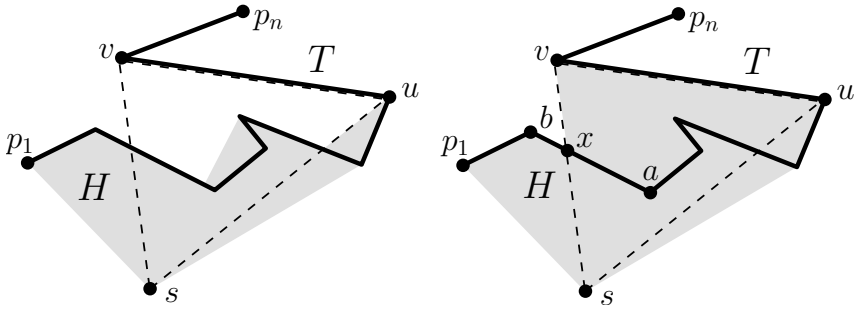

Figure 3.20: Expansion step, case (ii), before and after changes to H.

[9] In each of the figures that follow, H has ccw orientation according to this rule.

(iii) x reaches the point where edge $\overline{ab}$ of H intersects $\overline{uv}$. Replace $\overline{ab}$ with $\overline{ux}$ and $\overline{xb}$ (Figure 3.21). Advance u to x, and go to step 2.

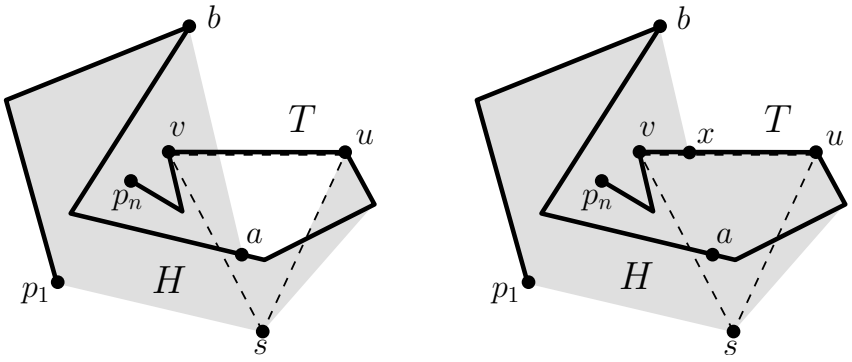

Figure 3.21: Expansion step, case (iii), before and after changes to H.

6. **Interior step.** Starting with $x = u$, advance x along P until the first of two events occurs:

   (i) x reaches p_n; stop.

   (ii) x reaches the point where P emerges from H's interior (Figure 3.22). Split $\overline{cd}$, the edge of H containing this point, into edges $\overline{cx}$ and $\overline{xd}$. Advance u to x, and go to step 2.

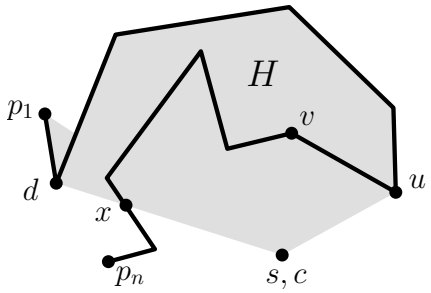

Figure 3.22: Interior step, case (ii).

We now examine the steps of the algorithm to prove that it generates the possible hull of s and P. We will use induction to show that at the start of each iteration of the algorithm, the following invariants hold:

1. H is the possible hull of s and $(p_1 \ldots u)$.

2. u lies on the boundary of H.

The invariants clearly hold for the base case, since the initial hull, H, is triangle $\triangle sp_1u$ (where $u = p_2$), and is thus equal to $PH(\{s, (p_1 \ldots u)\})$.

Suppose the invariants hold for every iteration until u reaches a particular position along P. We first consider the case where an expansion step is to be performed. If every edge of H lies within T (case i), then T is the possible hull of $(p_1 \ldots v)$, and the invariants are satisfied. Suppose instead that some edge of H does not lie within T. Consider the the boundary points of H following u. Since s lies in the kernel of H, the first such point where the boundary crosses an edge of T must lie on $\overline{sv}$ (case ii), or in the interior of $\overline{uv}$ (case iii). In the former case, modifying the boundary of H as stated has the effect of expanding H to include $\triangle suv (= T)$; and in the latter, it has the effect of expanding H to include $\triangle sux (\subset T)$. Since u is advanced to v in the former and x in the latter, we are thus adding exactly that portion of the hull contributed by those points of P between the old and new u, which satisfies invariant (1); and since the new u is not interior to H, invariant (2) is also satisfied.

Let us now consider the case where an interior step is to be performed. We first show that we can ignore the points on $(u \ldots x)$. Since H is the possible hull of s and $(p_1 \ldots u)$, and $(u \ldots x) \subset H$, we have (by Lemma 3.17):

$$\begin{aligned} PH(\{s, (p_1 \ldots x)\}) &= \{\overline{sp} \mid p \in (p_1 \ldots x)\} \\ &= \{\overline{sp} \mid p \in (p_1 \ldots u)\} \cup \{\overline{sp} \mid p \in (u \ldots x)\} \\ &= H \cup \{\overline{sp} \mid p \in (u \ldots x)\} \\ &= H \ . \end{aligned}$$

The only change we make to H is to split edge $\overline{cd}$ at x. As this does not actually change the boundary of H, and x (the new location of u) lies on this boundary, both invariants are satisfied.

We have succeeded in proving the correctness of the algorithm. Before analyzing its running time, we will disclose some additional implementation details. Since P is immutable, we can store its vertices in an array. The vertices of H are

stored in a doubly-linked list, so that inserting or removing a vertex, or accessing a vertex's neighbor, can be done in constant time. Since u occurs in both P and on the boundary of H, we maintain a pointer to its location in both data structures.

In step 4, we need to determine if the points of P immediately following u lie in the interior of H. This is true iff the vertex of H following u is to the right (or, if the hull has cw orientation, to the left) of $\overline{uv}$, and hence can be determined in constant time.

In the interior step, we must determine which edge of H contains the point x where P emerges from H's interior. To do this efficiently, we first characterize each boundary edge of H as being either a *polygonal* edge (one lying on an edge of P) or a *radial* edge (one lying on a ray from s through a vertex of P).

Lemma 3.25 *At the start of any iteration in which an interior step occurs, the following conditions hold: (i) exactly one of the edges of H incident with u is a radial edge; and (ii) if this radial edge is not incident with s, then x (the point where P emerges from H's interior) must lie on this edge as well; otherwise, x must lie on an edge incident with s.*

Proof. At the start of an interior step, the edges of H incident with u cannot both be polygonal edges, otherwise (since an interior step is about to occur) this would imply that u is incident to three edges of P, which is impossible. Suppose instead that that they are both radial edges. Note that each radial edge of H is induced by a distinct vertex of P (unless a radial edge has just been split in step 6(d); but each such step is immediately followed by an expansion step that removes one of these two edges). Hence, s and two distinct vertices of P must be collinear, contradicting our general position assumption.

To prove (ii), we first note that since P is simple, x must lie in the interior of $\overline{cd}$, a radial edge of H. Let $\overline{ab}$ be the radial edge of H incident with u. Assume by way of contradiction that $\overline{ab}$ and $\overline{cd}$ are distinct edges, and that at least one of them is not incident with s. This edge must then be adjacent to (distinct) polygonal edges y_1 and y_2 (see, for example, edge $\overline{ab}$ in Figure 3.23). Now observe that $(u \ldots x)$ partitions H into two pieces, and since $\overline{ab} \neq \overline{cd}$, y_1 and y_2 must lie on opposite sides of $(u \ldots x)$. We now have a contradiction, since the interiors of

paths $(p_1 \ldots u)$ and $(u \ldots x)$ must intersect, which implies that P is nonsimple. □

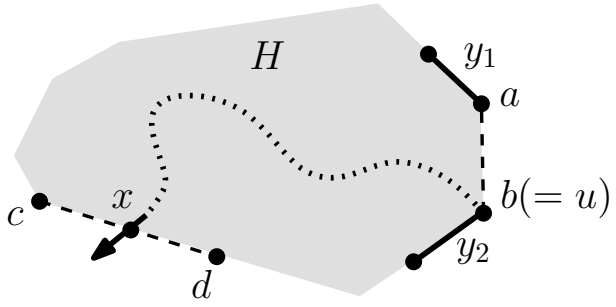

Figure 3.23: Lemma 3.25.

Lemma 3.25 implies that in order to find x while moving along edges of P during an interior step, we need check for intersections of P with at most two edges of H: the single radial edge incident with the point of entry u, and (if that edge is also incident with s) the other edge incident with s.

Every step of the algorithm can be done in constant time, except for the expansion and interior steps, which can be done in time proportional to the number of vertices that are: (i) visited on the chain; (ii) inserted into the hull; or (iii) removed from the hull. Since a vertex can be removed from the hull only once, the total running time of the algorithm is thus bounded by the number of iterations (which is $O(n)$, since each iteration advances u to a distinct vertex of either H or P), plus the number of vertices processed by the algorithm.

The vertices processed include those of P, plus any vertices that ever appear in H. Consider the start of a particular iteration, where H is the current hull, u is the current position on P, and v is the vertex of P following u. We will prove a linear bound on the total vertices by showing that at most three vertices are introduced to H by the addition of triangle $T = \triangle suv$.

Each new vertex (other than v) is a point where the boundaries of H and T cross, and hence must lie on either $\overline{uv}$ or $\overline{vs}$ (since $\overline{su} \subset H$). Suppose for the sake of contradiction that two new vertices, p and q, lie in the interior of $\overline{uv}$. We can assume that $\overline{uv}$ first enters H at p, then exits H at q. Since $\overline{uv} \subset P$, and P is simple, p and q must lie on radial edges of H. These radial edges must lie on rays $\overrightarrow{sa}$ and

$\overrightarrow{sb}$ respectively, where a and b are vertices of P preceding u (Figure 3.24). The

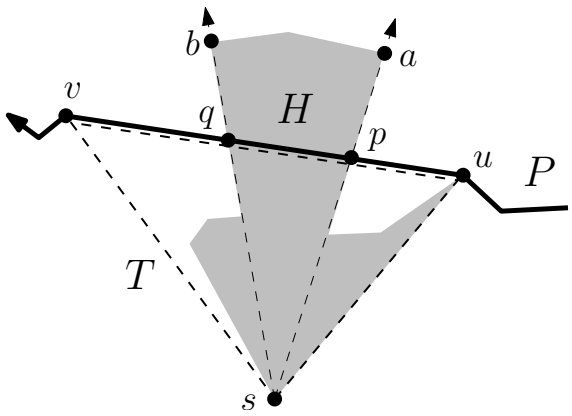

Figure 3.24: The interior of $\overline{uv}$ cannot contain two new vertices of H.

path $(a \ldots b \ldots u)$ (or $(b \ldots a \ldots u)$) in P cannot cross rays $\overrightarrow{pa}$ or $\overrightarrow{qb}$ (otherwise p or q would lie in H's interior), nor can it cross $\overline{pq}$ (since P is simple). We now have a contradiction, since this implies that $(a \ldots b)$ is not connected to $(u \ldots v)$ within P. Hence, at most one new vertex lies in the interior of $\overline{uv}$; and since s is in the kernel of H, at most one of the new vertices lies in the interior of $\overline{vs}$. We can therefore conclude:

Theorem 3.26 *The above algorithm generates the possible hull of a point and a simple polygonal chain of n vertices in $O(n)$ time.*

3.5.2 Two Polygons

Our second algorithm constructs the possible hull of a pair of uncertain polygons A and B. It starts with a polygon H equal to the convex hull of the pair, then modifies the boundary of H until H is equal to the possible hull of the pair. Such an approach seems promising, since $\text{CH}(A, B)$ and $PH(\{A, B\})$ will typically have many boundary edges in common. Consider Figure 3.25. If a boundary edge of $\text{CH}(A, B)$ is a polygonal segment or a bridge segment, then (by Lemmas 3.17 and 3.18) it lies on the boundary of $PH(\{A, B\})$ as well. Otherwise, its vertices must be nonadjacent vertices from the same polygon, and it will not be a boundary

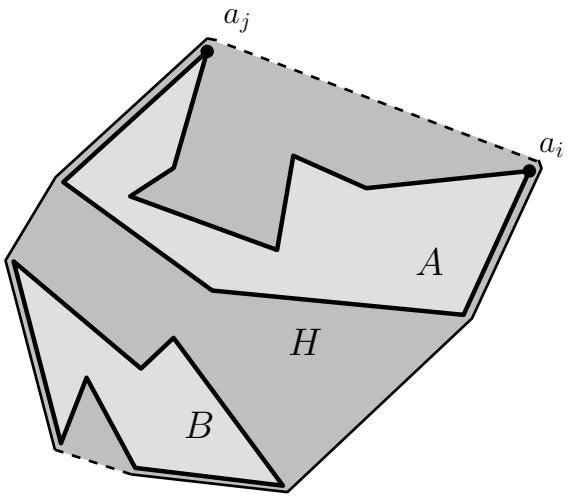

Figure 3.25: Initial polygon H; pocket lids shown as dashed lines.

edge of $PH(\{A, B\})$. As we manipulate H, both a_i and a_j will remain vertices of H, but the path $(a_i \ldots a_j)$ on H's boundary will change. We will refer to this path as a *pocket* of H, and to segment $\overline{a_i a_j}$ as the pocket's lid.

Our algorithm has these steps:

1. Initialize H to $\text{CH}(A \cup B)$.
2. **Hull Contraction.** For each pocket lid $\overline{a_i a_j}$ of H, perform the following steps:
   (a) Construct J, the possible hull of $(a_i \ldots a_j)$ (on the boundary of A) and any point s from B.
   (b) Replace $\overline{a_i a_j}$ with path $(a_i \ldots a_j)$ on the boundary of J.
3. Repeat the hull contraction steps with the roles of A and B reversed.
4. **Hull Expansion.** Perform the following steps:
   (a) Set u to h_1, any vertex of $\text{CH}(A \cup B)$.
   (b) Determine ray T_u as follows. If u is a vertex of A, then set T_u to the ray from u that is left-tangent to B; otherwise, set T_u to the ray from u that is left-tangent to A.

(c) If the vertex v of H following (i.e., in ccw direction) u is not left of T_u, then go to (f).

(d) Let x be the point where the boundary of H next crosses T_u.

(e) Replace boundary edges $(u \ldots x)$ of H with $\overline{ux}$.

(f) Advance u to the next convex vertex of H. If $u \neq h_1$, go to (b).

5. Repeat the hull expansion steps, substituting cw for ccw, and right for left.

The hull contraction steps use the algorithm of the previous section to replace each pocket lid with a portion of a possible hull associated with the pocket. This contracts the hull by 'taking bites' out of the convex hull of the two polygons (Figure 3.26). Corollary 3.21 implies that after being modified in this way, each pocket lies within PH. As we will see, each pocket also has a particular monotonicity quality that both ensures the correctness of the remaining steps of the algorithm, and contributes to the algorithm's efficiency. The hull expansion steps traverse the boundary of H, find tangent rays that potentially contain bridge segments of PH, and modify H to incorporate these segments (Figure 3.27). This has the effect of expanding the hull, by adding back some portions that were removed in the hull contraction steps.

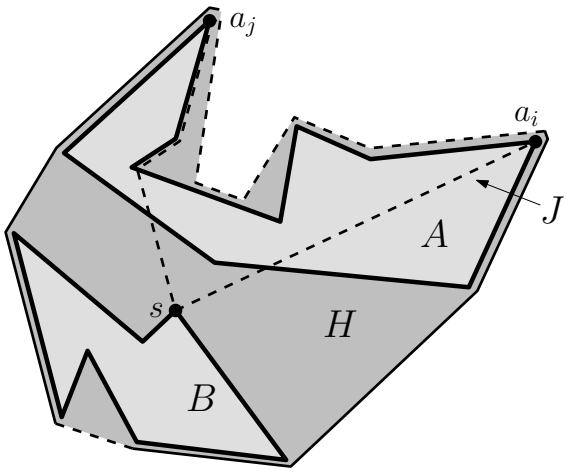

Figure 3.26: Hull contraction: pocket lid $\overline{a_i a_j}$ has been replaced by $(a_i \ldots a_j)$ of J.

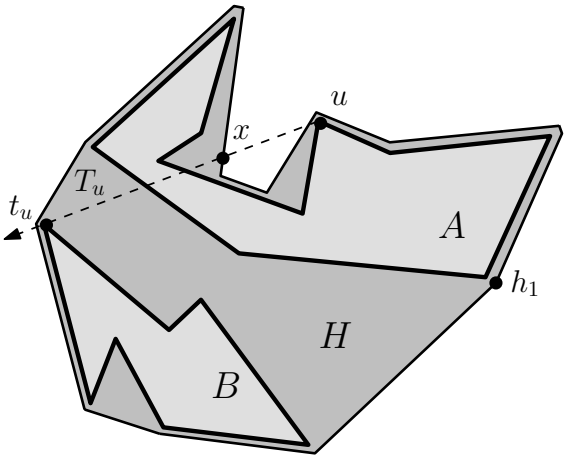

Figure 3.27: Hull expansion (ccw direction).

