Deconstruction of Approximate Offsets via Minkowski Sums

Thesis submitted in partial fulfillment of the requirements for the M.Sc. degree in the School of Computer Science, Tel-Aviv University by
Roza Pogalnikova

This work has been carried out at Tel-Aviv University under the supervision of Prof. Dan Halperin

31st January, 2012
To the memory of my father, Alexander.
Acknowledgements

I would like to deeply thank my advisor, Prof. Dan Halperin for introducing me to this particular problem and for his invaluable insight, guidance and support throughout this work.

I express my deep gratitude to our collaborators Eric Beberich and Michael Kerber for their contribution to this work and the indispensable encouragement and guidance in the process of our collaboration.

I would like to thank Eyal Flato (Plataine Ltd.) who initiated this work by raising the offset deconstruction problem in connection with wood cutting. I also thank Tim Bretl (UIUC) for suggesting the digital-pen offset deconstruction problem.

I am very grateful to the people from Applied Computational Geometry Lab at Tel-Aviv University, who nudged and helped and encouraged me (in addition to making sure that I will not forget about meals throughout this work). Thank you Efi Fogel, Barak Raveh, Oren Zaltsman, Michael Hemmer, Michal Balas and Kirill Solovey.

Work on the thesis has been supported in part by the 7th Framework Programme for Research of the European Commission, under FET-Open grant number 255827 (CGL—Computational Geometry Learning), by the German-Israeli Foundation (grant no. 969/07), and by the Hermann Minkowski–Minerva Center for Geometry at Tel Aviv University.

This work would not be possible without my family: my husband Anatoly Vorobey, who have always supported and encouraged me, our daughter Julia, who is the light of my life, and my mother Luba, who took care of her while I was working. I am dedicating this work to my late father, Alexander.
Abstract

We consider the offset-deconstruction problem: Given a polygonal shape $Q$ with $n$ vertices, can it be expressed, up to a tolerance $\varepsilon$ in Hausdorff distance, as the Minkowski sum of another polygonal shape $P$ with a disk of fixed radius? If it does, we also seek a preferably simple-looking solution $P$; then, $P$'s offset constitutes an accurate, vertex-reduced, and smoothened approximation of $Q$. We give an $O(n \log n)$-time exact decision algorithm that handles any polygonal shape, assuming the real-RAM model of computation. For convex shapes, the complexity of the exact decision algorithm drops to $O(n)$, which is also the time required to compute a solution $P$ with at most one more vertex than a vertex-minimal one. A variant of the general algorithm, which we have implemented using the CGAL library, is based on rational arithmetic and answers the same deconstruction problem up to an uncertainty parameter $\delta$; its running time additionally depends on $\delta$. If the input shape is found to be deconstructable, this algorithm computes an approximate solution for the problem. The implementation allows us to solve parameter-optimization problems induced by the offset-deconstruction problem.
# Contents

1 Introduction ........................................... 1
   1.1 Background ......................................... 2
   1.2 Related work ........................................ 3
   1.3 The problem ......................................... 4
   1.4 Contribution ......................................... 6
   1.5 Organization of the thesis ......................... 7

2 Decision Problem ....................................... 9
   2.1 Definitions .......................................... 9
   2.2 Decision algorithm ................................ 10
   2.3 The shape of offsets and insets ................... 11

3 Convex Case ......................................... 15
   3.1 Decision algorithm for convex inputs ............... 15
   3.2 Reducing the number of vertices .................... 16

4 Rational Approximation ............................... 19
   4.1 Interior approximation ............................. 19
   4.2 Exterior approximation ............................. 21
   4.3 Filtering algorithm via rational approximations ... 22
   4.4 Complexity analysis ................................ 22

5 Implementation Details ............................... 25
   5.1 CGAL demo ............................................ 25
   5.1.1 CGAL packages ................................... 26
   5.2 CONVEXDECIDE implementation ..................... 28
   5.3 APPROXDECIDE implementation ...................... 30
   5.3.1 Offset and inset approximation ................. 30
   5.3.2 Disk approximation ................................ 31
   5.3.3 Decision procedure .............................. 32
CONTENTS

5.3.4 Rational approximation illustrations ............................... 32

6 Optimizing $\varepsilon$ and $r$ ................................................. 35
   6.1 Searching for a minimal valid $\varepsilon$ ................................. 35
   6.2 Searching for a maximal valid radius .................................. 37
   6.3 Searching for a good combination of $r$ and $\varepsilon$ .................... 40

7 Conclusions ......................................................................... 43
The \textit{r-offset} of a polygon, for a real parameter $r > 0$, is the set of points at distance at most $r$ away from the polygon. Technically, it is usually computed as the \textit{Minkowski sum}, i.e., vector sum of points, of the polygon and a disk of radius $r$. The resulting shape is bounded by straight-line segments and circular arcs, as can be seen in Figure 1.1 below. However, a customary practice is to model the disk in the Minkowski sum with a \textit{tight}\footnote{\textit{Tight} (or \textit{interior}) approximation is contained within the approximated shape.} polygon, which yields a piecewise-linear approximation of the offset, as demonstrated in Figure 1.1 (c).

![Figure 1.1: Example of offset and offset approximation.](image)

(a) Polygon  
(b) Polygon offset  
(c) Offset approximation
1.1 Background

Computing the offset of a polygon is a fundamental operation. The offset operation is, for instance, used to define a tolerance zone around the given polygon [HA92], or to dilute details for clarity of graphic exposition [Mat74, Ser83, DvKS08]; see Figure 1.2 (a) and (b).

(a) Tolerance zone for cutting a pocket in the solid part. The blue path in the picture on the right is the offset of the pocket boundaries and defines the area where the cutting tool cannot go. (From CATIA V5 Online Training: Power Machining Parameters at http://www.catiaonline.com/B15doc/English/online/mpgug_C2/mpgugrf0100.htm)

(b) Building map (on the left) transformed to a version with diluted details (on the right) for display on a larger map. The transformation performed by a closure operation (defined via offsets), that “closes” tight passages and small holes. (From DvKS08.)

Figure 1.2: Offset usage examples.

Manufacturing companies in various industries are burdened with legacy 2D drawings in many obsolete formats. These drawings contain critical engineering information but become unusable due to quickly advancing software systems and waning human expertise. The migration of 2D drawings to modern formats is a costly and time consuming process, requiring technical expertise and automation of this process provides many challenges. In many cases only the geometry of the original drawing is accessible, and the initial design intent is lost.
There are cases where an offset approximation forms the legacy data which a program has to deal with — the original shape before offsetting is unknown — so it is natural to ask what is the original polygon whose approximate offset we have at hand? Of course, finding the exact original polygon, or even its topology, is impossible in general, because the offset might have blurred small features like holes or dents.

(a) Are these polygons an (approximate) offset of an unknown source polygon?

(b) What polygon can be a possible offset source?

Figure 1.3: The polygons in (a) are approximate offsets with increasing radius of the same input polygon. However blurring of certain details (gecko tail and legs) make exact reconstruction of the original polygon impossible, as demonstrated by the possible polygon’s source shown in (b).

In this thesis we reformulate this question. We explore whether a given polygon is “close” to the exact $r$-offset of some unknown source polygon, and aim to find a solution (valid source polygon) with good properties when possible.

1.2 Related work

Deciding whether a given polygon is an approximate offset of some unknown shape can be seen as a special case of the Minkowski decomposition problem which asks whether a set can be composed in a non-trivial way as the Minkowski sum of two sets—disallowing a summand to be a homothetic copy of the input set. A general criterion for decomposability of convex sets in arbitrary dimension has been presented in [Sal82], where Salle shows that a wide class of convex sets is decomposable, if only they have on their surface a neighborhood which is sufficiently “nice”.
1.3 The problem

A particularly well-studied case is the case of planar lattice polygons — convex polygons with integral vertices. Several questions in geometric modeling reduce to testing whether a given convex lattice polygon can be presented as a Minkowski sum of two such polygons (so called integral decomposability question), and, if so, to finding one or all such decompositions. Integral decomposability is closely related to problems in algebra, for instance, polynomial factorization [Ost21].

\[
\begin{align*}
(a) & \quad f = xy + x + y + 1 \\
(b) & \quad g = x + y + 1 \\
(c) & \quad h = f \cdot g
\end{align*}
\]

It has been shown that deciding decomposability is NP-complete for lattice polygons [GL01]. In [ET06], decomposability of lattice polygons is investigated under the constraint that one of the summands is a line segment, a triangle, or a quadrangle.

1.3 The problem

Our study is motivated by two applications, where we are interested in a more compact and smooth representation of the approximate offset. In case of legacy data we might not know whether the polygon at hand is in fact an approximate offset. Since our input is only an approximation of an exact shape it seems reasonable to allow a controlled deviation from it in the required solution.

The first relevant problem concerns cutting polygonal parts out of wood. A wood-cutting machine, which can smoothly cut along straight line segments and circular arcs, is given a plan to cut out a certain shape. This shape was designed as a polygon expanded by a small offset, but with circular arcs approximated by polygonal lines comprising many tiny line segments. Thus instead of moving smoothly along circular arcs, the cutting tool has to move along a sequence of very short segments, and make a small turn between every pair of segments. The process becomes very slow, the tool heats up, and occasionally it causes the wood to burn.
Moving the cutting tool smoothly and fast enough is the way to keep it cool. If this were the only issue, other smoothing techniques like *arc-spline approximation* [DRS08, HH08] may have been applicable. However, we may also wish to reduce the offset radius if a more accurate cutting machine is available—in this case, it seems desirable to find the original shape first and then to re-offset with a smaller radius.

A motivation to study this question from a different domain is to recover shapes sketched by a user of a digital pen and tablet. The pen has a relatively wide tip, and the input obtained is in fact an approximate offset (with the radius of the pen tip) of the intended shape. The goal is to give a good polygonal approximation of the intended shape. More broadly, as the offset operation is so commonplace, it seems natural to ask, given only an (approximated) offset shape, what could be the original shape before the offsetting.

We pose the *(offset-)deconstruction problem* which comes in two variants:

**Problem 1: the decision problem** Given a polygonal shape $Q$, and two real parameters $r, \varepsilon > 0$, decide if there exists a polygonal shape $P$ such that $Q$ is within (symmetric) Hausdorff-distance$^2$ $\varepsilon$ to the $r$-offset (i.e., offset with radius $r$) of $P$. We refer to $Q$ as *deconstructable* with the given parameters if the outcome is positive.

**Problem 2: finding a valid source polygon** If the answer to Problem 1 is YES, compute a valid polygonal shape $P$. We refer to $P$ as a *solution*$^3$ of the deconstruction problem. Note that $P$ might be disconnected, even if $Q$ is connected (Figure L.6). Typically, we would like to construct a valid $P$ that has some desirable properties; for example, having a small number of vertices.

The approaches described in Section 1.2 discuss the exact decomposition problem. Our

---

$^2$A standard metric of closeness between two sets, defined and discussed in Section 2.1.

$^3$Of course usually there are several (or infinitely many) solutions. We wish to obtain one of them and from now on refer to it in single form to simplify the presentation.
1.4. Contribution

This thesis is based on the collaborative work with Eric Berberich and Michael Kerber and its main results were introduced in the Proceedings of the 27th Annual Symposium on Computational Geometry (SoCG 2011) [BHKP11].

We first present an efficient decision algorithm for Problem 1: For a shape $Q$ with $n$ vertices, the algorithm reports the correct answer in $O(n \log n)$ time in the real-RAM model of computation [PS90]. It constructs offsets with increasing radii in three stages; the intermediate shapes arising during the computation are in general more difficult to offset than polygons, as they are bounded by straight-line segments and “indented” circular arcs (namely, the shape is locally on the concave side of the arcs). The main observation is that for certain classes of such shapes, these circular arcs can be ignored when computing the next offset (see Theorem 2.8 for the precise statement). This observation bounds the time required by each offset computation by $O(n \log n)$, which is the key to the efficiency of the decision algorithm. Our proof is constructive, that is, if a solution exists it can be computed within the same running time.

For a convex shape $Q$ with $n$ vertices, we reduce the running time for solving Problem 1 to the optimal $O(n)$ (in the real-RAM model). Moreover, we describe a greedy algorithm within the same time complexity that returns a solution $P^*$ which minimizes, up to one extra vertex, the number of vertices among all solutions, if there are any. Our algorithm technically resembles an approach for the different problem of finding a vertex-minimal polygon in the annulus of two nested polygons [ABO+89]. We also remark that the $r$-offset of $P^*$ has a tangent-continuous boundary and therefore constitutes a special case of an arc-spline approximation of $Q$ where all circular arcs have the same radius.

