A WEIGHT-DISTRIBUTING PLACEMENT ALGORITHM FOR LARGE TEAMS OF LIFTING ROBOTS

by

Michael B. Doherty

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

MASTER OF COMPUTER SCIENCE

Carleton University
Ottawa, Ontario

Your file Votre référence
ISBN: 978-0-494-81619-6
Our file Notre référence
ISBN: 978-0-494-81619-6

Abstract

The vast majority of established work involving teams of lifting robots overlooks an important aspect of teamwork: sharing the weight evenly among the lifters. Distributing weight evenly among the robots ensures that no robots are overburdened while still allowing the minimum number of robots to be used.

This thesis develops an algorithm for determining placements of robots around a polygon such that the weight borne by each robot is near-equal. Our algorithm aims to maximize the stability of the lifted object by placing robots strategically to maximize the lifting base. By placing robots in teams of two directly across the object's centre of mass, we are able to ensure a stable lift with equal weight distribution. The initial algorithm, has a runtime of $O(mn)$ and determines robot placements for convex polygons. This algorithm is then extended to work for all simple polygons with a runtime increase to $O(n^4)$.

Acknowledgements

First and foremost, I'd like to thank my supervisor, Dr. Mark Lanthier, without whom this work would not have been possible. His guidance and input have proven invaluable. I'd also like to thank my friends and family for their ever-continuing support. Without them, I'd never have made it through this.

Table of Contents

List of Figures

Chapter 1

Introduction

The use of robots for object manipulation has been widespread for many years. In many instances, such as manufacturing environments, these robots follow a fixed set of commands and motions in order to perform the same action repeatedly. Other times, they are directly controlled by a human operator, following commands as they are input. Both of these situations are extremely limiting in their own ways. In the former, the robot is restricted to a preset action and is unable to adapt to new objects or environments, while the latter still requires the use of a human to control the robot as it works. The goal, then, is to combine the best aspects of both types (autonomy and versatility) to build robots which can adapt, learn and work on their own.

This is the motivation behind the field of robotic manipulation. In more recent years, a growing trend has been to move away from single, large robots toward teams of smaller robots, as they present a number of advantages, such as increased versatility in grasping configurations and mobility.

One of the most important aspects of robotic manipulation is the question of how an object should be "grasped". When dealing with a single robot, this usually means identifying how to get the best grip using a robotic hand or claw. When dealing with multiple robots, it usually entails determining how to place robots in relation to the object to push, pull or lift it.

The identification and selection of good lifting points provides the motivation for this thesis, which presents a novel approach to determining a placement for multiple robots attempting to lift an object.

1.1 Contributions

The main contribution of our work is an algorithm which is capable of distributing weight equally or near-equally amongst a large team of robots while attempting to

lift an object, based on knowledge of its shape and centre of mass. Additionally, our iterative algorithm seeks to stabilize the object as the robots lift it using different stability metrics which each affect the size and shape of the lifting base. Finally, our algorithm is not specific to any particular robotic hardware, making it versatile and implementable on a variety of platforms.

Our algorithm works for both objects that have an even mass-distribution and objects which do not. An object, for instance, which is constructed of a non-homogeneous material would have an uneven mass-distribution. Our algorithm runs in $O(nm)$ time for convex polygons with n vertices using m robots. For non-convex polygons, our algorithm runs in $O(n^4)$ time.

The most significant difference between work done in the past and the research presented in this thesis is that our work identifies specific points at which to place large (greater than three) teams of lifting robots. Previous work generally either works with small teams of three robots or less, pushing robots as opposed to lifting, or it does not determine specific points at which to place the robots, preferring to borrow from ideas such as swarm logic.

Our algorithm ensures first that all candidate lifting points result in a near-equal weight distribution, with variability determined by the implementation. After identifying the candidate points, placements are selected based on how they affect the size and shape of the lifting base using one of three different metrics. To the best of our knowledge no such methodology for lifting point selection has been used before.

Given knowledge of an object's weight and the lifting capabilities of the robots, the minimum number of robots required to lift the object can be determined. Our algorithm can work to use as few robots as necessary by ensuring that all robots lift near to their maximum capabilities.

1.2 Overview of Results

Our algorithm was implemented and a series of tests were conducted. These tests compared the results of our algorithm against those using two alternative methods for selecting lifting points. Comparisons against these alternative methods were made in the areas of weight distribution, stability and form-closure of the polygon. Our

algorithm performed superior to both alternatives in the area of weight-distribution, and comparatively in both stability and closure, despite neither alternative method suffering from the same weight-distributing restrictions that our algorithm requires.

1.3 Organization of Thesis

This thesis will begin by giving some background information in chapter 2, including necessary definitions, a brief explanation of the physics principles involved throughout the work, and a survey of the previous work done in the area of robotic manipulation.

Chapter 3 introduces the reader to our algorithm as it was designed for convex polygons. It will discuss the properties the algorithm aims to achieve and the computational model used to achieve these properties, followed by a detailed discussion of the algorithm itself with runtime analysis.

Chapter 4 will discuss what challenges are raised in order to extend the algorithm to non-convex polygons. After identifying these issues, the necessary changes to the algorithm will be discussed, followed by a runtime analysis of the modified algorithm.

Chapter 5 outlines the experiments used to test our algorithm against a series of polygons and the results are then broken down and analyzed to show how the algorithm achieves its goal.

Finally, chapter 6 discusses our conclusions and future work.

Chapter 2

Background

This chapter discusses the background material necessary to understand this thesis. After surveying previous work and discussing how our work differs from it in Section 2.1, Sections 2.2 and 2.3 will give definitions and an outline of the basic physics principles which form the basis of our algorithm.

2.1 Previous Work

There has been extensive work in the area of robotic manipulation in the last twenty years. The work in this area can be broken down into several categories which relate to our work: (i) the robotic model in use, (ii) grasp synthesis and analysis, and (iii) the computational model used. This section will discuss previous research broken down into these respective areas.

2.1.1 Robotic Model

While there are countless different models in use, they can nearly all be broken down into one of two categories: Robotic hands or graspers [5, 8, 21, 22, 26, 28, 29, 37, 47], and teams of robots which work in conjunction with one another [1, 10, 14, 16, 33, 40, 41, 42]. While our work deals solely with teams of robots, work in the former category addresses many of the same challenges involved in moving objects with multirobot teams such as where to grasp an object, stability and equilibrium, and in some cases, the force required to move it. The previous work in hand grasping and team grasping will now be explained.

Hand Grasping

A significant portion of the work in the area of robotic grasping has been directed specifically toward addressing the constraints of robotic hands. Parallel jaw grippers compose a significant portion of this research, due in large part to their simplicity and ease of use. Many other hand configurations are used in both experimental and practical settings, however, so there is some merit in the analysis of these more complicated configurations.

Parallel jaw grippers are a popular choice for robotic manipulators. They can be used to grip a wide variety of different objects, including all convex polyhedra [8]. Furthermore, several different classes of two-dimensional polygons have been proven to be able to be grasped with this type of gripper (e.g.: convex, free, and sail polygons) [8].

Most literature in this area models the grasped object as a k-dimensional polyhedron and the grasping fingers as points. Bendiksen and Hager [5], however, propose an alternative, sampling-based methodology, using projected edge point data as the input and modeling the fingers as line segments. Moreover, they chose to use a sampling method to compute a number of different potential grips, evaluating each according to a predefined metric, before attempting to physically grasp the object using a parallel jaw gripper. Using a number of assumptions (such as known mass of object, centre of mass and possible squeezing force of grippers, among others) the grasps are evaluated based on required squeezing force to keep the object in a state of equilibrium.

The GraspIt! system outlined in [26] describes an interactive grasping simulator designed to implement some of the work done in the area of robotic grasp synthesis over the last 30 years. Its designers created it to allow easy implementation of any robotic hand configuration, and the system already has several different hands preconfigured (including parallel jaw grippers, a Puma 560 arm and a "simplified Nomadics XR4000 mobile robot"). The system is designed to permit users to test potential grasping algorithms without the need to do so in the physical world.

An extensive survey of robotic hands and related algorithms is presented by Bicchi [6], wherein over 190 papers are reviewed. The foci in this survey are in three

different areas: manipulative dexterity, grasp robustness and human operability. The second of these three areas, which is the one most related to our work, is further broken down into discussions about the design of robotic hands for grasping, the properties of grasping (including form- and force-closure and stability metrics) and hand kinematics.

In his discussion on grasp properties, Bicchi discusses in relative detail the concepts of force-closure and form-closure, as both have received a fair amount of attention in the literature and are important concepts for grasping. A number of works are presented which cover both the synthesis and analysis of form-closure and force-closure grasps.

While our work is centred around the use of multirobot teams, research in robotic hands provides insight into choosing points to grasp and lift an object, as well as the constraints and challenges associated with the task.

Multirobot Teams

Most researchers in the area of multirobot teams focus on robots that collectively push/pull an object [10, 14, 33, 41, 42, 43, 46], rather than lift and carry it [2, 3, 17]. The former case is inherently easier to accomplish, since in the latter case multiple robots must grip an object before navigating as a highly coordinated group, all the while maintaining their hold of the object.

With the increased simplicity, pushing teams can focus less on the need to balance an object and more on the how to direct the object and coordinate the robots' movements in the plane.

Brown and Jennings [10] present arguably the simplest model. A simple two-robot system, wherein one robot provides the locomotive force required to move the object while the other steers it. The steering robot is the only one with knowledge of the path to be taken, and the pushing robot adjusts its direction to follow the object's orientation.

Rus, Donald and Jennings [33] present four different algorithms for using small teams of robots to manipulate objects via pushing. Each successive algorithm removes one further constraint. The first algorithm, labeled an "off-line, global control

protocol" has the robots sending and receiving signals from a global controller, and moving one at a time to push an object around. Two robots remain stationary while one moves at each step of the process. The remaining three algorithms remove the need for a i) global controller (by using communication directly between robots), ii) a planner for initial robot placements and movement schedules, and iii) synchronization between robots, respectively.

Sudsang et al. [40, 41, 42] take a different approach to manipulation; they use a team of three robots to immobilize an object, then determine the inescapable configuration space (ICS) before attempting to move it. The ICS determines the potential range of movement for the robots such that the object remains imprisoned between the three robots. By sequentially moving the robots, an object can be both rotated and translated in the plane, achieving the desired manipulation.

A novel approach to object manipulation is taken in [14] whereby a pair of robots encircle an object or group of objects with a rope. The rope can then be pulled to move the objects or "flossed" back and forth to rotate the objects.

Amadabadi and Eiji [1] address a notable issue when lifting three-dimensional objects: the tilt angle of the object can cause its projected centre of mass into the plane defined by the robots' contact points (this is known as the *zero moment point* or ZMP) to shift. This, in turn, can cause the object to become unstable and thus dropped. By determining the object's tilt angle as they are lifting, the robots are able to ensure the object's tilt stays within a given threshold [1].

Ghaderi and Ahmadabadi [16] propose two fault-clearance methods for small teams of lifting robots, neither of which require a reassessment of the robots' positions. The first method relies on redundancy where the weight borne by a faulty robot is redistributed to the others by simple nature of the orientation of the team. This is clearly limiting (especially in the example case presented in the paper of a team of four robots), and the authors thus propose a second fault-handling method of redistribution in which the zero moment point is moved by tilting the object away from the faulty robot. In this way, the mass of the object can be redistributed among the remaining non-faulty robots.

2.1.2 Grasp Synthesis

When attempting to lift or move an object, the first task that must be accomplished is determining where and how the robots should orient themselves or their hands in order to effectively grip the object. There are a number of key concepts that arise in the literature for grasp synthesis and analysis. The first and most prevalent is the need to constrain the movements of the object. Discussions of form-closure and force-closure abound [11, 12, 18, 20, 25, 27, 30, 31, 44, 48], and there are several algorithms for computing them [47]. Moreover, there are a number of different methods for evaluating what makes one grasp better than another [12, 20, 48].