We will prove that the algorithm is correct by showing that after all its steps have been performed, H is within PH, and PH is within H. To support these claims, we will need some additional terminology. We will denote by T_x the ray from point x that is left-tangent to a particular region (usually A or B), and by t_x its point of tangency. If $P = (u \ldots v)$ is a path, and R is a region, then we will say that P is *left-monotonic with* R if (i) neither u nor v penetrate R, (ii) v lies in the right half-plane of T_u, and (iii) the sequence of rays from points of P that are left-tangent to R have nondecreasing polar angles.[10] We define right-monotonicity analogously. One property of these paths is that any subpath can be replaced by a line segment, and monotonicity is preserved:

Lemma 3.27 *If path $(u \ldots v)$ is left-monotonic (resp., right-monotonic) with region R, then segment $\overline{uv}$ is left-monotonic (resp., right-monotonic) with R.*

Proof. We will prove only the case for left-monotonicity, since the two cases are symmetric. Suppose $(u \ldots v)$ is left-monotonic with R, and w is any interior point of $\overline{uv}$. We need only show that the third property holds, since the truth of the first two does not depend upon w. Now, t_v must lie in the left half-plane of T_w,

[10] Points of P that lie on the boundary of R may lie on infinitely many lines tangent to CH(R). In this case, we can choose a tangent line that maintains the monotone property.

which implies that $\theta(\overrightarrow{wt_v}) \geq \theta(T_w)$; and since $w \in \overline{uv}$, $\theta(T_v) \geq \theta(\overrightarrow{wt_v})$. Thus, $\theta(T_v) \geq \theta(T_w)$. A symmetric argument shows that $\theta(T_w) \geq \theta(T_u)$. □

At the start of the hull expansion steps, the boundary of H consists of native segments, bridge segments, and pockets derived from the possible hulls J of the hull contraction steps. Together, these three sets of edges form a closed path that satisfies the conditions of Corollary 3.21; hence, at the start of step 4, H lies within PH. Observe that step 4 (and, by a symmetric argument, step 5) affects only pockets of H, for if $\overline{uv}$ is a native or bridge segment on the boundary of CH$(A \cup B)$, then v will not lie to the left of the ray constructed in step 4(b), and no change to H's boundary will be made. Hence, in the rest of our analysis, we will focus mainly on the pockets of H.

Lemma 3.28 *At the end of the hull expansion steps, each pocket $(a_i \ldots a_j)$ of H is both left- and right-monotone with B.*

Proof. We will show only that at the end of step 4, each pocket is left-monotone with B. A symmetric proof can then be used to show that step 5 induces right-monotonicity, and (by Lemma 3.27) preserves left-monotonicity.

The proof is by induction on the length of path $(a_i \ldots u)$, a subpath of $(a_i \ldots a_j)$. When $u = a_i$, this path is a single point, exterior to B, and is thus left-monotone with B. Suppose $(a_i \ldots u)$ is left-monotone with B, for some $u \in (a_i \ldots a_j)$. The vertex v following u lies either to the left or right of T_u (Figure 3.28).

Suppose v lies to the left of T_u. Observe that at the start of step 4, pocket $(a_i \ldots a_j)$ is left-monotone with some point $s \in B$. Since the only changes made to the pocket by the hull expansion steps involve replacing subpaths with individual line segments, the pocket must (by Lemma 3.27) remain left-monotone with s throughout these steps. As a consequence, the pocket lies within triangle $\triangle s a_i a_j$, and v lies to the right of $\overline{us}$. Now, a_j cannot lie to the left of T_u; otherwise, $\theta(T_u) < \theta(\overline{a_i a_j}) < \theta(T_{a_i})$, violating $(a_i \ldots u)$'s monotonicity with B. It follows that path $(u \ldots a_j)$ must cross T_u to reach a_j, which implies the existence of point x of step 4(d). If x precedes t_u on T_u, then $\theta(T_x) = \theta(T_u)$; and if x follows t_u on T_u, then we can rotate T_u ccw around x until it reaches T_x that is left-tangent to B,

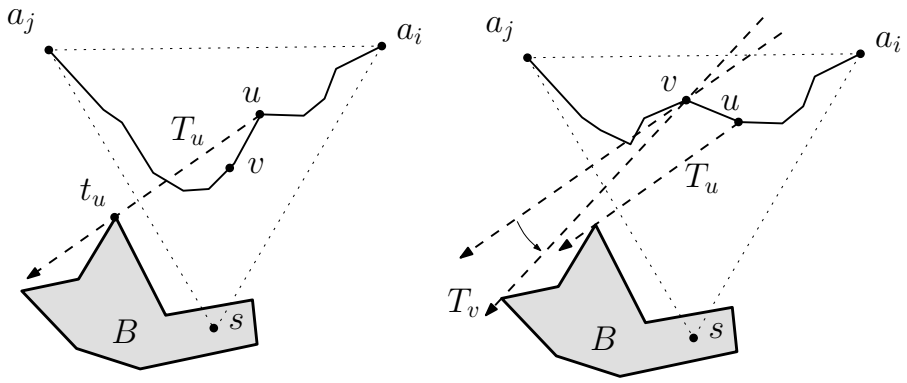

Figure 3.28: Lemma 3.28.

which implies that $\theta(T_x) \geq \theta(T_u)$. In both cases, $(u \ldots x)$ is left-monotone with B.

Now consider the case where v lies to the right of T_u. If we start with a line through v that is parallel to T_u, and rotate it ccw around v, it must become left-tangent to B before it has rotated by π radians; hence, $\theta(T_v) > \theta(T_u)$, which implies that $(u \ldots v)$ is left-monotone with B.

We can use a very similar argument to this second case to show that this monotonicity exists for every point following u (or x, if the first case applied) up to the pocket's next convex vertex. Thus, after step 4(f), $(u \ldots v)$ is left-monotone with B. □

Let us prove our first claim, that H is within PH at the end of the hull expansion steps. We showed earlier that this is true at the start of these steps, so we will consider the changes made to a particular pocket $(a_i \ldots a_j)$ in step 4 (the proof for step 5 is symmetric). Consider one iteration of step (e), the only step that changes H. Observe that u cannot have been added to H by a previous iteration of step (e), since u is convex and these steps can only add reflex vertices. Thus u must have been a vertex of J. Now, $\overline{sa_i}$ and $\overline{sa_j}$ are both within J, which implies that if s is a vertex of J it must be a reflex vertex. Hence, $u \neq s$.

In fact, u must be a vertex of A. For if it is not, then by Lemma 3.23, u must be a point of intersection of L_1 and L_2, where L_1 and L_2 are either edges of A, or bridge segments between vertices of A and s. Since u is neither s nor a vertex

of A, u lies in the interior of both L_1 and L_2. But L_1 and L_2 both lie within PH, by Lemmas 3.17 and 3.18, which implies that u is not a convex vertex of J, a contradiction.

In the proof of Lemma 3.28, we showed that point x of step (d) must exist. There are two cases (Figure 3.29): x is either strictly between u and t_u, or it is at or beyond t_u. We can now apply Corollary 3.21 to show that the region added to

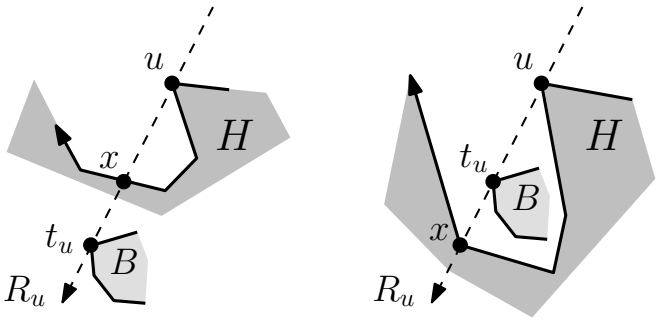

Figure 3.29: Expanding H remains within PH.

H in step (e) lies within PH. In the former case, the region is bounded by path $(u \ldots x)$ and $\overline{ut_u}$. In the latter case, since u and a_j are both vertices of A, path $(a_i \ldots a_j)$ on the boundary of A must cross T_u at a point y which is at or beyond x; hence, the region is bounded by segments $\overline{ut_u}$, $\overline{t_u y}$, and path $(u \ldots x)$.

We are now ready to prove our second claim, that $PH \subseteq H$. Let $(a_i \ldots a_j)$ be any pocket of H, and let L_i and L_j be the lines through a_i and a_j respectively that are left-tangent to B. These lines intersect at a point c to form a double wedge. B lies in one of these wedges, and Lemma 3.28 implies that pocket $(a_i \ldots a_j)$ lies in the other wedge, within triangle $\triangle c a_i a_j$ (Figure 3.30). If some point $x \in PH$ is not within H, then by Lemma 3.17, x lies on some segment $\overline{ab}$, where $a \in A, b \in B$. Since $x \in \text{CH}(A \cup B)$, and all non-pocket edges of H lie on the boundary of $\text{CH}(A \cup B)$, x must lie within the triangle associated with some pocket; without loss of generality, we can assume that the pocket is $(a_i \ldots a_j)$. Since B lies in the wedge opposite to $\triangle c a_i a_j$, a must lie in this triangle. In fact, either a, or some point farther from b on $\overrightarrow{ba}$, must be a point on pocket $(a_i \ldots a_j)$. We can now apply Lemma 3.28 to show that $x \in H$, a contradiction.

The algorithm is correct; we now show that it is efficient as well. Let n be the

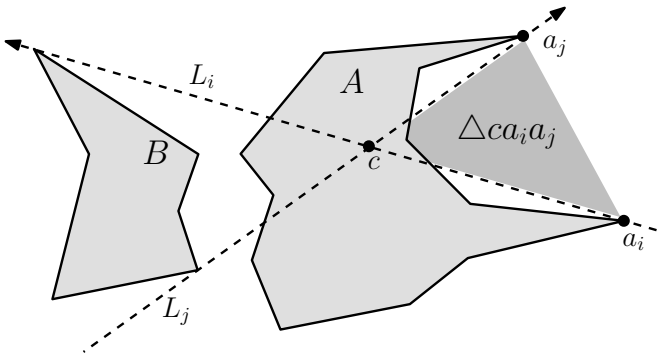

Figure 3.30: Wedges induced by pocket $(a_i \ldots a_j)$.

total number of vertices of A and B. To perform step 1, we start by constructing the convex hulls of both A and B. This can be done in $O(n)$ time, e.g., by using Melkman's algorithm [50]. Once these hulls are known, the convex hull of $A \cup B$ can be constructed, in $O(n)$ time, using the rotating calipers paradigm [72].

In the hull contraction steps, for each pocket $\overline{a_i a_j}$, we construct the corresponding subset of the boundary from A, then calculate the possible hull of this boundary and an arbitrary point of B. Since each edge of A appears in only one of these subsets, and the possible hulls can be constructed in time linear in the size of the subset (Theorem 3.26), we can perform these steps in $O(n)$ time.

In step 4(b), we find the ray through a point $u \in A$ that is left-tangent to B. Using reasoning very similar to that of the proof of Lemma 3.28, we can show that by remembering the vertices of $\text{CH}(A)$ and $\text{CH}(B)$ last used to calculate a ray tangent to one of these polygons, each iteration of this step can be performed in $O(1 + m)$ time (by advancing the appropriate pointer until it reaches the vertex supporting a tangent line), where the total m over all iterations is linear in the complexity of the hulls, or n in the worst case. Since an edge can only be removed once, steps 4(d) and 4(e) can be performed in $O(1+k)$ time, where the total k over all iterations is linear in the number of vertices at the start of the expansion steps. By Theorem 3.24, H at this time (as well as at the start of step 5) has no more than n vertices; hence, the total k over all iterations is linear in n. Each of the remaining substeps of step 4 require constant time per iteration. Since each iteration advances u to a distinct vertex of H, there are $O(n)$ iterations; thus the total time spent in

step 4 (as well as in step 5) is $O(n)$.

We have shown that each step of the algorithm can be performed in linear time. Hence,

Theorem 3.29 *The possible hull of a pair of polygons with n total vertices can be constructed in $O(n)$ time.*

A natural question to ask is whether the possible hulls J need to be constructed within the hull contraction steps. It would be simpler, after all, if we could just contract each pocket $\overline{a_i a_j}$ to the corresponding path $(a_i \ldots a_j)$ on the boundary of A.

One reason why this cannot be done is that if this simplification is made, the pocket is not guaranteed to be left-monotone to any particular point of B at the start of the hull expansion steps, which impacts the efficiency of those steps. Consider the pair of polygons in Figure 3.31. Each of the 'spikes' in the boundary of A have a pair of convex vertices u that are not left-monotone with B. As noted earlier, the existing algorithm calculates R_u by advancing a pointer m times until it finds a vertex supporting tangent line R_u. If we use $(a_i \ldots a_j)$ as the initial pocket, then m can be $\Theta(n)$ for each tangent line calculation, resulting in an $O(n^2)$ running time. We could improve the running time to $O(n \log n)$ by using a binary search to find R_u, but this is still inferior to the $O(n)$ running time of the original algorithm.

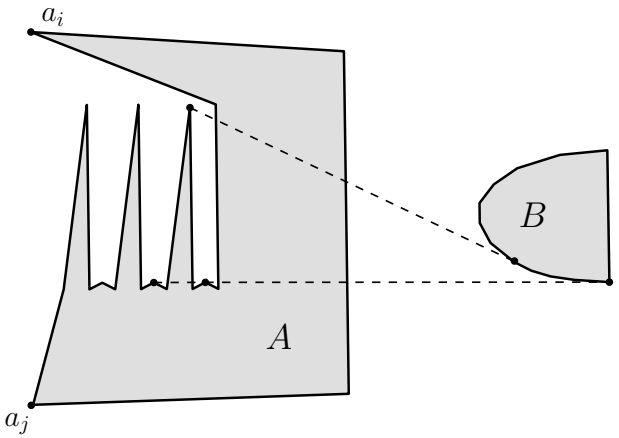

Figure 3.31: Lemma 3.28 will not hold.

3.5.3 Multiple Polygons

We can leverage the results of the previous sections to produce an algorithm for the possible hull of any number of polygons, with minimal extra effort.

Theorem 3.30 *The possible hull of k polygons with a total of n vertices can be constructed in $O(n \log k)$ time.*

Proof. By Theorem 3.29, the possible hull for a pair of polygons can be constructed in $O(n)$ time. Lemma 3.19 implies that we can apply this algorithm recursively, in a divide-and-conquer manner, to construct the possible hull of k polygons. In doing so, we increase the running time by a factor that is logarithmic in the height of a binary tree of k elements. □

3.6 Closing Remarks

This chapter was concerned with two variants of the convex hull problem involving uncertain sites: guaranteed hulls, and possible hulls. We provided an improved bound on the complexity of guaranteed hulls, and presented a novel algorithm for constructing guaranteed hulls of a particular class of uncertain regions. We also demonstrated how an existing output sensitive convex hull algorithm of point sites can be adapted for such regions, and showed how three planar guaranteed hull algorithms can be adapted to construct guaranteed hulls in $\mathbb{R}^3$. In addition to identifying some properties of possible hulls of (not necessarily convex) uncertain regions, we presented an optimal algorithm for constructing possible hulls of uncertain polygons. To the best of our knowledge, these are the first results related to possible hulls of nonconvex uncertain regions.

We can identify some open problems and areas of future research with respect to guaranteed hulls. All of the guaranteed hull algorithms that we have encountered (including the ones we have introduced here) operate by constructing a set of hull bitangents, then generating the guaranteed hull as the intersection of the bitangents' half-planes. The complexity of this intersection may be much smaller than the number of bitangents (the intersection may even by empty), which suggests that any truly output-sensitive algorithm for guaranteed hulls will probably require a different approach.

Another open problem relates to the number of hull bitangents induced by a set of uncertain polygons. We have shown that the complexity of a guaranteed hull is strictly linear in the number of uncertain regions, regardless of the regions' shapes (Theorem 3.6); however, the number of hull bitangents that produce this guaranteed hull is not necessarily linear. While we have proven linear bounds for the number of bitangents induced by aligned similar uncertain regions (Theorem 3.7), we have been unable to improve the $O(n\alpha(n))$ bound given by [55] for uncertain polygons.

The algorithms discussed in Section 3.4.3 rely on Theorem 3.14, and hence can only be relied upon to construct guaranteed hulls that are known to have nonzero volume. This situation could be improved by strengthening Theorem 3.14 to be a direct analog of Theorem 3.1 (we leave this as an open problem). In addition, the algorithm we presented for constructing guaranteed hulls of uncertain spheres is perhaps not optimal. Recall that one step of the algorithm involves constructing the convex hull of a set of spheres. As shown in [11], these hulls can have $\Theta(n^2)$ complexity; but we are only interested in the planar facets of these hulls. We do not know if the number of these facets is always subquadratic. If so, then modifying the algorithm of [11] (or some other algorithm) to generate only planar facets of these hulls may yield a subquadratic guaranteed hull algorithm. We also leave as an open problem the complexity of guaranteed hulls of general uncertain regions in $\mathbb{R}^3$.