Figure 1.6: For a given $Q$, the red $P$ is a candidate summand whose exact $r$-offset is shaded. Left: For a given $\varepsilon$, deconstruction is ensured iff $\phi_1 \leq \varepsilon$ and $\phi_2 \leq \varepsilon$. Note that, when $r$ decreases, $\phi_1$ decreases, but $\phi_2$ increases. Middle: An example where $Q$ can be approximated by an $r$-offset of a $P$ that has much fewer vertices than $Q$. Right: An example where $Q$ can be approximated by the $r$-offset of a disconnected shape $P$.

scenario of being Hausdorff-close to a particular decomposition seems to not have been addressed in the literature. Allowing tolerance raises interesting algorithmic questions and at the same time makes the tools that we develop more readily suitable for applications, which typically have to deal with inaccuracies in measuring and modeling.
Chapter 1. Introduction

The computation of the exact decision procedure requires the handling of algebraic coordinates of considerably high degree. As an alternative, we devise a filtering algorithm that, working exclusively with rational numbers, can both give a certified decision (solving Problem 1) and construct a valid solution (solving Problem 2). The filtering uses a rational approximation scheme to replace the offset disks by polygonal shapes of similar diameter, whose precision is determined by another parameter $\delta < \varepsilon$. We prove a bound $\Delta$ that depends on $\hat{\varepsilon}$, the minimal $\varepsilon$ for which the answer to the decision problem is $\text{YES}$ (for a given $Q$ and $r$), such that the filtering returns the certified result for all $\delta \leq \Delta$. If the input shape is found to be deconstructable, this algorithm also outputs a solution. The computation of $\hat{\varepsilon}$ up to any desired precision is still possible. We believe that our investigation of the relation between $\Delta$ and $\hat{\varepsilon}$ is of independent relevance, mostly to the study of certified algorithms that approximate geometric objects with algebraic coordinates by means of rational arithmetic.

The deconstruction problem leads to natural optimization questions: if $Q$ and $r$ are given, how to compute $\hat{\varepsilon}$, the minimal tolerance for which a solution exists? Similarly, if $Q$ and $\varepsilon$ are given, what are the possible radii such that a solution exists? For the first question, we provide a certified and efficient solution based on binary search, using the filtering algorithm. For the second question, we prove that the set of possible radii forms an interval and propose an algorithm to compute it. We also provide a heuristic to find a reasonable combination of $r$ and $\varepsilon$ when only polygon $Q$ is given.

1.5 Organization of the thesis

An exact decision algorithm for the deconstruction problem (solving Problem 1) is discussed in Chapter 2 along with a way to find a valid source (solving Problem 2). For convex input, Chapter 3 exposes a specialized deconstruction algorithm with improved complexity and the computation of an almost vertex-minimal valid source. Chapter 4 describes a rational-approximation algorithm for the deconstruction problem that outputs a valid source in case the input is deconstructable (solving Problem 2). Implementation details are given in Chapter 5. The effectiveness of our implementation is illustrated by application to solving optimization problems in Chapter 6. Chapter 7 summarizes our work and concludes with open problems.
For a set $X \subset \mathbb{R}^2$ denote its boundary by $\partial X$ and its complement by $X^C := \mathbb{R}^2 \setminus X$. A polygonal region or polygonal shape $X \subset \mathbb{R}^2$ is a set whose boundary consists of finitely many line segments with disjoint interiors. The endpoints of these straight-line segments are the vertices of the polygonal region. We assume henceforth that the input shapes that we deal with are bounded (but not necessarily connected). Although the techniques seem to go through also for unbounded shapes, this assumption simplifies the exposition and is sufficient for the real-life applications we have in mind.

2.1 Definitions

Definition 2.1. The Minkowski sum of two sets $X, Y$ is $X \oplus Y := \{x + y \mid x \in X, y \in Y\}$.

With $d(\cdot, \cdot)$ the Euclidean distance function, and any $c \in \mathbb{R}^2, r \in \mathbb{R}$, we write $D_r(c) := \{p \in \mathbb{R}^2 \mid d(c, p) \leq r\}$ for the (closed) $r$-disk around $c$, and $D_r := D_r(O)$ for the $r$-disk centered at the origin.

Definition 2.2. The $r$-offset of a set $X$, offset$(X, r)$, is the Minkowski sum $X \oplus D_r$.

We define another operation, $r$-inset (a.k.a. “erosion”), which is computationally similar to an offset:

Definition 2.3. For $r > 0$, and $X \subset \mathbb{R}^2$, the $r$-inset of $X$ is the set $\text{inset}(X, r) := \text{offset}(X^C, r)^C = \{x \in \mathbb{R}^2 \mid D_r(x) \subseteq X\}$.

For $p \in \mathbb{R}^2$ and $X$ a closed set, we write $d(p, X) := \min\{d(p, x) \mid x \in X\}$.

Definition 2.4. The (symmetric) Hausdorff distance of two closed point sets $X$ and $Y$ is $H(X, Y) := \max\{\max\{d(x, Y) \mid x \in X\}, \max\{d(y, X) \mid y \in Y\}\}$. 
We say that $X$ is $\varepsilon$-close to $Y$ (and $Y$ to $X$) if $H(X,Y) \leq \varepsilon$, which can also be expressed alternatively:

**Proposition 2.5.** For $X,Y$ closed, $X$ is $\varepsilon$-close to $Y$ if and only if $Y \subseteq \text{offset}(X,\varepsilon)$ and $X \subseteq \text{offset}(Y,\varepsilon)$.

### 2.2 Decision algorithm

We fix $r > 0$, $\varepsilon > 0$, and a polygonal region $Q$, and consider the following question: Is there a polygonal region $P$ such that $Q$ and the $r$-offset of $P$ have Hausdorff-distance at most $\varepsilon$? We can assume that $r > \varepsilon$; otherwise, we can choose $P := Q$, because $\text{offset}(Q,r)$ and $Q$ have Hausdorff-distance at most $\varepsilon$.

We present the decision algorithm:

<table>
<thead>
<tr>
<th>Algorithm 1 DECIDE($Q, r, \varepsilon$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) $Q_\varepsilon \leftarrow \text{offset}(Q, \varepsilon)$</td>
</tr>
<tr>
<td>(2) $\Pi \leftarrow \text{inset}(Q_\varepsilon, r)$</td>
</tr>
<tr>
<td>(3) $Q' \leftarrow \text{offset}(\Pi, r + \varepsilon)$</td>
</tr>
<tr>
<td>(4) if $Q \subseteq Q'$ then return YES else return NO</td>
</tr>
</tbody>
</table>

The steps of the algorithm are illustrated in Figure 2.1 on the example of the $M$-shaped polygon $Q$ with $\varepsilon = 0.4r$. The algorithm returns YES with these parameters, that is input polygon $Q$ is found to be $\varepsilon$-close to the $P_r$ (depicted in Figure 2.2):

![Figure 2.1: Decision algorithm steps illustrated on an non-convex polygon Q.](image)

We next prove that DECIDE (Algorithm 1) correctly decides whether $Q$ is $\varepsilon$-close to some $r$-offset of a polygonal region. A first observation is that for any polygonal region $P$, $\text{offset}(P, r) \subseteq Q_\varepsilon$ if and only if $P \subseteq \Pi$. This is an immediate consequence of the definition of the inset operation. This shows that for any offset($P, r$) that is $\varepsilon$-close to $Q$, $P$ must be inside $\Pi$. Moreover, it shows that any choice of $P \subseteq \Pi$ already satisfies one of Proposition 2.5’s inclusions. It is only left to check whether $Q \subseteq \text{offset}(\text{offset}(P, r), \varepsilon) = \text{offset}(P, r + \varepsilon)$. We summarize:

**Proposition 2.6.** $Q$ is $\varepsilon$-close to $\text{offset}(P, r)$ if and only if $P \subseteq \Pi$ and $Q \subseteq \text{offset}(P, r + \varepsilon)$.
To prove correctness of DECIDE, we have to show that $Q \subseteq \text{offset}(\Pi, r + \varepsilon)$ already implies that there also exists a polygonal region $P \subseteq \Pi$ with $Q \subseteq \text{offset}(P, r + \varepsilon)$. The main difficulty in proving this is that $\Pi$ is not polygonal in general; we have to study its shape closer to prove that we can approximate it by a polygonal region, maintaining the property that the offset remains $\varepsilon$-close to $Q$.

Figure 2.2: Decision algorithm correctness: we prove that algorithm returns YES iff there exists a polygon $P$, such that $P$ and $Q$ are $\varepsilon$-close.

### 2.3 The shape of offsets and insets

For a polygonal region $Q$, it is not hard to figure out the shape of $Q_\varepsilon = \text{offset}(Q, \varepsilon)$: It is a closed set bounded by straight-line segments and by circular arcs, belonging to a circle of radius $\varepsilon$. It is important to remark that all circular arcs are bulges:

**Definition 2.7.** Let $X \subset \mathbb{R}^2$ be a closed set with some circular arc $\gamma$ on its boundary. Then, $\gamma$ is called a dent with respect to $X$, if each line segment connecting two distinct points on $\gamma$ is not fully contained in $X$. Otherwise, the arc is called a bulge.

We call $X$ a bulged (resp. an indented) region with radius $r$, if $\partial X$ consists of finitely many straight-line segments and bulges (resp. dents) that are all of radius $r$, interlinked at the vertices of the region.

Note that a bulged region (left) is not necessarily convex. The $r$-offset of a polygonal region $P$ is a bulged region with radius $r$. The heart of this section is Theorem 2.8 showing that the same also holds if $P$ is an indented region (right) with radius smaller than $r$:

**Theorem 2.8.** Let $P$ be an indented region with radius $r_1$, and let $r_2 > r_1$. Then, there is a polygonal region $P_L \subseteq P$ such that $\text{offset}(P, r_2) = \text{offset}(P_L, r_2)$. In particular, $\text{offset}(P, r_2)$ is a bulged region with radius $r_2$.

**Proof.** After possibly splitting circular arcs into at most four parts, we can assume that each circular arc spans at most a quarter of the circle. For such a circular arc, we denote its endpoints by $x_1$ and $x_2$, and define the linear cap of the circular arc to be the (closed) indented region enclosed by the circular arc, and the two lines tangent to the circle through...
2.3. The shape of offsets and insets

\[ x_1 \] and \( x_2 \) (the shaded area in Figure 2.3(a)). The extended linear cap is the (polygonal) region spanned by the two tangents just mentioned, and the two corresponding normals at \( x_1 \) and \( x_2 \). Clearly, the normals meet in the center of the circle that defines the arc.

We iteratively replace an indented arc of an indented region \( P' \) with radius \( r_1 \) (initially set to \( P \)) by a polyline \( \ell \) ending in the endpoints of the circular arc, such that \( \ell \) does neither leave \( P' \) nor the linear cap of the circular arc, and such that other boundary parts of \( P' \) are not intersected. This yields another indented region \( P'' \) with radius \( r_1 \), where one indented arc is replaced by a polyline, as depicted in Figure 2.3(a). Iterating this construction, starting with \( P \), until all indented arcs are replaced, we obtain a polygonal region \( P_L \).

We show that in each iteration, the \( r_2 \)-offsets of \( P' \) and \( P'' \) are the same. For that we consider any point \( x' \in P' \setminus P'' \), in the region that is cut off by \( P'' \), and consider \( y = x' + v' \) for an arbitrary \( v' \in D_{r_2} \). We show that in all cases, \( y \) can also be written by \( y = x'' + v'' \), with \( x'' \in P'' \) and \( v'' \in D_{r_2} \).

Since the circular arc spans at most a quarter of the circle, it is easily seen that \( D_{r_1}(x_1) \cup D_{r_1}(x_2) \) covers the whole extended linear cap. Therefore, for any \( y \) that lies within the extended linear cap, selecting \( x'' = x_1 \) or \( x'' = x_2 \), we get \( y = x'' + v'' \) with \( v'' \in D_{r_1} \).

We distinguish two other cases: for \( y \) that lies outside of the extended linear cap \( v' = \overrightarrow{xy} \) crosses either \( \ell \) or the circular arc. In the former case, we can simply pick the crossing point as \( x'' \), and set \( v'' \in D_{r_2} \) accordingly (Figure 2.3(b)). In the latter case, let us denote the crossing point as \( x^* \) (Figure 2.3(c)). We consider the set of points that is closer to \( x^* \) than to \( x_1 \) and \( x_2 \). Clearly, that region is bounded by the two corresponding bisectors, which meet in the center of the circle that defines the circular arc and is therefore completely contained within the extended linear cap. It follows that \( y \) is closer to one of the endpoints of the arc, say \( x_1 \), than to \( x^* \). Selecting \( x'' = x_1 \) we ensure that \( y \) is closer to \( x'' \) than to \( x' \), which proves that \( y = x'' + v'' \) with some \( v'' \in D_{r_2} \) in this case as well.

The proof of Theorem 2.8 implies that offset(\( P, r_2 \)) for such a region \( P \) is completely determined by the offset of its linear segments, and the offset of the endpoints of circular
arcs: the interior of the indented circular arcs can be ignored.\footnote{The arc-splitting points are not actually necessary. They are used in the proof for the convenience of presentation, to demonstrate the existence of the polygon with the desired offset property.}

**Corollary 2.9.** Algorithm \( \textsc{Decide} \) returns \textsc{YES} if and only if there exists a polygonal region \( P \) such that \( \text{offset}(P, r) \) is \( \varepsilon \)-close to \( Q \).

