Offset Polygon and Annulus Placement Problems

Alex Goryachev
Offset Polygon and Annulus Placement Problems

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

Alex Goryachev

Submitted to the Senate of
the Technion — Israel Institute of Technology

Elul 5765 Haifa September 2005
The Research Thesis was done under the supervision of Prof. Gill Barequet in the Faculty of Computer Science.

I would like to thank Gill Barequet for his guidance and support.

The generous financial help of the Technion during my studies is gratefully acknowledged.
# Contents

Abstract .................................................. 1

Notation and Abbreviations .......................... 3

1 Introduction ........................................... 5
  1.1 Motivation ........................................ 5
  1.2 Offset Polygons and Annuli ...................... 6
  1.3 Skeletons .......................................... 8
    1.3.1 Straight skeleton ............................ 8
    1.3.2 Voronoi diagram ............................. 12
  1.4 Statement of the Problems ....................... 15
  1.5 Previous Work .................................... 16
    1.5.1 Max-Cover ................................... 17
    1.5.2 Containment .................................. 19
    1.5.3 Partial Containment .......................... 22
  1.6 Our Results ...................................... 24
  1.7 Thesis Outline ................................... 27

2 The Translation-Rotation-Offset Diagram ......... 29
  2.1 Preface ........................................... 29
  2.2 Structure of the Diagram ......................... 31
3.2.7 Total running-time complexity ........................................ 85
3.3 Extensions ................................................................. 86
  3.3.1 Linearized offset ..................................................... 86
  3.3.2 True offset ............................................................ 87

4 The Number of Stable Poses ............................................... 89
  4.1 Two-Point Stable Poses ............................................... 89
  4.2 Three-Point Stable Poses ............................................. 91
  4.3 Four-Point Stable Poses .............................................. 94

5 Conclusion ........................................................................ 101

Bibliography ........................................................................ 103
### List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Algorithms for the Max-Cover problem (running times)</td>
<td>19</td>
</tr>
<tr>
<td>1.2</td>
<td>Algorithms for the Containment problem for scaling (running times)</td>
<td>22</td>
</tr>
<tr>
<td>1.3</td>
<td>Algorithms for the Containment problem for offsetting (running times)</td>
<td>23</td>
</tr>
<tr>
<td>1.4</td>
<td>Algorithms for the Partial Containment problem for scaling (running times)</td>
<td>25</td>
</tr>
<tr>
<td>1.5</td>
<td>Algorithms for the Partial Containment problem for offsetting (running times)</td>
<td>25</td>
</tr>
</tbody>
</table>
List of Figures

1.1 Offset polygons and annuli ........................................ 7
1.2 A polygon $P$ and its straight skeleton .......................... 10
1.3 A polygon $P$ and its Voronoi diagram .......................... 13

2.1 A triangle sliding on two lines ................................. 32
2.2 The polygon $P$ and the ellipse $E_{p3}$ ......................... 33
2.3 The parameters $t$ and $\theta$ ................................... 37
2.4 A sample diagram of containing regions ....................... 39
2.5 The angles $\alpha_{ij}$ ........................................ 40
2.6 The angles $\beta_{ij}$ ........................................ 42
2.7 The obtainable region is the intersection of two unions of pairs of ranges 42
2.8 Relative movement of $\ell(e_1)$ and $\ell(e_2)$ .................. 45
2.9 The angles $\alpha_{11}$ and $\alpha_{12}$ ............................... 46
2.10 The $\alpha$ and $\beta$ curves and the obtainable region .......... 47
2.11 The diagram and the optimum placement ...................... 54
2.12 A horizontal cross-section of a translation-rotation-offset diagram is a translation-rotation diagram .................. 56
2.13 A vertical cross-section of a translation-rotation-offset diagram is a translation-offset diagram .................. 57
2.14 Covering points ............................................. 62
2.15 The polygon $\hat{P}$ ........................................ 63
2.16 A four-bar linkage ........................................... 66

3.1 Stable and nonstable poses involving a vertex and an edge .......... 72
3.2 Stable and non-stable poses involving two vertices .................. 72
3.3 Stable poses involving two edges ................................ 73
3.4 Three points on three lines ...................................... 78
3.5 Finding stable poses for three points .............................. 81
3.6 Four points on four lines ........................................ 83

4.1 \(\Omega(n^2m^2)\) two-point stable poses ............................. 90
4.2 \(\Omega(n^3m^3)\) three-point stable poses ............................ 92
4.3 A polygon \(P\) and its reflection \(P^R\) ............................. 95
4.4 \(\Omega(n^4m^3)\) four-point stable poses ............................ 97
Abstract

The $\delta$-annulus of a polygon $P$ is the closed region containing all points in the plane at distance at most $\delta$ from the boundary of $P$. An inner (resp. outer) $\delta$-offset polygon is a polygon defined by an inner (resp. outer) boundary of its $\delta$-annulus.

In this thesis we address three major problems of covering a given point set $S$ by an offset version or a polygonal annulus of a polygon $P$. First, the Max-Cover objective is, given a value of $\delta$, to cover as many points from $S$ as possible by the $\delta$-offset (or by the $\delta$-annulus) of $P$, allowing translation and rotation. Second, the Containment problem is to minimize the value of $\delta$ such that there is a transformation (rotation and translation) of the $\delta$-offset (or the $\delta$-annulus) of $P$ that covers all points from $S$. The Partial Containment dilemma is very similar to the Containment problem, only this time we seek the minimum offset of $P$ covering $k \leq |S|$ points. We address several variants of these problems, including convex and simple polygons, as well as polygons with holes and sets of polygons.

These problems arise in many applications where one needs to match a given polygonal figure (a known model) to a set of points (usually, obtained measures.)

In order to resolve the above mentioned issues, we exploit two approaches. One approach is based on “translation-rotation-offset diagrams,” and the other is based on “stable poses.”

In our first solution we present a two-point translation-rotation-offset diagram, which accumulates the information of all the placements of a given polygon under
translation, rotation, and offset that keep two points of $S$ on the boundary of the polygon. We analyze the structure and complexity of this diagram, present an algorithm to build it, and show how it can be used to solve the above problems.

The second solution is based on so-called stable poses. We solve the Containment and Partial Containment problems by enumerating all the “suspectable” placements of the polygon. The resulting algorithm is rather simple and space-efficient.

We analyze the complexity of our approaches, and also prove that our diagram-based solution is almost optimal among the algorithms that enumerate all the placements of the polygon.
Notation and Abbreviations

$P$ an input polygon
$S$ an input point set
$m$ the complexity of $P$
$n$ the cardinality of $S$
$k$ the number of points in $S$ covered by $P$
$p_1, p_2, p_3, p_4$ points in $S$
$e_1, e_2, e_3, e_4$ edges of $P$
$p_1p_2$ the line segment $p_1p_2$
$\text{SS}(P)$ the straight skeleton of $P$
$\text{VD}(P)$ the Voronoi diagram of $P$
Chapter 1

Introduction

1.1 Motivation

Imagine a robot that navigates in a polygonal environment using a map. The robot is equipped with a ranging device that allows it to sample the environment. Its goal is to move from one point to another. For this purpose it has to locate itself in the given map. This task is referred to in the robotics literature as the robot global localization problem (see, e.g., [BEF96]). The problem is to match the map, which is given as a polygon, to the obtained measures—a set of points sampled from the real environment.

The problem of matching a given polygon to a set of points arises in several other fields too. One of its applications is the pattern matching problem in Computer Vision [HU90]. Another problem is geometric tolerancing [CY97], where one compares objects produced by some manufacturing process to a desired (polygonal) pattern. The manufactured object is measured by some sampling device to obtain a set of points, which are then matched to the given polygon.

In all the applications mentioned above, the measures introduce an error that may be significantly large for some part of the sampled points. That is why the matching
process usually uses some tolerance $\delta > 0$ and even discards points with a large error. Thus, the samples should be matched not to the given polygon, but rather to an annulus of width $\delta$, for some value of $\delta$, that is similar to the original polygon.

### 1.2 Offset Polygons and Annuli

**Definition 1 (Offset Annulus)** The $\delta$-annulus of a polygon $P$ (for $\delta \geq 0$) is the closed region containing all points in the plane at distance at most $\delta$ from the boundary of $P$.

Figure 1.1(a) shows a sample offset annulus of a simple polygon.

In various applications we want to fix the inner or outer boundary of a polygonal annulus. For example, in geometric tolerancing, if a manufactured object is to fit inside a sleeve, then its outer boundary should be fixed. If an object must fit over a peg, then its inner boundary must be fixed. Thus we define:

**Definition 2 (Constrained Offset Annulus)** The inner (resp., outer) constrained $\delta$-annulus of a polygon $P$ is the closed region containing all points inside (resp., outside) $P$ at distance at most $\delta$ from the boundary of $P$.

**Definition 3 (Offset Polygon)** Given a polygon $P$ and $\delta < 0$, the $\delta$-offset polygon is the boundary portions of the $|\delta|$-annulus of $P$ that are properly contained by $P$. Similarly, for $\delta > 0$, the $\delta$-offset polygon is the boundary portions of the $\delta$-annulus of $P$ outside of (i.e., properly containing) $P$.

The 0-offset polygon identifies with $P$ by definition. A $\delta$-offset polygon, for $\delta < 0$, is often referred to as inner offset polygon, and for $\delta > 0$ as outer offset polygon.

Note that both outer and inner offset polygons of $P$ are made up of line segments and circular arcs, where the latter appear as the offsets of convex (resp., concave)
Figure 1.1: Offset polygons and annuli

(a) $\delta$-annulus

(b) True (solid line) and linearized (dotted line) offset polygons

(c) A linearized offset polygon with a hole

(d) A linearized offset annulus with two holes
vertices in outer (resp., inner) offsets. We also define a linearized version of these
offset polygons. In what follows, we shall refer to an offset polygon as a true offset
to distinguish it from the linearized version.

**Definition 4 (Linearized Offset Polygon)** The linearized $\delta$-offset polygon of $P$
is created from the $\delta$-offset polygon of $P$ by extending each of the straight edges until
they meet the extensions of other straight edges.

Note that a more precise definition would be obtained by using the offset process;
see below. See Figure 1.1(b) for examples of true and linearized offset polygons.

An offset polygon is not necessarily simple. See, for example, Figure 1.1(c), where
the linearized offset polygon has a hole. It may even consist of several polygons.
An offset annulus is also not necessarily a simply-connected polygonal region. Fig-
ure 1.1(d) shows an example of a linearized offset annulus (shown in gray) bounded
by three polygons: one boundary and two holes.

Throughout this thesis we use the following notation: $O_{P,\delta}$ is the true $\delta$-offset of
a polygon $P$, while $O^L_{P,\delta}$ denotes the linearized $\delta$-offset of $P$. We denote by $A_{P,\delta}$ the
$\delta$-annulus of $P$. Similarly, $A^L_{P,\delta}$ stands for the linearized version of $A_{P,\delta}$.

### 1.3 Skeletons

#### 1.3.1 Straight skeleton

Consider a simple polygon $P$. Imagine that the boundary of $P$ is shrunk toward
its interior in a self-parallel manner and at the same speed for all edges. In other
words, each edge moves in the direction orthogonal to itself, and all edges move with
equal speed. The lengths of the edges change during this process. Each vertex of $P$
moves along the angular bisector of its incident edges. The process continues until
the boundary changes topologically. It can happen in one of the following two ways:
• A vertex crashes into an edge. As a result, the edge is split, breaking the polygon into two parts, and the process continues separately for each part.

• An edge shrinks to a point. Its neighboring edges become adjacent and the process continues.

The process ends when the polygon (or each of its parts, if the polygon is broken into pieces during this process) collapses to a point (or a line segment in the degenerate case). This has been coined by [AAAG95] as the shrinking process.

Now imagine that edges of $P$ start to move outward according to the same rules as before. We call this process, together with the shrinking process, the offset processes; see Figure 1.2(a) for an example. We see that the linearized offset polygons of $P$ are “snapshots” of a continuous offset process. Note in the figure how during the offset process an edge $e \in P$ is split into two pieces. At each offset, we call the union of these pieces an edge $e$, and each one of them—a piece of $e$.

The union of all polygonal paths followed by vertices of $P$ and all newly formed vertices during the offset process is called the straight skeleton of $P$. It is defined more formally as [AA98]:

**Definition 5 (Straight Skeleton)** The straight skeleton of a simple polygon $P$ is the union of the pieces of angular bisectors traced out by polygon vertices during the offset process.

We denote the straight skeleton of $P$ by $SS(P)$. Figure 1.2(b) shows the straight skeleton of a simple polygon.

The following properties of the $SS(P)$ are stated in [AAAG95, AA98]:

• The $SS(P)$ is a unique structure defining a polygonal partition of the plane.

• Each edge $e \in P$ sweeps two areas: one inside $P$ and one outside $P$. It is proven in [AA98] that each one of these two areas is monotone with respect to
Figure 1.2: A polygon $P$ and its straight skeleton
its defining edge. We call their union the *skeletal face of e*. There are exactly \( m \) simply-connected skeletal faces. It is easy to prove that the united face is also monotone with respect to its defining edge.

- Bisector pieces are called *arcs*, and their endpoints, which are not vertices of \( P \), are called *nodes* of \( \text{SS}(P) \).

- During the shrinking process, \( P \) gets smaller until it disappears. The point(s) to which \( P \) shrinks is(are) called the *center of \( \text{SS}(P) \)*. The center may be a point, a set of distinct points, a line segment, or a set of points and segments. The center of the polygon shown in Figure 1.2(b) is a single point denoted by \( C(P) \). We denote by \( \delta_{\text{min}} \) the (negative) offset at which \( P \) vanishes.

- The total complexity of \( \text{SS}(P) \) is \( \Theta(m) \). Note that the worst-case complexity of a single face of \( \text{SS}(P) \) is also \( \Theta(m) \), but the total complexity of all \( m \) faces is still \( \Theta(m) \).

- During the offset process, an edge of \( P \) may be split into \( \Theta(m) \) pieces. However, the number of all pieces of edges created during the offsetting is also \( \Theta(m) \).

There has been a lot of work on computing the straight skeleton. We briefly summarize it below:

- Aichholzer et al. [AAAG95] developed the first algorithm that constructed the inner part of the skeleton of a simple \( m \)-gon; it runs in \( O(m^3) \) time and requires linear space. They called this algorithm “trivial,” and mentioned a different variant of it that runs in \( O(m^2 \log m) \) time and uses \( O(m^2) \) storage.

- Aichholzer and Aurenhammer [AA98] introduced another algorithm that builds the entire skeleton of any planar polygonal figure (possibly consisting of several polygons and polylines) of total complexity \( \Theta(m) \). Its running time is
\(O(m^3 \log m)\),\(^1\) and it uses linear storage. Although in the worst case it is slower than the trivial algorithm, for most practical applications the authors observed running times close to \(O(m \log m)\).

- Felkel and Obdržálek [FO98] presented an algorithm that works for simple \(m\)-gons and for polygons with holes. It constructs the inner part of the skeleton in \(O(m(r + \log m))\) time, where \(r\) is the number of reflex vertices of the polygon. \((r = \Theta(m)\) in the worst case, in which case the algorithm runs in \(\Theta(m^2)\) time). The algorithm uses \(\Theta(m)\) space.

- Eppstein and Erickson [EE99] introduced another algorithm for constructing the inner part of the straight skeleton of a simple \(m\)-gon, which runs in \(O(m^{1+\varepsilon} + m^{8/11+\varepsilon} r^{9/11+\varepsilon})\) time and space, where \(r\) is the number of reflex vertices of the polygon. The running time is, thus, \(O(m^{17/11+\varepsilon})\) in the worst case. Although in theory it is fast, this algorithm is impractical and the authors suggested two practical variants that run in \(O(m(r + \log m))\) time \((O(m^2)\) in the worst case) using \(O(m + r^2)\) space, and in \(O(m(r + \log m) + r^2 \log r)\) time \((O(m^2 \log m)\) in the worst case) using linear space.

1.3.2 Voronoi diagram

In the previous section we saw a process that produces the linearized offsets of a polygon. Now we turn to true offsets. The process is essentially the same, with one difference: we consider a vertex as an arc of a circle of an infinitely-small radius. This arc connects the endpoints of the adjacent straight edges. When the offset process starts, the circles grow with the same speed as the edges move. The process continues in the same manner until one of the following occurs:

\(^1\)An erroneous bound of \(O(m^2 \log m)\) was given in [AA96].
• An edge crashes into another edge. As a result, both edges are split and two new vertices are created. The polygon is broken into two parts, and the process continues separately for each part.

• An edge shrinks to a point. Its neighboring edges become adjacent and the process continues.

Preparata and Shamos [PS85] described this process as a “prairie fire,” which is set simultaneously at the entire boundary of the polygon and propagates at constant speed in all directions. See Figure 1.3(a) for an example. During this offset process, the original and newly created vertices of $P$ draw a structure that is called the Voronoi diagram of $P$ [PS85, p. 262].

![Offset process and Voronoi diagram](image)

Figure 1.3: A polygon $P$ and its Voronoi diagram

**Definition 6 (Voronoi Diagram)** The Voronoi diagram of a set of $k$ sites in the plane is a subdivision of the plane into $k$ cells, such that a cell of a site $p$ contains the points that are closer to $p$ than to any other site.
By the Voronoi diagram of a simple polygon \( P \) we mean the Voronoi diagram whose sites are vertices and edges of \( P \). We denote it by \( VD(P) \); see Figure 1.3(b) for an example.

The Voronoi diagram is a well-studied structure (see, e.g., [OBSC00, AK00]). Below we mention some of the properties of the Voronoi diagram of a simple \( m \)-gon \( P \):

- \( VD(P) \) is a unique structure that partitions the plane into \( 2m \) connected cells, one for each site (vertex or open edge). These cells are called Voronoi regions.
- The boundary edges of the Voronoi regions are called Voronoi edges, and the vertices of the diagram are called Voronoi vertices.
- Voronoi edges are either straight line segments or parabolic arcs.
- The total complexity of \( VD(P) \) is \( \Theta(m) \). The worst-case complexity of one region of \( VD(P) \) is also \( \Theta(m) \).

Note that a parabolic arc of \( VD(P) \) always represents the motion of two vertices during the offset process. Thus, we divide it into two parts, each one corresponding to one of the vertices. There has been a lot of work on computing the Voronoi diagram of a simple polygon given as an ordered list of vertices. We summarize it briefly:

- There is a variety of \( O(m \log m) \)-time and \( O(m) \)-space algorithms for building \( VD(P) \): divide-and-conquer approaches [Kir79, Lee82, Yap87], plane sweep [For86], and randomized incremental insertion [BDS+92, KMM93]. All these methods work also for general planar straight-line graphs.
- Devillers [Dev92] devised a randomized \( O(m \log^* m) \)-time algorithm, where \( \log^* m \) is a function that calculates how many times one would need to take
the logarithm of \( m \) before going below 2. It is an extremely slow-growing function and can be regarded as a constant for any practical application. This algorithm also works for any connected planar straight line graph.

- Chin et al. [CSW95], and Klein and Lingas [KL95] achieved optimal linear-time deterministic algorithms.

With a slight abuse of notation, we shall call both variants (true and linearized) “offset processes”. The exact meaning will be clear from the context.

1.4 Statement of the Problems

Problem 1 (Max-Cover) Given a set \( S \) of \( n \) points in the plane, a polygon \( P \), and an offset \( \delta \), find a placement \( \tau \) that maximizes the number of points of \( S \) contained in \( \tau(O_{P,\delta}) \).

Problem 2 (Containment) Given a set \( S \) of \( n \) points in the plane and a polygon \( P \), find the smallest value of \( \delta \), such that there exists a placement \( \tau \) with all \( n \) points of \( S \) being contained in \( \tau(O_{P,\delta}) \).

Problem 3 (Partial Containment) Given a set \( S \) of \( n \) points in the plane, a polygon \( P \), and an integer \( k \leq n \), find the smallest value of \( \delta \), such that there exists a placement \( \tau \) with \( k \) points of \( S \) being contained in \( \tau(O_{P,\delta}) \).