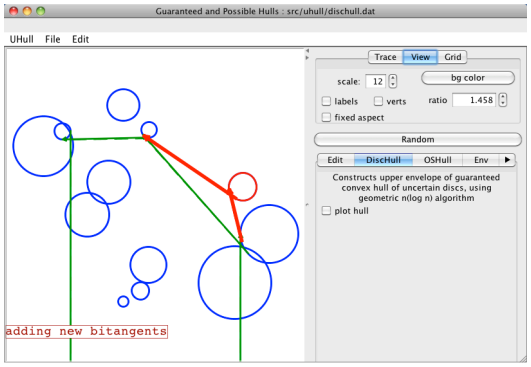

Figure 3.32: Applet for Chapter 3.

An applet that implements some of the algorithms described in this chapter can be found at http://www.cs.ubc.ca/~jpsember/uh.html. The applet (Figure 3.32)

includes the following sample files:

1. `dischull.dat`: Constructs the guaranteed hull of uncertain discs, using the algorithm of Section 3.4.1.
2. `oshull.dat`: Constructs the guaranteed hull of uncertain discs, using the algorithm of Section 3.4.2.
3. `poly_pt.dat`: Constructs the possible hull of a point and a nonconvex polygon, using the algorithm of Section 3.5.1.
4. `valley.dat`: Constructs the possible hull of a pair of nonconvex polygons, using the algorithm of Section 3.5.2.
5. `valley_multiple.dat`: Constructs the possible hull of several nonconvex polygons, using the algorithm of Section 3.5.3.

Chapter 4

Smallest Bounding Disc of Imprecise Points

4.1 Introduction

In the 1-center problem (or, as it is sometimes called, the minmax location problem), we are given a set S of points in the plane, and we need to find the location of an additional point f such that the maximum distance from f to any point of S is minimized. The 1-center problem is a classic facility location problem from the field of operations research. In that setting, the points S are considered *demand* points, and the point f is considered the location of a facility to service the demand points. The 1-center problem has application in many other fields as well, such as computer graphics, mechanical engineering, biology, and even political science [28].

The 1-center problem can be generalized in many ways. For instance, the distance between the facility and demand points may use other metrics (e.g., the L^1 or Manhattan metric), or may be replaced by a more complicated *transportation cost*, perhaps involving individual weighting factors on the demand points. Another generalization is the k-center problem. This differs from the 1-center problem in that more than one facility can be specified, and each demand point is free to use the nearest facility.

In this chapter, we will consider only the 1-center problem, where the distances

are Euclidean (i.e., employ the L_2 metric). In this setting, the 1-center problem is equivalent to what we will call the *smallest bounding disc* problem, in which the goal is to find the smallest disc that contains a point set.

We will denote the smallest bounding disc of a point set S by $\mathbb{B}(S)$. There are two fundamental conditions that a disc C must satisfy in order to be a smallest bounding disc of S. The *intersection* condition holds if S is a subset of the disc, and the *support* condition holds if every section of the boundary of the disc that subtends an angle of at least π intersects the set. Clearly, the intersection condition must hold for C to be a bounding disc of S. The support condition must hold as well, for if some boundary arc of C subtends an angle of at least π and does not contain any points of S, then we can move the origin of C away from this arc and reduce its radius, yielding a smaller bounding disc of S.

It is easy to show that any disc C that satisfies these conditions must be unique: if p is the origin of C, and q is any other point in the plane, then the support condition implies that at least one point of S must be strictly farther from q that it is from p. Hence, any disc centered at q that satisfies the intersection condition must be larger than C.

An important property of the smallest bounding disc of a point set is that it is determined by a small number of points. If we start with a point set S, and repeatedly remove points from this set until no points can be removed without changing its smallest bounding disc, then we will be left with some subset of S. We will call this subset a *critical subset* of S. When it is clear from the context, we may refer to a member of a critical subset as a *critical point*. There may not be a unique critical subset for a point set. For instance, the vertices of a regular pentagon have five distinct critical subsets.

We can make some observations concerning critical subsets. First, every critical subset of a point set S lies on the boundary of $\mathbb{B}(S)$. Second, every critical subset of S (where $|S| \geq 2$) consists of either two or three points: if it has two, then the midpoint of the two critical points will be the 1-center of S (and the critical points are the endpoints of a diameter of the smallest bounding disc of S); and if it has three, then the angles subtended by the arcs between neighboring critical points will be less than π. The left set of points in Figure 4.1 has two critical points, and the right set of points has three.

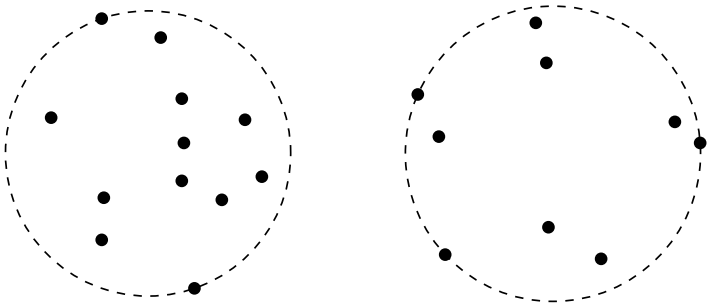

Figure 4.1: Smallest bounding discs of point sets.

This chapter is concerned with a variant of the 1-center problem, one involving imprecise points. We define the *possible 1-centers* of uncertain regions $\mathcal{D}$ to consist of those points in the plane that are the 1-center for some feasible set S of $\mathcal{D}$. We will denote the possible 1-centers by $\mathbb{C}^\circ(\mathcal{D})$ (or simply $\mathbb{C}^\circ$).

4.1.1 Related Work

The smallest bounding disc problem was first posed by Sylvester in 1857 [68]. A linear time algorithm for its solution was given by Megiddo [49], and Welzl provided a simple randomized algorithm to solve the problem in expected linear time [74]. Prior to Megiddo's result, the fastest algorithm to solve this problem was given by Shamos and Hoey [65], who observed that the smallest bounding disc of a set of points must be centered on a vertex of the farthest Voronoi diagram of the points, which can be constructed in $O(n \log n)$ time.

Some work has been done related to smallest bounding discs of uncertain points. The *intersection radius* of a set of uncertain regions is the radius of the smallest disc that is a feasible smallest bounding disc of the regions. Bhattacharya *et al* [10] showed that the intersection radius for a set of uncertain lines and points can be found in linear time. This result was later extended to sets of uncertain regions that may also include convex polygons, rays, and wedges [32]. For uncertain discs, Löffler and van Kreveld [48] have shown that an adaptation of Welzl's algorithm [74] can find the intersection radius in expected linear time.

The problem of computing the *largest* feasible smallest bounding disc for uncertain regions has been investigated by Löffler and Kreveld [48] as well. They

show that for uncertain discs or uncertain axis-aligned squares, the problem can be solved in linear time.

4.1.2 Contributions

In this chapter, we introduce a new geometric figure: $\mathbb{C}^\circ(\mathcal{D})$, the possible 1-centers of uncertain regions $\mathcal{D}$. We identify some properties of this figure, and prove that its boundary points are contained by three families of curves induced by the uncertain regions.

We demonstrate a connection between $\mathbb{C}^\circ(\mathcal{D})$ and the guaranteed hull of $\mathcal{D}$, one that can be exploited to optimize operations involving $\mathbb{C}^\circ(\mathcal{D})$.

For the cases where the uncertain regions $\mathcal{D}$ are discs or axis-aligned rectangles, we present two algorithms: an $O(n \log k)$ *query* algorithm to determine if a query point lies within $\mathbb{C}^\circ$, and an expected $O((n + k^6) \log k)$ *construction* algorithm to construct $\mathbb{C}^\circ$. The value k is the number of regions of $\mathcal{D}$ that are not interior to the guaranteed hull of $\mathcal{D}$. This value never exceeds n, and can be substantially smaller than n.

4.2 Possible 1-Centers

In this section, we identify some properties of the possible 1-centers of a set of general uncertain regions, and show that each boundary point of this figure lies on one of three types of curves induced by the regions. We also highlight a connection between possible 1-centers and guaranteed hulls. Finally, we look at the specific case where the uncertain regions are discs, and determine what form the three curves take in this case.

Lemma 4.1 *If $\mathcal{D}$ is a set of connected uncertain regions, then $\mathbb{C}^\circ(\mathcal{D})$ is a connected, compact set.*

Proof. We can view the 1-center of a set of n uncertain regions as a mapping from a connected subset of $\mathbb{R}^{2n}$ to $\mathbb{R}^2$. This mapping is continuous, hence $\mathbb{C}^\circ$ is compact ([70], Theorem 13.3.1) and connected (Ibid., Theorem 13.4.6). □

To describe $\mathbb{C}^\circ$, we will investigate where points on its boundary can lie. We will say that an element s_i of a feasible set S is a *boundary site* if it lies on the

boundary of its corresponding region D_i.

Lemma 4.2 *Let $\mathcal{D}$ be a set of convex uncertain regions. If S is a feasible set of $\mathcal{D}$ whose 1-center lies on the boundary of $\mathbb{C}^\circ(\mathcal{D})$, then two elements of S are boundary sites that also lie on the boundary of $\mathbb{B}(S)$.*

Proof. Let S be a feasible set whose 1-center, p, lies on the boundary of $\mathbb{C}^\circ$. We will first modify S by replacing any boundary sites $s_i \in S$ lying in the interior of $\mathbb{B}(S)$ with points s_i' interior to both $\mathbb{B}(S)$ and D_i. Note that after this procedure, (i) every boundary site of S will lie on the boundary of $\mathbb{B}(S)$, (ii) $\mathbb{B}(S)$ (and p) have not changed, and (iii) all of the original boundary sites of S lying on the boundary of $\mathbb{B}(S)$ will remain. Hence, if we can prove the claim for the modified set, we have proven it for the original set.

If S has no boundary sites, then observe that for any θ, there exists a nonzero-length vector with polar angle θ (denoted τ_θ) such that each site $s_i \in S$ can be replaced by the point $s_i + \tau_\theta$. This yields a new feasible set S' whose 1-center is $p + \tau_\theta$. As this is true for any choice of θ, p cannot lie on the boundary of $\mathbb{C}^\circ$; a contradiction.

If S has a single boundary site s_i, then s_i can be translated by some nonzero-length vector τ_θ, and each $s_{j\neq i} \in S$ can be translated (if necessary) to an interior point of D_j so that the resulting set S' is (i) feasible, (ii) has the same 1-center as S, and (iii) has no boundary sites (Figure 4.2). We can now apply the previous argument to reach a contradiction. □

It is necessary at this point to define a variant form of tangency. We will say that region R is *locally inside-tangent* to disc C at point p if (i) p lies on the boundaries of both R and C, and (ii) every point of R within some distance $\epsilon > 0$ of p lies within C. We may also say that C has local inside-tangent R at p. In Figure 4.3, polygon R is locally inside-tangent to disc C at p. Note that if R is a disc, then local inside tangency is the same as inside tangency.

If D_f and D_j are distinct regions of $\mathcal{D}$, then we define the *type I curve of D_f and D_j* to be the set of points that are the centers of discs which have outside-tangent D_f and locally inside-tangent D_j. We denote this curve by I_{fj}. The subscript 'f' is chosen since D_f will usually be the farthest region from points of interest on this curve, as we will see later.

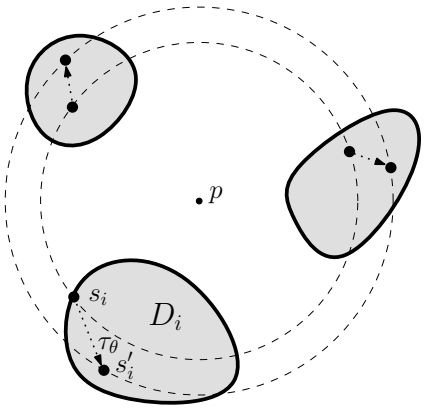

Figure 4.2: Lemma 4.2.

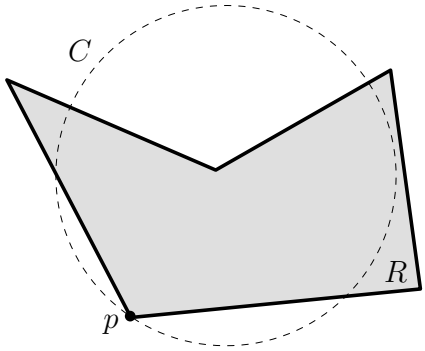

Figure 4.3: R is locally inside-tangent to C at p.

Lemma 4.3 *If $\mathcal{D}$ is a set of convex uncertain regions, and if p is both a point on the boundary of $\mathbb{C}^\circ(\mathcal{D})$ and the 1-center of a feasible set S that has a critical subset of three points, then p must lie on a type I curve of $\mathcal{D}$.*

Proof. Let p and S be such a point and feasible set respectively. We will show that p lies on a type I curve by proving that some region of $\mathcal{D}$ is outside-tangent to $\mathbb{B}(S)$, and another region of $\mathcal{D}$ is locally inside-tangent to $\mathbb{B}(S)$.

If no region is outside-tangent to $\mathbb{B}(S)$, then observe that we can construct another feasible set S' with the same 1-center as S, by replacing each point of S on the boundary of $\mathbb{B}(S)$ with a point that is (i) slightly closer to p, and (ii) interior to its respective region. Lemma 4.2 then implies that p is not on the boundary of

$\mathbb{C}^\circ$, a contradiction. If no region is locally inside-tangent to $\mathbb{B}(S)$, then we can use a similar procedure to construct a feasible set S' of points that are slightly farther away from p, and again apply Lemma 4.2 to arrive at a contradiction. □

We now consider points s on the boundary of $\mathbb{C}^\circ$ associated with feasible sets that have only two critical points. We will show that such points lie on one of two additional types of curves. We define the *type II curve* of distinct regions D_i and D_j of $\mathcal{D}$ to be the boundary of the Minkowski average of the regions, and denote this curve by II_{ij}. The Minkowski average of regions D_i and D_j is the set

$$\left\{\frac{1}{2}(p_i + p_j) \mid p_i \in D_i, p_j \in D_j\right\}.$$

We define the *type III curve* of distinct regions D_f, D_i, and D_j of $\mathcal{D}$ to consist of origins of discs C_p where (i) C_p has outside-tangent D_f; and (ii) p is the mid-point of boundary points of D_i and D_j (see Figure 4.4). We denote this curve by III_{fij}.

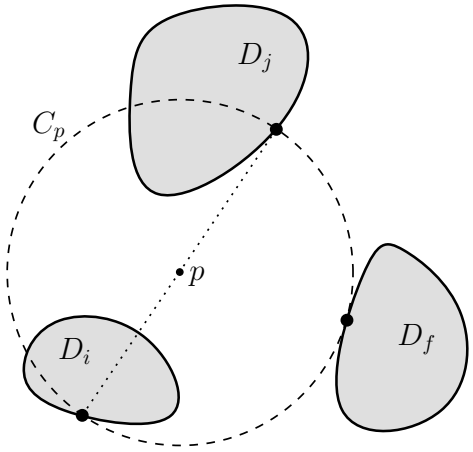

Figure 4.4: Point p is on the type III curve of D_f, D_i, and D_j.

Let $\mathcal{D}$ be a set of convex uncertain regions. We now show that each boundary point of $\mathbb{C}^\circ(\mathcal{D})$ must lie on a type I, II, or III curve of $\mathcal{D}$. Let p be a boundary point of $\mathbb{C}^\circ$ that does not lie on a type I curve. There must exist a feasible set S whose 1-center is p; and by Lemma 4.3, every critical subset of S must have

exactly two points, with p as their midpoint. Let one such critical subset be $Q = \{q_i \in D_i, q_j \in D_j\}$. We first consider the case where no third region $D_{k \notin \{i,j\}}$ of $\mathcal{D}$ is outside-tangent $\mathbb{B}(S)$.

Let us perform a procedure similar to that of the proof of Lemma 4.2, if necessary, to ensure that no sites other than q_i and q_j lie on the boundary of $\mathbb{B}(S)$. We can then apply Lemma 4.2 to prove that both q_i and q_j lie on their respective regions' boundaries. Observe that there is a continuous, closed range of polar angles of directed lines through q_i that are right-tangent D_i. We denote this range by Right(i). Note also that each angle $\theta \in$ Right(i) has a corresponding angle $\theta + \pi$, the polar angle of a directed line through q_i that is left-tangent D_i. We denote this corresponding range by Left(i). The Right(i) and Left(i) ranges are separated by disjoint, open ranges, representing the polar angles of lines through q_i that are about to either leave D_i, or enter the interior of D_i, at that point. We denote the former range by Out(i), and the latter range by In(i). Note that for convex uncertain regions with smooth (that is, continuously differentiable) boundaries, both Right(i) and Left(i) contain a single value, while for other regions (e.g., polygons), this may not be the case; see Figure 4.5.