In addition to providing a very clear explanation of both form-closure and force-closure, Bicchi [6] surveys the work and provides excellent examples of research in both the synthesis and analysis of grasps. Shimoga [36] compares and contrasts a number of older grasp generation algorithms which deal with specific hand configurations.

An often used alternative to form-closure and force-closure grasps in the area of multirobot teams is the use of caging [19, 40, 41, 42, 45] or object-closure [24], wherein a group of robots surround an object, leaving insufficient space between the robots to allow the object to escape the "cage".

Vongmasa and Sudsang [45] formalize and propose an algorithm for computing the *coverage diameter* of a polygon, which is the "longest possible distance between two points through which the polygon cannot pass". This can then be applied not only to individual objects, but to groups of objects that can then be transported as a whole. By calculating the smallest coverage diameter of a set of objects, the robots can then be placed such that the distance between neighbouring robots is less than this coverage diameter. If the robots then form a cage around the entire group of objects, they can move the entire group as a whole.

For teams of three robots, using their concept of the *inescapable configuration space*, the work of Sudsang et al. [40, 41] differs from the use of coverage diameters in that the relative orientation of the object with the robots is also maintained. This allows an object to be translated or rotated by three robots. They also discuss the further step of motion and path planning for the manipulation of the object [42].

Kumar and Wang [24] do not restrict their system to three robots, but rather

present an algorithm for computing the object-closure with an arbitrary number of robots. Their approach is to place the robots by using a "2D force vector" made up of two components: "a potential force from the object and attracting or repelling forces to/from other robots." The result is that the robots encircle the object and all separations between robot neighbour pairs will be identical. The robots can then move as a team, dragging the object along between them. The work demonstrates simulations done with four and with twenty robots.

Another important consideration in grasp synthesis is whether friction is to be taken into account. Some research does consider the forces due to friction [4, 5, 7, 20, 27, 30, 31, 34], though the work usually deals with robotic hands or grasp synthesis rather than multirobot systems. When dealing with multirobot systems, considerations of friction are necessary only in certain circumstances, such as when it is used to keep a carried object stationary [3, 4].

A final and notably relevant topic that is rarely discussed in the literature is the challenge of lifting an object with more than three robots due to the difficulty in maintaining planarity among more than three physical points. Ahmadabadi et al. [3] partially address this concern with a system designed to lift with four robots. In a manner similar to maintaining a small tilt angle with three robots in Ahmadabadi and Nakano's earlier paper [1], the robots either speed up or slow down their lifting based on their positions relative to the ZMP of the object.

2.1.3 Computational Model

There are two main facets of the computational model which need to be addressed: (i) the use of centralized vs. decentralized computation for teams of robots, and (ii) the evaluation of what makes a good grasp.

The first major issue to be addressed in the computational model is relevant only for multirobot teams: whether the computational work should be done in a centralized or distributed manner. Each has its merits, and while the latter has received much more attention in the literature in recent years [3, 2, 23, 39, 43], centralized approaches have not yet been abandoned completely [15, 42].

In the work of Ahmadabadi et al. [1, 2, 3], the decentralized approach is achieved

without any explicit communication between robots. To accomplish this, each robot looks at how the object being lifted is oriented in its own coordinate system. When the tilt angle of the object increases (decreases) too far, the robot increases (decreases) its lifting speed in order to level off the object. Stilwell and Bay [39] use a system in which a "leader" is elected from a group of homogeneous robots, and all others become followers. If the leader fails for any reason, a new leader is chosen and the system continues to function. Kube [23] implemented a system based on that of ants in which a group of robots move toward a box, then push it toward a lit goal. Robots repeatedly execute a series of queries and actions which result in the group slowly (and indirectly) pushing a box toward the goal. Sugar et al. [43] use a different leader-follower system than that of Stilwell and Bay. Their system has a single lead robot, and all other robots are followers. Note, however, that a follower robot need not necessarily follow the lead robot; they can, in fact, follow another follower robot. Each follower uses one of two position controllers to maintain the group arrangement: they maintain the desired relative angle and distance to a leading robot or they maintain the desired distance to two leading robots.

The work of both Sudsang et al. [42] and Erickson et al. [15] use knowledge of the shape of the object to be manipulated in order to compute where to place three robots. A centralized controller directs the robots to the proper locations, then directs them where to move in order to manipulate the object. Note that while Sudsang et al. [42] implemented a system involving robots and conducted physical experiments, Erickson et al. [15] implemented the algorithm and ran simulated tests to determine placement without actually running tests on physical robots.

Evaluating the quality of robotic grasps is a very subjective area and the method of doing so will depend largely on the desired end-goal. In many cases, especially in those where the desired property is of a binary nature (e.g., a grasp can be said to either achieve form-closure or not), the metric can simply be the algorithmic efficiency. In some cases, however, the goal is to minimize the forces required to retain a stable grip [25]. Cornella and Suarez [12] present two algorithms which give quantitative evaluations of a grasp based on the position and wrenches at the grasp points. Van Der Stappen [44] presents an algorithm to find all placements of fingers that achieve

form closure or second-order immobility, but makes no claims about the value of one grasp over another. Stansfield [38] reviews some of the more complex measures of grasp performance in his work.

2.1.4 Differences to Previous Work

The most glaring difference between work done in the past and the research presented in this thesis is in the use of large (greater than three) teams of robots. Ahmadabadi and Nakano [2] use four robots, but the system works with precisely four robots, and no more. Some work discusses the use of an arbitrary number of robots [4, 23, 24], but in those cases the work does not specify a grasp, but rather often borrows from swarm logic to determine how to cooperatively move an object. Our goal is to specify the points at which to place a larger number of robots in order to lift an object such that the weight distribution among the robots is equal or near-equal.

Since the primary goal is to equalize weight distribution, the algorithm uses this as its primary selection method for grasp points, and will not select grasping points that do not fall within an acceptable distribution. As our secondary goal is to produce a *stable* grasp, we chose to look at how much of the object lies outside of the convex hull defined by the grasping points. This will be discussed in depth in Section 3.2, but to our knowledge, no such metric has been used before.

Given that a part of the original motivation for this research was the use of non-specific robotic hardware, a centralized approach was adopted. Any implementing system would be able to use any robot type, as long as it meets several basic requirements (see Section 2.2).

The majority of robot teams work to push and/or pull an object along the ground [10, 14, 24, 41, 42, 45, 46] rather than lift it. Those that lift an object often rely on a specific hardware configuration [2, 3]. By designing an algorithm which identifies the points at which robots should lift, implementation is possible on any robotic system that meets very simple constraints.

2.2 Definitions and Assumptions

The following terms and symbols will be used throughout this thesis.

- H denotes the convex hull of the lifting points, and will be used throughout the later parts of this thesis. Any addition of downward vertical forces inside this area will not de-stabilize the object (assuming that the increased weight is not too much for the lifters to bear).

- Disturbing forces are those vertical forces which act outside of H. These have the potential to de-stabilize the object, if they are sufficiently large.

- Throughout this work, we will be using m as an indicator of the number of lifting robots being used.

Additionally, the following assumptions are made and used throughout this thesis:

- We use the variable n to indicate the number of vertices of a polygon, and the variable m to indicate the number of lifting robots used. Moreover, we make the assumption that $m \leq n$. Note that the minimum number of robots required to lift an object is determined by the object's weight and the lifting capabilities of the robots.

- We assume that all robots are able to lift simultaneously with a desired amount of force. Hardware restrictions are beyond the scope of this work.

- Our algorithm assumes that the object to be lifted has a planar polygonal base (e.g., see Figure 2.1). The majority of the work in this thesis discusses planar polygons in two dimensions, and in such cases refers to the base of the object. Note, however, that our algorithm is based on physical forces that occur in three dimensions, and there are therefore sections of this thesis that work in 3D space.

- Our algorithm assumes that each robot lifts the object at a single point on its boundary, with no accommodations for physical robot size.

- Since we make the assumption that all robots are able to lift simultaneously, we can also make the assumption that our object is "flat" because the projection of the object (and its centre of mass) in the plane of the lifting points remains constant.

- Our algorithm makes no assumptions about the material properties of the object. Depending on whether the object is constructed of a homogeneous or non-homogeneous material, the centre of mass may or may not be at the geometric centre of the object.

2.3 Mechanics

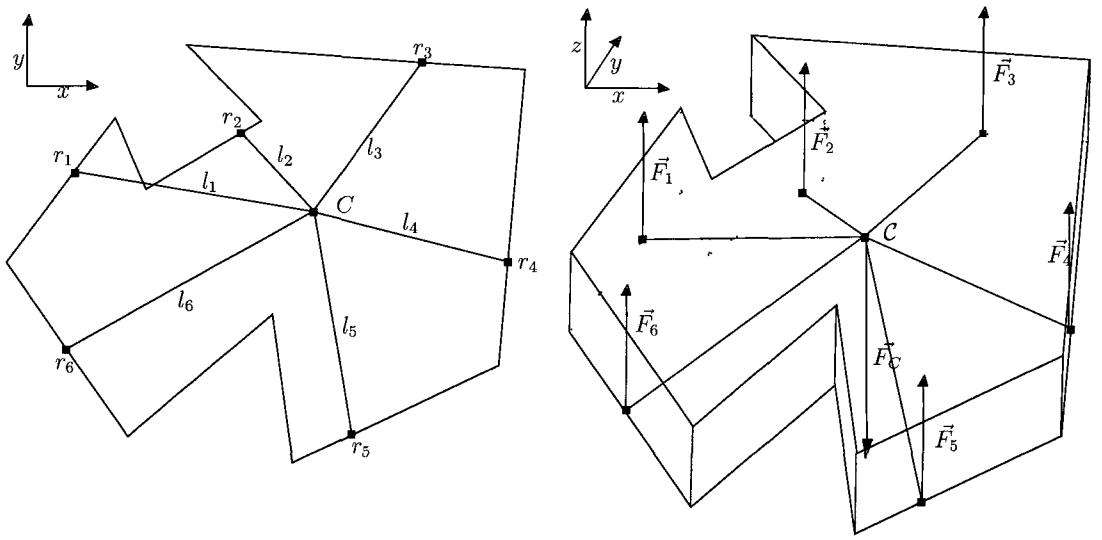


Figure 2.1: **a)** the polygonal base of an object, showing the centre of mass (C), robot lifting points (r_i), and vectors from C to r_i ($\vec{l_i}$). **b)** A 3-dimensional object represented by the base shown in Figure 2.1a. The forces ($\vec{F_C}$ and $\vec{F_i}$) acting on the object at the centre of mass and the robot lifting points are all in the positive or negative z direction.

Let P be a 2-dimensional polygon with n vertices, representing the base of an object, $\mathcal{O}$. Gravity exerts a force on $\mathcal{O}$ downward which is proportional to its mass (which is assumed to be non-zero). While in reality, gravity acts on each individual part of the object, for our purposes, it can assumed that this force acts at $\mathcal{O}$'s centre of mass projected into P at C [35]. Recall also that we assume that all robots lift synchronously, and that the base of our object will therefore remain horizontal, and we look at how the object moves vertically. It therefore makes no difference whether our reference frame is attached to our object or its environment.