One may consider a few variants of these problems:

- \( P \) can be a convex, simple, or even not simply-connected polygon.

- Scaling can be used instead of offsetting.

- If offsetting is used, it can be true or linearized.
• The points can be covered by the entire $\delta$-offset polygon, by its $\delta$-annulus, by its constrained inner $\delta$-annulus, or by its constrained outer $\delta$-annulus.

• The placement $\tau$ can be a translation only, a rotation only (where the center of rotation is given), or a composition of a translation and a rotation.

• The algorithm can be off-line or on-line. That is, the points can all be known in advance, or can be given and processed one at a time. (In this thesis we consider only off-line problems.)

1.5 Previous Work

Some of the problems and their variants mentioned above have already been addressed in the literature. However, most work was done for scaling rather than offsetting, although offsetting is a more appropriate way to describe errors in the applications mentioned in the introduction. For example, in a manufacturing process, the absolute error of a production tool (milling head, laser beam, etc.) is independent of the location of the produced feature relative to some artificial reference point (the origin). Thus, a tool is more likely to allow (and expect) local errors bounded by some tolerance, rather than scaled errors relative to some (arbitrary) center. In addition, most of the researchers limited themselves to convex polygons, while most real-life problems involve non-convex, and sometimes even not simply-connected regions. Covering points by a polygonal annulus has received less attention than covering by an entire polygon, although it arises more frequently. That is, in most applications one tries to match a given point set to the boundary of a polygon—in other words, to cover the set by an annulus.
1.5.1 Max-Cover

- **Convex polygons**

  - For convex polygons under translation only, the Max-Cover problem was solved by Barequet et al. [BDP97]. Their algorithm runs in $O(nk \log(km) + m)$ time and $\Theta(n + m)$ space, where $k$ is the maximum number of points from $S$ that can be covered simultaneously by $P$. Their algorithm proceeds as follows. Fix a vertex $v \in P$. Place $n$ copies of $P$ reflected around $v$, such that for each copy, $v$ identifies with some point from $S$. Find a deepest point in the resulting arrangement by using an “anchored sweep.” The maximum depth of the arrangement is $k$.

  Scharf [Sch04] gave another algorithm to solve this problem, which runs in $O(n^2 \log(nm) + m)$ time and linear space.

  - For convex polygons under translation and rotation, the Max-Cover problem was solved by Dickerson and Scharstein [DS96]. They noticed that for the target position of $P$, there is at least one point of $S$ on the boundary of $P$. Thus, they built $n$ “rotation diagrams,” where each diagram represents all the placements (translation and rotation) of $P$ that are in contact with a fixed point from $S$. In these diagrams they searched for a maximal-depth region. The complexity of this approach is $O(n^2 km^2 \log(nm))$ time and $\Theta(n + m)$ space, where $k$ is the maximum number of points from $S$ that can be covered simultaneously by $P$. The authors claimed that using bucketing, the running time of the algorithm can be reduced to $O(nk^2 m^2 \log(nm))$, which is still $O(n^3 m^2 \log(nm))$ in the worst case, in which $k$ and $n$ are comparable.

  - For annuli of convex polygons under translation only, the Max-Cover problem was solved by Barequet et al. [BBDG98] (by a simple extension of the
technique of [BDP97]) in $O(n^2 \log(nm) + m)$ time and $\Theta(n + m)$ space.

- For annuli of convex polygons under translation and rotation, the Max-Cover problem was solved by Barequet et al. [BBDG98] (by a simple extension of the rotation-diagram technique of [DS96]) in $O(n^3 m^2 \log(nm))$ time and $\Theta(n + m)$ space.

- For constrained annuli of convex polygons under translation only, the Max-Cover problem can be solved in $O(n^2 \log(nm) + m)$ time and $\Theta(n + m)$ space for both inner and outer offsets (by simply extending [BDP97].)

- For constrained annuli of convex polygons under translation and rotation, the Max-Cover problem can be solved (by simply extending [DS96]) in $O(n^3 m^2 \log(nm))$ time and $\Theta(n + m)$ space for both inner and outer offsets.

- Simple polygons

  - For simple polygons under translation only, the Max-Cover problem can be solved by trivially extending the technique of [BDP97] for convex polygons. However, the running time and the space complexity will increase to $O(n^2 m^2 \log(nm))$ and $O(nm)$, respectively.

  - For annuli of simple polygons under translation only, the Max-Cover problem was solved by Barequet et al. [BBDG98] under some “fatness” condition, that is, only offsets that keep the offset polygons simple are considered. The authors used the same technique of [BDP97]. The resulting algorithm runs in $O(n^2 m^2 \log(nm))$ time and $O(nm^2)$ space. (However, the space complexity can be reduced to $O(nm)$ by keeping only the needed events in the event queue.)

  - For constrained annuli of simple polygons and under translation only,

\footnote{An erroneous bound of $O(n^3 \log(nm) + m)$ was given in [BBDG98].}
the Max-Cover problem can be solved (by simply extending [BDP97]) in $O(n^2m^2 \log(nm))$ time and $O(nm)$ space for both inner and outer annuli.

Table 1.1 summarizes the known results for variants of the Max-Cover problem.

<table>
<thead>
<tr>
<th>Type</th>
<th>Operation</th>
<th>Polygon</th>
<th>Annulus</th>
<th>Constrained annulus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex</td>
<td>Translation</td>
<td>$O(nk \log(km) + m)$ [BDP97]</td>
<td>$O(n^2 \log(nm) + m)$ [BDDG98]</td>
<td>$O(n^2 \log(nm) + m)$ extending [BDP97]</td>
</tr>
<tr>
<td></td>
<td>Translation and rotation</td>
<td>$O(nk^2m^2 \log(nm))$ [DS96]</td>
<td>$O(n^3m^2 \log(nm))$ [BBDG98]</td>
<td>$O(n^3m^2 \log(nm))$ extending [DS96]</td>
</tr>
<tr>
<td>Simple</td>
<td>Translation</td>
<td>$O(n^2m^2 \log(nm))$ extending [BDP97]</td>
<td>$O(n^2m^2 \log(nm))$ extending [BDDG98]</td>
<td>$O(n^2m^2 \log(nm))$ extending [BDP97]</td>
</tr>
<tr>
<td></td>
<td>Translation and rotation</td>
<td>$O(n^3m^3 \log(nm))$† (this thesis)</td>
<td>$O(n^3m^3 \log(nm))$† (this thesis)</td>
<td>$O(n^3m^3 \log(nm))$† (this thesis)</td>
</tr>
</tbody>
</table>

Table 1.1: Algorithms for the Max-Cover problem (running times)

1.5.2 Containment

- Convex polygons
  - Scaling
    
    * For convex polygons under translation only, the Containment problem for scaling was addressed in [BD00], where the authors solved it in $O(n^2(n+m) \log(nm))$ time and $O(n(n+m))$ space. They built the so-called “translation-scale diagram,” which is a version of the “rotation diagram” introduced in [DS96]. The region of depth $n$ in this diagram

†Θ(·) in the worst case.
represents all the placements of the polygon that contain all the $n$ points. All that is left to find is a point in this region that represents the minimal scaling.

Scharf [Sch04] improved the running time of the deterministic algorithm of [BD00] to $O(n \log h + h^2(\log h + m))$, where $h$ is the complexity of the convex hull of $S$. In the worst case this time bound is $O(n^2(\log n + m))$. This algorithm uses $\Theta(n + m)$ space.

Another solution, also presented in [Sch04], is to find the convex hull of $S$ first, and then find the maximal scaled version of it that can be placed inside $P$. The latter problem was solved by Toledo [Tol91] in linear time. Thus, the total time complexity of this approach is $O(n \log h + m)$. In the worst case this time bound is $O(n \log n + m)$.

Barequet et al. [BBDG05] solved this problem by a randomized algorithm, which runs in $O(n \log m + m)$ expected time and linear space.

They used an extension of a randomized incremental algorithm for finding the smallest enclosing circle (see [Wel91] and [BKOS97, §4.7]).

* For convex polygons under translation and rotation, the Containment problem for scaling can be solved using the above technique of finding the placement of the largest copy of the convex hull of $S$ inside $P$. Under translation and rotation, the problem of finding the placement of the largest copy of a convex $n$-gon that can be placed inside another convex $m$-gon was solved by Agarwal et al. [AAS98] in $O(nm^2 \log n)$ time using $O(nm^2)$ storage. Thus, this is also the complexity of the entire algorithm.

Barequet and Scharf [BS05] presented the “translation-rotation-scale” diagram, which is an extension of the rotation diagram of [DS96]. The authors solved the Containment problem for scaling and convex
polygons under translation and rotation in $O(n^4m^4 \log(nm))$ time and $O(n^3m^4)$ space using this diagram.

- For constrained scale-annuli of convex polygons under translation only, the Containment problem was solved in [BBDG05]. This was done in $O((n + m) \log(nm))$ time for inner annuli, and in $O(nm \log(nm))$ time for outer annuli. For an inner annulus, the algorithm builds a feasible region (that is, the region in which the center of $P$ may lie) and the nearest-site Voronoi diagram of $S$ with respect to the polygon-scale distance function. Then, it is left to check vertices and intersection points of the two structures. For the outer annulus, the technique is essentially the same except using the furthest-site Voronoi diagram.

- Offsettings

  - For convex polygons under translation only, the Containment problem for offsetting was solved in [BBDG05] by a randomized algorithm that runs in $O(n \log^2 m + m)$ expected time and linear space.

  - For annuli of convex polygons under translation only, the Containment problem was solved by Barequet et al. [BBDG98] in $O(n \log m \log(nm) + m)$ time and linear space. The authors built the nearest- and furthest-site Voronoi diagrams of $S$ with respect to convex polygon-offset distance function (introduced in [BDG01]). Next, they applied a technique similar to [DGR97] (where the authors minimized the width of a circular annulus).

  - For constrained annuli of convex polygons under translation only, the Containment problem was solved in [BBDG05] in $O(n(\log n + \log^2 m) + m \log(nm))$ time for inner annuli and in $O(nm \log(nm))$ or $O(n \log n \log(nm) + m)$ time for outer annuli.
Tables 1.2 and 1.3 summarize the known results for variants of the Containment problem for scaling and offsetting, respectively.

<table>
<thead>
<tr>
<th>Type</th>
<th>Operation</th>
<th>Polygon</th>
<th>Annulus</th>
<th>Constrained annulus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex</td>
<td>Translation</td>
<td>$O(n^2(n + m) \log(nm))$ [BD00]</td>
<td></td>
<td>inner: $O((n + m) \log(n + m))$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$O(n \log m + m)$ expected [BBDG05]</td>
<td></td>
<td>outer: $O(nm \log(nm))$ [BBDG05]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$O(n \log h + h^2(\log h + m))$ [Sch04]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$O(n \log h + m)$ using [Tol91]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Translation and rotation</td>
<td>$O(nm^4 \log n)$ [AAS98]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$O(n^4m^4 \log(nm))$ [BS05]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple</td>
<td>Translation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Translation and rotation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1.2: Algorithms for the Containment problem for scaling (running times)

1.5.3 Partial Containment

- Convex polygons
  - Scaling

  * For convex polygons under translation only, the Partial Containment problem for scaling was solved in [BD00] using the translation-scale diagram in $O(n^2(n + m) \log(nm))$ time and $O(n(n + m))$ space.
<table>
<thead>
<tr>
<th>Type</th>
<th>Operation</th>
<th>Polygon</th>
<th>Annulus</th>
<th>Constrained annulus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex</td>
<td>Translation</td>
<td>$O(n \log^2 m + m)$ [BBDG05]</td>
<td>$O(n \log m \log(nm) + m)$ [BBDG98]</td>
<td>inner: $O(n \log n + \log^2 m + m \log(nm))$ [BBDG05]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>outer: $O(nm \log(nm))$ or $O(n \log n \log(nm) + m)$ [BBDG05]</td>
</tr>
<tr>
<td></td>
<td>Translation and rotation</td>
<td>$O(n^4m^3 \log(nm))$ (this thesis)</td>
<td>$O(n^4m^3 \log(nm))$ (this thesis)</td>
<td>$O(n^4m^3 \log(nm))$ (this thesis)</td>
</tr>
<tr>
<td>Simple</td>
<td>Translation</td>
<td>$O(n^4m^3 \log(nm))$ (this thesis)</td>
<td>$O(n^4m^3 \log(nm))$ (this thesis)</td>
<td>$O(n^4m^3 \log(nm))$ (this thesis)</td>
</tr>
<tr>
<td></td>
<td>Translation and rotation</td>
<td>$O(n^4m^3 \log(nm)) \dagger$ (this thesis)</td>
<td>$O(n^4m^3 \log(nm)) \dagger$ (this thesis)</td>
<td>$O(n^4m^3 \log(nm)) \dagger$ (this thesis)</td>
</tr>
</tbody>
</table>

Table 1.3: Algorithms for the Containment problem for offsetting (running times)
Scharf [Sch04] introduced another algorithm that also runs in $O(n^2(n+m) \log(nm))$ time and $O(n(n+m))$ space.

[ESZ93]’s approach is based on parametric-searching technique. They provided a solution that runs in $O(nk\log^2 n)$ time and uses $O(nk)$ space, assuming that $m$ is constant.

* For convex polygons under translation and rotation, the Partial Containment problem for scaling was solved by Barequet and Scharf [BS05], who presented an algorithm that uses the translation-rotation-scale diagram, and runs in $O(n^4m^4 \log(nm))$ time and $O(n^3m^4)$ space.

- Offsetting

No results have ever been published for convex (all the more so for simple) polygons under translation (or translation and rotation) for the Partial Containment problem.

Tables 1.4 and 1.5 summarize the known results for variants of the Partial Containment problem for scaling and offsetting, respectively.

1.6 Our Results

In this thesis we address for the first time the Max-Cover, Containment, and Partial Containment problems for simple polygons under translation, rotation, and offset. We also address the variants of these problems for convex polygons, polygonal annuli, constrained annuli, polygons with holes, and sets of polygons.

We present the “two-point translation-rotation-offset” diagram that accumulates the information about all the placements of a given polygon $P$ under translation, rotation, and offset, which keep $P$ in contact with at least two points of a given set. We analyze the complexity of the diagram and present an algorithm for building it. Then, we develop an algorithm that solves the Containment and Partial Containment
### Table 1.4: Algorithms for the Partial Containment problem for scaling (running times)

<table>
<thead>
<tr>
<th>Type</th>
<th>Operation</th>
<th>Polygon</th>
<th>Annulus</th>
<th>Constrained annulus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex</td>
<td>Translation</td>
<td>$O(n^2(n + m) \log(nm))$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[BD00, Sch04]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$O(nk \log^2 n)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ESZ93] (assuming $m$ constant)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Translation</td>
<td>$O(n^4m^4 \log(nm))$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>and rotation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple</td>
<td>Translation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Translation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>and rotation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 1.5: Algorithms for the Partial Containment problem for offsetting (running times)

<table>
<thead>
<tr>
<th>Type</th>
<th>Operation</th>
<th>Polygon</th>
<th>Annulus</th>
<th>Constrained annulus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex</td>
<td>Translation</td>
<td>$O(n^3m^2(m + \log n))$ (this thesis)</td>
<td>$O(n^3m^2(m + \log n))$ (this thesis)</td>
<td>$O(n^3m^2(m + \log n))$ (this thesis)</td>
</tr>
<tr>
<td></td>
<td>Translation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>and rotation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple</td>
<td>Translation</td>
<td>$O(n^3m^3 \log(nm))$ (this thesis)</td>
<td>$O(n^3m^3 \log(nm))$ (this thesis)</td>
<td>$O(n^3m^3 \log(nm))$ (this thesis)</td>
</tr>
<tr>
<td></td>
<td>Translation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>and rotation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[\text{Table 1.4: Algorithms for the Partial Containment problem for scaling (running times)}\]

\[\text{Table 1.5: Algorithms for the Partial Containment problem for offsetting (running times)}\]

25
problems and their variants in $\Theta(n^4 m^3 \log(nm))$ time and $\Theta(n^2 m^2)$ space in the worst case by using this diagram.

Using a horizontal cross-section for a fixed offset of the diagram, we solve the Max-Cover problem for simple polygons and annuli under translation and rotation in $\Theta(n^3 m^3 \log(nm))$ time and $\Theta(nm)$ space in the worst case. For convex polygons we obtain $O(n^3 m^2 (m + \log(nm)))$ running time.

Using a vertical cross-section for a fixed rotation angle of the diagram, we solve the Containment and the Partial Containment problems for simple polygons under translation only in $O(n^3 m^3 \log(nm))$ time and $O(nm)$ space. For convex polygons we solve these problems in $O(n^3 m^2 (m + \log n))$ time and $\Theta(n + m)$ space. We consider the variants for polygonal annuli, constrained annuli, polygons with holes, and sets of polygons.

We also present another approach for solving the Containment and the Partial Containment problems, which is based on identifying and checking all possible “stable poses” of the polygon, and runs in $O(n^4 m^3 (n + m) \log m)$ time and $\Theta(n + m)$ space.

We show that the number of “stable poses” of a simple polygon is $\Theta(n^4 m^3)$ in the worst case, establishing a lower bound of $\Omega(n^4 m^3)$ on the running time of any algorithm that enumerates all possible placements of the polygon. Thus, we conclude that our diagram-based solution to Containment and Partial Containment problems is near optimal in this class of algorithms. We also derive a lower bound of $\Omega(n^3 m^3)$ on the running time of algorithms that solve the Max-Cover problem by traversing all possible placements of the polygon. Hence, our algorithm for the Max-Cover problem is also near optimal in this sense.
1.7 Thesis Outline

In Chapter 2 we introduce the two-point translation-rotation-offset diagram, describe its structure, and analyze its complexity. We also show how this diagram is used to solve the Containment and Partial Containment problems. We discuss a variant of the diagram that is used to solve the Max-Cover problem. In Chapter 3 we present a stable-poses-based algorithm for the Containment and Partial Containment problems. In Chapter 4 we establish tight bounds on the number of stable poses of a simple polygon. Chapter 5 presents our conclusions and possible directions for further work.
Chapter 2

The Translation-Rotation-Offset Diagram

2.1 Preface

We now present a solution to the Containment problem for simple polygons under translation, rotation, and linearized offset. At the end of this chapter we show how this solution can be modified to work also for true offset and polygonal annuli. We show that we can solve the Partial Containment problem by using the same algorithm. We can also modify our algorithm to solve the Max-Cover problem.

Dickerson and Scharstein [DS96] solve the Max-Cover problem using the fact that the optimum placement of $P$ contains at least one point of $S$ on its boundary. They
built a so-called “rotation diagram” that describes all the placements of $P$ in contact with one fixed point of $S$. We note that a more appropriate name for this diagram would be the “one-point translation-rotation” diagram. This diagram is two-dimensional: the two axes parameterize the translation and rotation.

Barequet and Dickerson [BD00] offered a “one-point translation-scale” diagram. This diagram is two-dimensional as well, but this time the axes parameterize the translation and scale of the polygon. As the previous diagram, it describes all the placements of $P$ (under the allowed transformations, i.e., translation and scale) that are in contact with one fixed point of $S$.

Barequet and Scharf [BS05] combined the two approaches and introduced a “one-point translation-rotation-scale” diagram. As its name suggests, this diagram has three dimensions and describes all the placements of $P$ under translation, rotation, and scale, that keep one fixed point from $S$ on the boundary of $P$.

In our approach we present a “two-point translation-rotation-offset” diagram, which accumulates the information on all the placements of $P$ under translation, rotation, and offset, which keep two fixed points from $S$ on the boundary of $P$. This diagram obviously has two dimensions. Note that this diagram describes less placements than its “one-point” counterpart: A one-point translation-rotation-offset diagram contains many two-point translation-rotation-offset diagrams as its cross-sections. However, as we noted at the beginning of this section, the “two-point” diagram suffices for solving our problem.