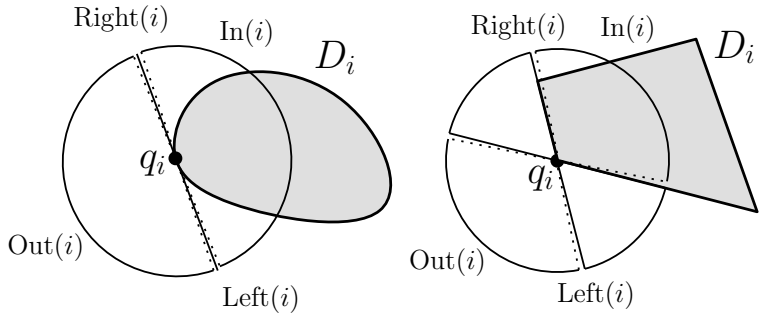

Figure 4.5: Polar angle ranges associated with $q_i \in D_i$.

If Right(i) and Right(j) are not disjoint, then there exist parallel lines through q_i and q_j that are right-tangent D_i and D_j respectively (Figure 4.6). It follows that p lies on the boundary of the Minkowski average of D_i and D_j; hence, $p \in II_{ij}$.

If Right(i) and Right(j) are disjoint, then it is easy to show that In(i) and Out(j) intersect. It follows that there exists an angle θ where $\theta \in$ In(i) and $\theta + \pi \in$ In(j) (Figure 4.7). Note that this implies that s_i and s_j can be replaced by points

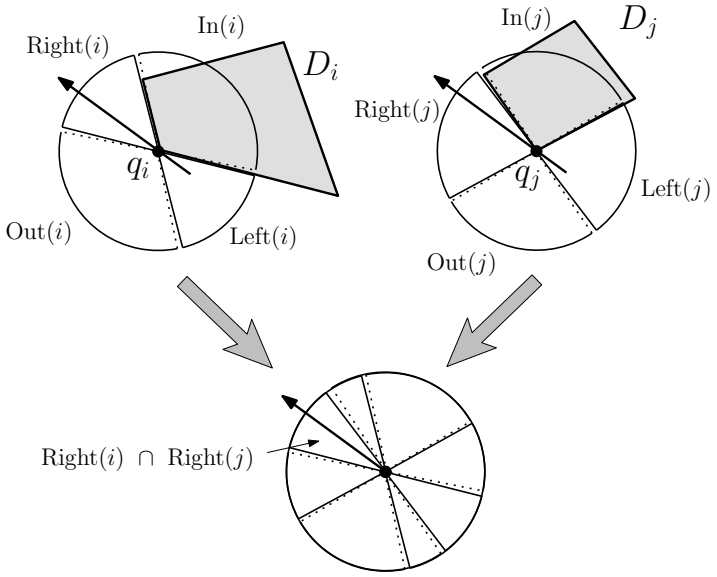

Figure 4.6: Right$(i) \cap$ Right$(j) \neq \emptyset$.

s_i', s_j' interior to their respective regions, such that their midpoint remains at p. The resulting feasible set S', whose 1-center is also at p, has no boundary sites. Lemma 4.2 then implies that p is not on the boundary of $\mathbb{C}^\circ$, a contradiction.

Now suppose some D_k is outside-tangent $\mathbb{B}(S)$. To prove that p lies on III_{kij}, We need only show that q_i and q_j are boundary sites. If this is not the case, then (without loss of generality) q_i lies in the interior of D_i. Note that D_j cannot be locally inside-tangent $\mathbb{B}(S)$, otherwise p lies on I_{kj}, which we know to be false. It follows that we can move q_i and q_j directly away from each other and into their respective regions' interiors, to produce a feasible set S', centered at p, but with no boundary sites. We can then apply Lemma 4.2 again to reach a contradiction.

We have proven the following result.

Theorem 4.4 *If $\mathcal{D}$ is a set of convex uncertain regions, then every point on the boundary of $\mathbb{C}^\circ(\mathcal{D})$ lies on a type I, II, or III curve of $\mathcal{D}$.*

The guaranteed hull of a set of uncertain regions can be used to identify some regions that will not contribute to the possible 1-centers of the regions.

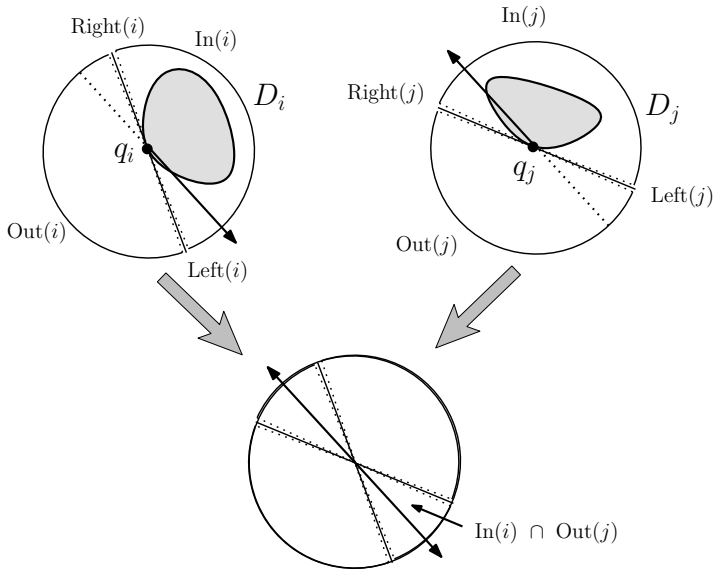

Figure 4.7: In$(i) \cap$ Out$(j) \neq \emptyset$.

Theorem 4.5 *If D_i is a member of $\mathcal{D}$, a set of uncertain regions, and D_i lies in the interior of GH, the guaranteed hull of $\mathcal{D}$, then $\mathbb{C}^\circ(\mathcal{D}) = \mathbb{C}^\circ(\mathcal{D} \setminus \{D_i\})$.*

Proof. Let q be an interior point of GH. We start by showing that every directed line through q has at least one region of $\mathcal{D}$ lying strictly to its right. Let L be one such line. If no region of $\mathcal{D}$ lies strictly to the right of L, then there exists a feasible set S of $\mathcal{D}$ that lies in the left half-plane of L. But since $GH \subseteq \text{CH}(S)$, q is not an interior point of GH, a contradiction.

Let us denote $\mathbb{C}^\circ(\mathcal{D} \setminus \{D_i\})$ by $\overline{\mathbb{C}}^\circ$. Suppose, by way of contradiction, that some point $p \in \overline{\mathbb{C}}^\circ$ is not within $\mathbb{C}^\circ$. Then there exists a disc C_p, centered at p, that is a feasible smallest bounding disc of $\mathcal{D} \setminus \{D_i\}$ but not $\mathcal{D}$. Note that this implies that D_i does not intersect C_p, which means that there exists a directed line L that (i) is left-tangent to D_i at a point q, and (ii) has C_p strictly to its left. Now note that since q is interior to GH, some region of $\mathcal{D} \setminus \{D_i\}$ lies strictly to the right of L, and hence does not intersect C_p, a contradiction (since C_p must intersect every region of $\mathcal{D} \setminus \{D_i\}$). Hence, $\overline{\mathbb{C}}^\circ \subseteq \mathbb{C}^\circ$.

Now suppose some point $p \in \mathbb{C}^\circ$ is not within $\overline{\mathbb{C}}^\circ$. Then there exists a disc C_p, the smallest bounding disc of a feasible set S of $\mathcal{D}$, that is not also the smallest

bounding disc of $S \setminus \{s_i\}$. This implies that C_p does not satisfy the support condition for $S \setminus \{s_i\}$, which implies that s_i is a boundary point of C_p. Now consider the line L left-tangent to C_p at s_i. Since s_i lies in the interior of GH, some region $D_j \in \mathcal{D}$ must lie strictly to the right of L. This is a contradiction, since it implies that $s_j \in S$ lies outside of C_p. Hence, $\mathbb{C}^\circ \subseteq \overline{\mathbb{C}}^\circ$. □

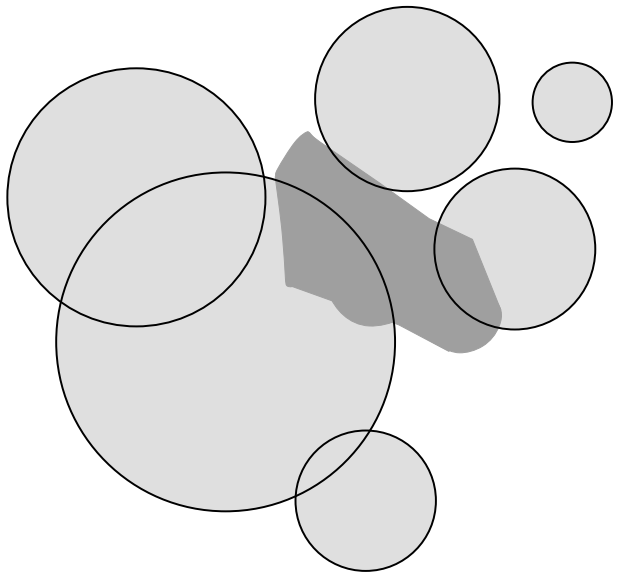

Figure 4.8: Possible 1-centers of uncertain discs.

We now consider the possible 1-centers of uncertain discs (Figure 4.8). We start by showing how Theorem 4.5 can be applied to 'filter out' some of the discs that do not contribute any possible 1-centers:

Lemma 4.6 *Let $\mathcal{D}$ be a set of n uncertain discs. The k discs that do not lie in the interior of $GH(\mathcal{D})$ can be found in $O(n \log k)$ time.*

Proof. We first use the algorithm of Section 3.4.2 to construct GH. The proof of Theorem 3.6 implies that h, the number of hull bitangents of $\mathcal{D}$, is not more than k; hence, GH is a convex polygon with at most k edges, and can be constructed in $O(n \log k)$ time. We then construct the Voronoi diagram of the edges of GH (in $O(k \log k)$ time); this allows us to determine, in $O(n \log k)$ time, which of the n discs of $\mathcal{D}$ lie in GH's interior. □

Next, we characterize the curves containing boundary points of the possible 1-centers of uncertain discs. We will need this information later in order to construct the possible 1-centers.

Lemma 4.7 *If $\mathcal{D}$ is a set of uncertain discs, then each curve I_{fi} lies on the arm of a hyperbola whose foci are o_f and o_i; and each curve II_{ij} lies on the boundary of a disc.*

Proof. By definition, every point $p \in I_{fi}$ is the center of a disc that has outside-tangent D_f and inside-tangent D_i. It is easy to show that p, o_f, and o_i satisfy the equation of a hyperbola of the stated type. Since the regions are discs, the Minkowski average of any pair of regions is also a disc; hence, each curve II_{ij} lies on a circle. □

Type III curves for uncertain discs are more complicated. In the following, we show how each such curve can be represented as a system of polynomial equations. Numerical methods can then be applied to this system to generate points on the curve.

Let p be a point on curve III_{fij}, C_p be the disc centered at p corresponding to the definition for III_{fij}, and a (resp., b) the critical point where the boundaries of C_p and D_i (resp., D_j) intersect. If we place the axes so that o_f is at the origin (Figure 4.9), then:

$$d(a, o_i) - r_i = 0 \tag{4.1}$$

$$d(b, o_j) - r_j = 0 \tag{4.2}$$

$$p = \frac{1}{2}(a + b) \tag{4.3}$$

$$d(p, 0) = d(a, b)/2 + r_f \; . \tag{4.4}$$

Substituting (4.3) into (4.4) leads to

$$\frac{1}{4}(a_x + b_x)^2 + \frac{1}{4}(a_y + b_y)^2 =$$
$$\frac{1}{4}(a_x - b_x)^2 + \frac{1}{4}(a_y - b_y)^2 + d(a, b) \cdot r_f + r_f^2 \; ;$$

and since $(u+v)^2 - (u-v)^2 = 4uv$, this can be rearranged to yield

$$a \cdot b - d(a,b) \cdot r_f - r_f^2 = 0\,, \tag{4.5}$$

where $a \cdot b$ is the scalar product of a and b.

Equations (4.1), (4.2), and (4.5) now form a system of polynomial equations defining pairs of points (a, b) whose midpoints lie on III_{fab}.

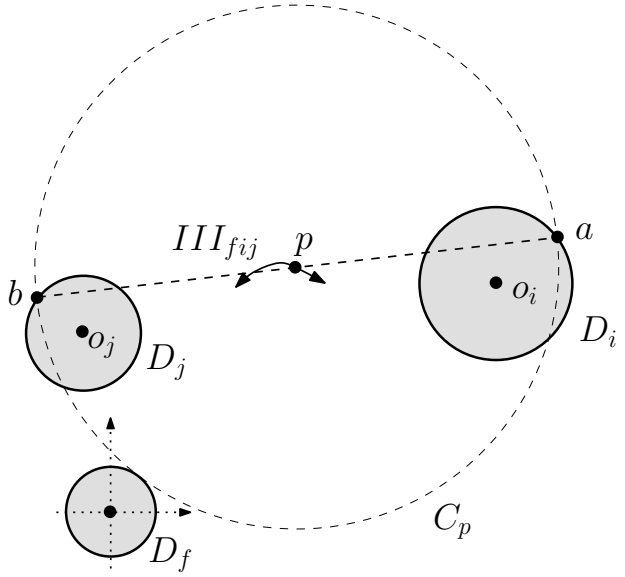

Figure 4.9: Parameterization of III_{fib}.

4.3 Query Algorithm

In this section, we present an efficient algorithm to determine whether or not a particular point is a 1-center of a feasible set of a set of uncertain discs. Later, we will extend the algorithm to the case where the uncertain regions are axis-aligned rectangles.

To gain some intuition into how the algorithm will work, we will require some additional terminology. See Figure 4.10. Let q be the query point, and C_q^r be the disc with radius r centered at q. For each boundary point x of C_q^r, let $\widetilde{x}$ denote its antipodal point. Let D_f be the farthest region of $\mathcal{D}$ from q, r^* be its distance from

q, and p_r be the point where the boundary of C_q^r crosses the ray from q through the closest point of D_f.[1]

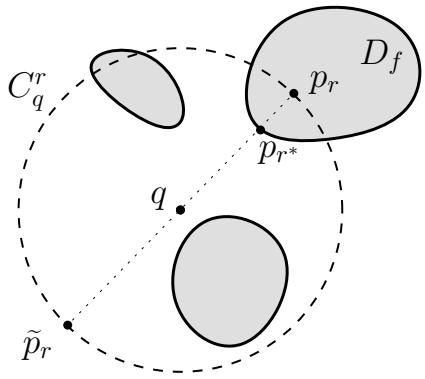

Figure 4.10: Terminology related to query point q.

As an optimization step, we will use the algorithm of Lemma 4.6 to remove any regions from $\mathcal{D}$ that do not contribute possible 1-centers.

Lemma 4.8 *D_f must be one of the k regions of $\mathcal{D}$ that is not interior to GH.*

Proof. Let D_i be any region of $\mathcal{D}$ that is interior to GH, and let L be the line that is (i) perpendicular to ray $\overrightarrow{qo_i}$, (ii) left-tangent to D_i, and (iii) has segment $\overline{qo_i}$ strictly to its left. By the proof of Theorem 4.5, some region of $\mathcal{D}$ lies strictly to the right of L, and is hence farther from q than D_i. □

The intersection and support conditions of smallest bounding discs can be extended to the situation where the points lie within a set of regions of uncertainty. The intersection condition extends directly: every smallest bounding disc of a feasible set of $\mathcal{D}$ must intersect every region of $\mathcal{D}$.

Lemma 4.9 *If $r^* = 0$, then $q \in \mathbb{C}^\circ$; otherwise, a particular C_q^r satisfies the intersection condition of smallest bounding discs if and only if $r \geq r^*$.*

Proof. If $r^* = 0$, then every region of $\mathcal{D}$ must intersect q. Our general position assumption then implies that there exists some feasible set of points very close to

[1] If $r^* = 0$, then this ray is not well-defined; but as we will see later, this will not be a problem.

q whose 1-center is at q. If $r^* > 0$, then only if $r \geq r^*$ will disc C_q^r intersect D_f (and every region of $\mathcal{D}$). $\square$

Extending the support condition is more challenging, since a particular uncertain region can contribute at most one point to the critical subset of each smallest bounding disc. To address this issue, we require some additional definitions. If u and v are two boundary points of disc C, then we denote the boundary arc from u ccw to v by arc(u, v). The *angle* of an arc is the angle subtended by the arc. If the boundary of C intersects at least two regions of $\mathcal{D}$, and s is a boundary point of C, then we define the *gap at* s to be the smallest arc(u, v) of this boundary that contains s and for which there are distinct regions $D_i, D_j \in \mathcal{D}$ such that $u \in D_i$ and $v \in D_j$. If the boundary of C intersects fewer than two regions of $\mathcal{D}$, then we will consider the gap at s to be the entire boundary of C, and to have an angle of 2π. A gap is large if its angle is greater than π, and small otherwise. In Figure 4.11, the gap at u is small, while those at v and w are large.