Let $\vec{F_C}$ be the force exerted by gravity on P, acting at C. When broken into its component parts in the x, y and z directions, these components will be indicated by F_{Cx}, F_{Cy} and F_{Cz}.

$$\vec{F_C} = \begin{bmatrix} F_{Cx} \\ F_{Cy} \\ F_{Cz} \end{bmatrix}$$

The goal is to lift P using a number of robot lifting points, each of which is represented by a single point on the edge of P. Let $R = r_1, r_2, ..., r_m$ be the set of mobile robots. When expressed in vector-notation, $\vec{r_i}$, $1 \leq i \leq m$, indicates a vector from a cartesian origin to the point r_i.

$$\vec{r_i} = \begin{bmatrix} r_{ix} \\ r_{iy} \\ r_{iz} \end{bmatrix}, 1 \leq i \leq m$$

The forces exerted by the robots are in the opposite direction to $\vec{F_C}$ and will be denoted by $\vec{F_1}, \vec{F_2}, ..., \vec{F_m}$, for robots $r_1, r_2, ..., r_m$ respectively. As in the case with $\vec{F_C}$,

$$\vec{F_i} = \begin{bmatrix} F_{ix} \\ F_{iy} \\ F_{iz} \end{bmatrix}, 1 \leq i \leq m$$

All the lifting done by the robots is subject to the basic laws of mechanics. While the actions are not complex, and thus a number of generalizations and simplifications can be made, a basic understanding of the fundamental physical laws and rules is required.

2.3.1 Newton's Laws

All interactions are subject to Newton's three laws, which form the foundation of mechanics and are used to predict how objects will perform under the influence of various forces. The following two laws relate to our work:

Law 2.1 *An object at rest will remain at rest and an object in motion will continue in motion with a constant velocity unless it experiences a net external force or resultant force [35].*

Law 2.2 *If object A interacts with object B, the force exerted on A by B is equal to and opposite the force exerted on B by A (or more commonly known as "Every action has an equal and opposite reaction") [35].*

Newton's laws give rise to our first rule of equilibrium.

Definition 2.1 *An object is in a state of equilibrium when it is stationary and the sum of all the forces acting on it is zero.*

Given an object $\mathcal{O}$ with l forces acting upon it $(\vec{F_1}, \vec{F_2}, ..., \vec{F_l})$ $\mathcal{O}$ is in equilibrium if

$$\sum_{i=1}^{l} \vec{F_i} = 0$$

For our purposes, these forces are those exerted by the robots and that of gravity. Thus, for a polygon P being lifted up by m robots, P is in equilibrium if

$$\vec{F_G} + \sum_{i=1}^{m} \vec{F_i} = 0 \tag{2.1}$$

Intuitively, if the sum of the forces from the robots is greater than that of gravity, the object will rise. Conversely, if the sum of the forces from the robots is less than that of gravity, the object will fall.

2.3.2 Torque

Another aspect of mechanics which is critical for our analysis is the idea of torque. Torque is the measure of the tendency of a body to rotate about a fixed axis under the influence of a force and is often denoted by $\vec{\tau}$. Let p be the point about which an object $\mathcal{O}$ pivots. Let $\vec{F}$ be the force applied at some point f on $\mathcal{O}$. Let $\vec{r}$ be the vector from p to f. The torque $\vec{\tau}$ is computed by using the following cross-product (Figure 2.2).

$$\vec{\tau} = \vec{r} \times \vec{F} \tag{2.2}$$

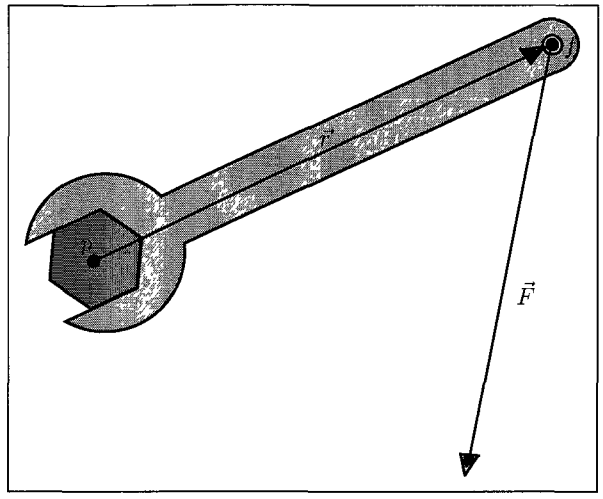


Figure 2.2: Torque is computed by taking $\vec{r} \times \vec{F}$

It is important to note that torque is a vector quantity. Moreover, it is always normal to the plane defined by the vectors $\vec{r}$ and $\vec{F}$. The value of Equation 2.2 can be computed in a number of ways, but for simplicity the following component-based methodology is used:

Let $\hat{i}$, $\hat{j}$ and $\hat{k}$ be unit-vectors in the x, y and z directions, respectively. Then,

$$\vec{\tau} = det \begin{bmatrix} \hat{i} & \hat{j} & \hat{k} \\ r_x & r_y & r_z \\ F_x & F_y & F_z \end{bmatrix}$$

The result of this determinant is then

$$\begin{aligned} \vec{\tau} &= \hat{i} r_y F_z - \hat{i} r_z F_y + \hat{j} r_z F_x - \hat{j} r_x F_z + \hat{k} r_x F_y - \hat{k} r_y F_x \\ &= \begin{bmatrix} r_y F_z - r_z F_y \\ r_z F_x - r_x F_z \\ r_x F_y - r_y F_x \end{bmatrix} \end{aligned} \tag{2.3}$$

or, in cartesian coordinates,

$$\vec{\tau} = ((r_y F_z - r_z F_y), (r_z F_x - r_x F_z), (r_x F_y - r_y F_x))$$

Our definition of equilibrium can now be extended:

Definition 2.2 *An object $\mathcal{O}$ with forces $\vec{F_1}, \vec{F_2}, ..., \vec{F_l}$ resulting in torques $\vec{\tau_1}, \vec{\tau_2}, ..., \vec{\tau_l}$, respectively, is in a state of equilibrium if the following hold true:*

1. *$\mathcal{O}$ is stationary,*

2. *The sum of all forces acting upon it is equal to zero,*

$$\sum_{i=1}^{l} \vec{F_i} = 0 \tag{2.4}$$

3. *The sum of all torques acting upon it is equal to zero,*

$$\sum_{i=1}^{l} \vec{\tau_i} = 0 \tag{2.5}$$

Equation (2.1) holds, and there is now an additional requirement for our robotic-lifting scenario: Let τ_C be the torque applied at C due to $\vec{F_C}$. Therefore, to ensure equilibrium:

$$\tau_C + \sum_{i=1}^{m} \tau_i = 0 \tag{2.6}$$

In all torque calculations, two vectors are required: the force-vector, and a vector from the pivot point to the point where the force is applied (known as the *moment arm*).

In our robotic-lifting scenario, we do not have a fixed axis of rotation from which to define the moment arm. Using standard $2D$ or $3D$ cartesian coordinates, it can be defined to extend from the cartesian origin to the point at which the force is applied.

Moreover, the origin can be defined to be any arbitrary point, since intuitively, the total torque about any point in the system must be equal to zero in order to achieve equilibrium. If the net torque acting on our object about any point is non-zero, the entire object would then rotate about that point.

Since the coordinate system is able to be defined arbitrarily without loss of generality, from this point forward (unless otherwise specified) it is assumed that the z-axis is always oriented such that the force due to gravity is parallel to, and in the same

direction as the negative z-axis (see Figure 2.1b). We also make the assumptions that the object being lifted is not weightless and that all robots are lifting the object (and not pushing down on it).

Chapter 3

Convex Polygons

This chapter discusses our work as it relates to convex polygons. Section 3.1 discusses the properties that we want to look for in our grasps. Section 3.2 discusses how our algorithm will account for these desired properties. A discussion and analysis of the algorithm's time complexity takes place in Sections 3.3 and 3.4.

3.1 Discussion of Desired Lift Properties

This section discusses the lift properties that our algorithm needs to address. The main goal of our work is to obtain a near-equal weight distribution among the robots. The secondary goal is to produce as "stable" a lift as possible. Finally, while our algorithm does not directly address immobilization of the object, it is a desirable property to have, and therefore will be discussed and examined.

3.1.1 Weight Distribution

Weight distribution among the robots is the primary goal of our algorithm. The vast majority of the papers discussed and cited throughout this work in which robots lift objects used no more than three robots. In fact, the only work in which more than three are explicitly used is that of Ahmadabadi et. al [3], wherein the authors look only at the angle of the object being lifted, and make no considerations for the weight of the object.

This work considers only the vertical components of the forces acting on the object. As such, the system of equations shown in Section 2.3 results in three equations once broken into its component parts: the z-component of Equation 2.4 (which has no x- or y-components), and the x- and y-components of Equation 2.5 (which has no z-component). The portion of the weight that is being supported by each of three lifting robots can be uniquely determined due to the system of three equations and three

unknowns (the force being lifted by each of the three robots). If, however, additional robots are added, the system becomes indeterminate, and it is therefore significantly more difficult to determine what portion of the weight is supported at each point. While methods exist for calculating the forces at multiple (i.e., greater than three) supports for an object, such as Castigliano's Theorem [32], the calculations involved are far from trivial and rely on advanced knowledge of material properties, such as elasticity of the object.

Our algorithm, however, will not be determining the weight for each of three robots, but rather for pairs of robots. Ignoring for the moment the idea of lateral stability, two sufficiently strong robots suffice to lift an object, as long as they lie directly opposite one another from the centre of mass.

Theorem 3.1.1 *Let P be a polygon with centre of mass C. Given two lift points r_1 and r_2 on the convex hull of P, P can be lifted in a state of equilibrium if and only if* $\angle r_1 C r_2 = 180°$

Proof Assume C is not on $\overline{r_1 r_2}$. Let p be a point on $\overline{r_1 r_2}$ such that $\overline{pC}$ is perpendicular to $\overline{r_1 r_2}$ (Figure 3.1). Without loss of generality, position and orient coordinate axes such that $p = (0, 0, 0)$ and the y-axis is parallel to $\overline{r_1 r_2}$.

From Equation 2.4, if P is in equilibrium then the following force equation must hold:

$$\vec{F_1} + \vec{F_2} + \vec{F_C} = 0$$

Additionally, the following torque equation based on Equation 2.5 must also hold:

$$(\vec{r_1} \times \vec{F_1}) + (\vec{r_2} \times \vec{F_2}) + (\vec{C} \times \vec{F_C}) = 0 \tag{3.1}$$

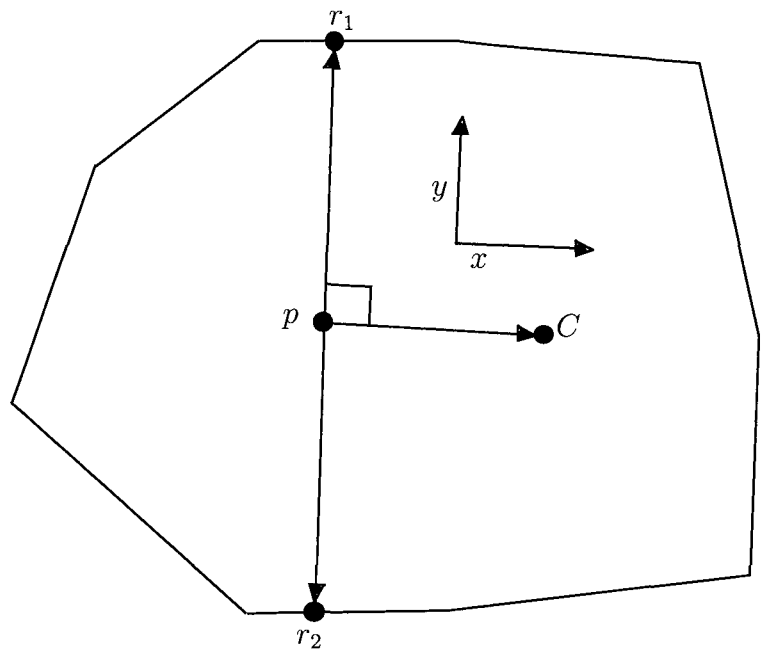


Figure 3.1: Two lifters attempting to lift a polygon at the unique points r_1 and r_2. The polygon's centre of mass, C, does not lie directly between the two lifting points.