As we do not know which points are on the boundary of $O_{P,\delta}^b$ at the optimum placement, we need to build these diagrams for all points in $S$. Thus, $n$ such diagrams (one for each point in $S$) are built in [DS96, BD00, BS05]. In our algorithm, we build \( \binom{n}{2} = \Theta(n^2) \) diagrams, one for each pair of points from $S$. 

30
2.2 Structure of the Diagram

Let us fix two arbitrary points \( p_1, p_2 \in S \) and two arbitrary edges \( e_1, e_2 \in P \). Throughout this section we consider only placements of \( P \) that put \( p_1 \) on \( e_1 \) and \( p_2 \) on \( e_2 \). We build a diagram that accumulates information about all the placements of \( P \) under translation, rotation, and offset that keep \( p_1 \) on \( e_1 \) and \( p_2 \) on \( e_2 \). We denote this diagram by \( D(p_1,p_2),(e_1,e_2) \).

2.2.1 Containing region

For some point \( p_3 \in S \), our goal is to classify all the placements of \( P \) according to whether or not they contain \( p_3 \). We first allow only translation and rotation of \( P \), and will consider offset later. We may consider these placements as if \( P \) were static and the point set \( S \) moved. This way, if we allow \( p_1 \) and \( p_2 \) to slide along \( e_1 \) and \( e_2 \), respectively, then \( p_3 \) will draw some curve. This is a portion of the curve that \( p_3 \) draws when we slide the triangle \( \triangle p_1 p_2 p_3 \) while keeping \( p_1 \) on the supporting line of \( e_1 \) (denoted by \( \ell(e_1) \)) and \( p_2 \) on the supporting line of \( e_2 \) (\( \ell(e_2) \)). The problem of computing this curve is known as “Van Schooten’s locus problem” [Dör65]. It turns out that \( p_3 \) draws an ellipse. For completeness, we provide a sketch of the proof in Dörrie’s book [ibid., pp. 214–215].

The proof begins with the following observation: If we mark three points on a straight line and slide two of them along the rays of a right angle, then the third moves along an ellipse. This observation is justified by using elementary tools of analytic geometry.

The solution for Van Schooten’s problem proceeds as follows. Figure 2.1 shows a triangle \( \triangle p_1 p_2 p_3 \) sliding along two straight lines \( \ell_1 \) and \( \ell_2 \) while keeping \( p_1 \) and \( p_2 \) in contact with \( \ell_1 \) and \( \ell_2 \), respectively. We draw a circle (denoted by \( C \)) passing through the points \( p_1, p_2, \) and \( O \) (the intersection point of \( \ell_1 \) and \( \ell_2 \)), and denote its center.
by $M$. By applying the law of sines to the triangle $\triangle p_1Op_2$, we find that the radius of $C$ is independent of the position of the triangle $\triangle p_1p_2p_3$. We now draw a straight line passing through $p_3$ and $M$. This line intersects the circle at points $N$ and $L$. Let us consider the circle $C$ along with the points $N$ and $L$ as being firmly connected to the triangle, so that it also participates in the motion of the triangle. Consequently, $C$ passes continuously through $O$. Since the radius of the circle does not change during this process, the position of $M$ relative to $p_1$ and $p_2$ is not altered, and thus, the segment $\overrightarrow{MP_3}$ is also fixed relative to the triangle $\triangle p_1p_2p_3$. Therefore, the angle $\angle p_3Mp_1$ also remains constant. Consequently, the length of the arc $\widehat{p_1L}$ is constant. Similarly, the length of the arc $\widehat{p_2N}$ is constant. Hence, during the described motion, the positions but not the magnitudes of the arcs $\widehat{p_2N}$ and $\widehat{p_1L}$ change continuously.

This entails the invariance of the peripheral angles $\angle p_2ON$ and $\angle p_1OL$, which implies the invariance of the directions $\overrightarrow{ON}$ and $\overrightarrow{OL}$. Since $\overrightarrow{NL}$ is a diameter of the circle $C$, the lines $ON$ and $OL$ are perpendicular to each other. We can, therefore, consider the motion of the vertex $p_3$ as the motion of the marked point $p_3$ of a rigid line $NLP_3$, while its two other marked points, $N$ and $L$, slide along the rays $\overrightarrow{ON}$ and $\overrightarrow{OL}$ of a right angle. According to the initial observation, the motion of $p_3$ describes an ellipse, which completes the proof. $\square$
We now return to our point-containment problem. We know that \( p_3 \) draws an ellipse (which we denote by \( E_{p_3} \)) when we slide the triangle \( \triangle p_1 p_2 p_3 \) with \( p_1 \) on \( \ell(e_1) \) and \( p_2 \) on \( \ell(e_2) \); see Figure 2.2(a) for an illustration. The thick elliptic arc represents the portion of \( E_{p_3} \) that \( p_3 \) draws when we slide \( p_1 \) and \( p_2 \) along the edges \( e_1 \) and \( e_2 \). The portions of \( E_{p_3} \) that lie inside \( P \) correspond to placements of \( P \) that contain \( p_3 \). Every edge of \( P \) crosses \( E_{p_3} \) in no more than two points, thus, \( E_{p_3} \) is split by \( P \) into \( O(m) \) arcs; see Figure 2.2(b).

We parameterize the ellipse and denote it as \( E_{p_3}(t) \). The parameter \( t \) represents
the angle between the directions $\overrightarrow{NL}$ and $\overrightarrow{OL}$ (refer to Figure 2.1), so that its range is $[0, 2\pi)$. Every value of $t$ corresponds to some placement of $P$.

Recall that so far we allowed only the translation and rotation of $P$. Let us now examine what happens when we offset $P$. The ellipse $E_{p_3}$ is defined solely by the lines $\ell(e_1)$, $\ell(e_2)$ and by the triangle $\triangle p_1p_2p_3$. During the course of the offset, the points $p_1$, $p_2$, and $p_3$ do not change, but the lines $\ell(e_1)$ and $\ell(e_2)$ move parallel to themselves with constant, equal velocities. Thus, $E_{p_3}$ does not change its shape and only moves (without rotating) in the direction of one of the bisectors of $\ell(e_1)$ and $\ell(e_2)$, depending on the relative motion of the lines.

We now draw a graph that plots the intersection points of $P$ and $E_{p_3}$ as a function of the offset. We get a curve that consists of several simple curves; see the illustration in Figure 2.2(c). The horizontal axis is the parameterization of $E_{p_3}$, and the vertical axis is the offset. Note that since $t = 0$ and $t = 2\pi$ represent the same placement, the diagram wraps around. All the points on the curve correspond to placements with $p_3$ on the boundary of $P$, and all the points above the curve correspond to placements with $p_3$ inside $P$. Using the terminology of [DS96], we call the area above this curve the containing region of $p_3$ and denote it by $\text{CR}_{(p_1, p_2), (e_1, e_2)}(p_3)$. See, for example, Figure 2.2(c), where the containing region of the point $p_3$ is shown in gray.

We note that the containing region is not necessarily connected—it may consist of several disconnected parts. In addition, each part does not necessarily have to be monotone with respect to the horizontal axis. Furthermore, each part may be closed or infinite.

The vertices of the boundary of a containing region represent those placements in which $E_{p_3}$ passes through a vertex of $P$. The ellipse moves in some straight direction with a constant velocity during the offset process. Vertices of $P$ move along the arcs of the straight skeleton of $P$ with constant velocities. Since arcs of $\text{SS}(P)$ are straight line segments, a vertex moving along such an arc (during the offsetting) will meet
no more than twice. There are \( \Theta(m) \) skeleton arcs; thus, there are \( O(m) \) vertices on the boundary of a containing region, which split the boundary into \( O(m) \) simple curves.

We now analyze the shape of these simple curves. Each such curve describes the position of the intersection of the ellipse and an edge of \( P \); the coordinates of the intersection in this graph are the position on the ellipse as a function of offset. Our goal is to find its equation \( t(\delta) \), where \( \delta \) is the offset.

We fix the direction \( \overrightarrow{OL} \) (in Figure 2.1) as the positive \( x \) axis, and direction \( \overrightarrow{NO} \) (the inverse of \( \overrightarrow{ON} \)) as the positive \( y \) axis. Let \( e \) be an edge of \( P \). Consider the intersection points of \( \ell(e) \) and \( E_{p_3} \). During the offset process, \( \ell(e) \) moves but keeps its orientation. The ellipse \( E_{p_3} \) also moves in some direction with constant velocity. Hence, the movement of \( \ell(e) \) relative to \( E_{p_3} \) is also along some straight line with a constant velocity. The component of this velocity that interests us is the component perpendicular to \( \ell(e) \), which we denote by \( \overrightarrow{v_e} \). Assume that \( \ell(e) \) is given in a polar-type form \( \langle \rho, \varphi \rangle \), where \( \rho \) is the distance from \( \ell(e) \) to \( O \), and \( \varphi \) is the angle between the normal to \( \ell(e) \) towards the interior of \( P \) and the positive \( x \) axis. During the offset process, \( \ell(e) \) moves parallel to itself, thus \( \varphi \) does not change and \( \rho \) changes linearly as a function of \( \delta \). \( \rho(0) \) is the known distance from \( \ell(e) \) to \( O \) at offset 0. Thus, we have

\[
\rho(\delta) = \rho(0) + v_e \delta.
\]

The line \( \ell(e) \) is the locus of points \((x, y)\) that satisfy

\[
x \cos \varphi + y \sin \varphi = \rho(\delta),
\]

and \( E_{p_3} \) is given in the parametric form for \( t \in [0, 2\pi) \)

\[
\begin{pmatrix}
x \\
y
\end{pmatrix} = \begin{pmatrix}
a \cos t \\
b \sin t
\end{pmatrix},
\]

(2.1)
where $a = |p_3 N|$ and $b = |p_3 L|$ (refer again to Figure 2.1). The values $a$ and $b$ are functions of only the edges $e_1$, $e_2$ and the points $p_1$, $p_2$, $p_3$, and can be calculated as follows. The angle $\angle Mp_1 p_2$ is known, so we can infer $|Mp_3|$ by applying the law of cosines to $\triangle Mp_1 p_3$. Now, we can obtain the radius of $C$ by applying the law of sines to the triangle $\triangle Op_1 p_2$. Then, $a$ (resp., $b$) equals $|Mp_3|$ plus (resp., minus) the radius of $C$.

The intersection points $(x(t), y(t))$ thus satisfy
\begin{equation}
    a \cos t \cos \phi + b \sin t \sin \phi = \rho(\delta). \tag{2.2}
\end{equation}

Let us define (note that $t_0$ is not $t(0)$)
\begin{align*}
    d &= \sqrt{a^2 \cos^2 \phi + b^2 \sin^2 \phi}, \tag{2.3} \\
    t_0 &= \arctan \left( \frac{b \sin \phi}{a \cos \phi} \right). \tag{2.4}
\end{align*}

Then, dividing Equation 2.2 by Equation 2.3 and substituting Equation 2.4, we have
\begin{equation}
    \cos(t - t_0) = \frac{\rho(\delta)}{d},
\end{equation}

and finally,
\begin{equation}
    t(\delta) = t_0 + \arccos \left( \frac{\rho(0) + v_e \delta}{d} \right).
\end{equation}

In a more general form we have
\begin{equation}
    t(\delta) = c_1 + \arccos(c_2 + c_3 \delta), \tag{2.5}
\end{equation}

where $c_1$, $c_2$, $c_3$ are constants, which depend only on the edges $e_1$, $e_2$, $e$ and the points $p_1$, $p_2$, $p_3$.

Remark: Equation 2.5 describes a sine curve unless the argument of the arccos function has a constant value. In this case, the equation describes a vertical line.

We conclude by stating the following theorem:
**Theorem 1** The boundary of the \( CR_{(p_1,p_2),(e_1,e_2)}(p) \) for any \( p,p_1,p_2 \in S \) and for any \( e_1,e_2 \in P \) is made of \( \Theta(m) \) sine curves and vertical lines in the worst case.

We would like now to build the diagram of all the containing regions \( CR_{(p_1,p_2),(e_1,e_2)}(p) \) of points \( p \in S \) (for \( p \neq p_1,p_2 \)). However, we notice that distinct points of \( S \) have different respective ellipses. Thus, we need to figure out a parametrization \( E_p(t') \), such that every value of the parameter \( t' \) corresponds to the same placement for the ellipses for all the points. As the parameter we use the angle between the segment \( \overline{p_1p_2} \) and \( e_1 \), and denote it by \( \theta \); see Figure 2.3. We reparameterize the ellipses by substituting \( t(\theta) \) for \( t \).

![Figure 2.3: The parameters t and \( \theta \)](image)

We denote by \( \sigma \) the angle between \( \ell(e_1) \) and \( \ell(e_2) \). The arc \( \widehat{NL} \) equals \( \pi \). Thus, the sum of the arcs \( \widehat{LP_1}, \widehat{P_1O}, \) and \( \widehat{ON} \) also equals \( \pi \). The arc \( \widehat{LP_1} \) equals half of the angle \( \angle LMp_1 \), which can easily be calculated by applying the law of cosines to \( \triangle p_3Mp_1 \). The arc \( \widehat{P_1O} \) equals \( \angle p_1p_2O \), which is \( (\pi - (\sigma + \theta)) \). Finally, the arc \( \widehat{ON} \) equals \( t \).

Thus, we get:

\[
t = \frac{\pi}{2} - \frac{\angle LMp_1}{2} - (\pi - (\sigma + \theta)),
\]

that is,

\[
t = \theta + (\sigma - \frac{\pi}{2} - \frac{\angle LMp_1}{2}).
\]
Figure 2.4(a) shows a sample polygon and a set of points. The diagram in Figure 2.4(b) shows the containing regions CR\((p_1,p_2),(e_1,e_2)\)(\(p_i\)) for \(3 \leq i \leq 5\). In Figure 2.4(c), some placement of \(P\) corresponds to the point marked on the diagram in Figure 2.4(b). The points \(p_1\) and \(p_2\) are on the edges \(e_1\) and \(e_2\), respectively. The point \(p_3\) is inside the polygon \(P\), while the points \(p_4\) and \(p_5\) are outside \(P\). The polygon appears smaller than the original \(P\) since it is an inner offset version of \(P\).

### 2.2.2 Obtainable region

So far we have built a diagram of containing regions of points from \(S\); refer to Figure 2.4(b). Every point inside the containing region in the two-dimensional diagram corresponds to a placement that puts \(p_1\) on \(\ell(e_1)\) and \(p_2\) on \(\ell(e_2)\). However, only a portion of these points represents placements that put \(p_1\) on \(e_1\) and \(p_2\) on \(e_2\); see, for example, Figure 2.2(a). If we slide \(p_1\) on \(e_1\) and \(p_2\) on \(e_2\), the point \(p_3\) will draw only the arc shown in the figure with a thick line. Thus, we actually need only a portion of the entire diagram. We call this part an **obtainable region** and denote it by \(\text{OR}_{(p_1,p_2),(e_1,e_2)}\). Recall that the parameterization that we chose is based on the angle \(\theta\) between \(p_1p_2\) and \(e_1\). (Note that the obtainable region does not depend on a choice of a point \(p_3\).)

Let us examine what the obtainable region for a fixed offset is. In other words, let us fix an offset and ask for which values of \(\theta\) the points \(p_1\) and \(p_2\) lie on \(e_1\) and \(e_2\), respectively. We first check for which values of \(\theta\) only one of the two conditions holds, that is, \(p_1\) lies on \(e_1\). Refer to Figure 2.5(a). We denote by \(\alpha\) the angle between \(\ell(e_1)\) and the normal to \(\ell(e_2)\). \(\alpha\) equals \(|\frac{\pi}{2} - \sigma|\). Now, we draw a circle of radius \(|p_1p_2|\) centered at the endpoint of \(e_1\) that is closer to the intersection point of \(\ell(e_1)\) and \(\ell(e_2)\). (This is the left endpoint in Figure 2.5(a).) Assume that the circle intersects \(\ell(e_2)\). Then, it intersects \(\ell(e_2)\) at angles \(\alpha_{11}\) and \(\alpha_{12}\) (for \(\alpha_{11} \leq \alpha_{12}\)). Note that \(\alpha - \alpha_{11} = \alpha_{12} - \alpha\). We repeat this operation for the right endpoint of \(e_1\) and denote
(a) A sample polygon and a point set

(b) The containing regions of the points $p_3$, $p_4$, and $p_5$

(c) A polygon placement

Figure 2.4: A sample diagram of containing regions
the respective angles by $\alpha_{21}$ and $\alpha_{22}$ (for $\alpha_{21} \leq \alpha_{22}$). Here, a similar equality holds: $\alpha - \alpha_{21} = \alpha_{22} - \alpha$. Figure 2.5(b) shows the possible ranges of these angles. Note that, in the example of Figure 2.5, all the $\alpha$ angles are in the range $[0, \pi)$. However, this is not necessarily true in the general case. For instance, if the left endpoint of $e_1$ coincides with the intersection point of $\ell(e_1)$ and $\ell(e_2)$ (in the same example), then $\alpha_{11}$ is $-(\pi/2 - \alpha)$, which equals $(3\pi/2 + \alpha)$ (if we normalize the angle to be greater than zero), that is, it is greater than $\pi$.

Imagine that we place the segment $\overline{p_1p_2}$, such that $p_1$ identifies with the left endpoint of $e_1$ and the angle between $\overline{p_1p_2}$ and $\ell(e_1)$ is $\alpha_{11}$. Now, if we slide $\overline{p_1p_2}$ to the right, the angle it forms with $\ell(e_1)$ changes continuously from $\alpha_{11}$ to $\alpha_{21}$. At the angle $\alpha_{21}$, the segment reaches the right endpoint of $e_1$ and cannot slide further to the right. We repeat this process, this time starting at the left endpoint and with the angle $\alpha_{12}$. Once again, when $\overline{p_1p_2}$ reaches the right endpoint, it stops, this time at the angle $\alpha_{22}$. Thus, the range of angles at which $p_1$ lies on $e_1$ is $[\alpha_{11}, \alpha_{21}] \cup [\alpha_{22}, \alpha_{12}]$.  

![Diagram of angles](image)

(a) Definition of $\alpha$ angles

(b) Range of $\alpha$ angles

Figure 2.5: The angles $\alpha_{ij}$
These ranges are marked by thick lines in Figure 2.5(b).

Let us now consider a few special cases:

1. The left circle does not intersect $\ell(e_2)$ at all. In this case the obtainable range is empty.

2. Only the left circle intersects $\ell(e_2)$. In this case the obtainable range is $[\alpha_{11}, \alpha_{12}]$.

3. The line $\ell(e_2)$ is tangent to the left circle. In this case $\alpha_{11} = \alpha_{12} = \alpha$. The obtainable range is a single angle $\alpha$.

Remark: As we know, an edge of $P$ may be split into several pieces during the offset process. Assume, for example, that at a given offset, an edge $e_1$ consists of $r$ disjoint pieces. Thus, we get $r$ quadruples of $\alpha$ angles, a quadruple for each piece. The above discussion still holds for each piece, and the obtainable range in this case is the union of $r$ obtainable ranges corresponding to the $r$ pieces of $e_1$.

We now turn to the second condition (alone): for which values of $\theta$ does $p_2$ lie on $e_2$. This condition is very similar to the previous one. We draw two circles of radius $|p_1p_2|$ centered at the endpoints of $e_2$. They intersect $\ell(e_1)$ at angles $\beta_{11}, \beta_{12}, \beta_{21},$ and $\beta_{22}$; see Figure 2.6(a). The only difference is that the angles $\beta$ are symmetric about $\pi/2$ (while the angles $\alpha$ are symmetric about $\alpha$); see Figure 2.6(b). Similarly to the previous condition, we conclude that the obtainable range is $[\beta_{12}, \beta_{22}] \cup [\beta_{21}, \beta_{11}]$. The special cases are also identical.

We now conclude by stating for which values of $\theta$ both conditions hold, that is, $p_1$ lies on $e_1$ and $p_2$ lies on $e_2$. This happens exactly in the intersection of the two ranges: $([\alpha_{11}, \alpha_{21}] \cup [\alpha_{22}, \alpha_{12}]) \cap ([\beta_{12}, \beta_{22}] \cup [\beta_{21}, \beta_{11}])$; see Figure 2.7.