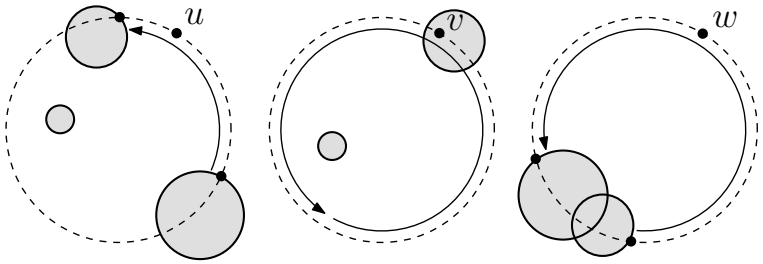

Figure 4.11: Examples of gaps at disc boundary points.

Lemma 4.10 *Point q lies within $\mathbb{C}^\circ$ iff there exists some C_q^r, with $r \geq r^*$, whose every boundary point lies within a small gap.*

Proof. If $q \in \mathbb{C}^\circ$, then there exists $r \geq r^*$ and a feasible set whose smallest bounding disc is C_q^r. Since C_q^r satisfies the support condition, every point on its boundary lies within an arc with angle not greater than π whose endpoints lie within distinct regions of $\mathcal{D}$, and hence lies within a small gap.

Suppose that for every $r \geq r^*$, at least one point p on the boundary of C_q^r lies within a large gap (Lemma 4.9 implies that we can ignore any $r < r^*$). If p lies

within some region of $\mathcal{D}$, then since p is within a large gap, no other region of $\mathcal{D}$ intersects the boundary of C_q^r; whereas if p is not within any region of $\mathcal{D}$, then p must lie within a boundary arc of C_q^r whose angle exceeds π, and which does not intersect any regions of $\mathcal{D}$. We can conclude that no feasible smallest bounding disc of $\mathcal{D}$ centered at q can have radius r and satisfy the support condition. Since this holds for all $r \geq r^*$, $q \notin \mathbb{C}^\circ$. □

We will now assume that $\mathcal{D}$ is a set of uncertain discs. Lemma 4.10 does not immediately motivate an algorithm for determining if $q \in \mathbb{C}^\circ$, since we cannot afford to examine every point on every disc C_q^r to see if it lies within a large gap. Fortunately, we can reduce our work by exploiting a close relationship between query point q, the support condition of feasible smallest bounding discs, and the farthest disc D_f from q.

Lemma 4.11 *Point q lies within $\mathbb{C}^\circ$ iff there exists some C_q^r, with $r \geq r^*$, such that the gaps at p_r and $\widetilde{p}_r$ are both small.*

Proof. We can use the proof of Lemma 4.10 to prove that if $q \in \mathbb{C}^\circ$, then some C_q^r exists (where $r \geq r^*$) with small gaps at p_r and $\widetilde{p}_r$. To prove the converse, suppose some C_q^r exists such that the gaps at p_r and $\widetilde{p}_r$ are both small, where $r \geq r^*$. If $r^* = 0$, we are done (by Lemma 4.9). Otherwise, we will show that $q \in \mathbb{C}^\circ$ by finding a critical subset for some disc centered at q whose radius is not less than r^*.

Let $\text{arc}(a, b)$ be the gap at p_r, $\text{arc}(c, d)$ be the gap at $\widetilde{p}_r$, and $\delta(x)$ denote the disc of $\mathcal{D}$ contributing gap endpoint $x \in \{a, b, c, d\}$. Since the gaps are minimal, the points $(p_r, b, c, \widetilde{p}_r, d, a, p_r)$ must appear in ccw sequence on C_q^r's boundary. Without loss of generality, there are five cases to consider.

Case 1: Each of the four points lie in distinct discs of $\mathcal{D}$. Three of the four points must form a critical subset for C_q^r.

Case 2: $\delta(a) = \delta(c)$. Either $\widetilde{b}$ or $\widetilde{d}$ must lie in $\delta(a)$, which implies that $(b, \widetilde{b})$ or $(d, \widetilde{d})$ is a critical subset for C_q^r.

Case 3: $\delta(a) = \delta(d)$. Either $\text{arc}(d, a)$ or $\overline{p_r \widetilde{p}_r}$ must lie within $\delta(a)$. In the former case, some $(x \in \text{arc}(d, a), b, c)$ is a critical subset for C_q^r. In the latter

case, $q \in \delta(a)$, which (since $r^* > 0$ and $q \notin D_f$) implies that $\delta(a) \neq D_f$, and $(p_{r^*} \in D_f, \widetilde{p}_{r^*} \in \delta(a))$ is a critical subset for $C_q^{r^*}$.

Case 4: $\delta(a) = \delta(c)$ and $\delta(b) = \delta(d)$. This can be dealt with using the same argument as case 2.

Case 5: $\delta(a) = \delta(d)$ and $\delta(b) = \delta(c)$. If $\text{arc}(d, a)$ does not lie within $\delta(a)$, or if $\text{arc}(b, c)$ does not lie within $\delta(b)$, we can apply the argument of case 3. Otherwise, some $(x \in \text{arc}(d, a), y \in \text{arc}(b, c))$ must be a critical subset for C_q^r. □

Lemma 4.11 does not hold for arbitrary antipodal pairs on the boundary of C_q^r. For example, in Figure 4.12, neither u nor $\widetilde{u}$ lie within large gaps, yet q is not within $\mathbb{C}^\circ(\mathcal{D})$.

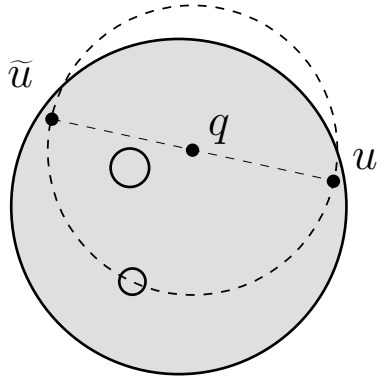

Figure 4.12: Lemma 4.11 does not hold for u, $\widetilde{u}$.

We now have the tools necessary to construct an efficient algorithm. Our first step is to apply the algorithm of Lemma 4.6 to remove any discs of $\mathcal{D}$ that lie within GH. Our second step will be to determine D_f, and from it, the value of r^*. If r^* is zero, then by Lemma 4.9, we return true. Otherwise, we perform additional steps to determine whether there exists an $r \geq r^*$ such that p_r and $\widetilde{p}_r$ both lie within small gaps. Per Lemma 4.11, we return true iff such an r is found.

We will construct *indicator gap functions* for both p_r and $\widetilde{p}_r$. The function for p_r, for example, is a function that indicates, for each $r \geq r^*$, whether the gap for p_r is large. We will actually derive the indicator gap function for a point by combining left and right *half gap functions*, each representing the angle of the gap lying to the left or right of ray $\overrightarrow{qp_{r^*}}$.

Our algorithm consists of the following steps:

1. Remove from $\mathcal{D}$ all but the k discs that may contribute possible 1-centers.
2. Determine D_f and r^*. If $r^* = 0$, return true.
3. Construct left and right half gap functions for p_r.
4. Combine half gap functions into indicator gap function for p_r.
5. Repeat steps 3 and 4 for $\widetilde{p}_r$.
6. Use the indicator gap functions to see if p_r and $\widetilde{p}_r$ both lie in small gaps for some $r \geq r^*$. If so, return true; else, return false.

Let us investigate the third step of the algorithm in more detail. We will consider only the left half gap function for p_r, as the others are symmetric. We construct a *disc curve* for each $D_i \in \mathcal{D}$ as follows. Let B be the portion of D_i lying to the left of $\overrightarrow{qp_{r^*}}$. If $B = \emptyset$, then we omit D_i. Otherwise, the disc curve is the portion of the boundary of B that lies to the right of $\overrightarrow{qo_i}$ (see Figure 4.13).

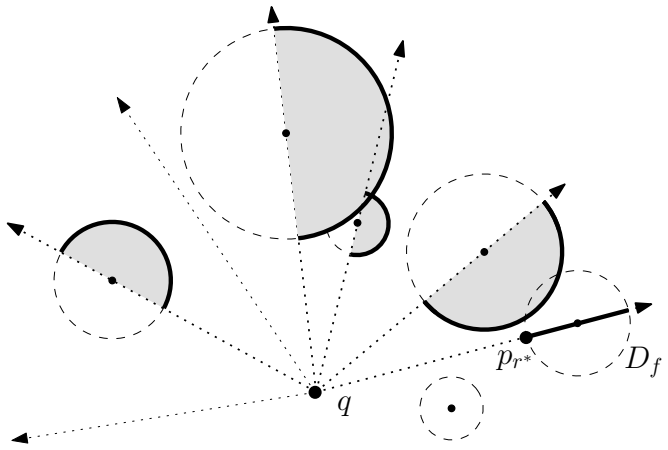

Figure 4.13: Disc curves of left half gap function for p_r.

Let Φ be the polar angle of $\overrightarrow{qo_f}$. Consider the homeomorphism H between polar coordinate points $(r \geq r^*, \Phi \leq \theta \leq \Phi + \pi)$ and cartesian coordinate points $(r \geq r^*, 0 \leq \theta \leq \pi)$. If we apply H to the disc curves, then the left half gap

function corresponds to the lower envelope, or 0-level, of the transformed disc curves.

By definition, the boundary points of the gap for p_r must lie in distinct discs of $\mathcal{D}$. This complicates matters, since it implies that the (full) gap function for p_r cannot be defined simply as the sum of the left and right half gap functions. To remedy this situation, we construct both the 0-level and 1-level of the arrangement of the transformed disc curves.

Note that for each value of r, a particular disc D_i will contribute a value to at most one of these levels. Hence, if we have these levels for both the left and right half gap functions, and we have recorded which disc has induced each edge of these levels, we can determine the value of the full gap function.

To clarify, for the left half gap function, we define three functions: $\theta_0^L(r)$, which returns the value of θ corresponding to the point of the 0-level at distance r from q; $\theta_1^L(r)$, which returns the corresponding value for the 1-level; and $\delta^L(r)$, which returns the disc of $\mathcal{D}$ corresponding to $\theta_0^L(r)$. We define analogous functions $\theta_0^R(r)$, $\theta_1^R(r)$, and $\delta^R(r)$ for the right half gap function. The value of the full gap function for a particular $r \geq r^*$ is now $\theta_0^L(r) + \theta_0^R(r)$ (if $\delta^L(r) \neq \delta^R(r)$) or $\min(\theta_0^L(r) + \theta_1^R(r), \theta_1^L(r) + \theta_0^R(r))$ (otherwise).

The fourth step of the algorithm, combining the left and right half gap functions for p_r into a full gap function, can be divided into two stages: a merge stage, and a convert stage. For the merge stage, we first take the four levels generated in the previous step (the 0-level and 1-level of the left and right half gap functions) and combine them to form the three functions $\theta_0^L(r) + \theta_0^R(r)$, $\theta_0^L(r) + \theta_1^R(r)$, and $\theta_1^L(r) + \theta_0^R(r)$. We modify the first of these so that for edges where the discs associated with the left and right 0-levels are the same (and are hence illegal), we return 2π. We next merge these three functions into the full gap function: for a particular r, it returns the gap angle for p_r as the minimum of the three constituent functions' values. In the convert stage, we convert the current half gap function (a continuous function of r) into a function that indicates whether the gap for p_r is large or small.

The final step of the algorithm is to perform a simultaneous scan of the indicator gap function sequences for p_r and $\widetilde{p}_r$, to see if there exists a value r where neither gap angle exceeds π. If one is found, we can return true immediately; otherwise,

when the scan is complete, we return false.

Let us determine the running time of our algorithm. By Lemma 4.6, step 1 requires $O(n \log k)$ time, where k is the number of discs not interior to GH. Step 2 clearly runs in $O(n)$ time. For step 3, we require the following lemma.

Lemma 4.12 *The 0-level of a half gap function of k uncertain discs has $O(k)$ complexity.*

Proof. Let $\mathcal{D}$ be a set of k discs. Without loss of generality, we assume we are dealing with the left half gap function of p_r, and that o_f lies on the positive x-axis. In addition, for ease of exposition, we will assume that no disc (other than D_f) intersects this axis. Let S be a sequence of representative interior points of each edge of the 0-level, ordered by their distance from the origin. Assume, by way of contradiction, that there exists a subsequence (a_1, b_2, a_3, b_4) of S, where a_1 and a_3 are points on the curve for $a \in \mathcal{D}$, and b_2 and b_4 are points on the curve for some other disc $b \in \mathcal{D}$. By proving that no such subsequence can exist, we will show that S is a Davenport-Schinzel sequence of length $\lambda_2(k)$, which implies that $|S| = O(k)$.

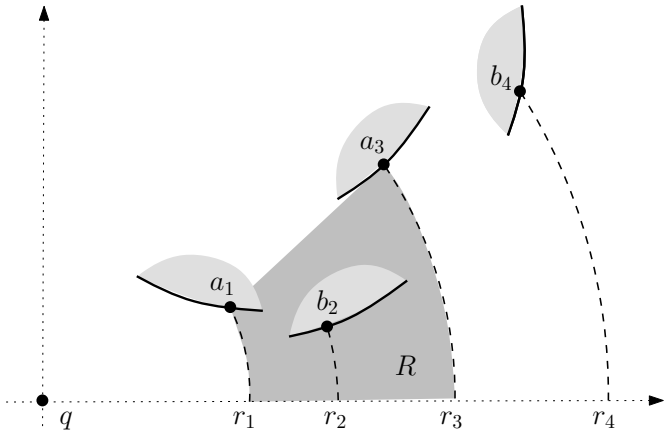

Figure 4.14: Lemma 4.12.

For each point of the subsequence, we construct an arc centered at q that extends from the positive x-axis to the point. Let the radii of these four arcs be $r_1 < r_2 < r_3 < r4$ (Figure 4.14). Consider the region R that is bounded by the

x-axis, segment $\overline{a_1a_3}$, and the arcs for r_1 and r_3. Note that b_2 must lie within R, otherwise it would not lie on a 0-level edge. It follows that segments $\overline{a_1a_3}$ and $\overline{b_2b_4}$ intersect. Now, these two segments are chords of a and b respectively, and it can be shown that if two such chords intersect, then at least one of the chord's endpoints must lie within the other chord's disc. This is a contradiction, since the endpoint in question cannot lie in the interior of a 0-level edge. □

Lemma 4.13 *The 1-level of a half gap function of k uncertain discs has $O(k\alpha(k))$ complexity.*

Proof. Observe that we can add almost-vertical rays extending from the endpoints of each transformed disc curve to produce an unbounded curve that yields the same 0- and 1-levels as the original transformed curves. Note also that each pair of such modified curves will intersect in at most two points. Hence, these curves can be viewed as *pseudoparabolas* [69], which implies that the 1-level of the half gap function has $O(k\alpha(k))$ complexity [3]. □

Lemma 4.12 implies that we can construct the 0-level for each half gap function in $O(k \log k)$ time, by modifying the standard divide-and-conquer algorithm of [6]. If we then clip away the edges forming the 0-level from the transformed disc curves, then the lower evelope of the curves that remain form the 1-level of the original transformed disc curves. This clipping can be performed in linear time, and Lemma 4.13 implies that we can use Hershberger's algorithm [31] to construct the 1-level in $O(k \log k)$ time.

The merge stage of step 4 can be performed in time linear in the complexity of the 0- and 1-levels, and hence runs in $O(k\alpha(k))$ time. The convert stage running time is linear in the size of its output, which is also $O(k\alpha(k))$:

Lemma 4.14 *Every indicator gap function of k uncertain discs has $O(k\alpha(k))$ complexity.*

Proof. The (output) indicator gap function is derived from the (input) original gap function. Since the input is represented by a sequence of length $O(k\alpha(k))$, we prove that the output sequence has the same size by showing that each input element generates $O(1)$ output elements.

We will represent the input gap function by a sequence of triples $((r_1, \delta_1^L, \delta_1^R)$, $(r_2, \delta_2^L, \delta_2^R), \ldots)$, where δ_i^L (resp., δ_i^R) is the disc inducing the left (resp., right) half gap angles for values $r_i \leq r < r_{i+1}$. Our task is to convert this sequence into an indicator gap function, a sequence of pairs $((r_1, f_1) \ldots)$ where each f_i indicates whether the gap is large for values immediately following r_i.

We can perform this task by iterating over the original gap function. Consider a particular step in this iteration, where the current triple is (r_u, D_i, D_j), and the following triple is $(r_v, \cdot, \cdot)$. At this point, we must determine those values of r between r_u and r_v where the gap function equals π. Geometrically, this is equivalent to finding points on the boundaries of discs D_i and D_j whose midpoint is q. Each such r corresponds to a point where the gap angle stops or starts being greater than π.

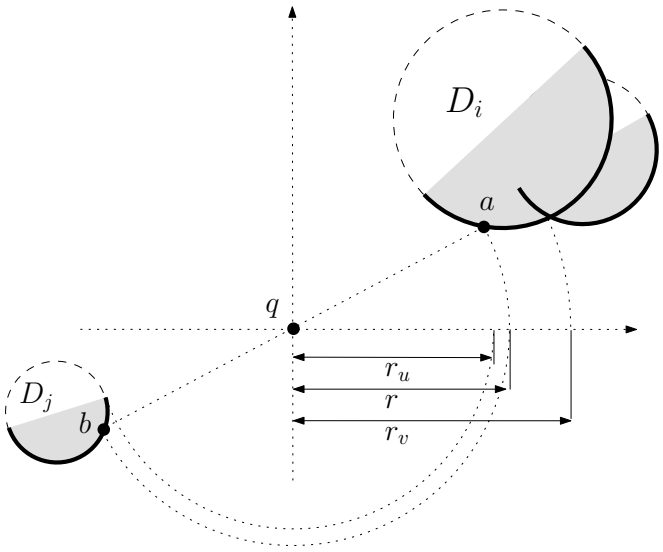

Figure 4.15: Lemma 4.14.