From the choice of origin and orientation of axes,

$$\vec{r_i} = \begin{bmatrix} 0 \\ r_{iy} \\ 0 \end{bmatrix} \quad ;i\epsilon\{1,2\}$$

$$\vec{C} = \begin{bmatrix} C_x \\ 0 \\ 0 \end{bmatrix}$$

$$\vec{F_i} = \begin{bmatrix} 0 \\ 0 \\ F_{iz} \end{bmatrix} \quad ;i\epsilon\{1,2,C\}$$

Substituting into equation (2.3) and using $\hat{i}$ and $\hat{j}$ as unit vectors in the x- and

y-directions, respectively,

$$\begin{aligned} \vec{r_1} \times \vec{F_1} = &\quad \hat{i}(r_{1y}F_{1z}) \\ \vec{r_2} \times \vec{F_2} = &\quad \hat{i}(r_{2y}F_{2z}) \\ \vec{C} \times \vec{F_C} = &\quad \hat{j}(C_xF_{Cz}) \end{aligned}$$

Substituting back into Equation 2.5 gives

$$\hat{i}(r_{1y}F_{1z}) + \hat{i}(r_{2y}F_{2z}) - \hat{j}(C_xF_{Cz}) \tag{3.2}$$

Since $C_x \neq 0$ and $F_{Cz} \neq 0$, the final term, $\hat{j}(C_xF_{Cz})$, cannot be zero. Since this is the only term which has a component in the y-direction, the resulting vector cannot be zero, which contradicts Equation 3.1. P cannot, therefore, be lifted in equilibrium if C is not on $\overline{r_1r_2}$.

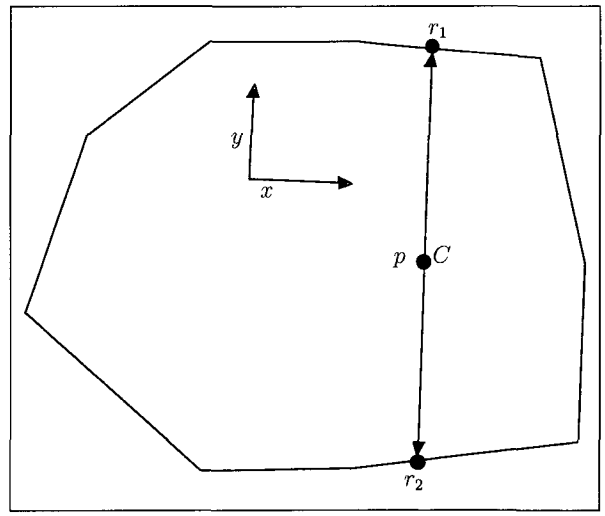


Figure 3.2: The point p lies at P's centre of mass, C.

Assume now that C lies on $\overline{r_1r_2}$. p is therefore at C, and $\vec{C} = 0$ (Figure 3.2). Therefore,

$$\vec{C} \times \vec{F_C} = 0$$

Since $\vec{r_1}$ and $\vec{r_2}$ have opposite directions,

$$\hat{i}(r_{1y}F_{1z}) + \hat{i}(r_{2y}F_{2z}) = 0 \tag{3.3}$$

is valid for all

$$\frac{r_{1y}}{r_{2y}} = \frac{F_{2z}}{F_{1z}} \tag{3.4}$$

and it is therefore possible for an object to be in equilibrium with two lifters placed opposite one another to the centre of mass. ∎

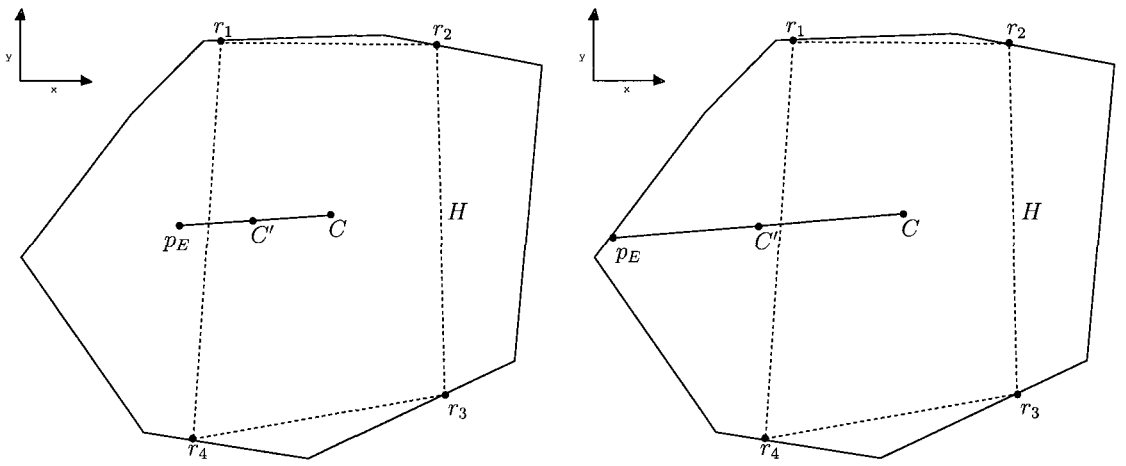
Consider for a moment a square table with an even weight distribution (i.e., the centre of mass lies in the geometric centre of the table) being lifted by four robots, such that robot r_1 and robot r_3 are on opposite diagonal corners, and robots r_2 and r_4 are on opposite diagonal corners. Theorem 3.1.1 shows that r_1 and r_3 can lift the table with no contribution from r_2 or r_4, or vice versa. Moreover, all four robots can apply *some* force and still have the table suspended stably. More specifically, the table can be stably suspended if r_1 and r_3 apply equal force, r_2 and r_4 apply equal force, and the total force is equal to the weight of the table. Therefore, as long as the robots apply vertical force such that the definition of equilibrium (Section 2.2) is satisfied, the object can be stably held with equal weight distribution.

3.1.2 Stability

Stability is a rather vague term often defined in different ways depending on the requirements of the situation. In our context, we want to minimize the likelihood that the object being lifted will be upset by small disturbances. Such disturbances can be modeled as forces applied at different points on the polygon. In our discussion of stability, we look only at vertical components of forces applied throughout the polygon, and want to ensure that they won't upset the equilibrium of the polygon while it is lifted. The goal, therefore, is to minimize the likelihood that an external force applied downward at an arbitrary point on the surface of the object is going to cause the object to tip.

By applying a force to the surface of the object, a non-uniform object density is created, thereby moving the centre of mass of the object to some point between the original centre of mass C and the point p_E at which the disturbing force is applied (see Figure 3.3). Precisely where between C and p_E the new centre of mass C' lies depends on the relative sizes of the external force F_E and the force due to gravity F_C

acting on the object. The goal is to ensure that C' does not lie outside the convex hull, H, determined by the lifting robots.



(a) (b)

Figure 3.3: A polygon whose original centre of mass, C, has been effectively moved to C' by an external downward force at p_E. (a) C' lies inside of H. (b) C' moves outside of H.

Figure 3.3 shows an example of an external force at p_E altering C. In 3.3a, the new centre of mass C' lies inside H. While the weights at each lifting point r_1 through r_4 will be altered in this instance, the object will not tip as long as the new weights do not exceed the capabilities of the robots. Figure 3.3b shows a case where C has moved outside of H. In this instance, regardless of the strength of the robots, the object will tip about the axis defined by r_1 and r_4 due to the force at C' causing a torque about that axis.

Since it is assumed that this disturbing force is applied at an arbitrary location on the surface of the object, with all points being equally likely targets, a logical goal is to enclose as much of the polygon as possible within H. Moreover, since the change in the centre of mass is dependent not only on the size of the disturbing force, but also its distance from the current centre of mass (see Figure 3.3), a second consideration is to minimize the distance from H to any point in the polygon which lies outside of H.

Therefore, the two quantities which will affect our evaluation of grasps will be: (1) the portion of the polygon's surface area which is inside of H, and (2) the distance from the vertices of the polygon to H.

3.1.3 Immobilization

The final property to be discussed is that of immobilization of the polygon (restricting its ability to move in the horizontal plane, without the robots penetrating its boundary). While our algorithm does not specifically aim to accomplish this, we will consider how well the robots can immobilize an object as a side-effect. Object immobilization (in the general sense) is a topic that has been discussed at length in the literature, albeit under the guise of a number of different specific requirements. Form-closure, force-closure and caging all address this topic in one form or another.

Force-closure means that the motions of the grasped object are restrained by virtue of large contact forces that the robots exert on the object. Since our algorithm is designed to apply only lifting forces (and not horizontal constraining forces), force-closure is a property that is far more constrained than we require. Indeed, it would require specific robotic hardware to ensure that it is attainable for a given grasp. Moreover, force-closure would require that the robots be able to exert potentially high horizontal forces in order to generate high enough friction forces to counter any external wrenches.

Caging, or positioning the robots near enough to one another that the object is too large to pass between them, is insufficiently restrictive. Despite its benefits, such as not requiring synchronicity amongst the robots and allowing the object to move inside the cage of lifters, there are a number of issues that would arise with the use of caging. First and foremost, if the object were to move within the cage, its centre of mass would move to a new location relative to the robots, thus redistributing the weight and requiring a recalculation of lifting points. Additionally, in order to be applicable to lifting robots, it would be necessary to limit the maximum distance the robots can deviate from the perimeter of the object to the length of the robots' respective arm lengths, which would differ for each type of robotic hardware.

The application of immobilization discussed in van der Stappen's work [44] is the

most ideal for our purposes. In principle, the resulting grasp is one of form-closure, wherein the robots are positioned such that any movement of the object requires one or more of the robots to penetrate the boundary of the object. Since it does not rely on frictional forces, but rather only on "unilateral, frictionless contact constraints", it can be calculated using only knowledge of the robot placements and object geometry.

3.2 Computational Model

Now that the desired properties have been explained, it remains to discuss how these properties are to be achieved. With respect to weight distribution, there are several key concepts that play a large part in our algorithm. First and foremost, it is necessary to ensure that each team (i.e., pair of robots) has its members placed directly across the centre of mass from one another. Since all robots are to be placed in teams of two, from Theorem 3.1.1 we know that placing them otherwise would not enable that team to be in equilibrium with their portion of the object's mass.