**Proof.** \( Q_\varepsilon \) is a bulged region with radius \( \varepsilon \). Therefore, \( Q_\varepsilon^C \) is an indented region with the same radius. Since \( r > \varepsilon \), Theorem 2.8 implies that \( \text{offset}(Q_\varepsilon^C, r) \) is a bulged region with radius \( r \), and so, \( \text{offset}(Q_\varepsilon^C, r)^C = \text{inset}(Q_\varepsilon, r) = \Pi \) is an indented region with the same radius. Using \( r + \varepsilon > r \) and applying Theorem 2.8 once more, there exists a polygonal region \( P \subseteq \Pi \) such that \( \text{offset}(\Pi, r + \varepsilon) = \text{offset}(P, r + \varepsilon) \). It follows that, if the algorithm returns \textsc{YES}, there is indeed a polygonal region \( P \) whose \( r \)-offset is \( \varepsilon \)-close to \( Q \). If the algorithm returns \textsc{NO}, it is clear that no such polygonal region can exist.

**Theorem 2.10.** Let \( P \) be an indented region with radius \( r_1 \) having \( n \) vertices, and assume \( r_2 > r_1 \). Then, \( \text{offset}(P, r_2) \) has \( O(n) \) vertices and it can be computed in \( O(n\log n) \) time.

**Proof.** By Theorem 2.8, it suffices to consider a polygonally bounded \( P_L \) instead of \( P \). We use trapezoidal decomposition of \( P \) to construct such a \( P_L \) with only \( O(n) \) vertices. The Voronoi diagram of \( P_L \)'s vertices and (open) edges can be computed in \( O(n\log n) \) time and has size \( O(n) \) \cite{Yap87}. From it, the offset polygon with the same asymptotic complexity can be obtained in \( O(n\log n) \) \cite{Hel91}.\footnote{The arc-splitting points are not actually necessary. They are used in the proof for the convenience of presentation, to demonstrate the existence of the polygon with the desired offset property.}

**Corollary 2.11.** Algorithm \( \textsc{Decide} \) decides \( \varepsilon \)-closeness with \( O(n\log n) \) operations.

**Proof.** Apply Theorem 2.10 in each step of Algorithm \( \textsc{Decide} \). The fourth step runs in \( O(n\log n) \) time as well using a simple sweep-line algorithm.

Note that \( \Pi_L \), if constructed for \( \Pi \) as in the proof of Theorem 2.8 during step (3) of Algorithm \( \textsc{Decide} \) is a solution to the deconstruction problem if \( \textsc{Decide} \) returns \textsc{YES}.\footnote{The arc-splitting points are not actually necessary. They are used in the proof for the convenience of presentation, to demonstrate the existence of the polygon with the desired offset property.}
Assume that the input $Q$ to Algorithm 1 is a convex polygon. We first improve the decision algorithm such that it runs in linear time (Algorithm 2). Then we look for a polygon $P$ with a minimal number of vertices (OPT) such that $Q$ is $\varepsilon$-close to $\text{offset}(P, r)$. We give a simple linear-time algorithm that produces a polygon with at most OPT + 1 vertices.

### 3.1 Decision algorithm for convex inputs

**Lemma 3.1.** If $Q$ is a convex polygonal region, then $\Pi$, as computed by DECIDE (Algorithm 1), is also a convex polygon, and it can be computed in $O(n)$ time.

**Proof.** $Q$ is the intersection of the half-planes bounded by lines that support the polygon edges. Observe that $\Pi$ can be directly constructed from $Q$ by shifting each such line by $r - \varepsilon$ inside the polygon, which shows that $\Pi$ is convex. For the time complexity, we divide the shifted edges of $Q$ into those bounding $Q$ from above, and those bounding $Q$ from below (we assume w.l.o.g. that no edge is vertical). Consider the former edges; the lines supporting those edges have slopes that are monotonously decreasing when traversing the edges from left to right. We have to compute their lower envelope; for that, we dualize by mapping $y = mx + c$ to $(m, -c)$, which preserves above/below relations, and compute the upper hull of the dualized points. Since we already know the order of the points in their $x$-coordinate, this can be done in linear time using Graham’s scan [And79,Gra72]. The same holds for the edges bounding $Q$ from below, taking the upper envelope/lower hull. Merging upper and lower envelopes requires constant time. □

DECIDE first computes $\Pi$ and checks whether $Q \subseteq \text{offset}(\Pi, r + \varepsilon)$. We replace the latter step for convex polygons: Let $q_1, \ldots, q_n$ be the vertices of $Q$ (in counterclockwise order) and
3.2 Reducing the number of vertices

We assume that offset(Π, r) is ε-close to Q. We prefer a simple-looking approximation of Q, thus we seek a polygon P ⊆ Π whose offset is ε-close to Q, but with fewer vertices than Π. Any such P intersects each of the bulged regions of radius r + ε: κᵢ := Kᵢ ∩ Π, i = 1, ..., n. We call these bulged regions Π’s eyelets. The converse is also true: Any convex polygon P ⊆ Π that intersects all eyelets κ₁, ..., κₙ has an r-offset that is ε-close to Q.

The following observation is a simple consequence of Proposition 2.6.

**Proposition 3.3.** If offset(P, r) is ε-close to Q, and P ⊆ P′ ⊆ Π, then offset(P′, r) is ε-close to Q.

We call a polygonal region P *(vertex-*)minimal, if its r-offset is ε-close to Q, and there exists no other such region with fewer vertices. Necessarily, a minimal P must be convex – otherwise, its convex hull CH(P) has fewer vertices and it can be seen by Proposition 3.3.

---

1 Iterate at most once to find the first intersection, and once again to find the rest of them.
that offset(CH(P), q) is also \( \varepsilon \)-close to \( Q \). By the next lemma, we can restrict our search to polygons with vertices on \( \partial \Pi \).

**Lemma 3.4.** There exists a minimal polygonal region \( P \subseteq \Pi \) the vertices of which are all on \( \partial \Pi \).

**Proof.** We pull each vertex \( p_i \notin \partial \Pi \) in the direction of the ray emanating from \( p_{i-1} \) towards \( p_i \) until it intersects \( \partial \Pi \) in the point \( p_i' \) (dragging \( p_i \)'s incident edges along with it); see the enclosed illustration. For \( P' = (p_1, \ldots, p_{i-1}, p_i', p_{i+1}, \ldots, p_m) \): \( P \subseteq P' \subseteq \Pi \), offset\((P', r)\) is \( \varepsilon \)-close to \( Q \) by Proposition 3.3.

We call a polygonal region \( P \) good, if \( P \subseteq \Pi \), all vertices of \( P \) lie on \( \partial \Pi \), and \( P \) intersects each eyelet \( \kappa_1, \ldots, \kappa_n \). Note that any good \( P \) is convex.

**Definition 3.5.** For two points \( u, u' \in \partial \Pi \), we denote by \([u, u'] \subset \partial \Pi \) all points that are met when travelling along \( \partial \Pi \) from \( u \) to \( u' \) in counterclockwise order (including \( u \) and \( u' \)). Likewise, we define half-open and open intervals \([u, u'), (u, u], (u, u')\).

Let \( \kappa_i = K_i \cap \Pi \) be \( q_i \)'s eyelet as before. Consider \( \kappa_i \cap \partial \Pi \). The portion of that intersection set that is visible from \( q_i \) (considering \( \Pi \) as an obstacle) defines a (ccw-oriented) interval \([v_i, w_i] \subset \partial \Pi \). We call \( v_i \) the spot of the eyelet \( \kappa_i \). Finally, for \( u, u' \in \partial \Pi \), we say that the segment \( \overline{uu'} \) is good, if for all spots \( v_i \in (u, u') \), \( \overline{uu'} \) intersects the corresponding eyelet \( \kappa_i \).

The figure on the right illustrates these definitions: The segment \( \overline{pp'} \) is good, whereas \( \overline{pp''} \) is not good, because \( v_2 \in (p, p'') \), but the segment does not intersect \( \kappa_2 \).

**Theorem 3.6.** Let \( P \) be a convex polygonal region with all its vertices on \( \partial \Pi \). Then, \( P \) is good if and only if all its bounding edges are good.

**Proof.** We first prove that if all the edges of \( P \) are good, then \( P \) is good. It suffices to argue that it intersects all eyelets \( \kappa_1, \ldots, \kappa_n \). Let \( p_1, \ldots, p_k \) be the vertices of \( P \) in counterclockwise order. Any spot \( v_i \) of an eyelet \( \kappa_i \) either corresponds to some vertex \( p_i \) of \( P \), or lies inside some interval \((p_i, p_{i+1})\). Since \( \overline{pp_{i+1}} \) is good, it intersects \( \kappa_i \).

For the converse, assume that \( \overline{pp_{i+1}} \) is not good. In conjunction with the interval \((p_i, p_{i+1})\) it encloses a polygonal region \( R \subseteq \Pi \setminus P \). Hence, there is a spot \( v_i \in R \) such that \( \overline{pp_{i+1}} \) does not intersect the eyelet \( \kappa_i \). It follows that the entire \( \kappa_i \) is inside \( R \) (see the illustration above, considering \( \overline{pp''} \) and \( \kappa_2 \)). Thus, \( P \cap \kappa_i = \emptyset \), and so \( P \) cannot be good.

For \( u \in \partial \Pi \), we define its horizon \( h_u \in \partial \Pi \) as the maximal point in counterclockwise direction such that that segment \( \overline{uh_u} \) is good. Consider again the figure above: The segment \( \overline{ph_p} \) is tangential to \( \kappa_2 \), so if going any further than \( h_p \) on \( \partial \Pi \) from \( p \), the segment would miss \( \kappa_2 \) and thus become non-good.

**Lemma 3.7.** Let \( P \) be a good polygonal region, and \( u \in \partial \Pi \). Then, \( P \) has a vertex \( p \in (u, h_u] \).
3.2. Reducing the number of vertices

Proof. Assume to the contrary that \( P \) has no such vertex, and let \( p_1, \ldots, p_\ell \) be its vertices on \( \partial \Pi \). Let \( p_j \) be the vertex of \( P \) such that \( u \in (p_j, p_{j+1}) \). Then, also \( h_u \in (p_j, p_{j+1}) \), because otherwise, \( p_{j+1} \in (u, h_u] \). Since \( P \) is good, the segment \( \overline{p_j u} \) is good, too. It is not hard to see that, consequently, both \( \overline{p_j u} \) and \( \overline{u p_{j+1}} \) are good. However, the latter contradicts the maximality of the horizon \( h_u \).

For an arbitrary initial vertex \( s \in \partial \Pi \), we finally specify a polygonal region \( P^s \) by iteratively defining its vertices. Set \( p_1 := s \). For any \( j \geq 1 \), if the segment \( \overline{p_j s} \), which would close \( P^s \), is good, stop. Otherwise, set \( p_{j+1} := h_p \). Informally, we always jump to the next horizon until we can reach \( s \) again without missing any of the eyelets. By construction, all segments of \( P^s \) are good, so \( P^s \) itself is good. The (almost-)optimality of this construction mainly follows from Lemma 3.7.

Theorem 3.8. Let \( P \) be a minimal polygonal region for \( Q \), having \( \text{OPT} \) vertices. Then, for any \( s \in \partial \Pi \), \( P^s \) has at most \( \text{OPT} + 1 \) vertices.

Proof. We first prove that \( P^s \) has the minimal number of vertices among all good polygonal regions that have \( s \) as a vertex. Let \( s := p_1, \ldots, p_m \) be the vertices of \( P^s \). There are \( m - 1 \) segments of the form \( \overline{p_i h_p} \). By Lemma 3.7 any good polygonal region has a vertex inside each of the intervals \( (p_i, h_p] \). Together with the vertex at \( s \), this yields at least \( m \) vertices, thus \( P^s \) is indeed minimal among these polygonal regions.

Next, consider any minimal polygonal region \( P^s \). We can assume that all its vertices are on \( \partial \Pi \) by Lemma 3.4. If \( s \) is not a vertex of \( P^s \), we add it to the vertex set and obtain a polygonal region \( P^s \) with at most \( \text{OPT} + 1 \) vertices that has \( s \) as a vertex. \( P^s \) has at most as many vertices as \( P \), so \( m \leq \text{OPT} + 1 \).

As each visit of an eyelet requires constant time, the construction of a horizon is proportional to the number of visited eyelets, and there are only linearly many eyelets. Thus, we can state:

Theorem 3.9. For an arbitrary initial vertex \( s \), computing \( P^s \) requires \( O(n) \) time.