Figure 4.15 shows one triple being processed by the convert stage. Every value

r that generates a new output element must satisfy the following equations:

$$\begin{aligned} d(a, o_i) &= r_i \\ d(b, o_j) &= r_j \\ b_x &= -a_x \\ b_y &= -a_y \\ r &= d(q, a) \end{aligned}$$

These equations can be simplified to yield

$$a_y = c_1 a_x + c_2 \ ,$$

the equation of a line with constant coefficients

$$\begin{aligned} c_0 &= o_i.y + o_j.y \\ c_1 &= -(o_i.x + o_j.x)/c_0 \\ c_2 &= (o_i.x^2 + o_i.y^2 - r_i^2 - o_j.x^2 - o_j.y^2 + r_j^2)/c_0 \ . \end{aligned}$$

Since a must be one of the two possible points of intersection of this line with D_i, and each r corresponds to a unique point a, there are at most two values of r (and hence at most two output elements) generated per triple. □

The proof of Lemma 4.14 implies that step 4 runs in $O(k\alpha(k))$ time. Step 5 can be charged to steps 3 and 4. Step 6 involves a simple simultaneous traversal of the two indicator gap function sequences, and hence has the same running time as step 4. The running time of the complete algorithm is thus dominated by step 1, and we can make the following claim.

Theorem 4.15 *If $\mathcal{D}$ is a set of n uncertain discs, then it can be determined if a point q is within $\mathbb{C}^\circ(\mathcal{D})$ in $O(n \log k)$ time, where k is the number of discs of $\mathcal{D}$ that do not lie in the interior of $GH(\mathcal{D})$.*

4.3.1 Uncertain Axis-Aligned Rectangles

In this section, we show how the query algorithm can be adapted to the case where the uncertain regions are axis-aligned rectangles. We will require the following analog to Lemma 4.11 (its proof can be found in Appendix A).

Lemma 4.16 *If $\mathcal{D}$ is a set of uncertain axis-aligned rectangles, then point q lies within $\mathbb{C}^\circ$ iff there exists some C_q^r, with $r \geq r^*$, such that the gaps at p_r and $\widetilde{p}_r$ are both small.*

The structure of the query algorithm is the same as before, but some of the steps are different. Let us consider step 3: constructing the left and right half gap functions for p_r. As we did for the case of uncertain discs, we will consider only the left half gap function for p_r. For each rectangle D_i, we construct a set of *boundary edges* (analogous to the disc curves for uncertain discs). These are segments from the boundary of D_i, after portions lying to the right of $\overrightarrow{qp_{r^*}}$ are removed; see Figure 4.16.

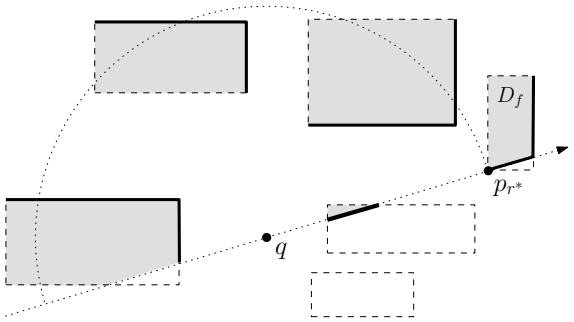

Figure 4.16: Boundary edges for axis-aligned rectangles.

Lemma 4.17 *The 0- and 1-levels of a half gap function of k uncertain axis-aligned rectangles have $O(k\alpha(k))$ complexity.*

Proof. There are $O(k)$ boundary edges induced by a set of k such rectangles, and since each such edge is a line segment, both levels of their arrangement have $O(k\alpha(k))$ complexity [3]. □

The 0- and 1-levels are derived from an arrangement of line segments. This implies that we can construct the 0-level in $O(k \log k)$ time, by using Hershberger's algorithm [31]. To construct the 1-level, we take the same approach as before: we clip away the portions of the line segments comprising the 0-level (in linear time), and construct the lower envelope of the remaining segments. By Lemma 4.17, we can use Hershberger's algorithm to construct the 1-level in $O(k \log k)$ time.

To prove that the algorithm for uncertain axis-aligned rectangles has the same running time as for uncertain discs, all that remains is to derive an analog to Lemma 4.14:

Lemma 4.18 *Every indicator gap function of k uncertain axis-aligned rectangles has $O(k\alpha(k))$ complexity.*

Proof. By Lemma 4.17, the gap function for such rectangles has $O(k\alpha(k))$ complexity. Hence, it will suffice to show that each element of this gap function induces $O(1)$ elements in the corresponding indicator gap function.

We can assume that the input gap function is represented by a sequence of triples $((r_1, \delta_1^L, \delta_1^R), (r_2, \delta_2^L, \delta_2^R), \ldots)$, where δ_i^L (resp., δ_i^R) is the boundary segment inducing the left (resp., right) half gap angles for values $r_i \leq r < r_{i+1}$. Suppose the gap function for some r is equal to π, where $(r_i, \delta_i^L, \delta_i^R)$ is an element in this sequence, and $r_i \leq r < r_{i+1}$. Observe that one of the boundary segments will be vertical, and the other will be horizontal.[2] Without loss of generality, we will assume that δ_r^L is horizontal, as in Figure 4.17. Then we have:

$$\begin{aligned} a_y &= c_1 \\ b_x &= c_2 \\ b_x &= -a_x \\ b_y &= -a_y \\ r &= d(q, a) \,, \end{aligned}$$

[2]Except when the two boundary segments lie on parallel lines equidistant from q; but in that case, the gap function for every r in the interval is equal to π, including the endpoints, and so does not contribute to the indicator gap function sequence.

where c_1 and c_2 are constants. These equations simplify to

$$r = \sqrt{c_1^2 + c_2^2}\,;$$

hence at most one value of r is generated for each element of the input gap function sequence. □

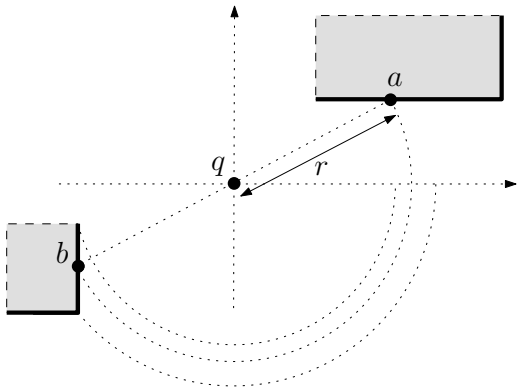

Figure 4.17: Lemma 4.18.

Since we have shown that the running time of each step of the algorithm is the same as it was in the case for uncertain discs, we can make the following claim.

Theorem 4.19 *If $\mathcal{D}$ is a set of n uncertain axis-aligned rectangles, then it can be determined if a point q is within $\mathbb{C}^\circ(\mathcal{D})$ in $O(n \log k)$ time, where k is the number of rectangles of $\mathcal{D}$ that do not lie in the interior of $GH(\mathcal{D})$.*

4.4 Construction Algorithm

In this section, we present our algorithm for constructing the possible 1-centers of a set of uncertain discs. By Theorem 4.4, the boundary of $\mathbb{C}^\circ$ lies on type I, II, and III curves. Our algorithm uses a plane sweep to construct the arrangement of these curves, then extracts the subset of the edges of the arrangement that form the boundary of $\mathbb{C}^\circ$. As we did for the previous algorithm, we first remove from $\mathcal{D}$ all but the k discs that do not lie in the interior of GH. For efficiency, we would like to minimize the number of possible intersections between the curves involved. Since

each curve III_{fij} lies in D_f's cell within the farthest disc Voronoi diagram of $\mathcal{D}$, if we restrict our attention to the portion of $\mathbb{C}^\circ$ lying within the cell for D_f, we need only consider intersections between pairs of $\binom{k}{2}$ possible type III curves (i.e., when f is fixed), as opposed to intersections between pairs of $\binom{k}{3}$ type III curves (i.e., when f can vary) within the complete $\mathbb{C}^\circ$. Since the number of intersections is a function of the square of the number of curves involved, this will improve our guaranteed worst-case running time by a factor of k, as we shall see.

Our algorithm has the following steps:

1. Remove from $\mathcal{D}$ all but the k discs that may contribute possible 1-centers.
2. Construct V, the farthest disc Voronoi diagram of $\mathcal{D}$.
3. For each cell $R_f \in V$:
   (a) Generate every type I, II, and III curve that may contain boundary points of $\mathbb{C}^\circ$ within R_f.
   (b) Construct A, the arrangement of these curves and the boundary of R_f.
   (c) For each edge E of A, choose point u in the interior of E, and points v and w lying very close to and on opposite sides of E. If u is within R_f, and exactly one of v and w lies within $\mathbb{C}^\circ$, output E.

Theorem 4.4 implies that the interior points of any particular cell of A are either all within, or all outside of, $\mathbb{C}^\circ$. Hence, we need only to determine which edges of A are boundary edges of $\mathbb{C}^\circ$. As we process each edge E of each cell R_f, we use the test of point u to verify that E lies within the cell. If this test is passed, then we verify that E is a boundary of $\mathbb{C}^\circ$ by testing points v and w. We can conclude that the algorithm constructs the boundary edges of $\mathbb{C}^\circ$.

The running time of step 1 is $O(n \log k)$ (Lemma 4.6). Step 2 takes $O(k \log k)$ time [60]. For a particular cell $R_f \in V$, there are $O(k)$ candidate hyperbolic arcs containing type I curves of the form I_{f*} that may intersect the cell, and $\binom{k}{2}$ pairs of discs (D_i, D_j) associated with curves II_{ij} and III_{fij} that may intersect the cell. The total number of curves generated in step 3(a) for all $O(k)$ cells of V is thus $O(k^2)$.

Since each curve is a polynomial curve of bounded degree, the number of intersections occurring between pairs of curves is at most $O(k^4)$. If we assume that (i) each curve can be split into $O(1)$ pieces, each monotonic in x, in $O(1)$ time, and (ii) all intersections between any pair of these pieces can be determined in $O(1)$ time, then A can be constructed in step 3(b) by using the plane sweep algorithm of Clarkson and Shor [16] or Mulmuley [53] in expected $O(k^4)$ time.

We can determine which of points u, v, and w lie within $\mathbb{C}^\circ(\mathcal{D})$, in $O(k \log k)$ time, by using the algorithm of Section 4.3. Since there are $O(k^4)$ edges in A, step 3(c) takes $O(k^5 \log k)$ time; and since there are $O(k)$ cells in V, the total expected time spent in step 3 is $O(k^6 \log k)$. The running time of the algorithm is thus dominated by steps 1 and 3, and we can make the following claim.

Theorem 4.20 *The possible 1-centers of n uncertain discs $\mathcal{D}$ can be constructed in expected $O((n+k^6)\log k)$ time, where k is the number of discs of $\mathcal{D}$ that do not lie in the interior of $GH(\mathcal{D})$.*

4.4.1 Uncertain Axis-Aligned Rectangles

We now consider the case where each $D_i \in \mathcal{D}$ is an (axis-aligned) rectangle (Figure 4.18). Observe that since each rectangle has nonzero area, a rectangle can be outside-tangent to a disc at either a vertex or a point along an edge, whereas it can be locally inside-tangent to a disc only at a vertex. Note also that with these rectangles, local inside tangency does not imply 'normal' inside tangency.

Lemma 4.21 *If $\mathcal{D}$ is a set of axis-aligned rectangles, then each curve I_{fi} lies on $O(1)$ lines and parabolas, and each curve II_{ij} lies on the boundary of a rectangle.*

Proof. By definition, each point $s \in I_{fi}$ is the center of a disc that has outside-tangent (at either a corner or an edge) D_f and locally inside-tangent (at a corner) D_i. Each corner/corner combination lies on the bisectors of the corners, whereas each edge/corner combination lies on a parabola induced by the edge and corner. The claim for II_{ij} follows from the fact that the Minkowski average of a pair of axis-aligned rectangles is also a rectangle. □

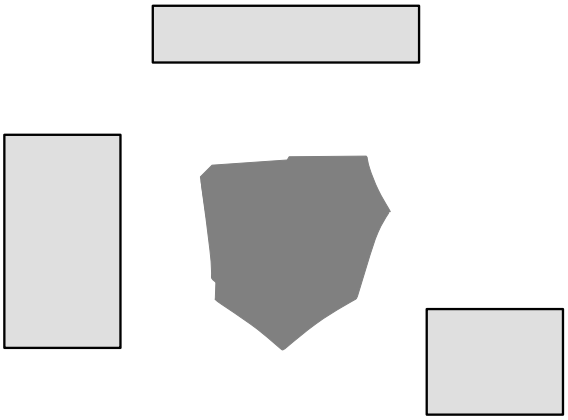

Figure 4.18: Possible 1-centers of uncertain rectangles.

Lemma 4.22 *If $\mathcal{D}$ is a set of axis-aligned rectangles, then every point on the boundary of $\mathbb{C}^\circ$ that is not on a type I or type II curve will lie on one of a small number of lines or parabolas.*

Proof. Let p be a point on the boundary of $\mathbb{C}^\circ$ that is not on a type I or type II curve. By Theorem 4.4, there must exist a feasible set S of $\mathcal{D}$ with a critical subset of boundary sites $\{s_i, s_j\}$, and $\mathbb{B}(S)$ must have outside-tangent some D_f. Without loss of generality, we can assume that $s_i.x \geq s_j.x$. Observe that s_i and s_j cannot both lie on the left sides of their respective rectangles, otherwise p would lie on I_{ij}. If s_i lies in the interior of its rectangle's left side, and s_j lies in the interior of its rectangle's right side, then it is possible to move s_i and s_j in opposite directions along these sides, to form a new feasible set S' with the same 1-center as S, but with a larger (smallest) bounding disc (Figure 4.19). It is then possible to move s'_i and s'_j directly away from each other to yield a feasible set S'' whose 1-center remains at p but that has no boundary sites, contradicting Lemma 4.2.

Symmetric arguments hold for the cases where s_i and s_j both lie on the right, top, or bottom sides of their respective rectangles, or when s_i and s_j both lie in the interior of horizontal sides. Hence it must be the case that s_i and s_j lie on sides of their respective rectangles that have opposite orientations (i.e., one vertical, and one horizontal).[3]

[3]This includes the rectangle corners, which are intersections of edges of both orientations.

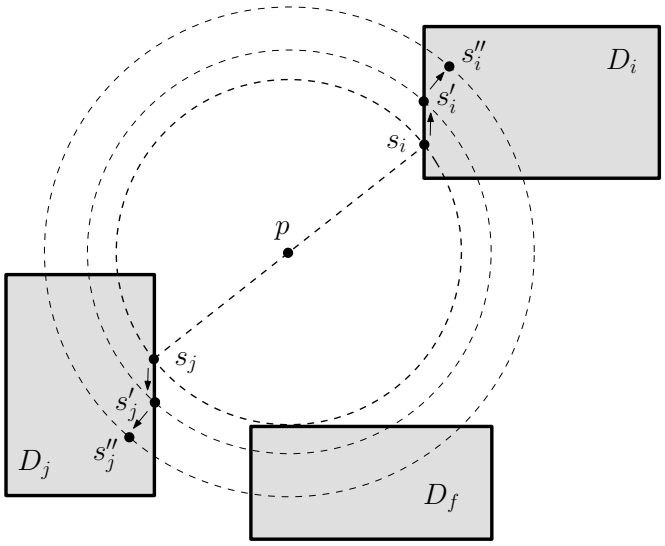

Figure 4.19: Lemma 4.22.