Equation 3.4 from Theorem 3.1.1 further shows that in order to distribute weight evenly between two lifters, they must be equidistant from the centre of mass. We must therefore ensure that such a placement is possible in order to properly distribute weight. To accomplish this, we look at edges of the polygon as lines in the plane, and show the following:

Theorem 3.2.1 *Given two non-parallel lines, L_1 and L_2, and a point p which lies between them, there exists a line segment l, connecting L_1 and L_2 through p, such that the portion of l from L_1 to p and the portion from L_2 to p are of equal length.*

Proof Assume there exist two non-parallel lines, L_1 and L_2, in the plane, with normals n_1 and n_2, and a point p which lies between them (figure 3.4). Assume further that there are two lifting points, r_1 and r_2, on L_1 and L_2, connected by a line segment l which passes through p. Let d_1 and d_2 be the perpendicular distances from p to L_1 and from p to L_2, respectively, θ be the angle between n_1 and l, and ϕ be the angle between n_1 and n_2 (figure 3.4). Without loss of generality, assume L_1 has a slope$= 0$ and ϕ has a range of $(0, \pi)$. Then

$$|\overline{r_1p}| = \frac{d_1}{Cos(\theta)}$$
$$|\overline{r_2p}| = \frac{d_2}{Cos(\theta + \phi)}$$

Therefore,

$$\lim_{\theta \to \pm\frac{\pi}{2}} |\overline{r_1p}| = \infty$$
$$\lim_{\theta \to \pm\frac{\pi}{2} - \phi} |\overline{r_2p}| = \infty$$

Which implies that

$$\theta \epsilon (-\frac{\pi}{2}, \frac{\pi}{2} - \phi)$$

As θ approaches $-\frac{\pi}{2}$, the length of $\overline{r_1p}$ approaches infinity while the length of $\overline{r_2p}$ approaches a fixed value, indicating that $|\overline{r_1p}| > |\overline{r_2p}|$. As θ approaches $\frac{\pi}{2} - \phi$, the length of $\overline{r_2p}$ approaches infinity while the length of $\overline{r_1p}$ approaches a fixed value, indicating that $|\overline{r_2p}| > |\overline{r_1p}|$. Since both functions are continuous on this interval, by the mean value theorem, they must cross at some point. At this point, their lengths must be equal. ∎

Looking at the value of $\frac{|\overline{r_1p}|}{|\overline{r_2p}|}$, gives the function

$$f(\theta) = \frac{d_1 Cos(\theta + \phi)}{d_2 Cos(\theta)} \tag{3.5}$$

whose derivative is

$$f'(\theta) = \frac{d_1 Sin(\theta + \phi)Cos(\theta) + d_1 Sin(\theta)Cos(\theta + \phi)}{d_2 Cos^2(\theta)}$$
$$= -\frac{d_1 Sin(\phi)}{d_2 Cos^2(\theta)}$$

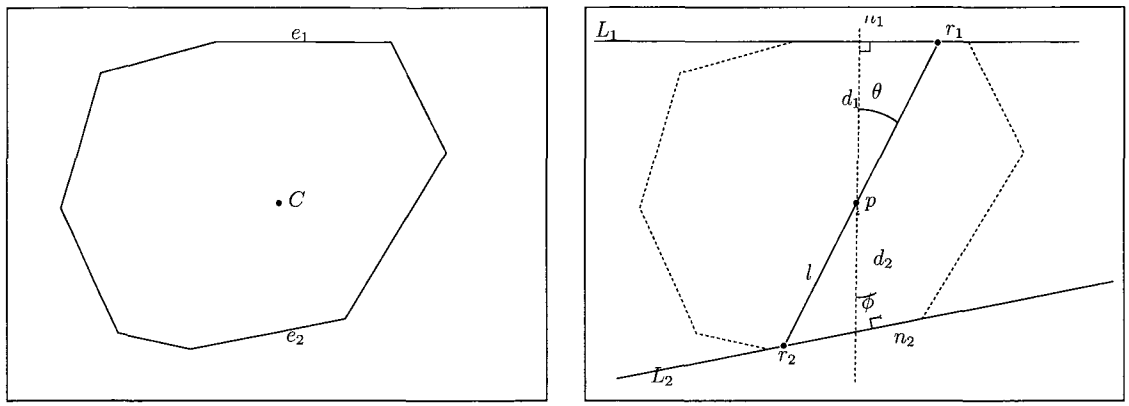


Figure 3.4: Finding an even weight distribution on two edges of a polygon (e_1 and e_2). We treat e_1 and e_2 as lines in the plane. The centre of mass is the point p across which the robots must be placed.

In the range $\left(-\frac{\pi}{2}, \frac{\pi}{2} - \phi\right)$, $Cos^2(\theta)$ is positive. Additionally, for any given pair of lines, ϕ, d_1 and d_2 are all constant. Therefore, $f'(\theta)$ is either always positive or always negative for all θ on $\left(-\frac{\pi}{2}, \frac{\pi}{2} - \phi\right)$, depending on the values of ϕ, d_1 and d_2. $f(\theta)$ is therefore either continuously increasing or continuously decreasing on the given range, indicating that there is exactly one value of θ for which $|\overline{r_1 p}| = |\overline{r_2 p}|$, at

$$\theta = Tan^{-1}\left(-\frac{d_2 - d_1 Cos(\phi)}{d_1 Sin(\phi)}\right) \tag{3.6}$$

Note the condition in Theorem 3.2.1 that L_1 and L_2 be non-parallel. If L_1 and L_2 are parallel, then the value of ϕ is zero, and Equation 3.6 becomes constant, implying that either $|\overline{r_1 p}| = |\overline{r_2 p}|$ for all θ, or $|\overline{r_1 p}| \neq |\overline{r_2 p}|$ for all θ.

Assume that L_1 is collinear with an edge e_1, and L_2 is collinear with an edge e_2 of P. Clearly, the edges e_1 and e_2 of the polygon are not lines, but rather line segments. Therefore, it is entirely possible that one or both of the points r_1 and r_2 for which $|\overline{r_1 p}| = |\overline{r_2 p}|$ lie beyond the endpoints of their respective edges. The monotonicity of Equation 3.5 indicates that if this is the case, then the closer θ can get to the value indicated in Equation 3.6, while keeping both r_1 and r_2 within e_1 and e_2, the closer this ratio will be to 1. In the case where both edges of the polygon are parallel, if the ratio of d_1:d_2 is within Δ, then the midpoint of the edges can be used in order to

increase the likelihood that good coverage is obtained.

All of the above result two key factors in the design of our algorithm:

1. It can be determined, given two edges of the polygon and the centre of mass, whether there exists a placement of two lifting robots, one on each edge, such that each robot bears an equal amount of the weight.

2. If such a placement doesn't exist, then the most even weight distribution exists at an endpoint of the edge. Whichever endpoint is closer to the position which would result in equal weight distribution is the nearest to equal.

While it is certainly possible for a given polygon to have many positions of equal weight distribution, each polygon is only guaranteed to have one:

Theorem 3.2.2 *Given a convex polygon P with centre of mass C, there exists at least one chord of P, $\overline{r_1 r_2}$ such that C bisects $\overline{r_1 r_2}$.*

Proof Let $f(\theta) = |\overline{Cr_1}|$ as $\overline{r_1 r_2}$ rotates about C. Let $g(\theta) = |\overline{Cr_2}|$ as $\overline{r_1 r_2}$ rotates about C. Since P is polygonal, and is therefore made up of line segments, both $f(\theta)$ and $g(\theta)$ are piecewise and continuous. Since P is convex and has a continuous boundary, $f(\theta) = g(\theta+\pi)$. Therefore, by the mean value theorem, $f(\theta)$ and $g(\theta)$ must cross somewhere on the range $[0, \pi)$. At this point, $f = g$, and therefore $|\overline{Cr_1}| = |\overline{Cr_2}|$. ∎

Since we can guarantee only one team placement which distributes weight evenly, an algorithm which relies only on completely even weight distribution may only be able to identify two lifting points. To that end, our algorithm allows for a definable parameter Δ which allows non-equal weight distributions, but keeps the difference in weight below this value. If a placement for equal weight distribution doesn't exist for a given pair of edges, but the edge endpoint closest to it gives a distribution that is within Δ, then it (and its corresponding point across the centre of mass) are included as candidates instead.

For perfectly even weight distribution throughout the robots, Δ is set to zero, and for a weight distribution that makes no consideration for equality, Δ is set to 1.

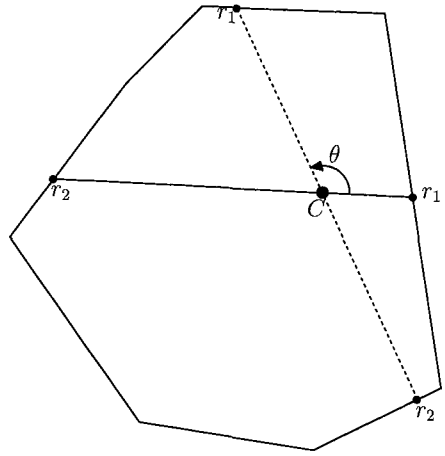


Figure 3.5: The chord r_1r_2 rotates about the centre of mass C. Somewhere on the range $\theta \in [0, \pi)$ there exists a chord $\overline{r_1r_2}$ such that $|\overline{Cr_1}| = |\overline{Cr_2}|$.

In order to generate a stable lift, the impact of disturbing forces must be minimized. This means two things need to be taken into consideration: the size of the area in which they can act (i.e., how much of the polygon lies outside of H) and how far they can act from H. The former is fairly self-explanatory; as H covers more of the polygon's area, the likelihood of an additional arbitrary vertical force becoming a disturbing force decreases. The latter part relates back to Section 2.3.2: as the distance from the edge of H to the disturbing force increases, the torque about that edge of H increases. This results in an increasing effect as this distance increases.

We look at multiple metrics in our algorithm, and will discuss comparisons in Chapter 5. The first metric looks only at the area covered by H when choosing successive placements of robots, while a second looks at the potential distance that disturbing forces can act from H. Each of these address one aspect of stability, and while the latter is more thorough than the former, neither one perfectly encompasses the other. For that reason, we introduce a third metric, which makes use of inscribed circles to help us place robots.

The *area metric* chooses lifting points based on which would make the largest area contribution to H. At each iteration candidate points are examined and the one which results in the largest increase in size to H is selected (Figure 3.6a). The *distance metric* is slightly more robust; rather than look solely at coverage of the polygon by H,

it seeks to minimize the possible torque generated by disturbing forces. The furthest point from any given edge of H must lie at a vertex of the polygon. Therefore, at each iteration and for each candidate lifting point, the distances from vertices to H', the set of points defining H with the inclusion of the candidate points, are examined (Figure 3.7). The candidate point that makes the overall largest impact is then selected.

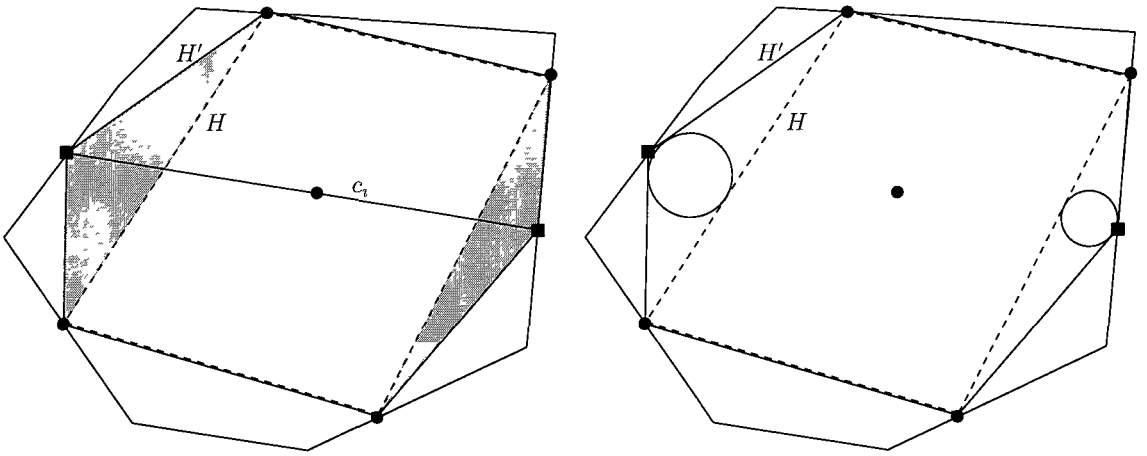
The *circle metric* is an attempt to account for both area and distance to some degree. Each candidate lifting point forms a triangle with the nearest edge of H. For each candidate point, a circle is inscribed within this triangle (Figure 3.6b). Whichever candidate point results in the largest circle is the one selected as the next lifting point. Intuitively, this should result in "fatter" triangles being added to extend the convex hull.

3.3 Algorithm Description

Several key definitions must be introduced before outlining and describing our algorithm. A *vertex-chord* is defined as any chord of the polygon which originates at a vertex and passes through the centre of mass (Figure 3.8). *Edge-pairs* consist of linear sections of the polygon's perimeter that lie directly across the centre of mass from one another. As a vertex-chord rotates about the centre of mass in either direction, it will reach the end of one of the polygon edges (Figure 3.9). An edge-pair contains the two sections of the polygon's perimeter that lie between neighbouring vertex-chords.