Proof. We prove that computing the horizon of a point \( u \) takes a number of operations proportional to the number of eyelets that are visited by the segment \( uh_u \). Let us consider an arbitrary \( u \in \partial \Pi \). By rotating appropriately, we can assume, without loss of generality, that \( u \) lies on a vertical edge of \( \Pi \) (or, if \( u \) is a vertex, that the next edge in counterclockwise order is vertical), and that the edge is traversed top-down. The horizon is determined by the slope of the edge at \( u \). Note that for each eyelet \( \kappa_1, \ldots, \kappa_n \), there is an interval of slopes \( I^{(u)}_1, \ldots, I^{(u)}_n \) such that the segment from \( u \) with slope \( \lambda \) intersects \( \kappa_i \) if and only if \( \lambda \in I^{(u)}_i \). Furthermore, each single \( I^{(u)}_i \) can be computed with a constant number of arithmetic operations. Assuming that the next eyelet to be travelled from the current \( p_i \) is \( \kappa_j \), we can iteratively compute the intersections \( I_j \cap I_{j+1} \cap I_{j+2} \ldots \) until \( I_j \cap \ldots \cap I_{j+k} \) is empty. In this case, we choose \( \lambda_i := \max(I_j \cap \ldots \cap I_{j+k-1}) \) as the slope for the next segment, which must be \( p_i h_p \), since it is good by construction, and any larger slope would produce a non-good segment. Based on this property, it is easy to show that computing \( P^s \) needs a number of operations which is proportional to \( n \), the number of eyelets.
Rational Approximation

A direct realization of Algorithm 1 runs into difficulties since vertices of the resulting offsets are algebraic numbers whose degrees become high in cascaded offset computations. We next describe two approximation variants of Algorithm 1, each producing a certified one-sided decision. All disks participating in the Minkowski sum operation (both offsets and insets) of the original algorithm are replaced by their polygonal approximation with rational coordinates. In order to make guaranteed statements about the exact \( \varepsilon \)-approximability by \( r \)-offsets, we have to approximate the disks by a “working precision” \( \delta \) which is even smaller than \( \varepsilon \). Recall that \( D_r \) is the disk of radius \( r \) centered at the origin.

**Definition 4.1.** For \( a, b \in \mathbb{R} \), \( a < b \) define \( \bar{D}_{a,b} \) to be a polygon with rational vertices whose boundary lies in the annulus \( D_b \setminus D_a \).

In the approximation algorithms, every disk is replaced with such a polygon lying inside a \( \delta \)-width annulus. We prove the correctness of these algorithms using a well-known relationship that is true for any polygonal region \( P \) and radius \( r \):

\[
\text{inset}(P, r) \subseteq \text{offset}(\text{inset}(P, r), r) \subseteq P \subseteq \text{inset}(\text{offset}(P, r), r) \subseteq \text{offset}(P, r).
\]

### 4.1 Interior approximation

In the first part of our algorithm, we ensure that the final approximation of \( Q' \) (see line (3) of Algorithm 1), denoted \( \hat{Q}' \), will be a subset of the exact \( Q' \). We achieve this by approximating \( D_s \) by \( \bar{D}_{s-\delta,s} \) when an offset is computed, and by approximating \( D_s \) by \( \bar{D}_{s,s+\delta} \) when an inset is computed, see Algorithm 3.
Algorithm 3 APPROXDECIDEINTERIOR \((Q,r,\varepsilon,\delta)\)

(1) \(\hat{Q}_\varepsilon \leftarrow Q \oplus \hat{D}_\varepsilon\) with \(\hat{D}_\varepsilon \leftarrow \hat{D}_{\varepsilon-\delta,\varepsilon}\).

(2) \(\hat{\Pi} \leftarrow \left(\hat{Q}_\varepsilon^C \oplus \hat{D}_r\right)^C\) with \(\hat{D}_r \leftarrow \hat{D}_{r,r+\delta}\).

(3) \(\hat{Q}' \leftarrow \hat{\Pi} \oplus \hat{D}_{r+\varepsilon}\) with \(\hat{D}_{r+\varepsilon} \leftarrow \hat{D}_{r+\varepsilon-\delta,r+\varepsilon}\).

(4) If \(Q \subseteq \hat{Q}'\), return YES, otherwise, return UNDECIDED.

**Lemma 4.2.** If APPROXDECIDEINTERIOR \((Q,r,\varepsilon,\delta)\) returns YES, then DECIDE\((Q,r,\varepsilon)\) returns YES as well, which means that there exists a polygonal region \(P\) such that offset\((P,r)\) is \(\varepsilon\)-close to \(Q\). In particular, \(P := \hat{\Pi}\) is a solution to the deconstruction problem.

**Proof.** Compare the execution of Algorithm\[3\] with the corresponding call of its exact version, Algorithm\[1\]. It is straightforward to check that for any \(\delta, \hat{Q}_\varepsilon \subseteq Q_\varepsilon, \hat{\Pi} \subseteq \Pi\), and \(\hat{Q}' \subseteq Q'\). The last inclusion shows that if \(Q \subseteq \hat{Q}'\), also \(Q \subseteq Q'\).

**Definition 4.3.** For fixed \(Q\) and \(r\), define \(\hat{\varepsilon} := \inf\{\varepsilon \mid \text{DECIDE}\((Q,r,\varepsilon)\) returns YES\}\).

Note that \(\hat{\varepsilon} \in [0,r]\), and that DECIDE\((Q,r,\varepsilon)\) returns YES for every \(\varepsilon \geq \hat{\varepsilon}\) and returns NO for every \(\varepsilon < \hat{\varepsilon}\). We do not have a way to compute \(\hat{\varepsilon}\) exactly. However, we show next that APPROXDECIDEINTERIOR \((Q,r,\varepsilon,\delta)\) returns YES for every \(\varepsilon > \hat{\varepsilon}\) for \(\delta\) small enough, and that the required precision \(\delta\) is proportional to the distance of \(\varepsilon\) to \(\hat{\varepsilon}\).

**Theorem 4.4.** Let \(\varepsilon > \hat{\varepsilon}\), and \(\delta < \frac{\varepsilon - \hat{\varepsilon}}{2}\). Then, APPROXDECIDEINTERIOR \((Q,r,\varepsilon,\delta)\) returns YES.

**Proof.** Let \(\varepsilon_0\) be such that \(\hat{\varepsilon} < \varepsilon_0 < \varepsilon_0 + 2\delta \leq \varepsilon\). Let \(Q_{\varepsilon_0}, \Pi\) and \(Q'\) denote the intermediate results of DECIDE\((Q,r,\varepsilon_0)\) and let \(\hat{Q}_\varepsilon, \hat{\Pi}, \hat{Q}'\) denote the intermediate results of APPROXDECIDEINTERIOR \((Q,r,\varepsilon,\delta)\). By the choice of \(\varepsilon_0\), YES is returned by DECIDE\((Q,r,\varepsilon_0)\), and thus \(Q \subseteq Q'\). The theorem follows from \(Q' \subseteq \hat{Q}'\), which we prove in three substeps:

1. offset\((Q_{\varepsilon_0},\delta)\) \(\subseteq \hat{Q}_\varepsilon\):
   Indeed, offset\((Q_{\varepsilon_0},\delta) = \text{offset}(Q,\varepsilon_0 + \delta) \subseteq \text{offset}(Q,\varepsilon - \delta) \subseteq Q \oplus \hat{D}_\varepsilon = \hat{Q}_\varepsilon\).

2. \(\Pi \subseteq \hat{\Pi}\):
   Starting with (1), we obtain
   \[
   \text{offset}(Q_{\varepsilon_0},\delta) \subseteq \hat{Q}_\varepsilon \\
   \Rightarrow \text{offset}(Q_{\varepsilon_0},\delta)^C \oplus \hat{D}_{r+\delta} \supseteq \hat{Q}_\varepsilon^C \oplus \hat{D}_r \\
   \Rightarrow \text{inset}(\text{offset}(Q_{\varepsilon_0},\delta),r+\delta) \subseteq \hat{\Pi}.
   \]
   We use the general fact inset\((\text{offset}(A,r),r) \supseteq A\) to obtain:
   
   \[
   \text{inset}(\text{offset}(Q_{\varepsilon_0},\delta),r+\delta) \\
   = \text{inset}(\text{inset}(\text{offset}(Q_{\varepsilon_0},\delta),\delta),r) \\
   \supseteq \text{inset}(Q_{\varepsilon_0},r) = \Pi.
   \]


(3) \( Q' \subseteq \widehat{Q}' \):
Using (2), we have that
\[
\text{offset}(\Pi, r + \varepsilon - \delta) = \Pi \oplus D_{r+\varepsilon-\delta} \subseteq \widehat{\Pi} \oplus \widehat{D}_{r+\varepsilon} = \widehat{Q}'.
\]
Note that \( r + \varepsilon - \delta > r + \varepsilon_0 \), and therefore, \( \text{offset}(\Pi, r + \varepsilon - \delta) \supset \text{offset}(\Pi, r + \varepsilon_0) = Q' \). □

4.2 Exterior approximation

In Algorithm 4, we ensure that \( \widehat{Q}' \) becomes a superset of the exact \( Q' \) by appropriately choosing approximate disks. Specifically, we approximate \( D_s \) by \( \widehat{D}_s, s + \delta \) when an offset is computed, and \( D_s \) by \( \widehat{D}_s - \delta, s \) when an inset is computed. Not surprisingly, we get a certified answer in the other direction, and a certified answer is guaranteed when \( \delta \) is sufficiently small. (Notice that in Algorithm 4 we redefine \( \widehat{Q}_\varepsilon, \widehat{\Pi} \) and \( \widehat{Q}' \).)

Algorithm 4 APPROXDECIDEEXTERIOR \((Q, r, \varepsilon, \delta)\)

1. \( \widehat{Q}_\varepsilon \leftarrow Q \oplus \widehat{D}_\varepsilon \) with \( \widehat{D}_\varepsilon \leftarrow \widehat{D}_{\varepsilon, \varepsilon+\delta} \)
2. \( \widehat{\Pi} \leftarrow (\widehat{Q}_\varepsilon^C \oplus \widehat{D}_r)^C \) with \( \widehat{D}_r \leftarrow \widehat{D}_{r-\delta, r} \)
3. \( \widehat{Q}' \leftarrow \widehat{\Pi} \oplus \widehat{D}_{r+\varepsilon} \) with \( \widehat{D}_{r+\varepsilon} \leftarrow \widehat{D}_{r+\varepsilon, r+\varepsilon+\delta} \)
4. if \( Q \subseteq \widehat{Q}' \), return UNDECIDED, otherwise, return NO

Lemma 4.5. If APPROXDECIDEEXTERIOR \((Q, r, \varepsilon, \delta)\) returns NO, then DECIDE\((Q, r, \varepsilon)\) returns NO as well, which means that there exists no polygonal region \( P \) such that \( \text{offset}(P, r) \) is \( \varepsilon \)-close to \( Q \).

Proof. Letting \( Q_\varepsilon, \Pi, \) and \( Q' \) denote the intermediate results of DECIDE\((Q, r, \varepsilon)\), it holds that for any \( \delta \), \( Q_\varepsilon \supset Q_\varepsilon, \widehat{\Pi} \supset \Pi, \) and \( \widehat{Q}' \supset Q' \). The last inclusion shows that if \( Q \not\subseteq \widehat{Q}' \), also \( Q \not\subseteq Q' \). □

Theorem 4.6. Let \( \varepsilon < \varepsilon \) and \( \delta < \frac{\varepsilon - \varepsilon_0}{2} \). Then, APPROXDECIDEEXTERIOR \((Q, r, \varepsilon, \delta)\) returns NO.

Proof. Let \( \varepsilon_0 \in \mathbb{R} \) be such that \( \varepsilon < \varepsilon_0 - 2\delta < \varepsilon_0 < \varepsilon \). Define \( Q_{\varepsilon_0}, \Pi, \) and \( Q' \) as in Theorem 4.4
By the choice of \( \varepsilon_0 \), DECIDE\((Q, r, \varepsilon)\) returns NO that is \( Q \not\subseteq Q' \). We show that \( \widehat{Q}' \subseteq Q' \) in three steps, thus proving that APPROXDECIDEEXTERIOR \((Q, r, \varepsilon, \delta)\) also returns NO:

1. \( \widehat{Q}_\varepsilon \subseteq \text{inset}(Q_{\varepsilon_0}, \delta) \):
   Note that
   \[
   \text{offset}(\widehat{Q}_\varepsilon, \delta) \subset \text{offset}(Q \oplus D_{\varepsilon+\delta}, \delta) = \text{offset}(Q, \varepsilon + 2\delta) \subset \text{offset}(Q, \varepsilon_0) = Q_{\varepsilon_0}.
   \]
The inclusions are preserved when applying the \( \delta \)-inset, and the result follows from \( \text{inset} \text{\( (\text{offset}(A, \delta), \delta) \supset A \)}. \)
4.3 Filtering algorithm via rational approximations

(2) $$\hat{\Pi} \subseteq \Pi$$:
Starting with (1), we obtain

$$\hat{Q}_\varepsilon \subseteq \text{inset}(Q_{\varepsilon_0}, \delta)$$
$$\Rightarrow \hat{Q}_\varepsilon^C \oplus D_r \supseteq \text{inset}(Q_{\varepsilon_0}, \delta)^C \oplus D_{r-\delta}$$
$$\Rightarrow \hat{\Pi} \subseteq (\text{inset}(Q_{\varepsilon_0}, \delta)^C \oplus D_{r-\delta})^C.$$

On the right-hand side, we observe that

$$(\text{inset}(Q_{\varepsilon_0}, \delta)^C \oplus D_{r-\delta})^C = \text{inset}(\text{inset}(Q_{\varepsilon_0}, \delta), r-\delta) = \text{inset}(Q_{\varepsilon_0}, r) = \Pi.$$  

(3) $$\hat{Q}' \subseteq Q'$$:
Using (2), we have that

$$\hat{Q}' = \hat{\Pi} \oplus \hat{D}_{r+\varepsilon} \subseteq \Pi \oplus D_{r+\varepsilon+\delta},$$

and since $$r + \varepsilon + \delta < r + \varepsilon_0$$, the right-hand side is a subset of $$\Pi \oplus D_{r+\varepsilon_0} = Q'$$.