Until now we discussed what the obtainable region for a fixed offset is. Now we consider the evolution of the angles $\alpha_{ij}$ and $\beta_{ij}$ as we change the offset. During this operation, each one of these angles draws a curve in our diagram. We investigate the structure of the curves of the $\alpha$ angles; the structure of the $\beta$ curves is similar.
(a) Definition of $\beta$ angles

(b) Range of $\beta$ angles

Figure 2.6: The angles $\beta_{ij}$

Figure 2.7: The obtainable region is the intersection of two unions of pairs of ranges
We start from the minimal offset—at which the polygon $P$ shrinks to a point (or several points and/or segments). As $P$ is offset outward, it grows continuously. During the offset process, edges of $P$ are created, change their lengths, and possibly disappear. Refer to a specific pair of edges $e_1$ and $e_2$, as in the above discussion. The $\alpha$ angles do not exist until the moment at which $e_1$ is created. (Similarly, the $\beta$ angles do not exist until $e_2$ is created.) At this moment, the $\alpha$ curves are created. As the length of $e_1$ changes, the $\alpha$ angles change as well—see the exact equations below. In addition, there are offsets at which these $\alpha$ curves change topologically. This occurs in either of the following cases:

1. $P$ undergoes a topological change that involves $e_1$. We call this a “pure topological” event. This happens in either of the following situations:

   - $e_1$ is created. Prior to this point there are no $\alpha$ curves. At this point if the circle defined by $e_1$ (initially a single point) crosses $\ell(e_2)$, then the $\alpha$ curves are created.

   - A neighboring edge of $e_1$ appears or disappears. From this moment, the vertex it shares with $e_1$ moves with a different velocity (and possibly in a different direction along $\ell(e_1)$). This is reflected by changes in the equations of the respective pair of $\alpha$ curves.

   - $e_1$ (or a piece of it in case $e_1$ consists of several disjoint pieces) is split into two pieces by a vertex that crashes into it. In the diagram, two new pairs of $\alpha$ curves, which correspond to the two newly created endpoints of pieces of $e_1$, are created.

   - $e_1$ consists of several disjoint pieces, and two of them unite. This event is the opposite of the previous event. In the diagram, the two pairs of curves, corresponding to the two edge endpoints that crashed one into another, meet at one point and disappear.
- $e_1$ vanishes when its endpoints meet. The $\alpha$ curves meet at one point and disappear.

2. The right circle becomes tangent to $\ell(e_2)$. If prior to this event the circle and $\ell(e_2)$ intersected, then they cease to intersect, and so the curves of $\alpha_{21}$ and $\alpha_{22}$ meet at the angle $\alpha$ and disappear. (Recall that $\alpha$ was defined as the angle between $\ell(e_1)$ and the normal to $\ell(e_2)$. Refer to Figure 2.5(a).) On the other hand, if the circle and $\ell(e_2)$ did not intersect prior to the event, they will now start to intersect, and so the curves of $\alpha_{21}$ and $\alpha_{22}$ are now created at the angle $\alpha$.

3. The left circle becomes tangent to $\ell(e_2)$. This means that the right circle does not intersect $\ell(e_2)$, thus, the curves of $\alpha_{21}$ and $\alpha_{22}$ do not exist prior to the event. If the left circle and $\ell(e_2)$ already intersected, they now cease to intersect and so the curves of $\alpha_{11}$ and $\alpha_{12}$ meet at the angle $\alpha$ and disappear. Otherwise, the curves of $\alpha_{11}$ and $\alpha_{12}$ are created at the angle $\alpha$.

We call the last two event types “pseudo topological.”

It remains to derive the equations of the simple curves that comprise the $\alpha$ and $\beta$ curves.

We begin by expressing the distance $s$ from one of the vertices of $e_1$ to the intersection point of the two lines as a function of $\delta$. We denote by $\eta$ the respective angle of $P$ at this vertex; see Figure 2.8 for an illustration.

There are three similar cases:

- The lines move away from each other (relative to the interior of the polygon). See Figure 2.8(a) for an illustration. In this case

\[ s = s_0 + \delta(\cot \frac{\sigma}{2} \pm \cot \frac{\eta}{2}), \]

where $s_0$ is the distance at the offset in which $e_1$ is created. The sign of $\cot \frac{\eta}{2}$ depends on which vertex of $e_1$ is considered.
The lines move toward each other (relative to the interior of the polygon). This case is very similar to the previous one. Here we have

\[ s = s_0 + \delta(-\cot \frac{\sigma}{2} \pm \cot \frac{\eta}{2}). \]

- The lines are chasing each other. See Figure 2.8(b) for an illustration. In this case we have

\[ s = s_0 + \delta(\pm \tan \frac{\sigma}{2} \pm \cot \frac{\eta}{2}). \]

The sign of \(\tan(\sigma/2)\) depends on the direction of the movement.

To conclude, the distance from the endpoint of \(e_1\) to the intersection point of the two lines is

\[ s = s_0 + c\delta, \quad (2.6) \]

where \(c\) and \(s_0\) are constants that change only at topological changes of \(P\) that involve this endpoint of \(e_1\).

Figure 2.9 illustrates the situation for the angles \(\alpha_{11}, \alpha_{12}\).

From the law of sines we have

\[ \frac{s}{\sin(\pi - \sigma - \alpha_{11})} = \frac{|p_1p_2|}{\sin \sigma}. \]

45
Thus,
\[ \alpha_{11} = \arcsin \left( \frac{s}{|p_1p_2|} \sin \sigma \right) - \sigma. \] (2.7)

Substituting Equation 2.6 in Equation 2.7, we obtain
\[ \alpha_{11} = \arcsin \left( \frac{s_0 + c\delta}{|p_1p_2|} \sin \sigma \right) - \sigma. \] (2.8)

The same equation holds for \( \alpha_{12} \). That is, \( \alpha_{11} \) and \( \alpha_{12} \) are the angles in the range \([0, 2\pi)\) that satisfy Equation 2.8.

The equations of \( \alpha_{21} \), \( \alpha_{22} \), and the \( \beta \) angles are similar, and their general form is \( \arcsin(c_1 \delta + c_2) + c_3 \), where \( c_1, c_2, \) and \( c_3 \) are constants that depend solely on the edges \( e_1, e_2 \) and the points \( p_1, p_2 \).

Figure 2.10 shows the obtainable region \( OR_{(p_1,p_2),(e_1,e_2)} \) for the polygon and point set shown in Figure 2.4(a). Also, \( \alpha \) (solid) and \( \beta \) (dotted) curves are shown.

We now analyze the complexity of an obtainable region. To simplify the analysis, we build separate diagrams for each pair of pieces, one of \( e_1 \) and one of \( e_2 \), instead of building a diagram for two edges \( e_1 \) and \( e_2 \). That is, each time an edge is split during the offsetting, we continue following one piece in the original diagram, and “start” a new diagram for the other piece. Each time two pieces of an edge unite, we follow the united part in only one of the diagrams.
The resulting diagrams together contain the same information as the one diagram for the entire edges $e_1$ and $e_2$. Note that the number of all the pieces of edges created during the offsetting is $\Theta(m)$, which is asymptotically the number of edges. Hence, we shall see later that building diagrams for pieces of edges instead of for entire edges will not affect the running time of our algorithm.

**Theorem 2** The boundary of an obtainable region $OR(p_1,p_2, (e_1,e_2))$, where $e_1$ and $e_2$ are two pieces of edges, is made of $O(m)$ sine curves and vertical lines.

**Proof** There are $\Theta(m)$ topological changes of $P$ during the entire offset process. Thus, for a fixed piece of edge $e_1$, there are $O(m)$ ($\Theta(m)$ in the worst case) pure topological events. Note that between two topological events, each endpoint of the piece of $e_1$ moves with a constant velocity along $\ell(e_1)$ relatively to the intersection point of $\ell(e_1)$ and $\ell(e_2)$. Hence, each angle $\alpha$ increases or decreases monotonically. Therefore, there is no more than one pseudo topological event for any circle between two consecutive pure topological events. We conclude that there are in total $O(m)$ events for this diagram, which means that all the $\alpha$ curves together consist of $O(m)$ simple curves. The same is true for the $\beta$ curves.

We saw that between any two consecutive topological changes, the $\alpha$ and $\beta$ curve equations are of the form $\arcsin(c_1\delta + c_2) + c_3$ for some constants $c_1, c_2, c_3$. Moreover,
the α and β angles are in the range [0, 2π). Thus, between any two consecutive
topological changes one α curve and one β curve intersect at most twice. Hence, the
total number of intersections in one diagram is \(O(m)\).

Remark: Unless \(c_1\) is 0, the equation describes a sine curve. Otherwise it describes
a vertical line. □

### 2.2.3 Diagram complexity

We now know the structure of the diagram \(D_{(p_1, p_2), (e_1, e_2)}\): it has \(n - 2\) containing
regions, and one obtainable region. See Figure 2.11(a) for an illustration. As already
noted before, we are interested only in the part of the diagram inside the obtainable
region. Thus, we analyze the complexity of this part only.

The complexity of the part of \(D_{(p_1, p_2), (e_1, e_2)}\) inside the obtainable region \(OR_{(p_1, p_2), (e_1, e_2)}\)
is the sum of the following terms:

- **Containing regions complexity.** As argued in the previous section (Theorem 1),
the complexity of one containing region is \(O(m)\). Hence, the complexity of \(n - 2\)
containing regions is \(O(nm)\).

- **Obtainable region complexity.** Theorem 2 states that the complexity of the
obtainable region is \(O(m)\).

- **The number of intersection points between the boundaries of various regions.**
Recall that the boundary of each region is made of \(O(m)\) sine curves of the
same frequency (refer to Equations 2.5 and 2.8) and vertical lines. Moreover,
for the sine curves, the diagram includes at most one period. Thus, each pair of
“simple” curves of two region boundaries intersect at most twice. As a result,
the overall number of intersection points between the boundaries of various
regions is \(O(n^2m^2)\).
Hence, the complexity of a two-point translation-rotation-offset diagram is $O(n^2 m^2)$. Corollary 27 in Chapter 4 shows a matching worst-case lower bound.

Thus, we have:

**Theorem 3** The complexity of a two-point translation-rotation-offset diagram $D(p_1, p_2, (e_1, e_2))$ is $\Theta(n^2 m^2)$ in the worst case.

To solve the Containment Problem, we build a translation-rotation-offset diagrams for each pair of points in $S$ and for each pair of pieces of edges of $P$. The total number of pieces of edges that are created during the offset process is $\Theta(m)$ as was stated in Section 1.3.1. Thus we build $\Theta(n^2 m^2)$ diagrams. Together, they describe all the placements of $P$ in contact with at least two points from $S$.

We analyze the total complexity of all these $\Theta(n^2 m^2)$ diagrams. A simple upper bound would be $O(n^2 m^2 \cdot n^2 m^2) = O(n^4 m^4)$. In fact, we can achieve a tighter upper bound by a more detailed analysis. The total complexity of all the diagrams is the sum of:

- The complexity of all the containing regions. This equals $O(nm \cdot n^2 m^2) = O(n^3 m^3)$.

- The complexity of all the obtainable regions. This equals $O(m \cdot n^2 m^2) = O(n^2 m^3)$.

- The overall number of intersection points of the boundaries of the obtainable and containing regions. Each such intersection point corresponds to a placement of $P$ under translation, rotation, and offset that puts three points of $S$ on the boundary of $P$ and at least one of these points on a vertex of $P$. However, every such placement corresponds to a vertex on a boundary of some containing region in one of the diagrams.

For instance, an intersection point in the diagram $D(p_1, p_2, (e_1, e_2))$ between the
boundary of OR\((p_1,p_2),(e_1,e_2)\) and the boundary of CR\((p_1,p_2),(e_1,e_2)\)(\(p_3\)) corresponds to a placement \(\tau\) of \(P\), at which (without loss of generality) \(p_1\) lies on a vertex of \(e_1\), \(p_2\) lies on \(e_2\), and \(p_3\) lies on some edge \(e_3\). Hence, in the diagram \(D_{(p_2,p_3),(e_2,e_3)}\), the point that corresponds to \(\tau\) will be on a vertex of \(CR_{(p_2,p_3),(e_2,e_3)}(p_1)\).

Thus, the number of these placements is bounded by the overall complexity of all the containing regions in all the diagrams, which is \(O(n^3m^3)\).

- The overall number of intersection points of the boundaries of the containing regions. Each such intersection point corresponds to a placement of \(P\) under translation, rotation, and offset that puts four points of \(S\) on the boundary of \(P\). In Chapter 4, Theorem 26 we show that the number of such placements is \(O(n^4m^3)\), which is tight in the worst case.

We conclude by the following theorem:

**Theorem 4** The total complexity of all the two-point translation-rotation-offset diagrams for a simple \(m\)-gon and a set of \(n\) points is \(\Theta(n^4m^3)\) in the worst case.

In the next section we describe how to build the translation-rotation-offset diagram for a pair of points of \(S\) and a pair of edges of \(P\).

### 2.3 Building the Diagram

We first build the straight skeleton of the given polygon by the algorithm of [FO98] mentioned in the introduction (Section 1.3.1). This algorithm runs in \(\Theta(m^2)\) time and uses \(\Theta(m)\) space. Its time and space complexities are much smaller than those of the algorithm that builds the diagrams.
2.3.1 Containing regions

We build \( \text{CR}_{(p_1,p_2),(e_1,e_2)}(p) \) for each point \( p \in S \) \((p \neq p_1, p_2)\) by using a plane-sweep procedure. The sweep is along the offset axis, that is, we sweep the diagram by a horizontal line that moves from the bottom position \((\delta = \delta_{\text{min}})\) upward.

The events are the offsets at which

1. \( E_p \) meets a vertex of \( P \); or
2. \( E_p \) becomes tangent to an edge of \( P \).

(Recall that \( E_p \) is the ellipse drawn by the point \( p \).)

We can precompute all the events of both types before running the algorithm.

For the first event type, consider a vertex \( u \in P \). During the offsetting it moves along an arc of the straight skeleton. Its motion relative to \( E_p \) is along a straight line with a constant velocity \( v = (v_x, v_y) \). \( E_p \) is given in a parametric form as in Equation 2.1. Assume that \( u = (x, y) \) is given (as a function of \( \delta \)) by

\[
\begin{align*}
  x(\delta) &= x_0 + v_x \delta, \\
  y(\delta) &= y_0 + v_y \delta,
\end{align*}
\]  

(2.9)

where \((x_0, y_0)\) are the original coordinates of \( u \) at the offset 0. The intersection between \( E_p \) and \( u(\delta) \) occurs when

\[
\frac{x^2(\delta)}{a^2} + \frac{y^2(\delta)}{b^2} = 1.
\]  

(2.10)

By substituting Equations 2.9 in Equation 2.10 we obtain

\[
\frac{(x_0 + v_x \delta)^2}{a^2} + \frac{(y_0 + v_y \delta)^2}{b^2} = 1.
\]  

(2.11)

This is a quadratic equation in the parameter \( \delta \), which we can solve analytically.

For the second event type, we find when \( E_p \) becomes tangent to a line supporting an edge \( e \) of \( P \). As in Section 2.2.1, we consider the motion of \( \ell(e) \) relative to \( E_p \), which
is also along some straight line with a constant velocity. We denote the component of this velocity perpendicular to \( \ell(e) \) by \( v_e \). As before, \( \ell(e) \) is given in a polar-type form \( \langle \rho, \varphi \rangle \), where \( \varphi \) is constant and \( \rho(\delta) = \rho_0 + v_e \delta \). As before, \( E_p \) is given in the parametric form.

The ellipse becomes tangent to the line \( \ell(e) \) at offset \( \delta \), in which we have

\[
\rho(\delta) = \pm \sqrt{a^2 \cos^2 \varphi + b^2 \sin^2 \varphi},
\]

that is

\[
\delta = \pm \frac{\sqrt{a^2 \cos^2 \varphi + b^2 \sin^2 \varphi}}{v_e} - \rho_0.
\]

There are \( O(m) \) such events as was already explained in the previous section. Using a straightforward implementation of a plane sweep, we obtain the running time \( O(m \log m) \). Building \( n - 2 \) containing regions thus takes \( O(nm \log m) \) time.

### 2.3.2 Obtainable region

We use the same plane sweep technique for building the obtainable region. The events are

1. Topological change of an \( \alpha \) or \( \beta \) curve (this is described thoroughly in Section 2.2.2); and

2. An intersection of \( \alpha \) and \( \beta \) curves. We can easily find offsets in which the intersections occurs, since we know the equations of the curves.

As we saw earlier, there are \( O(m) \) events in total. Each event requires \( O(\log m) \) time. Thus, building one obtainable region takes \( O(m \log m) \) time.

### 2.3.3 Building the entire diagram

We compute only the part of the diagram that is \textit{inside} the obtainable region. Thus, after the regions have been computed, we clip each containing region by the obtainable
region and build the diagram only in the interior of the latter. This computation is done, once again, by using a plane-sweep procedure. The number of events during this procedure is \(O(n^3m^3)\) for all the diagrams, as we have already shown in Section 2.2.3.

Then we build the parts of the diagrams that are inside the corresponding obtainable regions. Again, we use a plane-sweep procedure. The events are the vertices of the obtainable and containing regions, as well as the intersection points of the boundaries of these regions. We process each such event in \(O(\log(nm))\) time.

The overall number of events for building all \(\Theta(n^2m^2)\) diagrams equals the complexity of the diagrams which is shown in Theorem 4 to be \(\Theta(n^4m^3)\) in the worst case.

Thus, we conclude with

**Theorem 5** The total complexity of the algorithm for building all the two-point translation-rotation-offset diagrams for a simple \(m\)-gon and a set of \(n\) points is \(\Theta(n^4m^3 \log(nm))\) in the worst case.

### 2.4 Applications of the Diagram

#### 2.4.1 Containment and Partial Containment problems

For every location in the diagram, its depth plus 2 is the number of points of \(S\) covered by \(P\) in the corresponding placement. Thus, once we have all the diagrams, we can easily solve the Containment problem by just looking for the lowest point with depth \(n - 2\). Note that this can be done while building the diagrams without paying extra time. Also note that we can build one diagram at a time, find the lowest point in it, and compare to the minimum computed so far.

Figure 2.11(a) shows the diagram \(D_{(p_1,p_2),(e_1,e_2)}\). The marked point inside the diagram is the lowest point with depth 3 that is inside the obtainable region. The
placement of the offset version of $P$ that corresponds to this point in the diagram is shown in Figure 2.11(b). In this placement $p_1$ lies on $e_1$, $p_2$ lies on $e_2$, and the polygon covers all the points.

Similarly, given the translation-rotation-offset diagrams, we can solve the Partial Containment problem (of $k$ points) by looking for the lowest point with depth $k - 2$. 

Figure 2.11: The diagram and the optimum placement
We summarize our results so far by the following theorem:

**Theorem 6** One can solve the Containment and the Partial Containment problems for simple polygons under translation and rotation in $\Theta(n^4m^3\log(nm))$ time and $\Theta(n^2m^2)$ space in the worst case.

### 2.4.2 Max-Cover problem

Given the entire diagram, we can also solve the Max-Cover problem for any given offset. For a given offset $\delta$, if there is a placement of $O_{P,\delta}^k$ that covers $k$ points from $S$, then we can translate it until its boundary hits one of the covered points, and then rotate it about this point until it meets another point. The number of covered points does not change and we have a placement with two points on the boundary. Thus, this placement is in our diagram.

To find a placement that contains a maximum number of points, we find a point with maximal depth in all the diagrams at offset $\delta$. However, we should note that by using this approach we will not find all the placements covering $k$ points, but only those that put at least two points of $S$ on the boundary of $P$. Thus, we shall find only these kind of placements of $O_{P,\delta}^k$ that cover $k$ points, and shall not find placements that cover $k$ points but have only one point (or no points at all) on the boundary of $O_{P,\delta}^k$.