D_f can be outside-tangent $\mathbb{B}(S)$ in two ways. First, the point of tangency can be at a corner (x_f, y_f) of D_f. In this case, we can assume without loss of generality that (i) s_i lies on a horizontal side of D_i at position (x, y_i), (ii) s_j lies on a vertical side of D_j at position (x_j, y), where x_f, y_f, x_j, and y_i are constants (Figure 4.20). Then

$$\begin{aligned} p_x &= \frac{1}{2}(x + x_j) \\ p_y &= \frac{1}{2}(y_i + y) \\ \sqrt{(p_x - x_f)^2 + (p_y - y_f)^2} &= \frac{1}{2}\sqrt{(x - x_j)^2 + (y_i - y)^2}\,. \end{aligned}$$

This can be simplified to yield

$$p_y = c_1 p_x + c_2\,,$$

the equation of a line with constant coefficients

$$c_1 = (x_j - x_f)/(y_f - y_i)$$
$$c_2 = (x_f^2 + y_f^2 - x_j^2 + 1)/(2(y_f - y_i)) \,.$$

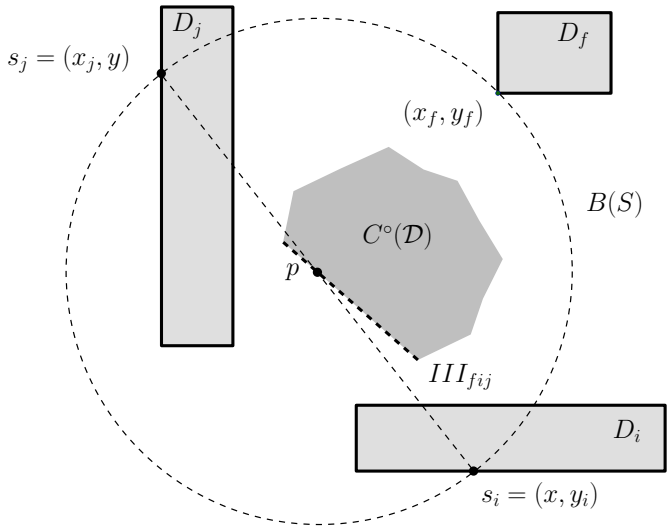

Figure 4.20: III_{fij} is a line.

Alternatively, D_f can be outside-tangent $\mathbb{B}(S)$ along a side of D_f. Then without loss of generality, we can make the same assumptions as in the previous case, except that now $\mathbb{B}(S)$ is outside-tangent a horizontal side of D_f lying on the line $y = y_f$ (Figure 4.21). Our system of equations now becomes

$$p_x = \frac{1}{2}(x + x_j)$$
$$p_y = \frac{1}{2}(y_i + y)$$
$$|p_y - y_f| = \frac{1}{2}\sqrt{(x - x_j)^2 + (y_i - y)^2} \,.$$

This can be simplified to yield

$$p_y = c_1 p_x^2 + c_2 p_x + c_3 \ ,$$

the equation of a parabola with constant coefficients

$$\begin{aligned} c_1 &= 1/(2(y_i - y_f)) \\ c_2 &= -2x_j c_1 \\ c_3 &= (x_j^2 - y_f^2 - 1)c_1 \ . \end{aligned}$$

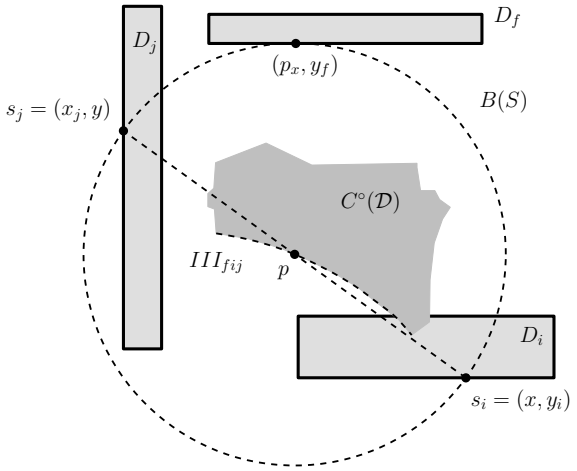

Figure 4.21: III_{fij} is a parabola.

As there is a small number of possible choices of sides of D_i and D_j and sides or corners of D_f associated with III_{fij}, every such boundary point p must lie on some small number of lines and parabolas. □

The preceding results allow us to make the following claim.

Theorem 4.23 *The possible 1-centers of n uncertain axis-aligned rectangles $\mathcal{D}$ can be constructed in expected $O((n + k^6) \log k)$ time, where k is the number of rectangles of $\mathcal{D}$ that do not lie in the interior of $GH(\mathcal{D})$.*

4.5 Closing Remarks

In this chapter, we studied a variant of a classical facility location problem, the smallest bounding disc of a set of points, in which each site lies in a region of uncertainty. We introduced a new geometric figure: the possible 1-centers of uncertain regions. In addition to presenting two algorithms related to this figure, we also demonstrated a connection between this figure and the guaranteed hull of the regions.

We have identified a number of open problems concerning possible 1-centers of uncertain discs. We do not, as yet, have bound on the complexity of these structures that is tighter than the $O(n^5)$ bound implied by our construction algorithm (Section 4.4). We also have not been able to prove that $\mathbb{C}^\circ$ is simply connected (i.e., has no holes), although we conjecture that this is true. Our construction algorithm can be simplified if this is the case.

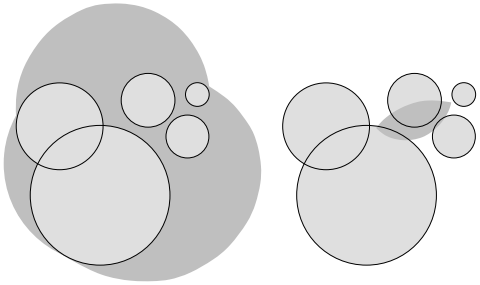

Figure 4.22: Possible bounding disc, guaranteed bounding disc (uncertain discs).

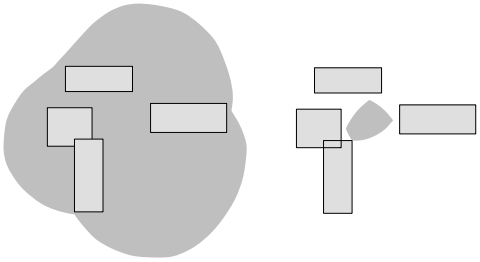

Figure 4.23: Possible bounding disc, guaranteed bounding disc (uncertain axis-aligned rectangles).

Two additional problems related to possible 1-centers that are candidates for future research involve what we call the *possible bounding disc* and the *guaranteed bounding disc* of uncertain regions (Figures 4.22 and 4.23). We define the first as the union of all feasible smallest bounding discs (in contrast to the origins of these discs, which form the regions' possible 1-centers). We define the second as the intersection of all feasible smallest bounding discs.

An applet exploring some of the geometric objects described in this chapter can be found at http://www.cs.ubc.ca/~jpsember/ocent.html. The applet (Figure 4.24) includes the following sample files:

1. `p1c.dat`: Generates possible 1-centers of a set of uncertain discs. The program uses a simple (and slow) random sampling process to do this, instead of the algorithm described in Section 4.4.

2. `p1c_rects.dat`: Generates possible 1-centers of a set of uncertain axis-aligned rectangles.

3. `psbd.dat`: Constructs the possible bounding disc of a set of uncertain discs or axis-aligned rectangles.

4. `g1c.dat`: Constructs the guaranteed bounding disc for a set of uncertain discs or axis-aligned rectangles.

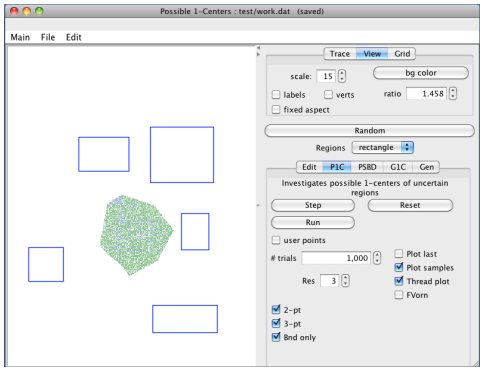

Figure 4.24: Applet for Chapter 4.

Chapter 5

Conclusion and Future Research

In this thesis, we considered problems from computational geometry involving point sets, and studied variants of the problems where each point's location was replaced by a region of uncertainty. We can often view the output of each of the original problems as being a partition of the plane into two sets: points that satisfy some property (e.g., lie within the convex hull of the points), and those that do not.[1] The partition of the plane associated with the output of our modified problems includes a third set: points that we cannot make any claims about with respect to the property. We have referred to the points within the first of these sets as a *guaranteed* object, and to the points in the first or third of these sets as a *possible* object.

In the introduction, we showed that imprecision is an unavoidable factor in most geometric algorithms, whether as a result of imprecision in the input, or as imprecision introduced by the representation and manipulation of numbers by the algorithm. By classifying the output of an algorithm as either a guaranteed or possible object, an algorithm may more precisely represent those properties that can be safely inferred from the input data. In each of the next three chapters, we applied this technique to a different fundamental computational geometry problem.

In Chapter 2, we focused on Voronoi diagrams of uncertain sites. We identi-

[1] Unless the output of the algorithm is combinatoric in nature, in which case the set being partitioned is also combinatoric. For example, the Delaunay triangulation of a point set is a subset of the edges of the complete graph of the sites.

fied as an area of future research the possibility of generating order-k guaranteed Voronoi diagrams, and discussed some of the issues likely to be encountered while doing so. We also left as future work the problem of constructing guaranteed Voronoi diagrams for non-disjoint uncertain polygons. We defined guaranteed variants of two combinatoric structures related to Voronoi diagrams: guaranteed Delaunay edges, and guaranteed Gabriel edges. We briefly discussed minimum spanning trees and relative neighborhood graphs, two other combinatoric structures associated with point sites that may yield interesting variants when the sites are imprecise.

In Chapter 3, we addressed two variants of the convex hull problem involving uncertain sites: guaranteed hulls, and possible hulls. We highlighted the fact that existing algorithms for generating guaranteed hulls are closely tied to hull bitangents. In some situations this may be undesirable, since there may be a large number of hull bitangents that do not contribute to the guaranteed hull. Finding a guaranteed hull algorithm with a reduced dependency on these hull bitangents is thus an open problem. We also left as an open problem the task of tightening the $O(n\alpha(n))$ bound on the number of hull bitangents of uncertain polygons.

In Chapter 4, we considered problems related to smallest bounding discs of imprecise points. Some fundamental questions regarding possible 1-centers of uncertain discs (or uncertain axis-aligned rectangles) remain: What is the complexity of this structure? And, is it simply connected?

In each of the three preceding chapters, we have referred to software programs that demonstrate the research presented in this thesis. The programs are written in Java, and are built upon a geometry editor and algorithm debugging framework that can be found at http://www.cs.ubc.ca/~jpsember/testbed.html.

This framework lets users create, manipulate, and experiment with sets of geometric objects in a visual manner, to gain insight into a wide range of computational geometry problems. Many of the results we presented in this thesis were sparked by observations made while using this software.

Bibliography

[1] M. Abellanas, F. Hurtado, and P. A. Ramos. Tolerance of geometric structures. In *Proceedings of the 6th Canadian Conference on Computational Geometry*, pages 250–255, 1994. → pages 16

[2] M. Abellanas, F. Hurtado, and P. A. Ramos. Structural tolerance and Delaunay triangulation. *Information Processing Letters*, 71(5-6):221–227, 1999. → pages 16

[3] P. K. Agarwal, B. Aronov, T. M. Chan, and M. Sharir. On levels in arrangements of lines, segments, planes, and triangles. *Discrete and Computational Geometry*, 19(3):315–331, 1998. → pages 147, 150

[4] M. E. Ali, E. Tanin, R. Zhang, and K. Ramamohanarao. Probabilistic Voronoi diagrams for processing moving nearest neighbor queries. Technical report, University of Melbourne, Melbourne, Australia, 2009. → pages 15

[5] T. Asano, J. Matoušek, and T. Tokuyama. Zone diagrams: Existence, uniqueness, and algorithmic challenge. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, volume 37, pages 756–765, 2007. → pages 15

[6] M. Atallah and C. Bajaj. Efficient algorithms for common transversals. *Information Processing Letters*, 25(2):87–91, 1987. → pages 85, 147

[7] F. Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, 1991. → pages 12

[8] F. Aurenhammer, G. Stöckl, and E. Welzl. The post office problem for fuzzy point sets. In *Proceedings of the International Workshop on Computational Geometry - Methods, Algorithms and Applications*, pages 1–11, London, UK, 1991. Springer-Verlag. → pages 14

[9] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The Quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996. → pages 71

[10] B. Bhattacharya, S. Jadhav, A. Mukhopadhyay, and J. Robert. Optimal algorithms for some intersection radius problems. *Computing*, 52(3): 269–279, 1994. → pages 129

[11] J. Boisson, A. Cérézo, O. Devillers, J. Duquesne, and M. Yvinec. An algorithm for constructing the convex hull of a set of spheres in dimension d. *Computational Geometry: Theory and Applications*, 6(2):123–130, 1996. → pages 99, 100, 125

[12] K. Q. Brown. Voronoi diagrams from convex hulls. *Information Processing Letters*, 9(5):223–228, 1979. → pages 11

[13] K. Buchin, M. Löffler, P. Morin, and W. Mulzer. Delaunay triangulation of imprecise points simplified and extended. In *Proceedings of the 11th International Symposium on Algorithms and Data Structures*, pages 131–143, Berlin, Heidelberg, 2009. Springer-Verlag. → pages 16

[14] E. W. Chambers, A. Erickson, S. P. Fekete, J. Lenchner, J. Sember, V. Srinivasan, U. Stege, S. Stolpner, C. Weibel, and S. Whitesides. Connectivity graphs of uncertainty regions. In *Proceedings of the 21st International Symposium on Algorithms and Computation*, 2010. → pages 68

[15] T. M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete and Computational Geometry*, 16:361–368, 1996. → pages 72, 75, 86, 93, 94

[16] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, ii. *Discrete and Computational Geometry*, 4: 387–421, 1995. → pages 154

[17] H. Davenport and A. Schinzel. A combinatorial problem connected with differential equations. *American Journal of Mathematics*, 87(3):684–694, 1965. → pages 6

[18] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, second edition, 2000. → pages 31

[19] O. Devillers. Delaunay triangulation of imprecise points, preprocess and actually get a fast query time. In *XIV Spanish Meeting on Computational Geometry*, 2011. → pages 16

[20] R. L. Drysdale and D. T. Lee. Generalized Voronoi diagrams in the plane. In *Proceedings of the 16th Annual Allerton Conference on Communications, Control and Computing*, pages 833–842, 1978. → pages 14

[21] A. Edalat and A. Lieutier. Foundation of a computable solid modelling. *Theoretical Computer Science*, 284(2):319–345, 2002. → pages 73

[22] A. Edalat, A. Lieutier, and E. Kashefi. The convex hull in a new model of computation. In *Proceedings of the 13th Canadian Conference on Computational Geometry*, pages 93–96, 2001. → pages 73

[23] A. Edalat, A. A. Khanban, and A. Lieutier. Computability in computational geometry. In *Conference on Computability in Europe*, pages 117–127, 2005. → pages 73, 74

[24] J. S. Ely and A. P. Leclerc. Correct Delaunay triangulation in the presence of inexact inputs and arithmetic. *Reliable Computing*, 6(1):23–38, 2000. → pages 15

[25] E. Ezra and W. Mulzer. Convex hull of imprecise points in $o(n \log n)$ time after preprocessing. In *Proceedings of the 27th European Workshop on Computational Geometry*, 2011. → pages 74

[26] S. Fortune. A sweepline algorithm for Voronoi diagrams. In *Proceedings of the 2nd Annual Symposium on Computational Geometry*, pages 313–322, New York, NY, USA, 1986. ACM. → pages 11, 14, 25, 36, 42, 51

[27] K. R. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Biology*, 18(3):259–278, 1969. → pages 14

[28] B. Gärtner. Fast and robust smallest enclosing balls. In *Proceedings of the 7th Annual European Symposium on Algorithms*, pages 325–338, 1999. → pages 127

[29] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132 – 133, 1972. → pages 71, 93

[30] L. Guibas, D. Salesin, and J. Stolfi. Epsilon geometry: Building robust algorithms from imprecise computations. In *Proceedings of the Fifth Annual Symposium on Computational Geometry*, 1989. → pages 4

[31] J. Hershberger. Finding the upper envelope of n line segments in $o(n \log n)$ time. *Information Processing Letters*, 33(4):169–174, 1989. → pages 147, 151

[32] S. Jadhav, A. Mukhopadhyay, and B. Bhattacharya. An optimal algorithm for the intersection radius of a set of convex polygons. *Journal of Algorithms*, 20(2):244 – 267, 1996. → pages 129

[33] R. A. Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2(1):18–21, 1973. → pages 71

[34] V. Karamcheti, C. Li, I. Pechtchanski, and C. Yap. A Core library for robust numeric and geometric computation. In *ACM Symposium on Computational Geometry, Applied Track*, 1999. → pages 3

[35] M. I. Karavelas and M. Yvinec. Dynamic additively weighted Voronoi diagrams in 2d. In *Proceedings of the 10th Annual European Symposium on Algorithms*, pages 586–598, 2002. → pages 41

[36] A. Khanban. *Basic Algorithms of Computational Geometry with Imprecise Input*. PhD thesis, Imperial College of London, 2005. → pages 15, 16

[37] A. Khanban and A. Edalat. Computing Delaunay triangulation with imprecise input data. In *Proceedings of the 15th Canadian Conference on Computational Geometry*, pages 94–97, 2003. → pages 15

[38] D. G. Kirkpatrick. Efficient computation of continuous skeletons. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 18–27, 1979. → pages 12, 14, 36

[39] D. G. Kirkpatrick and J. D. Radke. A framework for computational morphology. In G. T. Toussaint, editor, *Computational Geometry*, pages 217–248, Amsterdam, Netherlands, 1985. North-Holland. → pages 68