4.3 Filtering algorithm via rational approximations

Using a combination of APPROXDECIDEEXTERIOR and APPROXDECIDEINTERIOR we create a filtering algorithm that can produce a certified answer to the decision problem, using only rational numbers for intermediate calculations, if the selected working precision $$\delta$$ is sufficiently small. The complete rational approximation algorithm$^1$ is:

\begin{algorithm}
\begin{algorithmic}
\State \textbf{Algorithm 5 APPROXDECIDE} \((Q, r, \varepsilon, \delta)\)
\State \begin{align*}
\text{(1) if } & \text{APPROXDECIDEINTERIOR} (Q, r, \varepsilon, \delta) = \text{YES}, \text{ return YES} \\
\text{(2) if } & \text{APPROXDECIDEEXTERIOR} (Q, r, \varepsilon, \delta) = \text{NO}, \text{ return NO} \\
\text{(3) Otherwise, return UNDECIDED}
\end{align*}
\end{algorithmic}
\end{algorithm}

Combining Theorem 4.4 with Theorem 4.6 we guarantee that the exact answer will always be found for a sufficiently small $$\delta$$, that is for $$\delta < \Delta := \frac{|\varepsilon - \hat{\varepsilon}|}{2}$$.

4.4 Complexity analysis

The main task is to bound the number of vertices of $$\bar{D}_{a,b}$$. We will create a $$\bar{D}_{a,b}$$ with the additional property that all vertices lie on $$\partial D_b$$.

$^1$As in the DECIDE case for $$\varepsilon = 0$$ the return value is NO, and for $$\varepsilon = r$$ it is YES with $$P = Q$$. 
Chapter 4. Rational Approximation

As depicted on the right, two such points on $D_b$ are connected by a chord of the boundary circle that does not intersect $D_b$ if and only if the angle induced by the two points is at most $\psi := 2 \arccos \frac{b}{a}$, or equivalently, the length of the chord is less than $2\sqrt{b^2 - a^2}$. Note that we need at least $\frac{\theta}{\psi}$ points on $\partial D_b$ for a valid $D_{a,b}$, and $\frac{2\pi}{\psi} \in \Theta\left(\frac{b}{a}\right)$ as easily follows from L’Hopital’s rule.

Rational points on $\partial D_b$ can be constructed for an arbitrary $t \in \mathbb{Q}$ as $B_t := \left(b \frac{1-t^2}{1+t^2}, b \frac{2t}{1+t^2}\right)$ [CDR92]. For some positive $z \in \mathbb{Z}$, we define $C_i := B_{i/z}$ for $i = 0, \ldots, z$.

**Lemma 4.7.** For every $i = 1, \ldots, z - 1$, the chord $C_{i-1}C_i$ is longer than the chord $C_iC_{i+1}$. In particular, the length of each chord is bounded by the length of $C_0C_1$ which is shorter than $\frac{2b}{z}$.

**Proof.** W.l.o.g., we assume $b = 1$ for the proof, since the chord length scales proportionally when scaling the circle by a factor of $b$. The point $B_t = \left(\frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2}\right)$ can be constructed geometrically as the intersection point of $\partial D_b$ with the line $\ell_t$ through $S = (-1,0)$ and slope $t$ (see the figure above). In particular, the line $SC_i$ has slope $\frac{2}{z}$; we let $T_i$ denote the intersection point of that line with the line $x = 1$. We observe that the segment $T_iT_{i+1}$ has length $\frac{2}{z}$, and that $ST_i < ST_{i+1}$ for $i = 0, \ldots, z - 1$.

We are showing next that the chord $C_{i-1}C_i$ is longer than $C_iC_{i+1}$. For that, we consider the triangle $ST_{i-1}T_{i+1}$, and its bisector through $S$. This bisector intersects the line $x = 1$ at some point $I$. By the Angle Bisector theorem, $I$ divides the segment $T_{i-1}T_{i+1}$ proportionally to the corresponding triangle sides, that is, $\frac{ST_{i-1}}{ST_{i+1}} = \frac{IT_{i-1}}{IT_{i+1}}$. Because the left-hand side is smaller than 1, it follows that $IT_{i-1}$ is shorter than $IT_{i+1}$. Therefore, $I$ lies below $T_i$, and therefore, the angle $\alpha_{i-1} = \angle T_{i-1}ST_i = \angle C_{i-1}SC_i$ is larger than $\alpha_i = \angle T_iST_{i+1} = \angle C_iSC_{i+1}$. But the chord lengths $C_{i-1}C_i$ and $C_iC_{i+1}$ are defined by $2\sin(\alpha_{i-1})$ and $2\sin(\alpha_i)$, respectively, which proves that the chord lengths are indeed decreasing.

Finally, by Thales’ theorem, the triangle $SC_0C_1$ has a right angle at $C_1$. Therefore, the longest chord $C_0C_1$ is shorter than the segment $T_0T_1$, which has length $\frac{2}{z}$. $\square$
Note that all $C_i$’s lie in the first quadrant of the plane and that $C_0 := (b,0)$ and $C_z := (0,b)$. Therefore, we can subdivide the other three quarters of the circle symmetrically such that the length of each chord is bounded by $\frac{2b}{z}$, using $4z$ vertices altogether. To compute a valid $D_{a,b}$, it suffices to choose $z$ such that $2b^2 \leq 2\sqrt{b^2 - a^2}$, that is $z \geq \sqrt{\frac{b^2}{b^2 - a^2}}$. We choose $z_0 := \left\lceil \sqrt{\frac{b}{b-a}} \right\rceil$, indeed, since $0 < a < b$, we have that $z_0 \geq \sqrt{\frac{b}{b-a}} > \sqrt{\frac{b}{b-a} \cdot \frac{b}{b+a}} = \sqrt{\frac{b^2}{b^2-a^2}}$.

As stated above, we need at least $\Omega\left(\sqrt{\frac{b}{b-a}}\right)$ points, so $z_0$ is an asymptotically optimal choice.

We summarize the result

**Lemma 4.8.** For $a < b$, a polygonal region $D_{a,b}$ as above with $O\left(\sqrt{\frac{b}{b-a}}\right)$ (rational) points can be computed using $O\left(\sqrt{\frac{b}{b-a}}\right)$ arithmetic operations.

The Minkowski sum of an arbitrary polygonal region with $n$ vertices and a convex polygonal region with $k$ vertices has complexity $O(nk)$ and it can be computed in $O(nk \log^2(nk))$ operations by a simple divide-and-conquer approach, using a sweep line algorithm in the conquer step [KLPS86]. Using generalized Voronoi diagrams where the distance is based on the convex summand of the Minkowski sum operation [LS87], we obtain an improved algorithm, which requires only $O(nk \log(nk))$ operations.

According to Lemma 4.8, the disk approximation sizes in APPROXDECIDEINTERIOR and APPROXDECIDEEXTERIOR are bounded by $k_1 = |\widehat{D}_\varepsilon| = O\left(\sqrt{\frac{\varepsilon}{\delta}}\right)$, $k_2 = |\widehat{D}_r| = O\left(\sqrt{\frac{r}{\delta}}\right)$ and $k_3 = |\widehat{D}_r + \varepsilon| = O\left(\sqrt{\frac{\varepsilon}{\delta}}\right)$, that is the intermediate Minkowski sum sizes are $|\widehat{Q}_\varepsilon| = O(nk_1)$, $|\widehat{P}| = O(nk_1k_2)$ and $|\widehat{Q'}| = O(nk_1k_2k_3)$ accordingly. This leads to the following complexity bound for the two approximation algorithms.

**Theorem 4.9.** Algorithm APPROXDECIDE requires

$$O\left(n \frac{r}{\delta} \sqrt{\frac{\varepsilon}{\delta}} \cdot \log\left(n \frac{r}{\delta} \sqrt{\frac{\varepsilon}{\delta}}\right)\right)$$

arithmetic operations with rational numbers.

We remark that the $O(n \log n)$ bound for DECIDE refers to operations with real numbers instead. In Chapter 6 we present examples of approximative decisions.
We have implemented an application that demonstrates intermediate constructions and results of the decision algorithms CONVEXDECIDE (see Chapter 3) and APPROXDECIDE (see Chapter 4). The implementation of the algorithms employs the CGAL library\(^1\), while the application GUI uses the Qt framework\(^2\). In this chapter we give a brief description of the CGAL functionality we use and provide details about our implementation, illustrating the execution of the algorithms on several examples. All the illustrations are generated with our software.

### 5.1 CGAL demo

CGAL is an open source software library providing the implementation of efficient and reliable geometric algorithms. It is written in C++ and uses generic programming to achieve flexibility and genericity \(^\text{[Aus98]}\). CGAL classes are mostly templates that can be instantiated with various data types, according to the needs of the user.

Most computational geometry algorithms assume the real RAM model of computation, that is assume infinite-precision computation with real numbers. However, in modern hardware and software the standard representation of real numbers is discrete \(^\text{[46108]}\). CGAL employs the exact computation paradigm to guarantee correctness and ensure robustness of the implemented algorithms. Exact number types provided by external libraries (GMP, CORE, LEDA) can be plugged in the library algorithms for this purpose.

The work presented in this thesis is an extension of the 2D Minkowski Sums package. We have adapted CGAL and the Qt Graphics View Framework package to implement the

---

\(^1\) The Computational Geometry Algorithms, [www.cgal.org](http://www.cgal.org)

\(^2\) Cross-platform application and UI framework, [qt.nokia.com](http://qt.nokia.com)
GUI of our application. Figure 5.1 demonstrates a snapshot of our demo program.

![Figure 5.1: Snapshot of the CGAL demo for construction and deconstruction of approximate offsets. The M-shaped non-convex input polygon is shown in blue. The application shows the estimated offset source polygon (in green) and its r-offset approximating the input polygon ε-closely (in cyan), for ε = 0.38 · r (parameter values are displayed in the information line below the graphics area).](image)

5.1.1 CGAL packages

Our implementation is based on CGAL packages for 2D Polygons \[GW11\], 2D Minkowski Sums \[Wei11\] and 2D Regularized Boolean Set-Operations \[FWZH11\]. We proceed with an overview of the functionality we rely on.

Kernels and Traits

The algorithms and data structures that we use usually expect a template parameter in the form of traits, which provides basic operations and geometric predicates.

For representation of geometric objects with rational coordinates we use the \texttt{CGAL::Cartesian} kernel.\(^3\) The kernel is instantiated with exact rational number types (\texttt{CGAL::Gmpq} or \texttt{CORE::BigRat}). Operations on the underlying geometric objects produce exact results when these are also rational.

However, exact offset or inset of a rational polygon with rational offset radius typically produces non-rational results. For this kind of computation we use conic traits that can handle these operations exactly.\(^4\)

\(^3\)\texttt{CGAL::Cartesian} defines exact operations and predicates for geometric objects with coordinates remaining in \(\mathbb{Q}\).

\(^4\)Till the emergence of the \textit{Algebraic Kernel} \[BHK11\] in the recent CGAL versions this was the only available option for the exact computation of this kind. The new \textit{Algebraic Kernel} also provides a foundation for creation of cascaded offset and inset operations, that are not supplied by CGAL yet, and are required by the exact decision algorithm DECIDE. Alas, such an extension of CGAL functionality lies outside the scope of this thesis.
2D Polygons package

A polygon is a closed chain of edges. The Polygon\_2 class serves as a wrapper for a set of points that define a 2D Polygon. We use it to represent inputs (polygons with rational vertices), intermediate constructions and outputs of our algorithms when possible. We also use the functionality provided by 2D Polygons package to determine a polygon’s boundary orientation (clockwise/counterclockwise) and to find a polygon’s bounding box.

2D Regularized Boolean Set-Operations package

We use Join (the union of two sets \( S_1 \cup S_2 \)) and Difference (the difference of two sets \( S_1 \setminus S_2 \)) operations provided by 2D Regularized Boolean Set-Operations package via the \texttt{join} and \texttt{difference} global functions to perform Boolean operations on polygons.

In cases where the input or output of the performed operations cannot be represented as a single chain of segments, namely, when the input comprises a polygon with holes or a set of disjoint polygons with holes we use the Polygon\_with_holes\_2 or Polygon\_set\_2 classes provided by this package.\(^5\)

2D Minkowski Sums package

Non-convex inputs can produce results that are not simply connected, so the output of the Minkowski sum operation is represented by the Polygon\_with_holes\_2, provided by the 2D Regularized Boolean Set-Operations package. The Minkowski sum operation on two polygons is computed with the \texttt{minkowski\_sum\_2} function provided by the 2D Minkowski Sums package, which uses convolution method [Wei11].