Using our algorithm to solve the Max-Cover problem is an overkill. For this problem we only need the cross-sections of our diagrams that correspond to the given offset $\delta$. See Figure 2.12 for an illustration. That is, we need to build only one-dimensional two-point translation-rotation diagrams.

Let us analyze the complexity of this approach. An ellipse $E_p$ can cross the polygon in $\Theta(m)$ points in the worst case. Thus, the complexity of the cross-section of the containing region is $Theta(m)$ in the worst case. Recall that we build a diagram for each pair of pieces of edges. The complexity of the cross-section of the obtain-
Figure 2.12: A horizontal cross-section of a translation-rotation-offset diagram is a translation-rotation diagram.

The obtainable region is $\Theta(1)$ since it is merely the intersection of $\alpha$ and $\beta$ ranges for a given offset (refer to Figure 2.7). As a result, the complexity of the entire one-dimensional translation-rotation diagram for a fixed pair of points of $S$ is $O(nm)$.

We can build the translation-rotation diagram in $O(nm \log(nm))$ time: First, we build all the one-dimensional containing regions by intersecting the appropriate ellipses with the polygon $O^{L}_{P,\delta}$. This procedure takes $O(m \log m)$ time per containing region. Then, we construct the one-dimensional obtainable region by simply computing the $\alpha$ and $\beta$ regions and intersecting them. This is done in $\Theta(1)$ time. Finally, we scan the regions to obtain the translation-rotation diagram. Since the complexity of the diagram is $O(nm)$, this phase takes $O(nm \log(nm))$ time. We can find the deepest point in the diagram while we build it.

To find the optimum placement we need to find the deepest point in the diagrams for every pair of edge pieces. The complexity of all the diagrams is $O(nm \cdot n^2m^2) = O(n^3m^3)$. Note that the total complexity of all the one-dimensional translation-rotation diagrams is proportional to the number of placements of $O^{L}_{P,\delta}$ that put three points of $S$ on the boundary of $P$. Corollary 25 (see Chapter 4) states that the number of such placements is $\Theta(n^3m^3)$ in the worst case. It follows easily from this corollary that the complexity of the one-dimensional translation-rotation diagram for
a fixed pair of points of $S$ is $\Theta(nm)$ in the worst case.

Thus, we conclude:

**Theorem 7** One can solve the Max-Cover problem for simple polygons under translation and rotation in $\Theta(n^3m^3\log(nm))$ time and $\Theta(nm)$ space in the worst case.

### 2.5 Extensions

#### 2.5.1 Translation only

We address the Containment problem for simple polygons under translation only. Our main observation still holds for this case. That is, at the optimum placement, there are at least two points on the polygon’s boundary.

Let us fix two arbitrary points $p_1, p_2 \in S$ and two arbitrary edges $e_1, e_2 \in P$. And let us denote by $\theta_0$ the angle between $e_1$ and $p_1p_2$. This angle is constant for all placements of $P$ under translation and offset (since $e_1$ moves parallel to itself during the offset process). Thus, the vertical cross-section for $\theta = \theta_0$ of the diagram $D_{(p_1,p_2),(e_1,e_2)}$ represents all the placements of $P$ under translation and offset that keep $p_1$ on $e_1$ and $p_2$ on $e_2$. See Figure 2.13 for an illustration.

![Figure 2.13](image)

Figure 2.13: A vertical cross-section of a translation-rotation-offset diagram is a translation-offset diagram
Thus, we can solve the Containment problem by building the appropriate vertical cross-sections of all the diagrams (these should be called two-point translation-offset diagrams) and looking for a lowest point with depth $n - 2$. Note that $\theta_0$ is different for every diagram.

It is left to estimate the complexity of this approach. Recall that a containing region is not necessarily monotone with respect to the horizontal axis, and its boundary is made of $O(m)$ sine curves and vertical lines (refer to Theorem 1). Hence, the complexity of a vertical cross-section of a containing region is $O(m)$. Similarly, the complexity of a vertical cross-section of an obtainable region is also $O(m)$. Thus, the complexity of one translation-offset diagram is $O(nm)$ (since it has $n - 2$ containing regions). We can easily build this diagram in $O(nm \log(nm))$ time. Thus, building all the diagrams and solving the Containment problem takes $O(n^3 m^3 \log(nm))$ time.

We can similarly solve the Partial Containment problem by looking for a lowest point with depth $k - 2$.

**Theorem 8** One can solve the Containment and the Partial Containment problems for simple polygons under translation in $O(n^3 m^3 \log(nm))$ time $O(nm)$ space.

### 2.5.2 Special types of polygons

**Convex polygons**

Obviously, both our algorithms (recall Sections 2.4.1 and 2.4.2) will work for convex polygons. However, in this case the above analysis is probably not tight.

For the translation-rotation-offset diagrams, the complexity of one diagram is proportional to the number of intersections of containing regions. As already noted above, each such intersection corresponds to a placement of $O_{P,\delta}^L$ under translation, rotation, and offset with four points from $S$ on the boundary. For simple (nonconvex) polygons, the overall number of such placements is $\Theta(n^4 m^3)$ in the worst case. We
suspect a lower number for convex polygons. To the best of our knowledge, the
tightest lower bound is $\Omega(n^4m)$ [BS05]. Barequet and Scharf show an example of
a regular polygon achieving this lower bound for scaling. This example is valid for
offsets too since for regular polygons, scale and linearized offset are topologically
identical. The best upper bound is $O(n^4m^3)$, which is merely our result for simple
polygons from Theorem 26.

For the one-dimensional translation-rotation diagram $D_{(p_1,p_2),(e_1,e_2)}$, its complexity
is proportional to the number of placements of $O_{P,\delta}^L$ under translation and rotation,
that put $p_1$, $p_2$, and some other point from $S$ on $e_1$, $e_2$, and some other edge of $P$,
respectively. The overall complexity of all the translation-rotation diagrams for a
given polygon and offset $\delta$ is proportional to the number of placements of $O_{P,\delta}^L$ under
translation and rotation, that put some three points from $S$ on the boundary of $P$. We
show in Chapter 4 (Corollary 25) that this number is $\Theta(n^3m^3)$ for nonconvex polygons
in the worst case. However, no tight bound for convex polygons is known. (Recall that
an offset of a convex polygon is always convex.) One can easily construct an example
in which there are $\Theta(n^3m)$ such placements. The best known upper bound [DS96] is
$O(n^3m^2)$. As a result, the overall complexity of all the translation-rotation diagrams
for a given convex polygon is $O(n^3m^2)$.

Constructing each one-dimensional containing region requires $O(m)$ time since
we have to intersect some ellipse with the convex polygon. Since we build $\Theta(n^2m^2)$
diagrams, and each diagram has $\Theta(n)$ containing regions, building all the containing
regions takes $O(n^3m^3)$ time. The overall complexity of all the diagrams is $O(n^3m^2)$.
Therefore, building all the diagrams needs $O(n^3m^3 + n^3m^2 \log(nm)) = O(n^3m^2(m +
\log(nm)))$ time.

In conclusion:

**Theorem 9** One can solve the Max-Cover problem for convex polygons under trans-
lation and rotation in $O(n^3m^2(m + \log(nm)))$ time and $O(nm)$ space.
For the Containment and Partial Containment problems for convex polygons under translation only, we use the same translation-offset diagram-based algorithm as for the simple-polygon case. However, we get better time and space complexity. We present the analysis below.

It is easy to verify that for a convex polygon, each two-dimensional containing region in a translation-rotation-offset diagram is monotone with respect to the horizontal axis. Let us look at a containing region of a point $p \in S$ in a diagram $D_{(p_1,p_2),(e_1,e_2)}$. Let $\theta_0$ be some fixed angle. We argue that a vertical line passing through $\theta_0$ in the diagram crosses the containing region of $p$ at most twice. We fix a point $q$ on the ellipse $E_p$ that corresponds to the angle $\theta_0$, and follow it during the offset process.

We claim that if at some offset $q$ “leaves” the interior of $P$, it will never “enter” it again. During the entire offset process, $q$ moves with a constant velocity along a straight line. Suppose it moves up along a vertical line. Let us assume, without loss of generality, that at some offset $q$ leaves the interior of the polygon through an edge $e$, and that at that moment the polygon lies beneath $q$. Thus, $q$ moves up faster than $\ell(e)$. On the other hand, since $P$ is convex, it lies entirely beneath $\ell(e)$ and will always stay beneath it. The line $\ell(e)$ moves with the same velocity during the entire offset process. Thus, starting at this moment, $q$ will stay above $\ell(e)$ forever, and hence, outside the polygon.

During the offsetting of a convex polygon, its edges are getting longer constantly (sometimes with a different speed). Note that no edge is split during offsetting a convex polygon, thus, we build our diagrams for the entire edges. Hence, an obtainable region for a pair of edges and a pair of points constantly gets wider or does not change its width. Therefore, each two-dimensional obtainable region in a translation-rotation-offset diagram is monotone with respect to the horizontal axis as well.

Since the translation-offset diagram is a vertical cross-section of the translation-rotation-offset diagram, the complexity of an obtainable and each one of the contain-
ing regions in the former diagram is $\Theta(1)$. Hence, the complexity of each translation-offset diagram is $O(n)$. Building a containing or obtainable region takes $O(m)$ time. As a result, constructing the entire diagram takes $O(n(m + \log n))$ time, and the entire algorithm time complexity becomes $O(n^3 m^2 (m + \log n))$.

**Theorem 10** One can solve the Containment and Partial Containment problems for convex polygons under translation only in $O(n^3 m^2 (m + \log n))$ time $\Theta(n + m)$ space.

**Non-simply-connected polygons**

As we noted in the introduction, an offset of a simple polygon is not necessarily simple: it can have holes or consist of several simple polygons (or both). Since our algorithm works for simple polygons, it also works correctly for non simply-connected polygons.

Thus, we have:

**Theorem 11** One can solve the Containment and the Partial Containment problems for polygons with holes and sets of polygons under translation and rotation in $\Theta(n^4 m^3 \log(nm))$ time and $\Theta(n^2 m^2)$ space in the worst case.
One can solve the Max-Cover problem for these types of polygons under translation and rotation in $\Theta(n^3 m^3 \log(nm))$ time and $\Theta(nm)$ space in the worst case.

**2.5.3 Polygonal annuli**

We now describe the changes needed in our translation-rotation-offset diagram-based algorithm to adopt it for Containment and Partial Containment problems for polygon offset annuli. The changes in the algorithm for the Max-Cover problem are similar. See Figures 2.14(b) and 2.14(c) for an illustration of covering points by an offset polygon and a polygon annulus, respectively.
Figure 2.14: Covering points
Consider a simple polygon $P$, given by a counter-clockwise list of its vertices: $v_1, v_2, \ldots, v_{m-1}, v_m$. Now, let us think about $P$ as being a polygon with a hole $v'_m, v'_{m-1}, \ldots, v'_2, v'_1$, where $v'_i$ is a copy of the vertex $v_i$ moved infinitesimally towards the interior of $P$ in the direction of $v_i$'s angular bisector. See Figure 2.15 for an illustration. The distance from $v_i$ to $v'_i$ is infinitesimally small. We denote the new polygon with a hole by $\hat{P}$, to distinguish it from the original polygon $P$.

![Figure 2.15: The polygon $\hat{P}$](image)

Now, we can regard a $\delta$-annulus of $P$ as an outer $\delta$-offset polygon of $\hat{P}$. Note that $\hat{P}$ does not have an inner offset, since for every $\delta < 0$, its $\delta$-offset does not exist. In fact, $P$ is the center of the straight skeleton of $\hat{P}$.

Thus, solving the Containment or Partial Containment problem for an annulus of $P$ is merely solving the same problem for an offset polygon of $\hat{P}$. Although we double the number of vertices and edges of $P$, it does not change the asymptotic time complexity of the algorithm, which remains $\Theta(n^4m^3\log(nm))$ in the worst case.

The same holds for offset annuli of polygons with holes and of sets of polygons.

### 2.5.4 Constrained annuli

As mentioned in the introduction, in various applications one wants to fix the inner or outer boundary of a polygonal annulus. For example, in geometric tolerancing, if a manufactured object is to fit inside a sleeve, then its outer boundary should be fixed. If an object must fit over a peg, then its inner boundary must be fixed. For these applications, covering a set of points by a constrained inner or outer annulus
becomes a useful task.

For constrained inner and outer offset annuli of simple polygons, as well as of sets of polygons and polygons with holes, we use the same technique as in Section 2.5.3. We define \( \hat{P} \) as before except only one difference: For constrained inner annulus, the “hole” in \( \hat{P} \) is shrinking and the outer boundary of \( \hat{P} \) does not change during the offset process. Analogously, for constrained outer annulus, the hole is fixed and the outer boundary of \( \hat{P} \) is inflated.

Our algorithm can be easily adapted for these cases. In both cases we have a polygon with a hole, and during the offsetting, some of its edges move (with a constant and equal velocity), and others remain in their place. This change affects only the calculations of various motion speeds that we use in the algorithms of building obtainable and containing regions. For instance, when we build a containing region of a point \( p \) in a diagram \( D(p_1,p_2),(e_1,e_2) \), we should compute the velocity of \( E_p \) accordingly to whether or not each one of \( e_1 \) and \( e_2 \) is static.

**Theorem 12** One can solve the Containment and the Partial Containment problems for offset annuli and constrained inner and outer offset annuli of simple polygons, polygons with holes, and sets of polygons under translation and rotation in \( \Theta(n^4m^3\log(nm)) \) time and \( \Theta(n^2m^2) \) space in the worst case. One can solve the Max-Cover problem for these types of polygons under translation and rotation in \( \Theta(n^3m^3\log(nm)) \) time and \( \Theta(nm) \) space in the worst case.

### 2.5.5 True offset

Recall that a true offset polygon is made of straight line segments and circular arcs—see Figure1.1(b). Also refer to Figures 2.14(b) and 2.14(c) for an illustration of covering points by a true offset polygon and a true polygon annulus, respectively. In this case we use the Voronoi diagram of \( P \) as the appropriate skeleton.
For true offsets we build a translation-rotation-offset diagram, similar to that of the linearized-offset case, with the following differences:

1. When a triangle $\triangle p_1p_2p_3$ slides with $p_1$ (resp., $p_2$) along an edge $e_1$ (resp., $e_2$), the point $p_3$ describes an elliptic arc when $e_1$ and $e_2$ are straight line segments. When one (or two) of them is a circular arc, the motion of $p_3$ is no longer along the boundary of an ellipse, but along some other curve. Moreover, as the offset process continues, the circular edges change their radii (at offset $\delta$, all the circular edges have radius $|\delta|$). The curve traced by $p_3$ not only moves but also changes its shape.

It turns out that the trace of a moving triangle with two of its vertices sliding along two circles is a fundamental problem in kinematics, that is called the four-bar linkage (see, e.g, [WK98]). See Figure 2.16 for an illustration. McCarthy [McC00] provides a detailed solution of the problem, which specifies the equation of the curve traced by $p_3$ (a degree-six algebraic curve). He also analyzes the so-called angle limit, which is in our terminology the obtainable region.

The case where $e_1$ is circular and $e_2$ is straight can be solved in a similar way.

2. When the two edges are circular arcs, we can no longer use the parameterization of a curve proposed above. However, we can use as a parameter the angle between the segment $p_1p_2$ and any fixed straight line. Recall that choosing this line to be the line supporting one of the segments was arbitrary.

3. We should take into account the fact that there are parabolic arcs in $VD(P)$ when building a containing region and looking for intersection points of skeleton arcs and the ellipse $E_\nu$.

All these modifications complicate significantly the calculations but do not change
Figure 2.16: A four-bar linkage

The correctness of the algorithm, nor its asymptotic running time and space complexity.

**Theorem 13** One can solve the Containment and the Partial Containment problems for true offsets with respect to annuli and constrained inner and outer annuli of simple polygons, polygons with holes, and sets of polygons under translation and rotation in $\Theta(n^4 m^3 \log(nm))$ time and $\Theta(n^2 m^2)$ space in the worst case.

One can solve the Max-Cover problem in $\Theta(n^3 m^3 \log(nm))$ time and $\Theta(nm)$ space in the worst case.
Chapter 3

Stable Poses

3.1 Preface

3.1.1 Stable poses

Definition 7 A pose of a polygon is a pair $\langle \delta, \tau \rangle$, where $\delta$ is an offset and $\tau$ is a placement (in our context—$\tau$ is a translation and rotation).

Inspired by Chazelle [Cha83], who coined the term “stable placements,” we define:

Definition 8 (Two-point stable pose) A two-point stable pose is a pose $\langle \delta, \tau \rangle$ satisfying the following conditions:

1. There are two points $p_1, p_2 \in S$ on the boundary of $\tau(O_P, \delta)$; and

2. There exists $d > 0$, such that for every $\delta' \in (\delta - d, \delta)$, there is a small neighborhood of $\tau$, in which for every $\tau'$ in this neighborhood, at least one of the points $p_1, p_2$ does not lie on the boundary of $\tau'(O_{P,\delta'})$.

Definition 9 (Three-point stable pose) A three-point stable pose is a pose $\langle \delta, \tau \rangle$ satisfying the following conditions:
1. There are three points \( p_1, p_2, p_3 \in S \) on the boundary of \( \tau(O_P, \delta) \); and

2. There exists \( d > 0 \), such that for every \( \delta' \in (\delta - d, \delta) \), there is a small neighborhood of \( \tau \), in which for every \( \tau' \) in this neighborhood, at least one of the points \( p_1, p_2, p_3 \) does not lie on the boundary of \( \tau'(O_P, \delta') \).

**Definition 10 (Four-point stable pose)** A four-point stable pose is a pose \( \langle \delta, \tau \rangle \), for which there are four points of \( S \) on the boundary of \( \tau(O_P, \delta) \).

Suppose that we are given a simple polygon \( P \) and a point set \( S \), let us address the Containment problem. (The discussion below is valid for both true and linearized offsets.) We build translation-rotation-offset diagrams for every pair of points of \( S \) and for every pair of edges of \( P \). In these diagrams, we find regions of depth \( n - 2 \). The lowest point in all these regions represents the optimum pose. This point is one of the following:

- The (locally) lowest point on the boundary of an obtainable region. In this case it corresponds to a two-point stable pose.

- The (locally) lowest point on the boundary of a containing region. In this case it corresponds to a three-point stable pose.

- The intersection point of the boundaries of two containing regions. In this case it corresponds to a four-point stable pose.

- The intersection point between the boundaries of a containing and obtainable regions. An accurate analysis shows that in this case the intersection point corresponds to either a three-point stable pose or to a pose at which three points are on the polygon’s boundary, while two of these points identify with vertices of the polygon. The latter case is clearly degenerate. Thus, we do not consider it a stable pose. Nevertheless, we show how to check this case at the end of Section 3.2.4.
The stable poses for the linearized offsets are defined in the same way. We shall use the term stable poses for both types of offsets. The exact meaning will be clear from the context.

3.1.2 The approach

We start by presenting an algorithm for solving the Containment problem for a simple polygon under linearized offset, rotation, and translation. At the end of this chapter we show how this solution can be modified to work also for true offsets and polygonal annuli. We also show that we can solve the Partial Containment problem using the same algorithm.

We know that the solution of the Containment problem (the optimum pose) is one of the stable poses (as defined above). Thus, the basic idea of our approach is to compute all the stable poses of $P$ with respect to $S$ and choose the stable pose $\langle \delta, \tau \rangle$ with minimal $\delta$, for which $\tau(O_{P,\delta})$ contains all the points of $S$.

3.2 The Algorithm

3.2.1 Overview

The algorithm proceeds as follows.

1. Preprocess $P$;

2. Find all the stable poses;

3. For each stable pose $\langle \delta, \tau \rangle$, count how many points of $S$ are contained in $\tau(O_{P,\delta})$;

4. Return the stable pose with minimal $\delta$, for which all the points are contained in $\tau(O_{P,\delta})$. 
Remark: We assume that points of \( S \) are distinct. Otherwise, we can discard in \( O(n \log n) \) time redundant copies of points that appear more than once. This preprocessing step will not affect the total running time.