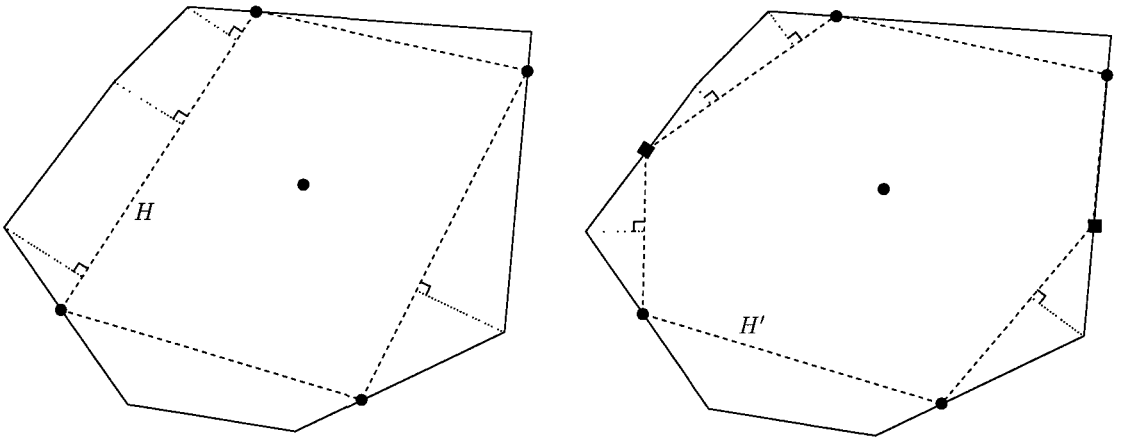
The algorithm maintains two lists of chords: the *candidate chords* (*cChords*) are those chords of the polygon whose endpoints represent a potential placement for a pair of robots. These chords always pass through the centre of mass of the polygon, and, as described in Section 3.2, the ratio of the two segments of the chord lying on either side of the centre of mass is close to 1:1. The candidate chords, therefore, are all those which satisfy the constraints for robot placement. The list of candidate chords is maintained in ascending order of slope.

The list of *selected chords* (*sChords*) is also maintained. Once a candidate chord has been selected by one of the three metrics (i.e., *areaMetric()*, *distanceMetric()*, or *circleMetric()*), it is moved from the *cChords* to *sChords*. Each iteration of the



(a) (b)

Figure 3.6: The candidate lifting points are represented by squares, and H and H' by dashed and solid lines, respectively. a)Area metric: the shaded regions represent the increase to H. b) Circle metric: the sizes of the inscribed circles are used to determine which candidate points will be used.



(a) (b)

Figure 3.7: The candidate lifting points, represented by squares, and the dashed lines show H and H'. The distances from the vertices to H and to H' are measured and compared.

algorithm must be able to examine previously selected robot placements in order to optimally select the next robot placement. The algorithm consists of several major steps, shown in Algorithm 3.3.1.

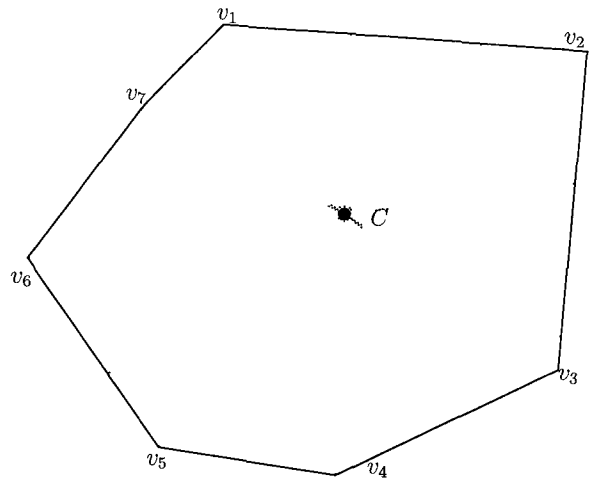


Figure 3.8: A polygon P with all its vertex-chords indicated by dotted lines. Each vertex-chord originates at a vertex of P and passes through the centre of mass to the other side of the polygon.

The first step is to identify all the vertex-chords of the polygon P. To begin this step, we first select an arbitrary vertex, v_i, whose vertex-chord is identified by finding the point at which a line passing through this vertex and C intersects an edge on the opposite side of C (see Figure 3.8). By keeping track of the edge on which each vertex-chord terminates and traversing the vertices in order, we can identify all vertex chords' end-points in linear time. Once all vertex-chords have been identified, they are sorted by increasing slope, with vertical chords given a slope of positive infinity.

Steps 3 and 4 of Algorithm 3.3.1 can be completed simultaneously using Algorithm 3.3.2. A step-by-step example is given in Figure 3.10. Once all vertex-chords have been identified and sorted, edge-pairs can be identified by traversing the set of vertex-chords and examining neighbouring pairs of them. The endpoints of the two edge-pair members are defined by the endpoints of the vertex-chords.

Each edge in the pair is represented as a line in the plane, L_1 and L_2. Equation 3.6 is used to determine the angle of the line segment (*candidate*) connecting the points r_1 and r_2 on L_1 and L_2, respectively, such that the length of $\overline{Cr_1} = \overline{Cr_2}$. Placement

Algorithm 3.3.1 High-level algorithm overview

Input: A polygon P

Output: a list of chords, *sChords* whose endpoints indicate robot placements

1: Get list of all vertex-chords, *vChords*
2. Get list of edge-pairs, *pairs*, from *vChords*
3. Get list of candidate chords, *cChords*, from *pairs*
4: **for all** $i = 1$ to $\frac{m}{2}$ **do**
5. Select one candidate, c, from *cChords* by calling Function 3.3.3, 3.3.4 or 3.3.5.
6: Move c from *cChords* to list of selected chords, *sChords*
7: **end for**
8: return *sChords*

Algorithm 3.3.2 Identifying candidate chords from vertex-chords

Input: a Polygon, P, and a list of vertex-chords, *vChords*, sorted by increasing slope

Output: a list of candidate chords, *cChords*

1: **for all** vc_i in *vChords* **do**
2: get the edge-pairs, e_i and e_j at the endpoints of vc_i and vc_{i+1}
3: using e_i and e_j as L_1 and L_2 respectively, find θ (Equation 3.6)
4: *candidate* = segment from L_1 to L_2 at angle θ
5: **if** endpoints of *candidate* are on e_1 and e_2 **then**
6: add *candidate* to *cChords*
7: **else if** vertex-chord nearest to *candidate* distributes weight within Δ **then**
8: add vertex-chord nearest to *candidate* to *cChords*
9: **end if**
10: **end for**
11: return *cChords*

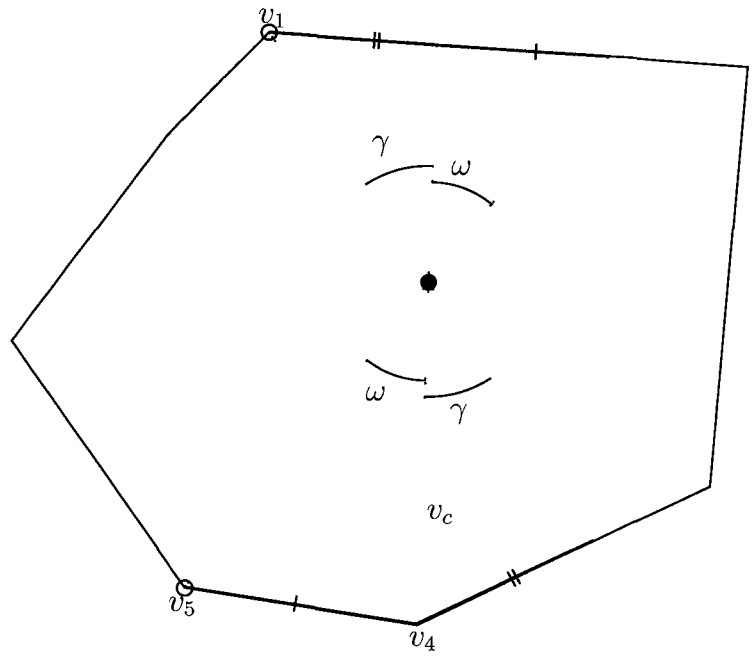
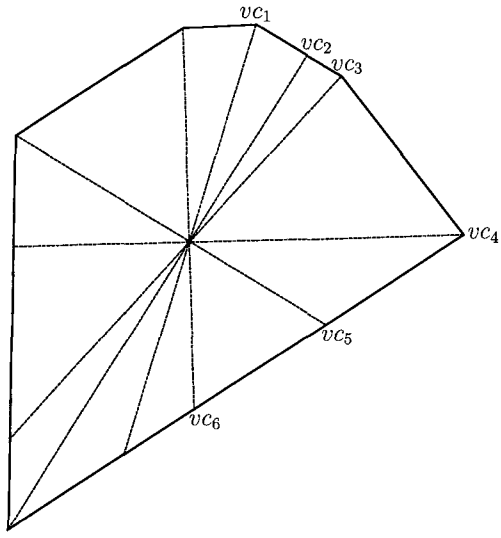


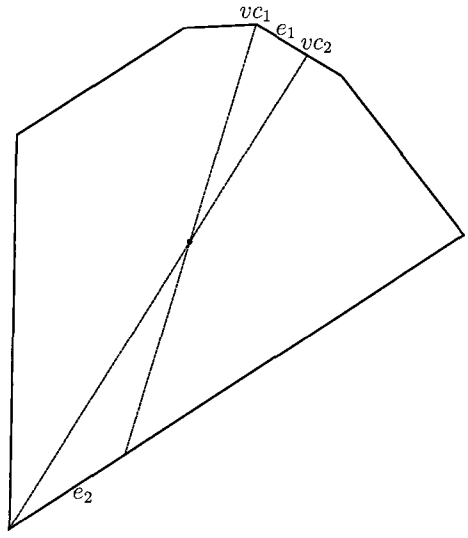
Figure 3.9: As the vertex chord v_c rotates through γ counterclockwise, it reaches the vertex v_1. As it rotates through ω clockwise, it reaches the vertex v_5. The members of the resultant edge-pairs are indicated by matching hash marks and appear in red and blue, respectively.

of robots at the endpoints of *candidate* would result in even weight-distribution between them. If the endpoints of *candidate* both lie within their respective edge-pair members, then the chord defined by *candidate* is added to the list of candidate chords.

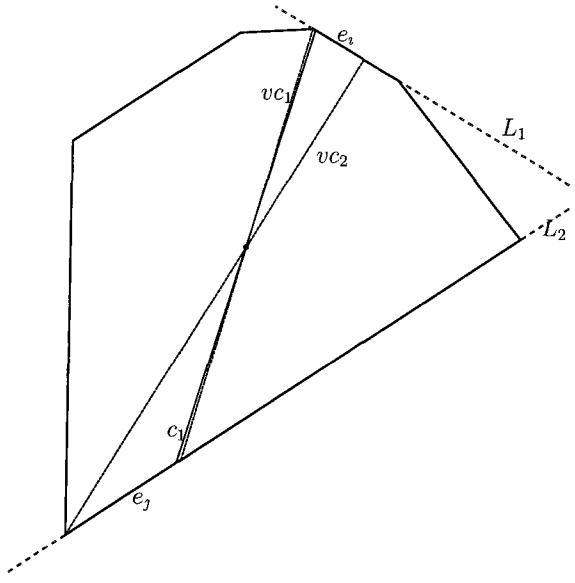
If, on the other hand, one or both of the endpoints of *candidate* lie outside the bounds of their edge-pair members, then the angle of *candidate* is either larger or smaller than that of both vertex-chords. In this case, whichever vertex-chord's angle is closer to that of *candidate* is examined. If the weight-distribution resulting from placing robots at the endpoints of this vertex-chord is within the user-defined Δ value, then the vertex-chord is added to the list of candidate chords.