The package also provides exact offset and exact inset operations with the \texttt{offset\_polygon\_2} and \texttt{inset\_polygon\_2} functions. Both operations receive Polygon\_2 as input. However the resulting shape cannot be represented as a chain (or even chains) of segments, because it can have circular arcs on the boundary, as demonstrated by the offset and inset of a non-convex polygon on the right.

Shapes that are bounded by segments and circular arcs can be represented by the General\_polygon\_2 and General\_polygon\_with_holes\_2 classes, which can have any \(x\)-monotone curve\(^6\) on their boundary.

In our case the coordinates of the input are represented by rational numbers, but the coordinates of the output are not rational in general. The resulting offset polygons can be represented exactly by the General\_polygon\_2 or General\_polygon\_with_holes\_2 that are instantiated with traits that can handle offset segment and circular arc intersections; we use the Arr\_conic\_traits\_2 to represent boundary edges as conic curves.

---

\(^5\)The current implementation does not support intermediate constructions with more than one chain, that is the computation is always carried out (with a due warning) on the first outer chain of the intermediate shape.

\(^6\)A continuous planar curve \(C\) is \(x\)-monotone if every vertical line intersects it at most once.
5.2 ConvexDecide implementation

Recall that the input to the algorithm is a polygon $Q$ with rational vertices, and to decide if a convex $Q$ can be $\varepsilon$-close to an unknown source polygon $r$-offset we have devised the CONVEXDECIDE algorithm (see Chapter 3), which follows the steps illustrated in Figure 5.2.

![Figure 5.2: ConvexDecide steps. The input polygon Q is shown in blue, II in light green. The decision procedure returns NO for the given example, because, as we can see in (c), the $(r + \varepsilon)$ disks around the leftmost and the rightmost $Q$ vertices do not intersect II.](image)

Notice that for a convex input $Q$ the initial construction steps can be united, since for $\varepsilon < r$, inset(offset(Q, $\varepsilon$), $r$) = inset(Q, $r - \varepsilon$). It is clear that the result of this operation is both polygonal and convex, so we can designate it $P$. It also follows that for any vertex of $Q$ the closest point on the boundary of $P$ is also a vertex. Consequently we do not need to actually construct $r + \varepsilon$ disks around vertices of $Q$, and intersect them with $P$. Instead we can simply find a closest vertex of $P$ for each vertex of $Q$ and verify that the distance between each pair is at most $r + \varepsilon$, that is if this condition holds the algorithm returns YES, and otherwise NO. It remains to demonstrate that such a correspondence between vertices of $Q$ and $P$ can be found in a linear time.

We use the 2D Polygons package functionality to arrange vertices of each polygon ($P$ and $Q$) in a counterclockwise order. Let $p_1...p_m$ denote the vertices of $P$, and $q_1...q_n$ the vertices of $Q$, with $m < n$. As a first step we iterate over all the vertices of $P$ and select the one closest to $q_1$ in $O(m)$. Assume that at step $i$ we have found vertex $p_j$ to be the closest to vertex $q_i$, in other words we have paired $q_i$ and $p_j$ and $|q_i p_j|$ is minimal. Consider the possibilities at step $(i + 1)$: either $|q_{i+1} p_{j+1}| < |q_{i+1} p_j|$, and then $|q_{i+1} p_{j+1}|$ is minimal; or $|q_{i+1} p_{j+1}| \geq \ldots \geq |q_{i+1} p_k| < |q_{i+1} p_{k+1}|$, that is the distances form $q_{i+1}$ to the vertices of $P$ are monotonously decreasing from $p_j$ up to $p_k$, and then increasing again, i.e., $|q_{i+1} p_{k+1}|$ is minimal. In any case for each following vertex of $Q$ we start the iteration at a previously selected vertex of $P$, and compute $P$’s vertices distances to the current vertex of $Q$ as long as they are decreasing, which is equivalent to selecting a minimal pair from $P$ for each vertex of $Q$. Since it is not possible that while iterating over all vertices of $Q$, we will rotate around $P$ more then once, the complexity of pairing vertices from both polygons is $O(m + n) = O(n)$. We check whether all these minimal distances are smaller then $(r + \varepsilon)$ in $O(n)$. Thus, the overall complexity of this simplified version of the algorithm is $O(n)$. 
(a) The input convex polygon is not approximable because the mappings shown in red (leftmost and rightmost vertices of Q) are not sufficiently close to the corresponding vertices of P.

(b) The approximation of the r-offset of the convex polygon from (a) (shown in green) is approximable with r and ε, since each vertex has a close enough mapping, shown in green.

Figure 5.3: Possible outcomes of ConvexDecide: NO in (a) and YES in (b).

Figure 5.3 illustrates the two possible outcomes of ConvexDecide with the same r and ε = 0.1 · r for a convex polygon and its ε-approximate r-offset.
5.3 ApproxDecide implementation

Recall that the input to the filtering algorithms **ApproxDecideInterior** and **ApproxDecideExterior** is a polygonal region $Q$, a radius $r$, a solution precision $\varepsilon$ and a working precision $\delta$. By construction of approximate offsets and insets with the working precision $\delta$, the algorithms combined determine whether $Q$ is $\varepsilon$-close to the $r$-offset of an unspecified polygonal shape $P$ and compute a three-valued answer: **YES** if a valid $P$ exists, **NO** if it does not, and **UNDECIDED** if a certified **YES** or **NO** answer cannot be determined for the given parameters.

<table>
<thead>
<tr>
<th><strong>ApproxDecideInterior</strong></th>
<th>#</th>
<th><strong>ApproxDecideExterior</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q^-<em>\varepsilon \leftarrow \text{offset}</em>{-\delta}(Q, \varepsilon)$</td>
<td>1</td>
<td>$Q^+<em>\varepsilon \leftarrow \text{offset}</em>{+\delta}(Q, \varepsilon)$</td>
</tr>
<tr>
<td>$P^- \leftarrow \text{inset}<em>{+\delta}(Q^-</em>\varepsilon, r)$</td>
<td>2</td>
<td>$P^+ \leftarrow \text{inset}<em>{-\delta}(Q^+</em>\varepsilon, r)$</td>
</tr>
<tr>
<td>$Q'^- \leftarrow \text{offset}_{-\delta}(P^-, r + \varepsilon)$</td>
<td>3</td>
<td>$Q'^+ \leftarrow \text{offset}_{+\delta}(P^+, r + \varepsilon)$</td>
</tr>
<tr>
<td>$Q \setminus Q'^- = \emptyset \Rightarrow \text{YES}$</td>
<td>4</td>
<td>$Q \setminus Q'^+ \neq \emptyset \Rightarrow \text{NO}$</td>
</tr>
</tbody>
</table>

Table 5.1: The **ApproxDecideInterior** cannot produce a certified answer, as seen in step 4, since $Q \subsetneq Q'^-$. The **ApproxDecideExterior** also cannot produce a certified answer, since $Q \subseteq Q'^+$. The combined algorithm returns **UNDECIDED** for this input. Tightening the $\delta$ parameter will bring $Q'^-$ and $Q'^+$ closer together and at some point (for $\delta < \frac{|\varepsilon - \hat{\varepsilon}|}{2}$) will result in a certified answer.

The steps of both algorithms are illustrated in Table 5.1 on the non-convex input.

5.3.1 Offset and inset approximation

The operations $\text{offset}_{+\delta}$ and $\text{offset}_{-\delta}$ (see Table 5.1) designate the (rational and polygonal) outer and inner $\delta$-approximations of the exact offset operation, respectively. We compute them via Minkowski Sum with outer and inner $\delta$-approximation of the offset disk.
The operations \( \text{inset}_{+\delta} \) and \( \text{inset}_{-\delta} \), on the other hand, designate (somewhat counterintuitively) the inner and outer \( \delta \)-approximations of the exact inset operation. This notation becomes clear when one remembers the definition of the inset as a complement of the offset applied to its input complement: to get the inner inset approximation one has to use outer offset approximation, and vice versa. Unfortunately, CGAL implementation of the Minkowski sum operation disregards polygon orientation, namely, it cannot be applied on the polygon complement.

Our implementation of the approximate inset operation relies on a “molding” technique. We turn the input polygon into a hole inside its bounding box, split the resulting polygon through this hole into two separate parts, offset and unite them back. The resulting holes are “molded” complements of the offset operation:

**Algorithm 6** \( \text{APPROXINSET}((Q, r, \delta)) \)

1. Create a complement of the input polygon by putting it in the (slightly enlarged) bounding box.
2. Cut the so obtained rectangle-with-hole into two halves by vertical cuts through the topmost and the bottommost points of the hole.\(^3\)
3. Create an approximate offset of each half.
4. Join the two offset polygons from the previous step.
5. The holes inside the resulting inflated rectangle are the result of the inset operation.

### 5.3.2 Disk approximation

The rational disk approximation that we create is used as a summand of the Minkowski sum operation. Notice that both the number of vertices in the approximation and the bit-lengths of their coordinates have significant impact on the time it takes to compute the result. Not surprisingly, there is a trade-off between the two. Since the output of this operation is used as an input for cascading constructions, it seems prudent to bound the bit-lengths of the resulting coordinates.

The inner disk approximation scheme described in Chapter 4 amounts to selection of a sufficiently small rational \( t \), such that the distance between consecutive rational points on an \( R \)-circle (where \( R \) can be \( \varepsilon \) or \( r \) or \( r + \varepsilon \)) generated by the parametrization \( p_t = (R_1 + t^2, R_2) \) is small enough to fit a polygon induced by these points into a \( \delta \)-band inside the circle. We have shown that selection of \( t < \tilde{t} := \left\lceil \sqrt{\frac{R}{\delta}} \right\rceil \) ensures this condition.

As can be seen from the parametrization \( p_t \), the bit-length of the resulting coordinates are bounded by \( O(\log(\frac{R}{\delta})) \).

In our implementation we have experimented with rounding \( t \) to the closest power of 2 or 10, that is have selected the biggest \( b \) (or \( d \)) such that \( t = t_{\text{bin}} = 2^b < \tilde{t} \) (or \( t_{\text{dec}} = 10^d < \tilde{t} \)). The actual selection is performed in the first quarter of the unit circle and then scaled by \( R \) and mirrored to the whole circle. The resulting circle points are guaranteed to be sufficiently close to each other, but not all of them are necessary to produce a valid approximation.

\(^3\)In case of several such points one can be chosen arbitrarily.
Table 5.2: Expected number of the unit disk approximation points \(4 \cdot t = 4 \cdot \left\lceil \sqrt{\frac{1}{\delta}} \right\rceil\) vs. greedily reduced number of points with \(t\) rounded to power of 2 \((t_{bin})\) and to power of 10 \((t_{dec})\).

We reduce the number of disk approximation vertices by greedily choosing points on the circle at maximal possible distance from each other, while keeping the resulting polygon inside the \(\delta\)-annulus. This reduces the number of points by about 20% to 40%, as shown in the Table 5.2.

The outer \(R\)-disk \(\delta\)-approximation is computed as the inner \(\delta\)-approximation of \((R + \delta)\)-disk.

### 5.3.3 Decision procedure

Having computed rational polygonal approximation of \(Q' (Q^- \text{ or } Q^+\) the decision is implemented via straightforward application of CGAL’s regularized Boolean operation \textit{difference}, followed by an emptiness test.\(^7\)

### 5.3.4 Rational approximation illustrations

We demonstrate the execution of our software on two examples in Figures 5.4 and 5.5. How the choice of \(\varepsilon\) and \(\delta\) influences this outcome is illustrated in Figure 5.4.

The decision procedures for the cases depicted in Figure 5.4 (a), (b), (c) and (d) took 0.252, 0.480, 0.388 and 0.844 seconds respectively on a 3GHz Intel Dual Core processor in our tests. See also Figure 5.5 for results on a larger polygon.

---

\(^7\)Limitation: we do not handle degeneracies in the results of Boolean operations, although they can affect the resulting decision. For example if the result of the \textit{difference} is a singular point, the regularized Boolean operation will return an empty set.
Chapter 5. Implementation Details

(a) \( \varepsilon = \frac{1}{3} \cdot r, \delta = \frac{1}{30} \cdot r \)  
(b) \( \varepsilon = \frac{1}{9} \cdot r, \delta = \frac{1}{36} \cdot r \)  
(c) \( \varepsilon = \frac{1}{6} \cdot r, \delta = \frac{1}{4} \cdot \varepsilon \)  
(d) \( \varepsilon = \frac{1}{6} \cdot r, \delta = \frac{1}{10} \cdot \varepsilon \)

Figure 5.4: Dependency of the algorithm outcome on \( \varepsilon \) and \( \delta \): The input polygon (wheel) appears in bold line. It is colored according to its approximability with the given parameters: green for YES, red for NO and yellow for UNDECIDED. The inset polygon \( P^- \) and its approximate \((r + \varepsilon)\)-offset \( Q^-\) are drawn in green and cyan respectively. Their outer-approximation counterparts \( P^+ \) and \( Q^+ \) are drawn in red and magenta. Figures (a) and (b) demonstrate how when \( \varepsilon \) is tightened from \( \frac{1}{3} \cdot r \) to \( \frac{1}{9} \cdot r \), with the same \( r \) and \( \delta \), the decision result changes from YES to NO. The green polygon inside the input polygon in (a) is a possible \( r \)-offset solution. The magnification in (b) highlights the area of the input polygon that does not fit inside the outer \( \delta \)-approximation (in magenta) of maximal possible \((r + \varepsilon)\)-offset. Figures (c) and (d) show how when \( \delta \) is decreased from \( \frac{1}{4} \cdot \varepsilon \) to \( \frac{1}{10} \cdot \varepsilon \), for the same \( r \) and \( \varepsilon \), the decision result changes from UNDECIDED to NO, namely in the latter case the algorithm is able to produce a certified negative answer.