### 3.2.2 Preprocessing

1. Find the straight skeleton of \( P \). We use the algorithm of [FO98], which runs in \( O(m^2) \) time in the worst case and in linear space.\(^1\)

2. Triangulate\(^2\) \( SS(P) \) and preprocess the triangulation for point location as in [EGS84].

   This takes \( O(m \log m) \) time. The resulting data structure uses \( O(m) \) space.

The total preprocessing time is \( O(m^2) \). The amount of space required for the preprocessing is \( O(m^2) \).

### 3.2.3 Point query

Given a point \( p \), a transformation \( \tau \), and an offset \( \delta \) of \( P \), we want to check whether \( p \in \tau(O_{P,\delta}^L) \).

The algorithm is thus:

1. Let \( p' = \tau^{-1}(p) \). The computation of \( p' \) requires \( \Theta(1) \) time.

2. Find which face of \( SS(P) \) contains \( p' \). This is performed by a point-location algorithm developed by Edelsbrunner et al. [EGS84], in \( O(\log m) \) time. Let us denote the defining edge (of \( P \)) of this face by \( e' \).

3. Compute the Euclidean distance \( d \) between \( p' \) and \( \ell(e') \) (the supporting line of \( e' \)). Let \( d \) be negative if \( p' \) is inside \( P \) and positive otherwise. Let \( d = 0 \) if \( p' \)

\(^1\)There are faster algorithms, but we use this one since it is the most space-efficient.

\(^2\)There is a linear-time algorithm [Cha91], but we can also use any of the \( O(m \log m) \)-time practical methods.
lies on the boundary of $P$. This step requires $\Theta(1)$ time, since we only need to
decide on which side of $\ell(e')$ the point $p'$ lies in order to determine the sign of
$d$.

4. Compare $d$ and $\delta$. If $d \leq \delta$, then $p \in \tau(O_{P,\delta}^L)$; otherwise $p \notin \tau(O_{P,\delta}^L)$. This step
is performed in constant time.

We conclude:

**Lemma 14** Given a point $p$, a transformation $\tau$, and an offset $\delta$ of $P$, we can check
whether $p \in \tau(O_{P,\delta}^L)$ in $O(\log m)$ time.

### 3.2.4 Finding two-point stable poses

Let $p_1$ and $p_2$ be two points of $S$. If for some pose $\langle \delta, \tau \rangle$, $p_1$ and $p_2$ lie on one edge
or on two nonparallel edges (but not on vertices) of $\tau(O_{P,\delta}^L)$, we can shrink $P$ a little
(by decreasing $\delta$) and change $\tau$ accordingly, such that $p_1$ and $p_2$ remain on one or two
edges. Thus, this pose $\langle \delta, \tau \rangle$ is nonstable. Consequently, at a two-point stable pose,
either at least one of the points $p_1$, $p_2$ lies on a vertex or they both lie on parallel
edges of $\tau(O_{P,\delta}^L)$.

Hence, at a two-point stable pose one of the following holds:

1. One of the points lies on a vertex $v$ and the other on an edge $e$. Obviously, the
circle of radius $|p_1p_2|$ centered at $v$ is tangent to $e$, otherwise it is not a stable
pose; see Figures 3.1(a) and 3.1(b).

2. Both points lie on vertices of $O_{P,\delta}^L$; see Figures 3.2(a) and 3.2(b).

3. Both points lie on two parallel edges $e_1, e_2 \in P$; see Figure 3.3. At first glance
it would seem that in this case we have infinitely many stable poses. However,
as we show later, only a constant number of them is of real interest.
Figure 3.1: Stable and non-stable poses involving a vertex and an edge

Figure 3.2: Stable and non-stable poses involving two vertices
For two given points $p_1, p_2 \in S$, our algorithm simply checks all the pairs of two vertices, a vertex and an edge, and two edges, and for each pair computes the respective offset $\delta$.

A vertex and an edge

Instead of checking the vertex and edge pairs, we check all the pairs composed of an arc of $SS(P)$ and an edge of $P$. (An arc $a$ represents the motion of a vertex $v$ of $P$ during the offset process.) $\langle \delta, \tau \rangle$ is a stable pose for a given pair of an arc $a$ and an edge $e$ if it satisfies the following conditions:

1. At the offset $\delta$, the distance between $v$ and $\ell(e)$ is $|p_1p_2|$—see Figure 3.1(a).

2. When the offset is decreased, the distance increases. Refer to Figure 3.1(a).

   In this example, when the offset decreases, $v$ moves to the left and $e$ moves to the right. If the distance decreases as the offset is decreased, then this is not a stable pose—see Figure 3.1(c).

   We shall refer later to the special case in which the distance stays constant during the offset.

3. Both edges incident to $v$ are directed “away” from $e$ (see Figure 3.1(a)); otherwise this is not a stable pose. For example, in Figure 3.1(d), one of the edges
incident to \( v \) is directed “toward” \( e \).

We can find the offset \( \delta \) at which the distance between \( v \) and \( e \) equals \( |p_1p_2| \) in a constant time. Let \( w \) denote the point where a circle of radius \( |p_1p_2| \) centered at \( v \) touches \( \ell(e) \) (see Figure 3.1(a) for an illustration). After finding the offset, it is easy to find the transformation \( \tau \) that puts \( p_1 \) on \( v \) and \( p_2 \) on \( w \).

After finding the pose, we need only to decide whether \( p_2 \) lies on \( e \) at a given offset (and not just on \( \ell(e) \)). We do it by checking whether \( w \) belongs to the face of \( e \) in \( SS(P) \). This can be performed in \( O(\log m) \) time using the point-location algorithm of [EGS84] (recall that we preprocess \( SS(P) \) to support this kind of operations).

We have two special cases:

- At the found pose, \( p_1 \) lies on a node of \( SS(P) \). In this case, we should also check the adjacent arcs of the skeleton to make sure that this is really a stable pose.

- The distance between \( v \) and \( e \) remains constant during the offsetting. A detailed analysis shows that we can discard this case, since it can only produce a two-vertices stable pose, which we handle separately, or a stable pose with \( p_1 \) on a node of \( SS(P) \), which we handled above.

There are \( O(m) \) arcs in \( SS(P) \), and we check each one against all the \( m \) edges of \( P \). Thus, checking all the pairs of arcs and edges takes \( O(m^2 \log m) \) time.

Thus we summarize:

**Lemma 15** All the two-point stable poses of type vertex-edge for a given pair of points \( p_1, p_2 \in S \) can be found in \( O(m^2 \log m) \) time. There are \( O(m^2) \) such stable poses.

**Two vertices**

Similarly to the previous case, instead of checking the two-vertices pairs, we check all the pairs of arcs of \( SS(P) \). \( \langle \delta, \tau \rangle \) is a stable pose for a given pair of arcs \( a_1, a_2 \) if it satisfies the following conditions:
1. At the offset $\delta$, the distance between $v_1$ (that moves on $a_1$) and $v_2$ (that moves on $a_2$) is $|p_1p_2|$—see Figures 3.2(a) and 3.2(b), in which the radius of the circle is $|p_1p_2|$.

2. The distance increases as the offset decreases, both edges incident to $v_1$ are directed “away” from $v_2$, and the edges sharing $v_2$ are directed away from $v_1$ (see Figure 3.2(b)). However, if there is an edge pointing to another direction, then this is not a stable pose—see Figure 3.2(c).

Or:

The distance decreases as the offset decreases, both edges incident to $v_1$ are directed “toward” $v_2$, and the edges sharing $v_2$ are directed toward $v_1$ (see Figure 3.2(a)). As in the previous case, if there is an edge pointing to another direction, then this is not a stable pose—see Figure 3.2(d).

Later we shall handle the special case in which the distance stays constant during the offsetting.

We can find the offset $\delta$ at which the distance between $v_1$ and $v_2$ equals $|p_1p_2|$ in a constant time. After finding the offset, it is easy to find the transformation $\tau$ that puts $p_1$ on $v_1$ and $p_2$ on $v_2$. Thus, we can find the stable pose in $O(1)$ time.

We have two special cases:

- At the found pose, $p_1$ or $p_2$ lies on a node of $SS(P)$. In this case, we should also check the adjacent arcs of the skeleton to make sure that this is really a stable pose.

- The distance between $v_1$ and $v_2$ remains constant during the offsetting. Thus, we only need to check the skeletal nodes of $a_1$ and $a_2$.

There are $O(m)$ arcs in $SS(P)$ and, consequently, $O(m^2)$ pairs of arcs. Checking each pair of arcs takes $O(1)$ time. Hence, we spend $O(m^2)$ time on finding stable poses of this type.
In conclusion, we obtain:

**Lemma 16** All the two-point stable poses of the two-vertices type for a given pair of points \( p_1, p_2 \in S \) can be found in \( O(m^2) \) time. There are \( O(m^2) \) such stable poses.

**Two edges**

As stated before, we have to check only the pairs of parallel edges (see Figure 3.3). Assume that \( e_1 \) is parallel to \( e_2 \), and at some offset \( \delta_0 \) the distance between them is exactly \( |p_1p_2| \). Now, transform \( O_{L,\delta_0}^L \) such that \( p_1 \) lies on \( e_1 \) and \( p_2 \) on \( e_2 \). Translate \( O_{L,\delta_0}^L \) parallel to \( e_1 \), such that \( p_1 \) and \( p_2 \) stay on their respective edges. The number of points contained by \( O_{L,\delta_0}^L \) does not change until the polygon hits another point of \( S \). If it does not, we can proceed until one of the points \( p_1, p_2 \) reaches an endpoint of its edge. Thus, from all the continuum of the stable poses, we are interested only in the ones that originate from the following situations:

- During the translation a point \( p_3 \in S \) is hit by the boundary of \( O_{L,\delta_0}^L \). This is actually a three-point stable pose, which we consider later on.

- \( p_1 \) (resp., \( p_2 \)) reaches an endpoint of \( e_1 \) (resp., \( e_2 \)). Note that we have already found this pose when searching for vertex-edge stable poses. Furthermore, if \( p_1 \) and \( p_2 \) hit the endpoints of their edges simultaneously, then this is a two-vertices stable pose, which we already handled before.

In conclusion, we do not need to check this type of two-point stable pose.

**Summary**

We iterate over all the pairs of points from \( S \): There are \( \binom{n}{2} = \Theta(n^2) \) of them. For each pair we are looking for all possible stable poses. Thus, finding all two-point stable poses takes \( O(n^2m^2 \log m) \) time.
There are $O(m^2)$ stable poses for each pair of points $p_1, p_2 \in S$. Thus, there are $O(n^2m^2)$ two-point stable poses.

The following theorem summarizes our results so far:

**Theorem 17** All the two-point stable poses for a simple $m$-gon and a set of $n$ points can be found in $O(n^2m^2 \log m)$ time. There are $O(n^2m^2)$ such stable poses.

In Chapter 4 we show that this bound is tight in the worst case.

Remark: To find all the poses in which three points of $S$ lie on the polygon boundary while two points identify with two vertices of the polygon, we consider all the poses that put some two points on two vertices of $P$. We can find these poses in $O(n^2m^2)$ time. For each such pose, we check all other points of $S$ against the boundary of $P$. (Note that we do this also for all the stable poses.)

### 3.2.5 Finding three-point stable poses

Let $p_1, p_2, p_3$ be three points of $S$. To find possible stable poses, we simply check all the triples of edges of $P$. For each triple of edges $e_1, e_2, e_3 \in P$ we find the minimal $\delta$ and a placement $\tau$, such that $p_1$, $p_2$, and $p_3$ lie on $e_1$, $e_2$, and $e_3$ (of $\tau(O_P, \delta)$), respectively.

Let us fix a triple of edges $e_1, e_2, e_3 \in P$. Their supporting lines $\ell(e_1)$, $\ell(e_2)$, and $\ell(e_3)$ are given in a polar-type form $\langle \rho_i, \varphi_i \rangle$ for $i \in \{1, 2, 3\}$. $\rho_i$ are functions of the offset $\delta$ and are given by:

$$\rho_i(\delta) = \rho_i^0 + c_i \delta,$$

where $\rho_i^0$ is the distance between $C(P)$ (the center of $P$, abbreviated in what follows by $C$) and $\ell(e_i)$ at offset 0, and $c_i$ equals 1 or $-1$ depending on whether $\ell(e_i)$ moves away or toward $C$, respectively, during the offsetting.

We wish to find a placement $\tau$ that puts $p_1$ on $\ell(e_1)$, $p_2$ on $\ell(e_2)$, and $p_3$ on $\ell(e_3)$—see Figure 3.4. The points $p_1, p_2, p_3$ form a triangle with angles $\alpha_1, \alpha_2$, and $\alpha_3$. 

77
Figure 3.4: Three points on three lines
$CR_1$, $CR_2$, and $CR_3$ are normals to $\ell(e_1)$, $\ell(e_2)$, and $\ell(e_3)$ respectively.

We denote the signed distances $|R_1p_1|$, $|R_2p_2|$, and $|R_3p_3|$ by $t_1$, $t_2$, and $t_3$, respectively. The value of $t_i$ is positive if the polygon lies to the left of $\overrightarrow{R_ip_i}$, for $1 \leq i \leq 3$. The angle $\theta$ is the rotation component of the placement $\tau$. We now find $t_1$, $t_2$, $t_3$, and $\theta$ as functions of $\delta$.

First, we express $t_1$ and $\theta$ in terms of $\delta$, such that the three points are on their corresponding lines.

$$R_1 = (\rho_1 \cos \varphi_1, \rho_1 \sin \varphi_1),$$
$$\overrightarrow{R_1p_1} = (-t_1 \sin \varphi_1, t_1 \cos \varphi_1).$$

Thus,

$$p_1 = R_1 + \overrightarrow{R_1p_1} = (\rho_1 \cos \varphi_1 - t_1 \sin \varphi_1, \rho_1 \sin \varphi_1 + t_1 \cos \varphi_1).$$

Now,

$$\overrightarrow{p_1p_2} = (|p_1p_2| \cos \theta, |p_1p_2| \sin \theta),$$
$$\overrightarrow{p_1p_3} = (|p_1p_3| \cos (\theta + \alpha_1), |p_1p_3| \sin (\theta + \alpha_1)).$$

And thus,

$$p_2 = p_1 + \overrightarrow{p_1p_2}$$
$$= (\rho_1 \cos \varphi_1 - t_1 \sin \varphi_1 + |p_1p_2| \cos \theta, \rho_1 \sin \varphi_1 + t_1 \cos \varphi_1 + |p_1p_2| \sin \theta),$$

$$p_3 = p_1 + \overrightarrow{p_1p_3}$$
$$= (\rho_1 \cos \varphi_1 - t_1 \sin \varphi_1 + |p_1p_3| \cos (\theta + \alpha_1), \rho_1 \sin \varphi_1 + t_1 \cos \varphi_1 + |p_1p_3| \sin (\theta + \alpha_1)).$$

Now, we demand that $p_2$ lie on $\ell(e_2)$ and $p_3$ on $\ell(e_3)$. That is, they must satisfy the line equations

$$x \cos \varphi + y \sin \varphi = \rho$$
of the lines $\ell(e_2)$ and $\ell(e_3)$, respectively. As a result, we get the following system of equations:

\[
\begin{align*}
\rho_1 \cos \varphi_1 - t_1 \sin \varphi_1 + |p_1 p_2| \cos \theta \cos \varphi_2 \\
+ (\rho_1 \sin \varphi_1 + t_1 \cos \varphi_1 + |p_1 p_2| \sin \theta) \sin \varphi_2 &= \rho_2, \\
\rho_1 \cos \varphi_1 - t_1 \sin \varphi_1 + |p_1 p_2| \cos(\theta + \alpha_1) \cos \varphi_3 \\
+ (\rho_1 \sin \varphi_1 + t_1 \cos \varphi_1 + |p_1 p_2| \sin(\theta + \alpha_1)) \sin \varphi_3 &= \rho_3.
\end{align*}
\]

After rearranging the equations, we get:

\[
\begin{align*}
\sin \theta(|p_1 p_2| \sin \varphi_2) + \cos \theta(|p_1 p_2| \cos \varphi_2) \\
+ t_1 \sin(\varphi_2 - \varphi_1) + (\rho_1 \cos(\varphi_2 - \varphi_1) - \rho_2) &= 0, \\
\sin \theta(|p_1 p_3| \sin(\varphi_3 - \alpha_1)) + \cos \theta(|p_1 p_3| \cos(\varphi_3 - \alpha_1)) \\
+ t_1 \sin(\varphi_3 - \varphi_1) + (\rho_1 \cos(\varphi_3 - \varphi_1) - \rho_3) &= 0.
\end{align*}
\]

This is a system of two equations with two unknowns: $t_1$ and $\theta$. We rewrite the system while substituting

\[
\begin{align*}
\sin \theta &= \frac{2 \tan \frac{\theta}{2}}{1 + \tan^2 \frac{\theta}{2}}, \\
\cos \theta &= \frac{1 - \tan^2 \frac{\theta}{2}}{1 + \tan^2 \frac{\theta}{2}}
\end{align*}
\]

and obtain a system of two equations with two unknowns: $t_1$ and $\tan(\theta/2)$. We can solve it analytically for $t_1$ and $\tan(\theta/2)$ in terms of the offset $\delta$ (recall that the $\rho$s are functions of $\delta$). Note that since the equations are quadratic, we can get two different solutions. In fact, there are two solutions in the general case: when $p_1$ and $p_2$ slide along $\ell(e_1)$ and $\ell(e_2)$, respectively, the motion of $p_3$ describes an ellipse, which $\ell(e_3)$ can intersect at two distinct points.

After obtaining $t_1$ and $\theta/2$, we can express $t_2$ and $t_3$ as functions of $\delta$.

Now we find (locally) minimal $\delta$, such that $t_1(\delta)$, $t_2(\delta)$, and $t_3(\delta)$ correspond to points inside the skeletal faces of $e_1$, $e_2$, and $e_3$.

Figure 3.5 shows an example of the skeletal faces of $e_1$, $e_2$, and $e_3$ together with the graphs of $t_1(\delta)$, $t_2(\delta)$, and $t_3(\delta)$. The regions where the graph of $t_i(\delta)$ is inside
the face of $e_i$ correspond to offsets at which $p_i$ lies on $e_i$ (and not only on $\ell(e_i)$) for each $1 \leq i \leq 3$. The regions where the three graphs lie inside their faces correspond to offsets at which the three points lie on the three edges.

In Figure 3.5, $\delta_1$ corresponds to a stable pose, while $\delta_4$ does not. $\delta_2$ and $\delta_3$ are candidates for being stable poses—see subsequent remarks. The number of stable poses is thus upper bounded by $O(k_1 + k_2 + k_3)$, where $k_i$ is the complexity of the face defined by $e_i$. We find these stable poses by computing the endpoints and intersections of each curve $t_\delta(e)$ with the face of $e_i$. The number of intersection points for the face of $e_i$ is $O(k_i)$, and the time required to find them is $O(\sum_{i=1}^{3} (k_i \log k_i))$, since we have to sort them according to the respective offsets. Then we can easily find the minimal offsets. (This may perhaps be done in a more efficient way, but even so, it will not affect the total time complexity.)

![Figure 3.5: Finding stable poses for three points](image_url)

We have four special cases:

- If the found stable pose for a triple of points $p_1, p_2, p_3$ and a triple of edges $e_1, e_2, e_3$ puts one of the points, say, $p_1$, on a vertex $v$ (endpoint of $e_1$), then we
should also check the triple of edges $e'_1, e_2, e_3$, where $e'_1 \neq e_1$ is the other edge adjacent to $v$, with the same triple of points. If, again, we find the same stable pose, then it is indeed a stable pose. Otherwise, it means that we can offset the polygon inward further while still keeping $p_1, p_2$, and $p_3$ on $e_1, e_2$, and $e'_1$, respectively. Thus, this is not a stable pose.