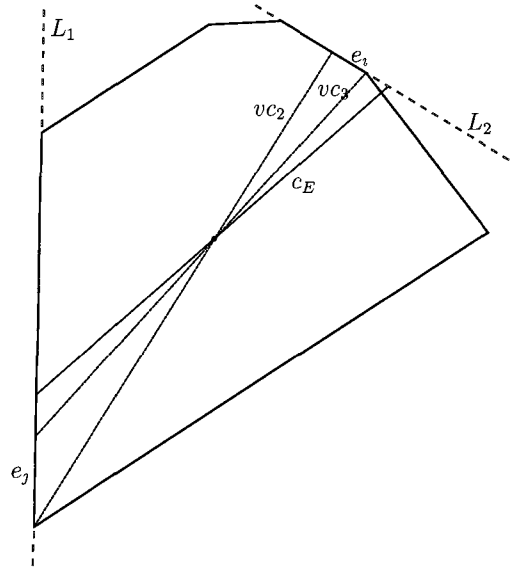
(a) Identify all the vertex-chords of the polygon.



(b) Using the endpoints of the first two vertex-chords (vc_1 and vc_2), identify the first edge-pair (e_1 and e_2).

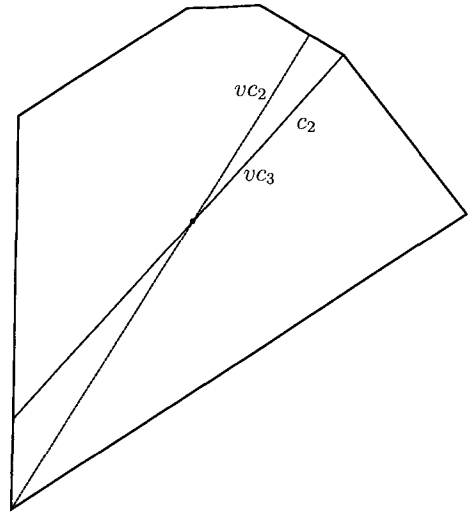


(c) Substituting lines for the edge-pair, use Equation 3.6 to find the chord (c_1) which would distribute weight evenly between two robots. Add this chord to the list of candidate chords, *cChords*.

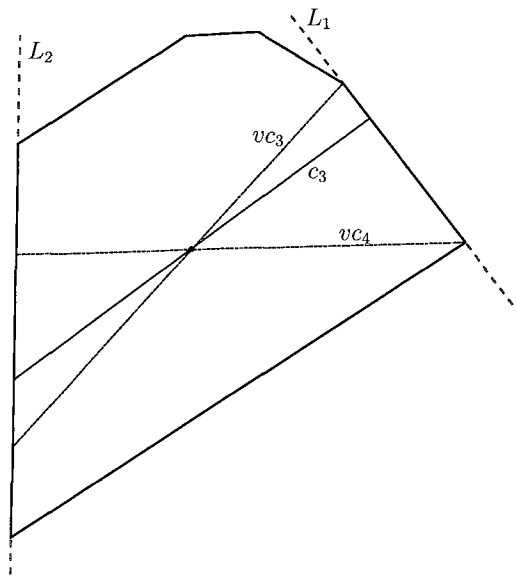


(d) Proceed to the next pair of vertex-chords (vc_2 and vc_3). In this case, the chord which would distribute weight evenly (c_E) lies outside the boundaries of e_i and e_j.

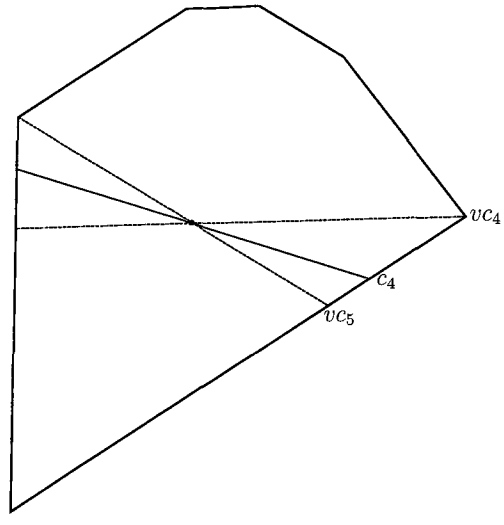
Figure 3.10: Algorithm walk-through (identifying candidate chords)



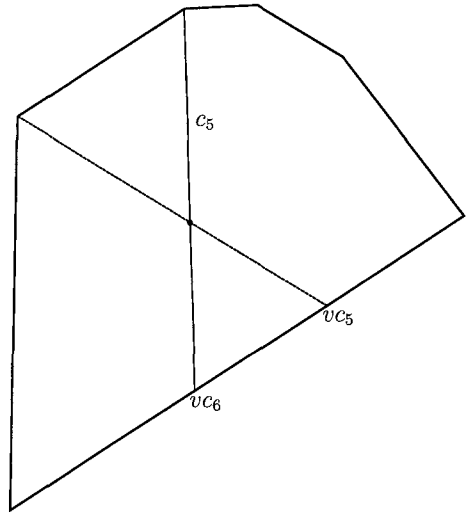
(e) Vertex-chord, vc_3 is nearest to c_E (previous image), and it's ratio is within the acceptable distribution. Add vc_3 to the candidate chords.



(f) Using vertex-chords vc_3 and vc_4, c_3 distributes weight evenly between two robots placed at its endpoints.

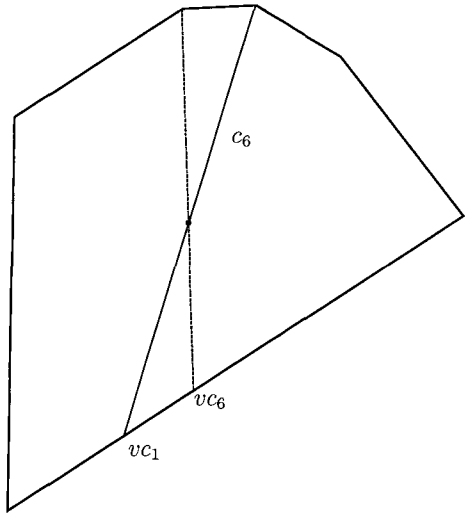


(g) c_4 is the even weight distributing chord between vc_4 and vc_5.

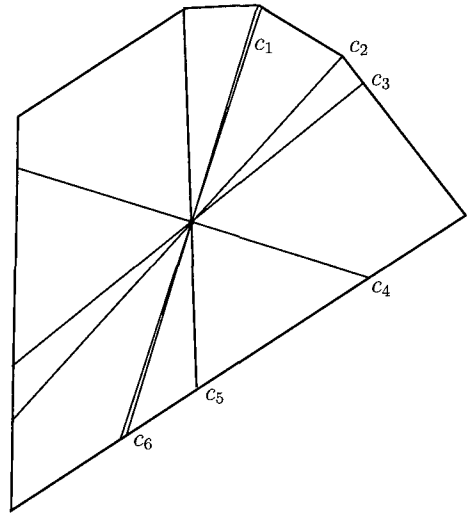


(h) As with vertex-chords vc_2 and vc_3, the chord which distributes weight evenly for the edge-pairs defined by vc_5 and vc_6 lies outside the boundaries of the edge-pairs. vc_6 is closest to distributing weight equally, and is added to the candidates (c_5).

Figure 3.10: Algorithm walk-through (continued)



(i) The final pair of vertex-chords to be examined is vc_6 and vc_1. Again, the chord which would distribute weight evenly is outside of the boundaries of the edge-pair, and the nearest vertex-chord vc_1 distributes the weight near-equally, so it is added to the candidates.



(j) The resulting list of candidate chords. These represent all the placements where a team of robots could be positioned (one at each end of a chord) such that weight is distributed evenly, or near-evenly between them. Note that in some cases, the same chord may appear twice. Duplicate chords should be removed.

Figure 3.10: Algorithm walk-through (continued)

Once all candidate chords have been identified, the algorithm begins choosing which candidate chords to use for each pair of robots based on one of the three metrics discussed in Section 3.2.

An important aspect to note, however, is that all three metrics rely on there being at least one robot-pair placement already selected. As such, the first chord selected from the candidates must be chosen based on some other metric. Given that all three metrics attempt, in one way or another, to enlarge H as much as possible, the most logical choice for the first placement of robots is to place them as far apart as possible while still keeping weight distribution equal. Therefore, the first chord chosen is the longest of those which provides equal weight distribution. Theorem 3.2.2 shows that there exists at least one such chord for all convex polygons.

All three metrics rely on a few pieces of data. First and foremost, they take as parameters two lists: the candidate chords that result in equal (or near-equal) weight distribution, and the list of currently selected chords. The chord chosen by the metric will be removed from the list of candidates (*cChords*) and placed in the list of selected chords (*sChords*). Each metric also makes use of "neighbouring chords". These neighbouring chords refer to the two chords in *sChords* which are radially closest, in each direction, to the candidate being examined. In the case where *sChords* has a size of 1, the neighbouring chords are in fact two instances of the single selected chord.

Function 3.3.3 provides a high-level overview of the area metric. Two triangles are generated for each candidate chord in the list. The vertices of one triangle are made up of the left endpoints of the candidate chord and the two neighbouring selected chords, and the second triangle's vertices are made up of the right endpoints (Figure 3.6a). Once both of these triangles are generated, their size is summed. The candidate chord that produces the largest cumulative size is returned and added to *sChords*.

The second metric, the distance metric, is outlined in Function 3.3.4. All vertices of P which lie between the neighbouring chords (in the sections containing the candidate chord) must be identified, since it is the distances from these vertices to the boundaries of H and H' which will be examined.

Once the affected vertices have been identified, the next step is to examine their

Function 3.3.3 *areaMetric()*

Input: a list of candidate chords, $cChords$, sorted by increasing slope
a list of selected chords, $sChords$, sorted by increasing slope

Output: a single chord to be selected

1: $largestTriangleSize = 0$;
2: **for all** chords c_i in $cChords$ **do**
3: $triangleSize = 0$;
4: find the chords, c_{cj} and c_{ck}, in $sChords$ which neighbour c_i;
5: **for** each endpoint r_i of c_i **do**
6: find the endpoints of c_{cj} and c_{ck}, r_j and r_k, which neighbour r_i;
7: $triangleSize = triangleSize + area(r_i, r_j, r_k)$;
8: **end for**
9: **if** $triangleSize \geq largestTriangleSize$ **then**
10: $largestTriangleSize = triangleSize$;
11: $bestChord = c_i$;
12: **end if**
13: **end for**
14: return $bestChord$;

distances to the boundaries of H and H'. For each affected vertex, the difference in the distance from the vertex to H and to H' is calculated. These differences are summed for all affected vertices for each candidate chord. Whichever candidate chord produces the largest difference is returned and added to the list of selected chords.

The circle metric is much like the area metric in its simplicity (Function 3.3.5). It operates in much the same way, but rather than calculating the size of the triangle's area, it calculates the size of the largest inscribable circle in the triangle [9]. The candidate chord which results in the largest combined circle area is the chord returned.