Figure 5.5: A map of Kazakhstan, represented as a polygon \( Q \) (in bold blue) with 1881 vertices, is approximable for \( \varepsilon = \frac{1}{3} \cdot r \) and \( \delta = \frac{1}{8} \cdot \varepsilon \). A solution polygon \( P \) (in green) has 335 vertices. \( \text{offset}(P,r) \) (shown as lightly-shaded gray \( r \)-strip around \( P \)) is inside the \( \varepsilon \)-offset of the input \( Q \) by construction. The \( \delta \)-approximation of the \( \varepsilon \)-offset of \( \text{offset}(P,r) \) (as computed in line (3) of \text{APPROXDECIDEINTERIOR}(Q,r,\varepsilon,\delta)) is drawn in cyan and has 261 vertices. Since the cyan polygon contains \( Q \), the \( \text{offset}(P,r) \) and \( Q \) have Hausdorff distance of at most \( \varepsilon \), that is, \( Q \) is approximable and \( P \) is a solution. Approximability computation took 3.868 seconds in this case on a 3GHz Intel Dual Core processor. The magnification on the left highlights some cavities in the input polygon that have no effect on the Hausdorff distance within this tolerance \( \varepsilon \). The magnification on the right demonstrates a sharp end that would prevent \( Q \)'s approximability with a tighter \( \varepsilon \).
In this chapter we demonstrate the effectiveness of the rational algorithms we have implemented by applying them to solving parameter optimization problems for approximate offset decomposition. This solution is implemented and illustrated by our CGAL demo as well.

So far, we have assumed that both $r$ and $\varepsilon$ are given as input parameters, and we posed the question of deconstructing a polygon with respect to these parameters. We now investigate three variants where $r$ and/or $\varepsilon$ are unknown. Specifically, we ask, for some input polygon $Q$:

1. Given $r$, what is $\hat{\varepsilon}$, the infimum of all $\varepsilon$-values such that the deconstruction problem has a solution (see Definition 4.3)?
2. Given $\varepsilon$, what is the set of radii for which the deconstruction problem has a solution?
3. Given neither $r$ nor $\varepsilon$, how to choose them in a “reasonable” way to obtain a solution?

Whereas the first two questions are formally posed, the third one is of a rather heuristic nature. In all three cases, we also ask for computing some polygonal shape $P$ that approximates the solution of the deconstruction problem for the relevant set of parameters.

We discuss the posed questions in the remainder of this section. Our main tool will be the decision algorithm for fixed $r$ and $\varepsilon$ as described earlier. Because we aim for a practical algorithm, we formulate our approach using the rational approximation algorithm Algorithm 5 from Chapter 4. We have implemented the proposed algorithms; the example at the end of this section has been produced with our implementation.

### 6.1 Searching for a minimal valid $\varepsilon$

If we use the exact decision procedure DECIDE, it is straight-forward to approximate $\hat{\varepsilon}$ to arbitrary precision $\Delta$ employing binary search: Start with the interval $[0, r]$ and choose $\varepsilon$
as the midpoint of the interval. If \( \text{DECIDE}(Q, r, \varepsilon) \) returns YES, recurse on the left subinterval, otherwise, on the right one. Obviously, the interval width is halved in every step, so \( O(\log(\frac{r}{\Delta})) \) steps are sufficient and necessary. Let \( \varepsilon_\Delta \) denote an approximation of \( \hat{\varepsilon} \), s.t. \( \varepsilon_\Delta - \hat{\varepsilon} \leq \Delta \). We demonstrate next that we can achieve the same approximation and produce with it a solution to the deconstruction problem using the rational approximation version \text{APPROXDECIDE}.

Let \( |I| \) denote the width of \( I \) henceforth and consider the pseudocode given in Algorithm 7. It computes an interval \( I \) of width at most \( \Delta \) that contains \( \hat{\varepsilon} \).

**Algorithm 7 APPROXSEARCHEPS\( (Q, r, \Delta) \)**

1. \( I \leftarrow [0, r] \)
2. while \( |I| > \Delta \) do
3. \( \varepsilon_{\text{no}} \leftarrow \) left endpoint of \( I \), \( \varepsilon_{\text{yes}} \leftarrow \) right endpoint of \( I \)
4. \( \varepsilon_{\text{mid}} \leftarrow \frac{\varepsilon_{\text{no}} + \varepsilon_{\text{yes}}}{2} \), \( \delta \leftarrow \frac{|I|}{8} \)
5. \( \text{res} \leftarrow \text{APPROXDECIDE}(Q, r, \varepsilon_{\text{mid}}, \delta) \)
6. if \( \text{res} = \text{YES} \) then \( I \leftarrow [\varepsilon_{\text{no}}, \varepsilon_{\text{mid}}] \)
7. otherwise, if \( \text{res} = \text{NO} \) then \( I \leftarrow [\varepsilon_{\text{mid}}, \varepsilon_{\text{yes}}] \)
8. otherwise, (\( \text{res} = \text{UNDECIDED} \)), \( I \leftarrow [\varepsilon_{\text{mid}} - \frac{|I|}{4}, \varepsilon_{\text{mid}} + \frac{|I|}{4}] \)
9. end while
10. return \( I \)

We prove the invariant that \( \hat{\varepsilon} \in I \) after each iteration of the while-loop, implying correctness of the whole algorithm. Trivially, \( \hat{\varepsilon} \in [0, r] \), and the invariant is obviously maintained if \( \text{APPROXDECIDE}(Q, r, \varepsilon_{\text{mid}}, \delta) \) returns YES or NO. For the case of UNDECIDED, recall that \( \text{APPROXDECIDE} \) is a combination of the two one-sided approximation algorithms \( \text{APPROXDECIDEEXTERIOR} \) and \( \text{APPROXDECIDEINTERIOR} \), and both returned UNDECIDED. Theorem 4.4 and Theorem 4.6 imply therefore that

\[
\frac{|I|}{8} \geq \left| \frac{\varepsilon_{\text{mid}} - \hat{\varepsilon}}{2} \right|.
\]

It follows that \( \hat{\varepsilon} \in [\varepsilon_{\text{mid}} - \frac{|I|}{4}, \varepsilon_{\text{mid}} + \frac{|I|}{4}] \) which proves that the invariant is maintained also in this case.

We next compute a solution \( P \) for the deconstruction problem for \( Q, r \) and \( \varepsilon_{\Delta} \). Recall that if \( \text{APPROXDECIDE} \) returns YES, the algorithm computes a solution for the deconstruction problem as a by-product. Let \( I \leftarrow \text{APPROXSEARCHEPS}(Q, r, \Delta) \) be the approximation interval for \( \hat{\varepsilon} \) and let \( \varepsilon_{\text{yes}} \) denote the right endpoint of \( I \), that is \( \hat{\varepsilon} \leq \varepsilon_{\text{yes}} \). We call \( \text{APPROXDECIDE}(Q, r, \varepsilon_{\text{yes}}, \frac{|I|}{4}) \). If the result is YES, then \( \varepsilon_{\Delta} = \varepsilon_{\text{yes}} \) and the polygon computed by \( \text{APPROXDECIDE} \) is a solution. Otherwise let us choose \( \varepsilon_{\Delta} = \varepsilon_{\text{yes}} + \frac{\Delta}{2} \) and produce a solution by calling \( \text{APPROXDECIDE}(Q, r, \varepsilon_{\Delta}, \frac{\Delta}{8}) \). Since the result of \( \text{APPROXDECIDE}(Q, r, \varepsilon_{\text{yes}}, \frac{|I|}{4}) \) was UNDECIDED we conclude from Theorem 4.4 that \( \hat{\varepsilon} \geq \varepsilon_{\text{yes}} - \frac{|I|}{2} \), that is \( \varepsilon_{\Delta} = \varepsilon_{\text{yes}} + \frac{\Delta}{2} \) is indeed a \( \Delta \)-approximation of \( \hat{\varepsilon} \). \( \text{APPROXDECIDE}(Q, r, \varepsilon_{\Delta}, \frac{\Delta}{8}) \) call is bound to yield YES because if it returned UNDECIDED, we would have that \( \hat{\varepsilon} \geq \varepsilon_{\text{yes}} - \frac{|I|}{2} \), that is \( \varepsilon_{\Delta} = \varepsilon_{\text{yes}} + \frac{\Delta}{2} \) is indeed a \( \Delta \)-approximation of \( \hat{\varepsilon} \).
\[ \tilde{\epsilon}_\Delta - 2\Delta \frac{\Delta}{8} = \varepsilon_{\text{yes}} + \frac{\Delta}{4}, \] a contradiction to \( \hat{\varepsilon} \leq \varepsilon_{\text{yes}} \). So, the polygon computed in this call is a solution.

An overall complexity analysis of approximating \( \hat{\varepsilon} \) (and computing a solution) is relatively straightforward: \( I \) is obviously halved in every iteration, so it takes \( O(\log(\frac{\Delta}{\varepsilon})) \) iterations to approximate \( \hat{\varepsilon} \). The running time of every iteration is bounded by the complexity given in Theorem 4.9. We omit further details of the proof:

**Theorem 6.1.** Approximating \( \hat{\varepsilon} \) to a precision \( \Delta > 0 \) requires

\[
O(n \frac{r}{\Delta} \sqrt{\frac{\varepsilon}{\Delta}} \cdot \log(n \frac{r}{\Delta} \sqrt{\frac{\varepsilon}{\Delta}}))
\]

arithmetic operations with rational numbers.

### 6.2 Searching for a maximal valid radius

We assume now that \( Q \) and \( \varepsilon \) are given, and discuss the question of what is the set \( R \) of radii such that the deconstruction problem has a solution. A priori, it is not clear what is the shape of \( R \), but we will prove that it is an interval of the form \([0, r^*]\). Having established this, we can apply another variant of binary search to approximate the extremal value \( r^* \).

In order to prove that \( R \) is an interval, we prove first that the deconstruction problem can always be solved for \( Q, r \) and \( \hat{\varepsilon} \). In other words, we can replace the infimum in Definition 4.3 by a minimum. The proof relies on two properties of infinite intersections of insets and offsets that we show first.

**Lemma 6.2.** Let \( (A_i)_{i \in \mathbb{N}} \) be a sequence of closed sets in \( \mathbb{R}^2 \). Then

\[
\text{inset}(\bigcap_{i=0}^{\infty} A_i, r) = \bigcap_{i=0}^{\infty} \text{inset}(A_i, r).
\]

**Proof.** The fact follows readily from the definition of insets: If \( a \in \text{inset}(\bigcap_{i=0}^{\infty} A_i, r) \), then \( D_r(a) \) is contained in \( \bigcap_{i=0}^{\infty} A_i \). In particular, it is contained in \( A_i \) for every \( i \) which proves one inclusion. The other direction is similar. \( \Box \)

**Lemma 6.3.** Let \( (A_i)_{i \in \mathbb{N}} \) be a sequence of closed sets in \( \mathbb{R}^2 \) with \( A_0 \supseteq A_1 \supseteq \ldots \). Moreover, let \( (\lambda_i)_{i \in \mathbb{N}} \) be a monotonously decreasing sequence of real numbers that converges to \( \lambda \in \mathbb{R} \). Then

\[
\text{offset}(\bigcap_{i=0}^{\infty} A_i, \lambda) = \bigcap_{i=0}^{\infty} \text{offset}(A_i, \lambda_i).
\]

**Proof.** The “\( \subseteq \)” inclusion is straight forward, so we concentrate on the “\( \supseteq \)” part. Fix some \( b \in \bigcap_{i=0}^{\infty} \text{offset}(A_i, \lambda_i) \). For every \( i \in \mathbb{N} \), there exists some \( a_i \in A_i \) such that \( (b-a_i) \in D_{\lambda_i} \). Now, the sequence \( (b-a_i)_{i \in \mathbb{N}} \) is a bounded sequence in \( \mathbb{R}^2 \) (bounded by \( D_{\lambda_0} \)) and therefore has a convergent sub-sequence by the well-known Bolzano-Weierstrass Theorem. Let \( r \) denote the limit point of this subsequence. In particular, the corresponding subsequence of \( (a_i)_{i \in \mathbb{N}} \)
2. Searching for a maximal valid radius

converges to \( a := b - r \). We show that \( a \in \bigcap A_i \) and \( r \in D_\lambda \) which suffices to prove the claim.