- If two of the three edges $e_1, e_2, e_3$ are parallel, we can still use the previous approach and find the stable poses.

- If all the three edges $e_1, e_2, e_3$ are parallel, then we cannot use the previous technique, since there may be a continuum of translations for the same rotation and offset that put the three points on the three edges. Thus, apparently, there is a continuum of stable poses. However, similarly to what is stated in a remark for two-point stable poses, we are interested only in those poses, that put another point $p_4 \in S$ on $O_{P,s}$'s boundary, or put one of the points $p_1, p_2, p_3$ on a vertex. The former is a four-point stable pose, which we handle later. In the latter case we need to check another triple of edges, as before.

- We also have to check triples in which one edge appears twice, e.g., $e_1, e_1, e_2$. In such cases at least one of the points $p_1, p_2, p_3$ must lie on a vertex. Otherwise, we can shrink $P$ in a way that keeps the three points on the boundary of $P$.

  The same is true for triples of only one edge, e.g., $e_1, e_1, e_1$.

  Each edge participates in $O(m^2)$ triples; thus the total time complexity of handling all the triples is $O(m^2(\sum_{i=1}^m k_i \log k_i)) = O(m^3 \log m)$.

  We iterate over all the triples of points from $S$; there are $\binom{n}{3} = \Theta(n^3)$ such triples. For each triple we search for all the stable poses. Thus, finding all three-point stable poses takes $O(n^3 m^3 \log m)$ time.

  There are $O(m^2(\sum_{i=1}^m k_i)) = O(m^3)$ stable poses for each triple of points $p_1, p_2, p_3 \in S$. Hence, there are $O(n^3 m^3)$ three-point stable poses.
In summary we have:

**Theorem 18** All the three-point stable poses for a simple $m$-gon and a set of $n$ points can be found in time $O(n^3m^3 \log m)$. There are $O(n^3m^3)$ such stable poses.

We show in Chapter 4, that this bound is tight in the worst case.

### 3.2.6 Finding four-point stable poses

For each quadruple $p_1, p_2, p_3, p_4$ of points of $S$ we check all the quadruples of edges of $P$ and find a pose $⟨\delta, \tau⟩$, such that $p_i$ lies on $e_i$ for $1 \leq i \leq 4$; see Figure 3.6.

![Figure 3.6: Four points on four lines](image)

As in the three-point case, $CR_1, CR_2, CR_3,$ and $CR_4$ are normals to $\ell(e_1)$, $\ell(e_2)$, $\ell(e_3)$, and $\ell(e_4)$, respectively. We denote the angle $\angle p_1p_2p_3$ by $α'_1$. We denote the
signed distances $|R_i p_i|$ by $t_i$, for $1 \leq i \leq 4$. The sign is defined as above. $\theta$ is the rotation component of the placement $\tau$. We want to find $t_1$, $t_2$, $t_3$, $t_4$, $\theta$, and $\delta$.

Similar to the three-point stable-poses case, we write a system of equations. This time we obtain a system of three equations (for the three points on three lines) with three unknowns: $t_1$, $\theta$, and $\delta$.

$$\sin \theta(|p_1 p_2| \sin \varphi_2) + \cos \theta(|p_1 p_2| \cos \varphi_2) + t_1 \sin(\varphi_2 - \varphi_1) + \delta(c_1 \cos(\varphi_2 - \varphi_1) - c_2) + (\rho_0^1 \cos(\varphi_2 - \varphi_1) - \rho_2^0) = 0$$

$$\sin \theta(|p_1 p_3| \sin(\varphi_3 - \alpha_1')) + \cos \theta(|p_1 p_3| \cos(\varphi_3 - \alpha_1')) + t_1 \sin(\varphi_3 - \varphi_1) + \delta(c_1 \cos(\varphi_3 - \varphi_1) - c_3) + (\rho_0^3 \cos(\varphi_3 - \varphi_1) - \rho_3^0) = 0$$

$$\sin \theta(|p_1 p_4| \sin(\varphi_4 - \alpha_1)) + \cos \theta(|p_1 p_4| \cos(\varphi_4 - \alpha_1)) + t_1 \sin(\varphi_4 - \varphi_1) + \delta(c_1 \cos(\varphi_4 - \varphi_1) - c_4) + (\rho_0^4 \cos(\varphi_4 - \varphi_1) - \rho_4^0) = 0$$

As before, we rewrite the system substituting for $\sin \theta$ and $\cos \theta$ their expressions in terms of $\tan(\theta/2)$, and get a system of three equations with three unknowns: $t_1$, $\delta$, and $\tan(\theta/2)$. We solve it analytically or numerically, and obtain the exact stable pose (placement and offset).

After solving the system, we use the values of $t_1$, $\theta$, and $\delta$ to get $t_2$, $t_3$, and $t_4$.

Now, it remains to check whether $t_1$, $t_2$, $t_3$, and $t_4$ represent points that lie on the corresponding edges (rather than on the supporting lines). As before, we check whether these points lie inside faces of $e_1$, $e_2$, $e_3$, and $e_4$. For each point, we can check it in $O(\log k_i)$ time, where $k_i$ is the complexity of the face defined by $e_i$.

We have four special cases:

- If two or three edges of the quadruple are parallel, we can still use the same technique and find the stable poses.

- As in the three-point case, we also have to check quadruples, in which one edge appears more than once.
• When the quadruple consists of only one edge \( e \) or the four edges are parallel, it seems that we can get infinitely many stable poses. But as in previous cases, we are only interested in a constant number of them: when there is an additional point on the boundary (and this is a different four-point stable pose) or when one of the points lies on a vertex.

• The case in which four edges are parallel is very similar to the case in which all the edges are identical.

Each edge participates in \( O(m^3) \) quadruples, thus, the total complexity of processing all the quadruples is \( O(m^3(\sum_{i=1}^{m} k_i \log k_i)) = O(m^4 \log m) \).

We process all the quadruples of points of \( S \): There are \( \binom{n}{4} = \Theta(n^4) \) of them. For each quadruple we search for all the stable poses. Thus, finding all four-point stable poses takes \( O(n^4 m^4 \log m) \) time.

In summary, we see that:

\textbf{Theorem 19} All the four-point stable poses for a simple \( m \)-gon and a set of \( n \) points can be found in \( O(n^4 m^4 \log m) \) time. There are \( O(n^4 m^4) \) such stable poses.

In Chapter 4 we show that there are actually \( O(n^4 m^3) \) four-point stable poses and that this bound is tight in the worst case.

\subsection{3.2.7 Total running-time complexity}

\[
\begin{align*}
O(m^2) & \quad \text{Preprocessing} \\
+ O(n^2 m^2 \log m + n^3 m^3 \log m + n^4 m^4 \log m) & \quad \text{Finding all stable poses} \\
+ O((n^2 m^2 + n^3 m^3 + n^4 m^3) \times n \log m) & \quad \text{Checking all points for each stable pose} \\
= O(n^4 m^3 (n + m) \log m) & \\
\end{align*}
\]

We conclude with the following theorem:
Theorem 20 One can solve the Containment problem for a simple polygon under linearized offsets, rotation, and translation in $O(n^4m^3(n + m)\log m)$ time and in $\Theta(n + m)$ space.

3.3 Extensions

3.3.1 Linearized offset

Partial Containment problem

Note that the solution of the Partial Containment problem is one of the stable poses as well. Thus, we can use the same algorithm, with the only difference that now we look for placements of $P$ that contain only $k \leq n$ points of $S$.

Different types of polygons

The form of the polygon is not an issue in our algorithm. We are only concerned with having a constant number of edges at any time.

Thus, our algorithm will also work for polygons with holes and for sets of polygons.

Polygon annuli and constrained annuli

For the polygon-annulus and constrained annulus problems, we use the polygon $\hat{P}$, the “polygon-with-hole-representation” of $P$, as shown in Chapter 2. With no further changes, our algorithm will work in this case too.

All these differences do not affect the running time of the algorithm.

Theorem 21 One can solve the Containment and Partial Containment problems for simple polygons, polygons with holes, and sets of polygons, for linearized offset polygons, linearized annulus, and linearized constrained annulus, under rotation and translation in $O(n^4m^3(n + m)\log m)$ time and in $\Theta(n + m)$ space.
3.3.2 True offset

We use the same approach for true offsets as we did for the linearized offsets. The main differences are the following:

- In the preprocessing phase we calculate $VD(P)$ and not $SS(P)$ using one of the algorithms mentioned in the Introduction.

- We cannot use the approach of [EGS84] to locate a point in the subdivision, since it has nonstraight edges (parabolic arcs). Thus, we use the trapezoidal-decomposition method of Preparata [Pre81], which works also for subdivisions with constant-description edges. The only condition that our parabolic arcs do not satisfy is that they may be not $x$-monotone. We fix this by dividing such arcs, if necessary, into two pieces at the tangency point. The overall complexity of the skeleton and of the subdivision obviously remains $\Theta(m)$.

This technique allows query time $O(\log m)$, but at the expense of a data structure that requires $O(m \log m)$ space. The preprocessing time is still $O(m \log m)$.

- For true offsets, the polygon has circular edges and the skeleton also contains parabolic arcs. These complicate the calculations of stable poses, but do not change the asymptotic time and space complexity of the algorithm.

In conclusion we have:

**Theorem 22** One can solve the Containment and Partial Containment problems for simple polygons, polygons with holes, and sets of polygons, for true offset polygons, true annulus, and true constrained annulus, under rotation and translation in $O(n^4m^3(n + m) \log m)$ time and in $O(n + m \log m)$ space.
Chapter 4

The Number of Stable Poses

In this chapter we establish tight bounds on the number of two-, three-, and four-point stable poses for a simple \( m \)-vertex polygon \( P \) and a set \( S \) of \( n \) points.

4.1 Two-Point Stable Poses

**Theorem 23** The number of two-point stable poses for a simple \( m \)-gon and a set of \( n \) points is \( \Theta(n^2m^2) \) in the worst case.

**Proof** In the previous chapter we showed that the number of two-point stable poses is \( O(n^2m^2) \). We will now show a matching lower bound.

The following example will serve for both linearized and true offsets. It shows that there can be \( \Omega(n^2m^2) \) two-point stable poses; see Figure 4.1. Figure 4.1(a) shows a polygon \( P \) with \( 2k \) spikes, where \( k \approx m/4 \). The maximum distance between a vertex of \( \{v_1, \ldots, v_k\} \) and a vertex of \( \{u_1, \ldots, u_k\} \) is \( |v_1u_k| \).

Figure 4.1(b) shows a point set \( S \), divided into two groups, \( A \) and \( B \), such that \( |A| = |B| = \frac{n}{2} \). All the points of \( A \) are contained in a circle of some small radius \( \varepsilon \), as are the points of \( B \). The value of \( \varepsilon \) is much smaller than both the distance...
between $A$ and $B$ and the lengths of the spikes of $P$. The minimum distance between
a point of $A$ and a point of $B$ is exactly $|v_1u_k|$. The gaps between the spikes (that is,
the distances $|v_iv_{i+1}|$ and $|u_iu_{i+1}|$, for $1 \leq i < k$) equal $\varepsilon$. Note that when we offset
$P$ inward, the distance between the offset version of the vertices $v_i$ and $u_j$, for any
$1 \leq i, j \leq k$, increases.

For every pair of points $a \in A$ and $b \in B$, and for every pair of vertices $v_i$ and
$u_j$ ($1 \leq i, j \leq k$), we can translate, rotate, and offset $P$ inwards in a way that will
identify $v_i$ and $a$, and $u_j$ and $b$; see Figure 4.1(c) for examples. Such a pose is stable,
since if we decrease the offset further, there will be no (local) placement of $P$ with $a$
and $b$ on its boundary in small vicinities of $v_i$ and $u_j$, respectively.

Since there is only a finite number of stable poses and infinitely many ways to
arrange the points (inside the radius-$\varepsilon$ circles), we can arrange the points in $A$ and
$B$ such that all the stable poses are different. That is, there is no placement, in
which \( a \in A \) (resp., \( b \in B \)) identifies with \( v_i \) (resp., \( u_j \)), while another point of \( S \) also identifies with some point \( v_t \) (or \( u_t \)).

Thus, in this example there are \( \Omega(n^2m^2) \) two-point stable poses. This completes the proof. \( \Box \)

4.2 Three-Point Stable Poses

**Theorem 24** The number of three-point stable poses for a simple \( m \)-gon and a set of \( n \) points is \( \Theta(n^3m^3) \) in the worst case.

**Proof** The upper bound \( O(n^3m^3) \) was shown in Section 3.2.5. We will now show a matching lower bound.

As in the previous section, the same example holds for both linearized and true offsets.

Consider the example shown in Figure 4.2. The polygon \( P \) includes two bundles \( E_1 \) and \( E_2 \) of closely-spaced parallel edges. The distances between edges in one bundle are much smaller than the distances between the endpoints of the lowest edge of \( E_1 \) and the highest edge of \( E_2 \). The number of edges in \( E_1 \) and \( E_2 \) is roughly \( m/3 \). The point set \( S \) consists of three subsets of points \( A \), \( B \), and \( C \), where \( |A| = |B| = |C| = \frac{n}{3} \). The points of each subset are contained in a circle of some small radius \( \epsilon \), which is much smaller than the distances between \( A \), \( B \), and \( C \). The distances between the endpoints of edges in \( E_1 \) and edges in \( E_2 \), and from points of \( A \) to points of \( B \), satisfy the following property: For every pair of points \( a \in A \), \( b \in B \), and for every pair of edges \( e_1 \in E_1 \), \( e_2 \in E_2 \), we can slide the segment \( \overline{ab} \) for a strictly positive distance, while \( a \) slides along \( e_1 \) and \( b \) slides along \( e_2 \). (It is obvious that such a condition is realizable.)

For each triple of points \( a \in A \), \( b \in B \), \( c \in C \), and for each pair of edges \( e_1 \in E_1 \), \( e_2 \in E_2 \), as we slide \( a \) on \( e_1 \) and \( b \) on \( e_2 \), \( c \) draws an elliptic arc. The edges in \( E_1 \) and
in $E_2$ are closely-spaced, and the points in each subset $A$, $B$, and $C$ are also close to each other, such that we get $\Theta(n^3m^2)$ closely-spaced elliptic arcs. Figure 4.2 shows a few of these arcs.

In addition, there is a third bundle $E_3$ of closely-spaced edges, where $|E_3|$ is also roughly $m/3$. Every edge of $E_3$ crosses all the elliptic arcs.

Let us fix a triple of points $a \in A$, $b \in B$, $c \in C$ and a triple of edges $e_1 \in E_1$, $e_2 \in E_2$, $e_3 \in E_3$. There is a placement of $P$ that puts these points on these edges in order. As we start to offset $P$ inward, the edge $e_1$ moves in the “north-west” direction, while $e_2$ moves to the “south-west.” Thus, the elliptic arc that they create moves to the left (“west”) and slightly changes its length. The edge $e_3$ moves to the right (“east”). As long as it intersects the elliptic arc, there is a placement of $P$ that puts $a$, $b$, $c$ on $e_1$, $e_2$, $e_3$, respectively. We may assume that there is no topological change in $P$ during the offsetting. The edge $e_3$ ceases to intersect the elliptic arc in one of two cases: the left endpoint of $e_3$ reaches the arc, or an endpoint of the arc reaches $e_3$. We can make the edges in $E_1$ and $E_2$ long enough and tightly-spaced enough to ensure that the latter case will never happen. In the former case, if we continue to shrink $P$, there will be no more contact between $e_3$ or its neighboring edge with the elliptic arc. Thus, there will be no placement of $a$, $b$, $c$ on the three respective edges. Hence, the instance in which the edge ceases to intersect the arc is a three-point stable pose.
In fact, we have shown that each triple, which contains an edge of \( E_1 \), an edge of \( E_2 \), and the left endpoint of an edge of \( E_3 \), creates a stable pose for every triple of points \( a, b, c \). Thus, there are \( \Omega(n^3m^3) \) three-point stable poses in this example.

It is easy to satisfy the assumption that no topological change occurs during the offsetting. We can make \( P \) very large, so that all the stable poses are reached before any topological change occurs.

For true offsets, the same example gives \( \Omega(n^3m^3) \) three-point stable poses in the same manner. □

**Corollary 25** *The number of placements of a simple \( m \)-gon \( P \) under translation and rotation only, which put three points of a set of \( n \) points on the boundary of \( P \), is \( \Theta(n^3m^3) \) in the worst case.*

Note that this corollary refers to placements of \( P \) under translation and rotation only, whereas Theorem 24 allows offset as well.

**Proof**  The example of Theorem 24 (see Figure 4.2) shows an \( \Omega(n^3m^3) \) lower bound on the number of placements. Since every elliptic arc intersects every edge of \( E_3 \), and there are \( \Theta(n^3m^2) \) elliptic arcs, we conclude that in this example there are \( \Theta(n^3m^3) \) placements of \( P \) under translation and rotation only, which put three points on its boundary.

The \( O(n^3m^3) \) upper bound is obvious. For every triple of edges and every triple of points, there are at most two placements of the polygon under translation and rotation that put the three points on the three edges, since an ellipse and a straight line can intersect at no more than two points. □
4.3 Four-Point Stable Poses

Theorem 26 The number of four-point stable poses for a simple $m$-gon and a set of $n$ points is $\Theta(n^4m^3)$ in the worst case.

Proof As before, we first consider linearized offsets of $P$.

Let us begin with the upper bound. Recall that a fixed quadruple of edges creates a constant number of stable poses for a fixed quadruple of points of $S$.

We fix a quadruple of points $p_1, p_2, p_3, p_4 \in S$. Assume that we have found a stable pose $\langle \delta, \tau \rangle$ with $p_i$ on $e_i \in P$, for $1 \leq i \leq 4$. The rotation part of $\tau$ is an angle $\theta$. We denote by $C$ the center point of $\tau(O_{P,\delta}^{L})$. We denote by $C(P)$ the center point of $P$. (If the center is a line segment, we randomly choose one point on it as the center.) Note that, in general, $C \neq C(P)$. Obviously, if we put $P$, rotated by $\theta$ with its center on $C$, then every point $p_i$ will lie inside the face of $e_i$ in $SS(P)$; see Figure 4.3(b) for an illustration. Note, for instance, that $p_3$ lies inside the face of $e_3$.

We denote by $P^R$ the reflection of $P$ about its center; see Figure 4.3(c). Now, if we put a copy of $P^R$ rotated by $\theta$ and such that its center lies on $p_i$, then $C$ lies inside the face of $e_i^R$ in $SS(P^R)$, where $e_i^R$ is the edge corresponding to $e_i$ in $P^R$. Figure 4.3(d) shows $P^R$ rotated by $\theta$ and placed with its center on $p_3$. Note that $C$ lies inside the face of $e_3^R$. Consequently, $C$ lies in the intersection of the four respective faces of $e_i^R$ in $SS(P^R)$, when four copies of $P^R$ are placed with their centers on $p_i$.

Let us take four copies of $SS(P^R)$ and center them at $p_1, p_2, p_3,$ and $p_4$. We get a two-dimensional map in which each region is the intersection of some four faces of $SS(P^R)$. Now, we add a third dimension to the map—the rotation. As we rotate the skeletons (each one by the same angle $\theta$ around its center), these regions change. As $\theta$ varies from 0 to $2\pi$, we get a three-dimensional map. Each region of this map corresponds to four edges of $P$. The number of four-point stable poses is comparable to the number of such (topologically different) regions.
Figure 4.3: A polygon $P$ and its reflection $P^R$
Let us count these regions. We begin with the two-dimensional map. It has $O(m^2)$ regions, since it is the overlay of four (straight-edged) maps, each one of complexity $O(m)$. Its vertices are nodes of skeletons and intersection points of arcs of different skeletons. As we rotate the four skeletons (each one around its own center), the intersection points slide along the arcs, but this does not cause any topological changes. The regions change topologically only when a vertex (a node or an intersection point) hits an arc.