[40] D. G. Kirkpatrick and R. Seidel. The ultimate planar convex hull algorithm. *SIAM Journal on Computing*, 15(1):287–299, 1986. → pages 72

[41] R. Klein, K. Mehlhorn, and S. Meiser. Randomized incremental construction of abstract Voronoi diagrams. *Computational Geometry: Theory and Applications*, 3(3):157–184, 1993. → pages 41

[42] D. E. Knuth. *The Art of Computer Programming, Volume III: Sorting and Searching*. Addison-Wesley, 1973. → pages 11

[43] D. T. Lee and B. J. Schachter. Two algorithms for constructing a Delaunay triangulation. In *International Symposium on Computer and Information Sciences*, 1980. → pages 11

[44] E. H. Lockwood. *A Book of Curves*. Cambridge University Press, Cambridge, England, 1967. → pages 57

[45] M. Löffler. *Data Imprecision in Computational Geometry*. PhD thesis, Utrecht University, 2009. → pages 1

[46] M. Löffler and J. Snoeyink. Delaunay triangulation of imprecise points in linear time after preprocessing. *Computational Geometry: Theory and Applications*, 43(3):234–242, 2010. → pages 16

[47] M. Löffler and M. J. van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 56(2):235–269, 2010. → pages 7, 74

[48] M. Löffler and M. J. van Kreveld. Largest bounding box, smallest diameter, and related problems on imprecise points. In *Proceedings of the 10th Workshop on Algorithms and Data Structures*, pages 447–458, 2007. → pages 129

[49] N. Megiddo. Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983. → pages 129

[50] A. A. Melkman. On-line construction of the convex hull of a simple polyline. *Information Processing Letters*, 25(1):11–12, 1987. → pages 109, 122

[51] R. E. Miles. On the homogeneous planar Poisson point process. *Mathematical Biosciences*, 6:85 – 127, 1970. → pages 14

[52] R. E. Moore. *Interval Analysis*. Prentice-Hall, 1966. → pages 4

[53] K. Mulmuley. A fast planar partition algorithm, ii. *Journal of the ACM*, 38 (1):74–103, 1991. → pages 154

[54] T. Nagai and N. Tokura. Tight error bounds of geometric problems on convex objects with imprecise coordinates. In *Revised Papers from the Japanese Conference on Discrete and Computational Geometry*, pages 252–263, London, UK, 2001. Springer-Verlag. → pages 74, 75, 82, 83, 86, 93

[55] T. Nagai, Y. Seigo, and N. Tokura. Convex hull problem with imprecise input. In *Revised Papers from the Japanese Conference on Discrete and Computational Geometry*, pages 207–219, London, UK, 2000. Springer-Verlag. → pages 74, 76, 83, 85, 93, 100, 101, 125

[56] F. Nielsen and M. Yvinec. An output-sensitive convex hull algorithm for planar objects. *International Journal of Computational Geometry and Applications*, 8:39–65, 1995. → pages 93

[57] F. P. Preparata. An optimal real-time algorithm for planar convex hulls. *Communications of the ACM*, 22(7):402–405, 1979. → pages 87

[58] F. P. Preparata and S. J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Communications of the ACM*, 20(2):87–93, 1977. → pages 71

[59] F. P. Preparata and D. E. Muller. Finding the intersection of n half-spaces in time $o(n \log n)$. *Theoretical Computer Science*, 8:45–55, 1979. → pages 93, 99

[60] D. Rappaport. Computing the furthest site Voronoi diagram for a set of discs (preliminary report). In *Proceedings of the Workshop on Algorithms and Data Structures*, pages 57–66, 1989. → pages 153

[61] D. Rappaport. A convex hull algorithm for discs, and applications. *Computational Geometry: Theory and Applications*, 1(3):171–187, 1992. → pages 73, 74, 99

[62] H. Rosenberger. Order-k Voronoi diagrams of sites with additive weights in the plane. *Algorithmica*, 6(4):490–521, 1991. → pages 14, 49, 51

[63] J. Sember and W. Evans. Guaranteed Voronoi diagrams of uncertain sites. In *Proceedings of the 20th Canadian Conference on Computational Geometry*, pages 207–210, 2008. → pages 17

[64] M. I. Shamos. *Computational Geometry.* PhD thesis, New Haven, CT, USA, 1978. → pages 87

[65] M. I. Shamos and D. Hoey. Closest-point problems. In *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, pages 151–162, Washington, DC, USA, 1975. IEEE Computer Society. → pages 11, 14, 26, 30, 129

[66] M. Sharir. Intersection and closest-pair problems for a set of planar discs. *SIAM Journal on Computing*, 14(2):448–468, 1985. → pages 12, 14, 24, 43

[67] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and their Geometric Applications*. Cambridge University Press, New York, NY, USA, 2010. → pages 6, 83, 84

[68] J. J. Sylvester. A question in the geometry of situation. 1, 1857. → pages 129

[69] H. Tamaki and T. Tokuyama. How to cut pseudo-parabolas into segments. In *Proceedings of the Eleventh Annual Symposium on Computational Geometry*, pages 230–237, New York, NY, USA, 1995. ACM. → pages 147

[70] T. Tao. *Analysis, Volume II*. Hindustan Book Agency, 2006. → pages 130

[71] G. T. Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern Recognition*, 12(4):261 – 268, 1980. → pages 67

[72] G. T. Toussaint. Solving geometric problems with the rotating calipers. In *Proceedings IEEE MELECON*, 1983. → pages 71, 122

[73] R. Wein. Efficient implementation of red-black trees with split and catenate operations. Technical report, Tel-Aviv University, Tel-Aviv, Israel, 2005. → pages 92

[74] E. Welzl. Smallest enclosing disks (balls and ellipsoids). *New Results and New Trends in Computer Science*, pages 359–370, 1991. → pages 129

[75] Y. Yang, M. Lin, J. Xu, and Y. Xie. Minimum spanning tree with neighborhoods. In *Proceedings of the 3rd International Conference on Algorithmic Aspects in Information and Management*, pages 306–316, Berlin, Heidelberg, 2007. Springer-Verlag. → pages 68

[76] C. K. Yap. An $o(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments. *Discrete and Computational Geometry*, pages 365–393, 1987. → pages 36

[77] C. K. Yap. Towards exact geometric computation. *Computational Geometry*, 7(1-2):3 – 23, 1997. → pages 4

Appendix A

Proof of Lemma 4.16

We will prove that if $\mathcal{D}$ is a set of uncertain axis-aligned rectangles, then point q lies within $\mathbb{C}^\circ$ iff there exists some C_q^r, with $r \geq r^*$, such that the gaps at p_r and $\widetilde{p}_r$ are both small. As was the case for Lemma 4.11, we can use the proof of Lemma 4.10 to prove that if $q \in \mathbb{C}^\circ$, then some C_q^r exists (where $r \geq r^*$) with small gaps at p_r and $\widetilde{p}_r$. We prove the converse as follows.

Without loss of generality, we will assume that p_r lies within quadrant 1. Our proof will manipulate a disc C_q that is centered at q and intersects D_f (and hence satisfies the intersection condition of smallest bounding discs). It will then show that either C_q, or some smaller disc centered at q, satisfies the support condition as well. This will allow us to conclude that $q \in \mathbb{C}^\circ$.

We can assume that p_{r^*} lies within a small gap arc(a, b), and that $\widetilde{p}_{r^*}$ lies within a small gap arc(c, d). We will make use of these facts:

(i) If a particular (axis-aligned) rectangle D_i intersects C_q's boundary in a particular quadrant, then every smaller disc at q that intersects D_i will also do so in that quadrant.

(ii) If some D_i intersects C_q's boundary in other than the first quadrant, then $D_i \neq D_f$.

(iii) If the boundary of some D_i intersects C_q at points u and v lying in the same quadrant, then arc(u, v) must lie within D_i as well.

(iv) If the boundary of some D_i intersects C_q at points u and v in opposing quadrants (i.e., quadrants 1 and 3, or 2 and 4), then D_i must contain q, and the boundary of every smaller disc at q will intersect D_i at points u' and v' that have the same polar angle (with respect to q) as u and v respectively (Figure A.1).

(v) If the boundary of some D_i intersects C_q at points u and v that lie in adjacent quadrants (e.g., 1 and 2), then every smaller disc at q whose boundary intersects D_i must to so at points u' and v', where u' and v' lie within the (non-reflex) wedge formed by q, u, and v.

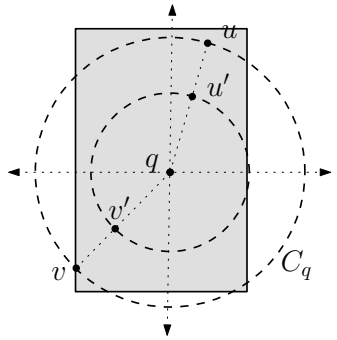

Figure A.1: Fact (iv).

Our proof will have a number of cases, based upon the quadrants containing points a, b, c, and d, and the rectangles of $\mathcal{D}$ associated with these points. For convenience, we will include a subscript with each of these points denoting its quadrant. For example, 'b_3' refers to a point b lying within quadrant 3.

Some of our arguments involve shrinking the original disc C_q. As the disc shrinks, its boundary will not contain any of the original gap endpoints (e.g., b_3); but by fact (i), its boundary will contain some other point from the same rectangle, and in the same quadrant. For brevity, we will refer to this new boundary point by the same name and subscript (b_3).

There are 15 distinct cases (once the symmetric ones have been removed), based upon the quadrants containing the gap endpoints a, b, c, and d. Each of these cases has seven subcases, based upon which rectangles are associated with these endpoints (Table A.1).

Subcase	Properties
1	$\delta(a) = \delta(c)$
2	$\delta(a) = \delta(d)$
3	$\delta(b) = \delta(c)$
4	$\delta(b) = \delta(d)$
5	$\delta(a) = \delta(c)$ and $\delta(b) = \delta(d)$
6	$\delta(a) = \delta(d)$ and $\delta(b) = \delta(c)$
7	a, b, c, and d lie in distinct rectangles of $\mathcal{D}$

Table A.1: Proof of Lemma 4.16, Subcases.

Case	Subcase	Proof steps
$a_1\ b_1\ c_1\ d_3$	1	One of (b_1, c_1, d_3) or (d_3, a_1, b_1) is a critical subset of C_q (to see this, consider rotating line $\overline{p_r \widetilde{p}_r}$ ccw around q until it reaches either b_1 or d_3).
	2,5,6	Shrink C_q until either (c_1, d_3) is a critical subset; or until D_f becomes outside-tangent to C_q, at which time (p_{r^*}, c_1, d_3) is a critical subset (d_3 must remain to the right of $\overrightarrow{qp_{r^*}}$, by fact (iv); c_1 must remain to its left, since arc(c_1, d_3) is not a large gap; and $D_f \notin \{D_{c_1}, D_{d_3}\}$, by fact (ii)).
	3,4	(a_1, c_1, d_3) is a critical subset.
$a_1\ b_1\ c_2\ d_3$	1	(b_1, c_2, d_3) or (d_3, a_1, b_1) (is a critical subset of C_q).
	2,4,5,6	Shrink C_q until (p_{r^*}, c_2, d_3) (is a critical subset of C_q).
	3	(a_1, c_2, d_3) (is a critical subset of C_q).
$a_1\ b_1\ c_2\ d_4$	1,2	(b_1, c_2, d_4).
	3,4	(a_1, c_2, d_4).
	5,6	Shrink C_q until (c_2, d_4) or (p_{r^*}, c_2, d_4).
$a_1\ b_1\ c_3\ d_3$	1	(b_1, c_3, d_3) or (d_3, a_1, b_1).
	2,6	Shrink C_q until (c_3, d_3) or (a_1, c_3) or (p_{r^*}, c_3, d_3).
	3,5	Shrink C_q until (c_3, d_3) or (p_{r^*}, c_3, d_3).
	4	(a_1, c_3, d_3) or (c_3, a_1, b_1).
$a_1\ b_2\ c_2\ d_3$	1,3	(d_3, a_1, b_2).
	2,6	Shrink C_q until (c_2, d_3) or (p_{r^*}, c_2, d_3).
	4	(a_1, c_2, d_3).
	5	Shrink C_q until (a_1, d_3), (c_2, d_3), (p_{r^*}, c_2, d_3), or (p_{r^*}, d_3, a_1).

Table A.2: Proof of Lemma 4.16, Cases $1 \ldots 5$.

In subcase 7, C_q must satisfy the support condition; hence we omit this subcase. The proofs for each of the 15 cases, and the remaining 6 subcases, are given in Tables A.2, A.3, and A.4.

Case	Subcase	Proof steps
$a_1\ b_2\ c_2\ d_4$	1	(b_2, c_2, d_4) or (d_4, a_1, b_2).
	2,5,6	Shrink C_q until (c_2, d_4) or (p_{r^*}, c_2, d_4).
	3,4	(a_1, c_2, d_4).
$a_1\ b_2\ c_3\ d_1$	1,2	(b_2, c_3, d_1).
	3,6	Shrink C_q until (c_3, d_1) or (p_{r^*}, c_3, d_1) (c_3 stays to the right of $\overrightarrow{qp_{r^*}}$ by fact (v)).
	4	(a_1, c_3, d_1) or (c_3, a_1, b_2).
	5	Shrink C_q until (c_3, d_1) or (p_{r^*}, c_3, d_1).
$a_1\ b_2\ c_3\ d_3$	1,3	(d_3, a_1, b_2).
	2,6	Shrink C_q until (p_{r^*}, b_2, d_3).
	4	(a_1, c_3, d_3) or (c_3, a_1, b_2).
	5	Shrink C_q until (a_1, d_3), (p_{r^*}, c_3, d_3), or (p_{r^*}, d_3, a_1).
$a_1\ b_2\ c_3\ d_4$	1	(b_2, c_3, d_4) or (d_4, a_1, b_2).
	2,3,5,6	Shrink C_q until (c_3, d_4), (p_{r^*}, c_3, d_4), or (p_{r^*}, b_2, c_3).
	4	(a_1, c_3, d_4) or (c_3, a_1, b_2).
$a_1\ b_3\ c_3\ d_1$	1	(d_1, a_1, b_3) or (b_3, c_3, d_1).
	2	by fact (iii), $(x \in \text{arc}(d_1, a_1), b_3, c_3)$.
	3	$(x \in \text{arc}(b_3, c_3), d_1, a_1)$.
	4	(a_1, c_3, d_1) or (c_3, a_1, b_3).
	5	Shrink C_q until (p_{r^*}, c_3, d_1).
	6	$(x \in \text{arc}(d_1, a_1), y \in \text{arc}(b_3, c_3))$.
$a_1\ b_3\ c_3\ d_3$	1	(d_3, a_1, b_3).
	2,6	Shrink C_q until (a_1, b_3) or (p_{r^*}, b_3, d_3).
	3	$(a_1, x \in \text{arc}(b_3, c_3))$ or $a_1, c_3, d_3)$.
	4,5	$(a_1, x \in \text{arc}(b_3, d_3))$.
$a_1\ b_3\ c_3\ d_4$	1,3	(d_3, a_1, b_3).
	2,6	Shrink C_q until (a_1, b_3), (p_{r^*}, a_1, b_3) (if b_3 is right of $\overrightarrow{qp_{r^*}}$), or (p_{r^*}, b_3, d_4) (otherwise).
	4	(c_3, a_1, b_3) or (a_1, c_3, d_4).
	5	Shrink C_q until (p_{r^*}, c_3, d_4).
$a_4\ b_2\ c_2\ d_3$	1,3	(d_3, a_4, b_2).
	2	Shrink C_q until (p_{r^*}, c_2, d_3).
	4,5	Shrink C_q until (p_{r^*}, c_2, d_3) (if d_3 is right of $\overrightarrow{qp_{r^*}}$) or (p_{r^*}, d_3, a_4) (otherwise).
	6	Shrink C_q until (p_{r^*}, c_2, d_3).

Table A.3: Proof of Lemma 4.16, Cases 6 . . . 13.

Case	Subcase	Proof steps
$a_4\ b_2\ c_2\ d_4$	1	(b_2, c_2, d_4).
	2	Shrink C_q until $(c_2, x \in \text{arc}(d_4, a_4))$ or (p_{r^*}, c_2, d_4).
	3	$(x \in \text{arc}(b_2, c_2), d_4, a_4)$.
	4	(c_2, a_4, b_2) or (a_4, c_2, d_4).
	5	Shrink C_q until (p_{r^*}, c_2, d_4).
	6	$(x \in \text{arc}(d_4, a_4), y \in \text{arc}(b_2, c_2))$.
$a_4\ b_2\ c_3\ d_3$	1,3	(d_3, a_4, b_2).
	2	Shrink C_q until (p_{r^*}, c_3, d_3) (if c_3 is left of $\overrightarrow{qp_{r^*}}$) or (p_{r^*}, b_2, c_3) (otherwise).
	4	(c_3, a_4, b_2).
	5	Shrink C_q until (p_{r^*}, b_2, c_3), (p_{r^*}, c_3, d_3), or (p_{r^*}, d_3, a_4).
	6	Shrink C_q until (p_{r^*}, c_3, d_3).

Table A.4: Proof of Lemma 4.16, Cases $14, 15$.