Assume that \( a \notin \bigcap A_i \). Then, there is some \( n_0 \) such that \( a \notin A_{n_0} \). Since \( A_{n_0} \) is closed, \( \varepsilon := d(a, A_{n_0}) > 0 \), where \( d \) is the Euclidean distance function. Moreover, because each \( A_n \) with \( n \geq n_0 \) is included in \( A_{n_0} \), \( d(a, A_n) \geq \varepsilon \). Because \( (a_i)_{i \in \mathbb{N}} \) converges to \( a \), we can find some \( N \geq n_0 \) such that \( d(a, a_N) < \varepsilon \). However, \( a_N \in A_N \), so

\[
d(a, A_N) \leq d(a, a_N) < \varepsilon = d(a, A_{n_0}) \leq d(a, A_N),
\]

which is a contradiction. The fact that \( r \in D_\lambda \) follows by a similar argument. \(\Box\)

Theorem 6.4. For arbitrary \( Q \) and \( r \), and \( \hat{\varepsilon} \) as in Definition 4.3, there exists a solution to the deconstruction problem.

Proof. Because of Corollary 2.9 we need to prove that

\[
Q \subseteq Q'_* := \text{offset}(\text{inset}(\text{offset}(Q, \hat{\varepsilon}), r), r + \hat{\varepsilon}).
\]

Let \( (\varepsilon_i)_{i \in \mathbb{N}} \) be a monotone decreasing sequence of real numbers that converges to \( \hat{\varepsilon} \). Because \( \varepsilon_i > \hat{\varepsilon} \) for each \( i \), \textsc{Decide} return \textsc{YES} for each \( \varepsilon_i \), which is equivalent to

\[
Q \subseteq Q'_i := \text{offset}(\text{inset}(\text{offset}(Q, \varepsilon_i), r), r + \varepsilon_i).
\]

It is therefore sufficient to prove that

\[
Q'_* = \bigcap_{i=0}^{\infty} Q'_i.
\]

For that, we apply Lemma 6.3 on the constant sequence \( (Q)_{i \in \mathbb{N}} \) and on \( (\varepsilon_i)_{i \in \mathbb{N}} \) to obtain

\[
\text{offset}(Q, \hat{\varepsilon}) = \bigcap_{i=0}^{\infty} \text{offset}(Q, \varepsilon_i).
\]

Applying Lemma 6.2 yields

\[
\text{inset}(\text{offset}(Q, \hat{\varepsilon}), r) = \bigcap_{i=0}^{\infty} (\text{inset}(\text{offset}(Q, \varepsilon_i), r),
\]

and applying Lemma 6.3 for the sequences \( (\text{inset}(\text{offset}(Q, \varepsilon_i), r))_{i \in \mathbb{N}} \) and \( (r + \varepsilon_i)_{i \in \mathbb{N}} \) yields

\[
\text{offset}(\text{inset}(\text{offset}(Q, \hat{\varepsilon}), r), r + \hat{\varepsilon}) = \bigcap_{i=0}^{\infty} \text{offset}(\text{inset}(\text{offset}(Q, \varepsilon_i), r), r + \varepsilon_i).
\]

\(\Box\)

Since \( r \) is no longer fixed, we now consider \( \hat{\varepsilon} \) as a function from \( \mathbb{R}^+ \) to \( \mathbb{R}^+ \) depending on \( r \); with a slight abuse of notation, we will let \( \hat{\varepsilon} \) by itself denote this function from now on.
Chapter 6. Optimizing $\varepsilon$ and $r$

**Theorem 6.5.** $\hat{\varepsilon}$ is a monotone increasing function. Moreover, $\hat{\varepsilon}$ is Lipschitz-continuous with Lipschitz factor 1.

**Proof.** We prove monotonicity first: Let $a < b$ denote two radii and $\varepsilon_a = \hat{\varepsilon}(a)$, $\varepsilon_b = \hat{\varepsilon}(b)$. We will show that DECIDE $(a, \varepsilon_b)$ returns YES, which proves that $\varepsilon_a \leq \varepsilon_b$.

We compare the intermediate results of DECIDE $(a, \varepsilon_b)$, denoted by $S_i$, to those of DECIDE $(b, \varepsilon_b)$, denoted by $B_i$:

- $S_1 = \text{offset}(Q, \varepsilon_b)$
- $S_2 = \text{inset}(S_1, a)$
- $S_3 = \text{offset}(S_2, a + \varepsilon_b)$

- $B_1 = \text{offset}(Q, \varepsilon_b) = S_1$
- $B_2 = \text{inset}(B_1, b) = \text{inset}(\text{inset}(S_1, a), b - a) = \text{inset}(S_2, b - a)$
- $B_3 = \text{offset}(B_2, b + \varepsilon_b) = \text{offset}(\text{offset}(B_2, b - a), a + \varepsilon_b)$

Since

$$\text{offset}(B_2, b - a) = \text{offset}(\text{inset}(S_2, b - a), b - a) \subseteq S_2$$

it follows that $B_3 \subseteq S_3$. Because DECIDE $(b, \varepsilon_b)$ returns YES by definition, it holds that $Q \subseteq B_3$, therefore $Q \subseteq S_3$ and DECIDE $(a, \varepsilon_b)$ also returns YES.

For Lipschitz continuity, let $a < b$ be such that $b - a \leq \delta$, and again $\varepsilon_a = \hat{\varepsilon}(a)$, $\varepsilon_b = \hat{\varepsilon}(b)$. We show that $\varepsilon_b - \varepsilon_a \leq \delta$. There exists a polygonal region $P$ that is a solution to the deconstruction problem for $Q$, $a$ and $\varepsilon_a$. In other words,

$$H(\text{offset}(P, a), Q) \leq \varepsilon_a.$$ 

Because of the general fact

$$H(A, B) \leq \varepsilon \Rightarrow H(\text{offset}(A, \delta), B) \leq \varepsilon + \delta,$$

and since $b \leq a + \delta$ we have that

$$H(\text{offset}(P, b), Q) \leq H(\text{offset}(P, a + \delta), Q) = H(\text{offset}(\text{offset}(P, a), \delta), Q) \leq \varepsilon_a + \delta.$$ 

Therefore, $P$ is a solution for the deconstruction problem for $Q$, $b$ and $\varepsilon_a + \delta$, so $\varepsilon_b \leq \varepsilon_a + \delta$. \hfill \Box

It follows from the monotonicity and Theorem 6.4 that $R$ is an interval which has 0 as its left endpoint. Thus, computing $R$ reduces to finding the maximal $r^* > 0$ such that $\hat{\varepsilon}(r^*) = \varepsilon$.

Using the exact decision procedure, we can perform a binary search similar to that for approximating $\hat{\varepsilon}$: First, we compute an interval $[0, r]$ containing $r^*$. Since $Q$ is finite, we can take $r$ to be the radius of the smallest enclosing circle of $Q$ plus $\varepsilon$. Then, we start the iterative process, deciding on the left or right subinterval depending on the result of DECIDE for $Q$, $\varepsilon$ and the midpoint of the interval.

What if we are using APPROXDECIDE instead? Unlike Algorithm 7, we can no longer guarantee that every execution of the approximation algorithm halves the search interval, because a return value UNDECIDED does not bound the distance of the current radius $r$ to the critical value $r^*$. Instead, we propose the following scheme: For an interval $I$ with midpoint $r$, APPROXDECIDE is called with some $\delta$, initially set to $\frac{\varepsilon}{2}$. If it returns UNDECIDED, $\delta$ is divided by 2 and APPROXDECIDE is recalled. Eventually, the algorithm returns YES or NO, and the interval $I$ can be halved.
Let $R'$ be the preimage of $\varepsilon$ under $\hat{\varepsilon}$. Note that $R'$ is an interval (which may consist of only one point). The algorithm from above is guaranteed to converge to some $r \in R'$. However, if $R'$ contains more than one point, it is not guaranteed to converge to the maximal one (because it gets stuck in an infinite loop as soon as the query value $r$ lies in $R'$). One way of avoiding this infinite loops is to decrease $\delta$ only to some threshold and choosing another query value $r$ from the interval if no decision was made. Nevertheless, we have not found an approximation algorithm with the formal guarantee of converging to the largest value in $R'$ eventually.

6.3 Searching for a good combination of $r$ and $\varepsilon$

We finally consider the question of how we can find a reasonable choice of $r$, $\varepsilon$ and a polygonal region $P$, such that $P$ is a solution for the deconstruction problem for $Q$, $r$, and $\varepsilon$. The meaning of “reasonable” depends on the application context, and possible prior information (for instance, a range of possible offset radii). We offer a basic generic approach and justify our choice with an example.

Generally, we expect from a reasonable pair $(r, \varepsilon)$ that $\varepsilon$ is small. So, in order to judge whether a good solution exists for radius $r$, we consider $\hat{\varepsilon}(r)$. However, $\hat{\varepsilon}(r)$ being small (or equivalently, $\frac{1}{\hat{\varepsilon}(r)}$ being large) is not a good criterion, because $\hat{\varepsilon}$ is monotone increasing according to Theorem 6.5, so $r = 0$ would always be the best solution.\footnote{Note that this is also formally correct, because $P := Q$ is the perfect solution for $r = 0$.} In order to remove the bias towards small radii, we scale the objective function and consider

$$J : \mathbb{R}^+ \rightarrow \mathbb{R}^+, r \mapsto \frac{r}{\hat{\varepsilon}(r)}.$$  

Note that $J$ is well-defined on the positive axis and continuous. Moreover, we can approximate the graph of $J$ in any finite interval of $r$-values by choosing a sample of the interval and approximating $\hat{\varepsilon}$ at each sample value using Algorithm 7.

We demonstrate by an example that the local maxima of $J$ yield radii that lead to good deconstruction results. Consider the polygonal region defined in Figure 6.1a and its (approximated) $J$-graph in Figure 6.1b. We can identify two local maxima $r_1$ and $r_3$; we have plotted the corresponding solutions in Figure 6.2. Indeed, we see that for the large radius $r_3$, we obtain a relatively simple solution whose offset blurs away the spikes of $Q$. For the smaller local maximum at $r_1$, we obtain a solution with more details such that the spikes can be approximated almost perfectly with the given radius. In contrast, the shape at the local minimum $r_2$ combines the disadvantages of the two previous cases: the solution is similarly complicated as the $r_1$-solution (it contains flattened versions of all the spikes), but its approximation quality is not significantly better than for the simpler $r_3$ solution which achieves the same with a much larger radius.
Chapter 6. Optimizing $\varepsilon$ and $r$

(a) The polygon

(b) The $J$-graph approximation with $\Delta = \frac{1}{\sqrt{2}}$

Figure 6.1: The Flower polygon example: 10 samples per radius unit for $r \leq 5$.

(a) $J$ maximum at $r_1 = 0.6$

(b) $J$ minimum at $r_2 = 1.4$

(c) $J$ maximum at $r_3 = 4.4$

(d) $\tilde{\varepsilon}_\Delta(r_1) \approx 0.082 \cdot r_1 = 0.049 \ldots$

(e) $\tilde{\varepsilon}_\Delta(r_2) \approx 0.202 \cdot r_2 = 0.283 \ldots$

(f) $\tilde{\varepsilon}_\Delta(r_3) \approx 0.099 \cdot r_3 = 0.439 \ldots$

Figure 6.2: Flower polygon approximations for the $(r, \tilde{\varepsilon}_\Delta(r))$ values at $J$-graph extrema. In the upper row, approximations of the solutions are shown in green, and their $r$-offsets ($\tilde{\varepsilon}_\Delta(r)$-close to the input) in dark blue. In the lower row the input polygon is shown in blue. The $\tilde{\varepsilon}_\Delta$-width cyan stripe around the $r$-offset demonstrates the quality of the approximation.
Conclusions

We have shown how to decide whether a given arbitrary polygonal shape $Q$ is composable as the Minkowski sum of another polygonal region and a disk of radius $r$, up to some tolerance $\varepsilon$. We have implemented an approximative variant of our decision procedure and demonstrated how it can be applied to the parameter optimization of the offset-deconstruction problem. Many related questions remain open.

- Deconstruction of Minkowski sums seems more difficult when both summands are more complicated than a disk; many practical scenarios may raise this general deconstruction problem.
- It would be interesting to analyze the deconstruction not only under the Hausdorff distance but for other similarity measures, such as the Fréchet or the symmetric distance.
- Can one remove the extra vertex when seeking an optimal (vertex minimal) polygonal summand $P$ in the convex case.
- Finding an optimal or near-optimal polygonal summand in the non-convex case seems challenging.
- As in polygonal simplification, we could also search for the polygonal region with a given number of vertices whose $r$-offset minimizes the (Hausdorff) distance to the given shape.
- The offset-deconstruction problem can be reformulated in higher dimensions. We consider especially the three-dimensional case to be of practical relevance.