If we look “from above,” when we rotate the skeletons, each skeletal node follows a circular trajectory. Each skeletal arc follows a circular trajectory too, in the sense that its endpoints move along circles. Thus, each node can hit an arc no more than two times. Each triple of arcs can intersect no more than once. Thus, there are $O(m^3)$ topological changes of the map. We conclude that the complexity of the three-dimensional map is $O(m^3)$.

Hence, the number of four-point stable poses for a given quadruple of points is $O(m^3)$. Thus, the total number of four-point stable poses is $O(n^4m^3)$.

Now we show an example that achieves a matching lower bound. Refer to the polygon $P$ shown in Figure 4.4(a). This polygon contains four bundles of closely-spaced parallel edges $E_1, E_2, E_3, E_4$, each one consisting of roughly $m/4$ edges. All the long edges in $E_2, E_3,$ and $E_4$ are parallel. When we “zoom in,” each bundle is similar to what is shown in Figure 4.4(b). We consider only the edges that are drawn with heavy lines in this figure. Each bundle contains roughly $m/16$ of these edges. The distance between each such pair of edges in $E_1$ and $E_2$ is $\varepsilon_1$. When we offset $P$ inward, each such edge moves a distance of about $\varepsilon_1/2$ until it hits another edge.

Figure 4.4(c) shows four points $a, b, c, d$. The distances between these points are much larger than $\varepsilon_1$. The points are fixed in a way that ensures that for every triple of edges $e_1 \in E_1, e_2 \in E_2, e_3 \in E_3$, there is a placement of $P$ that puts $a$ on $e_1$, $b$ on $e_2$, and $c$ on $e_3$. When we offset $P$ inward (for offsets smaller than $\varepsilon_1$), there is still a
Figure 4.4: $\Omega(n^4m^3)$ four-point stable poses
placement with the three points on the three respective edges. Imagine that $P$ does not move, but only shrinks, and that the points slide along their respective edges. Then, the fourth point $d$ draws some curve (its trace). In our example, the heights (vertical extents) of all these traces are slightly less than $\varepsilon_1/2$; see Figure 4.4(e). In the figure, the traces are shown as vertical lines. Of course, in reality, the traces are not straight-line segments, but this does not affect our analysis.

Let us fix two edges $e_2 \in E_2$ and $e_3 \in E_3$. For every edge in $E_1$ we have a distinct trace, but since $e_2$ is parallel to $e_3$, all these traces start at the same vertical distance from $e_3$. For an edge $e'_2 \in E_2$ that is different from $e_2$, the traces start at a different vertical distance from $e_3$. However, we can make $|cd|$ significantly smaller than $|bc|$, such that the vertical distance between the starting points of all the traces, $\mu$, is much smaller than $\varepsilon_1/2$. That is, for a fixed edge $e_3 \in E_3$ and for each pair of edges, one from $E_1$ and another from $E_2$, the endpoints of all the traces will be very close to some line parallel to $e_3$.

The distance between each pair of edges in $E_3$ is $\varepsilon_1$. For each edge we get a cluster of traces as in Figure 4.4(e). In total, we will get $|E_3|$ clusters of traces.

Now we add points to $S$ inside circles centered at $a, b, c,$ and $d$. The radius of each such circle is $\varepsilon_2$, which is much smaller than $\varepsilon_1$. We call the created sets of points $A, B, C, D$; see Figure 4.4(d). (The cardinality of each of these sets is $n/4$.) Now, for every triple of edges $e_1 \in E_1, e_2 \in E_2, e_3 \in E_3$ and for every quadruple of points $a' \in A, b' \in B, c' \in C, d' \in D$, we will get the trace of $d'$. The new traces will be very close to the original ones, since the points in each set are very close to each other; see Figure 4.4(f).

As we have seen above, the endpoints of the traces in each cluster are close to some horizontal line. For each of the $|E_3|$ clusters of the traces we add to $P$ a horizontal edge that intersects all the traces of this cluster near their lower endpoint. We call this bundle of edges $E_4$; see Figure 4.4(f). In the figure the vertical lines are the
traces and the horizontal lines are the edges of \( E_4 \). As we shrink \( P \), \( d' \) moves down along its trace. The edges in \( E_4 \) move up. There is an edge \( e_4 \) that at some offset will meet \( d' \).

This example clearly exhibits \( \Omega(n^4 m^3) \) four-point stable poses.

For true offsets we get the same upper bound \( O(n^4 m^3) \) by using the same analysis. The only difference is that in this case the skeleton we use is \( VD(P) \), which contains both parabolic and straight arcs. However, this does not change the fact that each pair or triple of arcs intersects a constant number of times. Thus, the overall complexity of the three-dimensional map is still \( O(n^4 m^3) \).

For the lower bound we use the same example. Thus, we conclude that there are \( \Theta(n^4 m^3) \) four-point stable poses in the worst case for true offsets. \( \square \)

**Corollary 27** The complexity of a two-point translation-rotation-offset diagram of a simple \( m \)-gon and a set of \( n \) points is \( \Omega(n^2 m^2) \) in the worst case.

**Proof** Recall that a two-point translation-rotation-offset diagram describes all the poses of the polygon, which put two fixed points of the set on two fixed edges.

The bound follows from the example given in the proof of Theorem 26. In this example, for each pair of points \( c \in C, d \in D \), there is a pair of edges \( e_3 \in E_3, e_4 \in E_4 \), such that there are \( \Omega(n^2 m^2) \) four-point stable poses of \( P \) putting \( c \) and \( d \) on \( e_3 \) and \( e_4 \), respectively. Thus, the complexity of the diagram \( D_{(c,d),(e_3,e_4)} \) is \( \Omega(n^2 m^2) \). \( \square \)

Recall that in Section 2.2.3 we showed a matching upper bound. Therefore, as stated in Theorem 3, this bound is tight in the worst case.
Chapter 5

Conclusion

In this work, we address the Max-Cover, Containment, and Partial Containment problems of points by a polygon under translation, rotation, and offset (true or linearized). We present an algorithm that solves the Max-Cover problem in $\Theta(n^3m^3 \log(nm))$ time and $\Theta(nm)$ space in the worst case. For convex polygons we obtain $O(n^3m^2(m + \log(nm)))$ running time. For the Containment and Partial Containment problems we develop an algorithm that runs in $\Theta(n^4m^3 \log(nm))$ time and $\Theta(n^2m^2)$ space in the worst case. Both algorithms work for simple polygons, as well as for polygons with holes and sets of polygons. With only minor modifications they can also be applied to polygonal annuli.

We also develop an algorithm for the Containment and the Partial Containment problems under translation only. For simple polygons this algorithm runs in $O(n^3m^3 \log(nm))$ time and $O(nm)$ space. For convex polygons it runs in $O(n^3m^2(m + \log n))$ time and $\Theta(n + m)$ space.

We present the stable-poses algorithm that runs in $O(n^4m^3(n + m) \log m)$ time and $\Theta(n + m)$ space ($O(n + m \log m)$ space for true offsets). This algorithm is less time-efficient, but more space-saving, than its diagram-based counterpart.

The high time complexity of all the algorithms stems from the fact that they
compute all the stable poses for a given polygon and a set of points. With this strategy, our diagram-based algorithms are nearly optimal in the worst case, since there are $\Omega(n^3m^3)$ placements of a simple polygon putting three points on the boundary of the polygon, and $\Omega(n^4m^3)$ stable poses in the worst case. Note that the analysis is not tight for convex polygons. Using the convexity of a polygon is a major subject for future research.

Thus, it is clear that in order to achieve a faster algorithm, a different approach should be taken. That is, the algorithm should not enumerate all the stable poses. One such approach for the Containment and Partial Containment problems may be using the Voronoi diagram of the points, based on the offset distance function defined by the polygon [BDG01].

Other further research directions include:

- Computing approximate solutions for the Containment and Partial Containment problems, trading accuracy for running time;
- Generalizing polygons to smooth shapes; and
- Solving similar problems in higher dimensions.

---
1 Chew and Kedem [CK93] solved the problem of placing the largest homothetic copy of a given convex polygon $P$ inside an environment $Q$ of polygonal obstacles, which does not intersect any obstacle, under translation, rotation, and scaling. They used a Voronoi-diagram approach to achieve an algorithm whose running time is $O(n\lambda_3(n)m^4 \log n)$, where $n = |Q|$, $m = |P|$, and $\lambda_3 = \Theta(n^{2^{\alpha(n)}})$ is a nearly linear function of $n$ related to Davenport-Schinzel sequences ($\alpha(n)$ is the inverse Ackerman function, which is an extremely slow-growing function). Note that Chew and Kedem sought a free placement of the largest copy of a polygon, that is, a placement in which the polygon contains no points of $S$, whereas we look for the smallest offset covering all the points. However, a similar approach may be used. Nevertheless, in our case, we should build the furthest site Voronoi diagram for a convex or non-convex polygon offset distance function. In addition, Chew and Kedem's analysis is probably not tight, and the dependence on $m$ is really subquartic.
Bibliography


103


המייקומיים העיינים של הפוליגוןemi מייקומיים את אחדו התנאים הבנאים:

- יש שית קחדות מ- $P$ לע הנבול של $S$- במובם, $P$ עוצר היסס כף יוזר (באם
  אינפיניטסימלי) או מייקום קרב (כפי) עוצר שיתי הקחדות İzנאית על
  הנבול של $S$- במובם.

- יש שלוש קחדות מ- $P$ לע הנבול של $S$- במובם, $P$ עוצר היסס כף יוזר (באם
  אינפיניטסימלי) או מייקום קרב (כפי) עוצר שלוש הקחדות İzנאית על
  הנבול של $S$- במובם.

- יש ארבע קחדות מ- $P$ לע הנבול של $S$- במובם.

 الأميرים השחפים של ביוות ההכלה והזנויות ההחלקה היא אחדו המייקומיים ámbלי.

כדי להבער את הפתרון, האלגוריתמים מחשב את כל המייקומיים הזניבים של הפוליגון,
עובר כל אחוד מהו או סופר מחמ קחדות מ- $S$- במובםعيי $P$ מייקום דואו שישיי $S$-
יאיגיט יוזר מרשימה ביטאול, איבי פישט אחד ו userEmail יוזר ביטאול.

ניקוד

אני מרשמ הקשי ביניصيبויו הрактиולת בצ 이후מי המייקומיים הזניבים. אני מוכחי
הසמס דואו למספי המייקומיים הזניבים, וביי משליבת הסמס דואו על הסיבים של
טבלאות הוזה-יסיבוב-היסט וחוזה-סיבוב.

אני מכרן, אני מרשמ сынוג הרצות של האלגוריתמים המובסע וטבלאות הוזה-סיבוב-
היסט או ממית אופטימלי ביוות ההכלה והתכליה המקיב כל האלגוריתמים
שבוניים בצומן צי Literal בין כל המייקומיים האפקטיביים של הפוליגון.
טבלת הזהה-סיבוב-היסט

ביותם פולינונים P וקבוצת קודומים S, ואנו מציגים את טבלת הזהה-סיבוב-היסט, המכללה מעיד על כל המיקומים האפשריים של הפולינון החת סיבוב, הזהה, היסט, המשארים שית קודוד קבוצה זו S על שית יציעים קבוצה של P. קל לארוג שיפורים בупитьה התחלה והתחלה התחילה מנוגים באת התחלות האות狸.

teבלת הזהה-סיבוב-היסט היא ד-ימיתית, כ.Assert העור האנכי ייצג את היסט הציר האופקית את הסיבוב הזהה. כל קודה בבלבל מתאימה למיקום של עברו. כל קדה 1/5 Shrako בבלבל המהובב את היסטים של P המכלים את הקדה. אוור זה מ-1 Interpreter בוטל המימי עגثم את המיקומיים של P המכלים את הקדה, בנא" על השנים את ב степ יעיחסים של הזהה שיל אאורי התחלה.

על מצה תלות אם ב степени התלעה של הזהה ואלה בבלבל את הקדה הנמוכיה יונת (המייזון ואת היסטים המימיים) שביצועי בחיתוך של כל אオリ התלעה. על מצה תלות אם ב степени התלעה התחילה, שיש למס או התלעה התחילה, בנא" בטיות הנמצאות בחיתוך של k אオリ התחלה.

התח אופק של הבלבל מייצג את כל המיקומיים של הפולינונים התלעה סיבוב הזהה בלע (עומר הטיס ק PropertyChanged). בנת אנוי זהות מחסן את סיבובים הזהה-סיבוב. בעור זהה בבצלב את ינוי טל על רעיית הסיסים הביסטים שליד מפיקות ציוד בחתך המסר מכוסמי של אオリ התחלה. התח אופק של טבלת הזהה-סיבוב-היסט מייצג את כל המניקומיים של הפולינונים התלעה הזהה היסט.

אנו חוקרים את מבנה הטבלאות, מתוחכם את סיבוכיונים, מעריצים שית לביגה של הטבלאות. אם כן, אנו מחקרים את הטרנאות של הטבלאות עבורי שיפורים במעハードב של הפולינונים,แก קונ פולינונים קומריס פולינונים עサポート, üret ברעשת היסט פולינון- תכלתים.
פואת

$\delta$-טביעה היחס של פוליגון $P$ היינו אזור שטור הממלך את כל הקדחות ב跻 ה하였
$\delta$-olygon $P$ של כל היוצר מתבגרת עם היחס $\delta$. היחס פוני (התחנה, המחלקה) זו פוליגון הממודר עלי
הגבולה הגמיש (התחנה, המחלקה) של $\delta$-polygon $P$.

בעבודה זו נגשו לפתרון שלוש בעיות עקריות של כיסוי קבוצת נקודות גוזה
$P$esty היחס של בטיעות פוליגונאות$S$esty-

- בטיעות היחסים הפっきりים$\delta$- HCI: החוזק הערך של $\delta$ שיש לבלוט כמה
-Shower $\delta$- HCI $P$-

- בטיעות ההכללה: מציא את הערכי המינימליים$\delta$-
-$\delta$-HCI $P$-

- פיזור$\delta$-HCI $P$-

בעיות ההכללה שקוליםتدخلים מואדים לטיעיות ההכללה, אלא שאינה יש לוגוא
At היחסים המינימליים הפключи

kiye מגורות הרטואות שגות של התتوج עלי: הפוליגון יכל להיתו קמור, פשט כולם, וא
פולים עם חוה; תטרופסים התמדות יכללה לחוף או סיבוב החוד.

בלעיות כל קיימות שימישים בחומציות בני:רובוטיק (כים מיקום אוטומטי של
робוט), אריא מומנטשב (כים ניתן הבנייה, ידית איזון של תחולים ציר אוטומטי
וכו). בעיה.

מחקרים רבים מספר על אראריוואות שגות של הביוית העי.
ורב המחקריםisko
בפוליוגונים קומרו, תחדלת/הקטנה (scaling) $P$-
$P$-
מלא בקומות על-ידי בטיעת היחס. וראה,钮, ררב האפליקציות האמונות ושיקות
בפוליוגונים לא קומים בעלותים בשתייה. מתרבה כיו היחס הוא מידה נמוכה יותר
מארש מודל/קוטנה באפליקציות בקרה אינית של תחלי ציר אוטומטי.
3.5 механизмы взаимодействия с шлангами воздуха

3.6 Арбуз кодирования и арбуз шифровалки

4.1 Микомпенсирует взаимодействие с штатным кодированием

4.2 Микомпенсирует взаимодействие с штатным кодированием

4.3 Полюса P и конфликт P

4.4 Микомпенсирует взаимодействие с штатным кодированием
רשימת איורים

1.1 פילוגון היסט וטבעון היסט
1.2 פילוגון גורשים-הצטלעות שול.
1.3 פילוגון דידאראומת של Voronoi
2.1 משלוח מחטיך על טרי ירבם
2.2 הפילוגון P האלפסה
2.3 הפרמטרים \( \theta \) \& אט
2.4 טבלת לדגנאות על גורשים התכלת
2.5 \( \alpha_{ij} \) גורשים
2.6 \( \beta_{ij} \) גורשים
2.7 גורשים כדי להפריד שני אינ.Xrדות של גורשים התחלמות
2.8 \( \ell(e_2) \) \& \( \ell(e_1) \) גורשים יסודי על \( \ell(e_1) \)
2.9 \( \alpha_{22} \) \& \( \alpha_{11} \) גורשים
2.10 \( \alpha \) התכונת היסוד
2.11 התרבויות והימיקו האופטימלי
2.12 התוכן האופקיני של טבלת גורשים-סיבוב-הייסט הוסבב טבלת גורשים-סיבוב
2.13 התוכן האנכי של טבלת גורשים-סיבוב-הייסט הוסבב טבלת גורשים-הייסט
2.14 היסטים קונקורד
2.15 הפילוגון \( \hat{P} \)
2.16 קשור ארבע-חולינית
3.1 מיקומים יוצוב ואז יוצוב הכילולים קדקוד וצלג
3.2 מיקומים יוצוב ואז יוצוב הכילולים שא הקדקודים
3.3 מיקומים יוצוב ואז יוצוב הכילולים שא קדקוד
3.4 שלוש קונקורד על שלושה יריבים
רשימת טבלאות

1.1 בניית מכשוף מקסימלי (זרעון)
1.2 בניית מכשוף ליציון התוכנית עבירה/ה GdkAH/ডড
1.3 בניית מכשוף ליציון התוכנית עבירה/ה GdkAH/डड
1.4 בניית מכשוף ליציון התוכנית עבירה/ה GdkAH/डड
1.5 בניית מכשוף ליציון התוכנית עבירה/ה GdkAH/डड
3 מיקומים ציביים

3.1 הקדמה
3.1.1 מיקומים ציביים
3.1.2 היגשה
3.2 האלגוריתמים
3.2.1 סקירה
3.2.2 ענדו מוקד
3.2.3 שאלות נקודת
3.2.4 מציאת מיקומי ציבים של שני נקודות
3.2.5 מציאת מיקומי ציבים של שלוש נקודות
3.2.6 מציאת מיקומי ציבים של ארבע נקודות
3.2.7 סיבוכיות וסולם
3.3 הרחבת
3.3.1 היסטים ו-אלגוריתמים
3.3.2 היסטים במיתיה

4 מftar מיקומי ציבים
4.1 מיקומי ציבים של שני נקודות
4.2 מיקומי ציבים של שלוש נקודות
4.3 מיקומי ציבים של ארבע נקודות
# חפצים עילגיים

<table>
<thead>
<tr>
<th>מספר</th>
<th>השם</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>תכ Fetish</td>
</tr>
<tr>
<td>3</td>
<td>רשימות סמלים וקצורות</td>
</tr>
<tr>
<td>5</td>
<td>מובהק 1</td>
</tr>
<tr>
<td>6-8</td>
<td>מוטיבציה 1,1,2,3</td>
</tr>
<tr>
<td>11-12</td>
<td>מוזיאון הביצות 1.4, 1.5</td>
</tr>
<tr>
<td>16-17</td>
<td>קונגו 1.5.1</td>
</tr>
<tr>
<td>19-21</td>
<td>חכלת 1.5.2</td>
</tr>
<tr>
<td>22-25</td>
<td>חכלית 1.5.3</td>
</tr>
<tr>
<td>26</td>
<td>חצצוצות חבל 1.6</td>
</tr>
<tr>
<td>29-31</td>
<td>טבלת האודה-סיבוב-היסס 2, 2.1, 2.2, 2.2.1</td>
</tr>
</tbody>
</table>

vii
המתקר עשו בהנחיית פרופסור גל ברקת ופקולטה למודי החشهاد

ברצוני להדות לני ברקת על הנחייה והתמיינה

אני מודע לסכינתי על התמיכת החסיפה והתנדבות בחשטלומטיים
בעיות микוס של היסטים ובעיות מיקוס פוליגונליות

היבר על מחקר

לשים מילוי חלקי של הדיריגוט לקבלי התווח החוזר מונגסטר
למודיעי בולעי המለש

אלכס גוריאצ'ב

הווש להסת העכוני – מרכז טכנולוגיה לישראל

חיפה

אפולותו 2005

ספטמבר
בעיות מיקום של היסט וטביעה היסט פוליגונאליות

אלכס גוריאצ'יב