Function 3.3.4 *distanceMetric()*

Input: $cChords$, sorted by increasing slope

$sChords$, sorted by increasing slope

Output: a single chord to be selected

set $largestDistanceSum = 0$;
for all chords c_i in $cChords$ **do**
  find the chords, c_{cj} and c_{ck}, in $sChords$ which neighbour c_i;
  set r_{jl} and r_{kl} to the left endpoints of c_{cj} and c_{ck};
  set r_{jr} and r_{kr} to the right endpoints of c_{cj} and c_{ck};
  set e_l and e_r to be the left and right edges of H connecting c_{cj} and c_{ck};
  for all vertices v_i of P **do**
    set $distanceSum = 0$,
    if v_i lies in between between r_{jl} and r_l **then**
      $l_{jl} = dist(v_i, e_l) - dist(v_i, \overline{r_l r_{jl}})$;
      $distanceSum = distanceSum + l_{jl}$
    else if v_i lies between r_{kl} and r_l **then**
      $l_{kl} = dist(v_i, e_l) - dist(v_i, \overline{r_l r_{kl}})$;
      $distanceSum = distanceSum + l_{kl}$
    else if v_i lies between r_{jr} and r_r **then**
      $l_{jr} = dist(v_i, e_r) - dist(v_i, \overline{r_r r_{jr}})$;
      $distanceSum = distanceSum + l_{jr}$
    else if v_i lies between r_{kr} and r_r **then**
      $l_{kr} = dist(v_i, e_r) - dist(v_i, \overline{r_r r_{kr}})$;
      $distanceSum = distanceSum + l_{kr}$
    end if
  end for
  if $distanceSum \geq largestDistanceSum$ **then**
    $largestDistanceSum = distanceSum$;
    $bestChord = c_i$;
  end if
end for
return $bestChord$;

Function 3.3.5 *circleMetric()*

Input: *cChords*, sorted by increasing slope

sChords, sorted by increasing slope

Output: a single chord to be selected

1: **for** all chords c_i in *cChords* **do**
2: $largestCircleSize = 0$;
3: find the chords, c_{cj} and c_{ck}, in *sChords* which neighbour c_i;
4: **for** each endpoint r_i of c_i **do**
5: find the endpoints of c_{cj} and c_{ck}, r_j and r_k, which neighbour r_i;
6: $circleSize = circleSize + inscribedCircleSize(r_i, r_j, r_k)$[9];
7: **end for**
8: **if** $circleSize \geq largestCircleSize$ **then**
9: $largestCircleSize = circleSize$;
10: $bestChord = c_i$;
11: **end if**
12: **end for**
13: return *bestChord*;

3.4 Runtime

This section will address the time complexity of our algorithm as it applies to convex polygons. It will address first the runtime requirements of identifying all the candidate placements before showing the requirements for the three metrics.

Lemma 3.4.1 *Given a convex polygon P with n vertices and centre of mass C, all candidate placements of robot teams can be found in $O(n)$ time.*

Proof Given the convex nature of our polygon, iterating over the vertices to get the vertex-chords gives us a list that, when split in two, produces two sorted lists, both in ascending order of slope. The first list is made up of those vertex-chords which originate at vertices on the left side of the polygon, and the second is made up of those originating at vertices on the right side of the polygon. We can, therefore, produce a single sorted list by merging the two in linear time.

Once all vertex-chords have been identified, the algorithm iterates over *vChords* to find the candidates. Each operation in Function 3.3.2 requires constant time. With a linear number of vertices (and therefore a linear number of vertex-chords), the resulting runtime for Function 3.3.2 is linear. Finding the entire set of candidate chords from the set of vertex-chords can therefore be completed in linear time. ∎

There is one final operation indicated in Algorithm 3.3.1, though in practice it would likely be implemented in the metric functions: moving the selected chord returned from the metric functions from the list of candidates to the list of selected chords. Since both lists of chords are maintained in ascending order of slope, the operation to move a chord from one list to the other can be completed in constant time if the appropriate pointers are maintained.

As each metric is different, the runtime for the algorithm will differ depending on which is used. The area and circle metrics have the same time complexity:

Lemma 3.4.2 *Given a convex polygon P with n vertices, a list of candidate robot team placements, and a list of robot teams already placed, the area and circle metrics can select a team placement in $O(n)$ time.*

Proof In both the area (Function 3.3.3) and circle (Function 3.3.5) metrics, each candidate chord must be examined and its neighbouring selected chords found. By ensuring that both sets of chords are maintained in the correct order, the neighbours can be found in constant time for each candidate. Once the three chords (candidate and both its neighbours) have been identified, determining the resultant triangle or inscribed circle is a constant-time operation. The result is a pair of metrics that will determine which among the list of candidate chords should be selected in linear time. ∎

The runtime of the distance metric is different, and slightly higher than that of the area and circle metrics.

Lemma 3.4.3 *Given a convex polygon P with n vertices, a list of candidate robot team placements, and a list of robot teams placements already selected, the distance metric can identify a team placement in $O(n^2)$ time.*

Proof As in the area and circle metrics, the selected chords which neighbour each candidate must be identified, and can be completed in constant time for each candidate. The list of vertices which lie between the neighbouring chords must be identified, and it must be determined whether they lie above or below the candidate chord. There can be a linear number of such vertices, and calculating the distance from each to a line (an edge on the boundary of H or H') is a constant-time operation. Each candidate chord, therefore, results in a linear number of operations, and there are a linear number of candidate chords. The distance metric, therefore, runs in quadratic time for each candidate chord to be selected. ∎

Theorem 3.4.4 *Given a convex polygon P with n vertices and m robot-lifting teams, the time complexity of our algorithm using either the area or circle metrics is $O(mn)$.*

Proof Lemma 3.4.1 shows how all candidate chords can be found in $O(n)$ time. Lemma 3.4.2 shows how, given a list of candidates, one can be selected in $O(n)$ time. Moving a chord from the list of candidates to the list of selected chords can be completed in constant time if pointers in both lists are maintained and the operation is done in Functions 3.3.3 or 3.3.5. Since a candidate must be selected for each of the m teams, the resulting runtime is $O(n) + O(mn) + O(m) = O(mn)$. ∎

Theorem 3.4.5 *Given a convex polygon P with n vertices and m robot-lifting teams, the time complexity of our algorithm using the distance metric is $O(mn^2)$.*

Proof Lemma 3.4.1 shows how all candidate chords can be found in $O(n)$ time. Lemma 3.4.3 shows how, given a list of candidates, one can be selected in $O(n^2)$ time. Moving a chord from the list of candidates to the list of selected chords can be completed in constant time if pointers in both lists are maintained and the operation is done in Function 3.3.4. Since a candidate must be selected for each of the m teams, the resulting runtime is $O(n) + O(mn^2) + O(m) = O(mn^2)$. ∎

3.5 Summary

This chapter discussed the desired properties of robot placement and shown how our algorithm addresses them in the case of convex polygons. An algorithm was presented based on three different metrics, which iteratively chooses placement points. The overall runtime our algorithm was shown to be either $O(mn)$ or $O(mn^2)$ given n vertices of the polygon and m robot teams, depending on which selection metric is used.

Chapter 4

Simple Polygons

This chapter will address how the algorithm described in Chapter 3 can be adapted to work for the more general class of simple polygons. After outlining the issues that arise from using these more general polygons in Section 4.1, the alterations to the algorithm to address these issues will be discussed in Section 4.2. Finally, Section 4.3 explains how these changes affect the algorithm's runtime.

4.1 Issues with Extending Algorithm

The algorithm, as previously listed, works for all convex polygons, but needs some adjustments in order to work for the more general class of simple polygons. While much of the theory on which the algorithm is based is still valid, some aspects need to be revisited and adjusted. This section will identify the issues concerned with extending the algorithm to work for all simple polygons.

The goal of the algorithm remains unchanged: to establish lifting points which will provide a stable grasp with near-even weight distribution. Theorem 3.2.1 showed how two robots placed opposite the centre of mass are able to lift a polygonal object in equilibrium. This theorem remains valid for non-convex polygons as well since it makes no assumptions about the shape of the polygon. Moreover, as the weight distribution is dependent solely on the lifters' respective distances from the centre of mass, it is still true that ratio of weight lifted is inversely proportional to the ratio of the distances from the centre of mass.

There are, however, some new considerations under the broader class of simple polygons. If a polygon P is convex, any line passing through its centre of mass intersects the boundary of P in exactly two places, each of which indicate a potential robot placement. If P is non-convex, however, a line passing through its centre of mass can pass through P's boundary in a number of places on either or both sides of

the centre of mass. Theorem 3.1.1 still requires that a placement of a pair of robots must ensure that each robot is on a different side of the centre of mass in order to achieve equilibrium.

Algorithm 3.3.1 makes the assumption that each vertex of the polygon contributes exactly one vertex chord. This assumption, though valid for convex polygons, is not valid for all simple polygons (Figure 4.1).

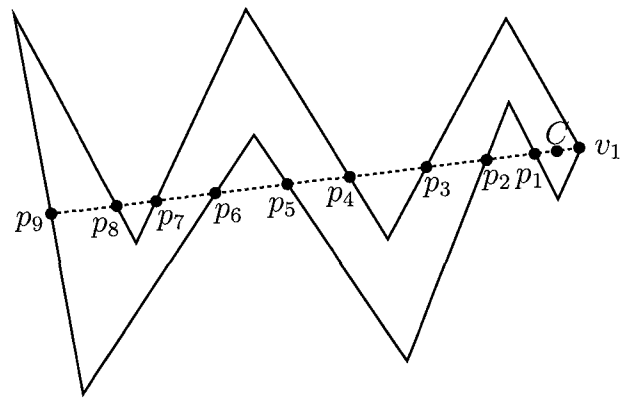
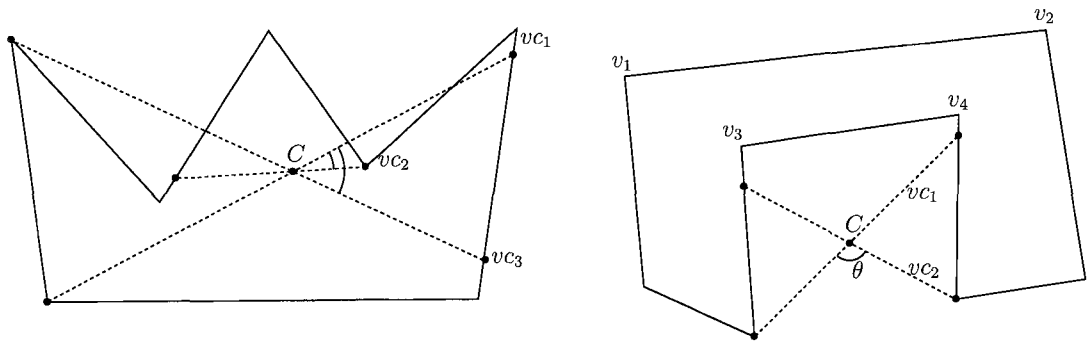


Figure 4.1: The vertex v_1 produces $n-2$ vertex chords: $\overline{v_1p_1}$, $\overline{v_1p_2}$, ..., $\overline{v_1p_9}$.

This, in turn, invalidates another useful assumption used in Algorithm 3.3.2: the knowledge that neighbouring vertex-chords (when sorted by increasing slope) define a single edge-pair. Figure 4.2 show simple polygons in which neighbouring vertex-chords do not define a valid edge-pair. Edge-pairs will therefore need to be found using a different methodology.

In a convex polygon P, a chord which originates on the boundary and passes through the centre of mass C *must* pass through the boundary of P on the other side. That is to say that for every possible placement of a robot on the boundary of P, there exists exactly one placement for a second robot such that the two can lift in equilibrium. This is not the case for all simple polygons. Figure 4.2b shows a polygon in which the centre of mass lies outside the boundary of the polygon. In such instances, it is possible that a ray drawn from the centre of mass does not pass through



(a) (b)

Figure 4.2: Two polygons wherein neighbouring vertex-chords do not produce a valid edge pair. (a) vertex-chords vc_1 and vc_2 are neighbours when sorted by increasing slope, but the valid edge-pair is actually defined by vc_1 and vc_3. (b) There is no edge pair in the range θ, as vertices v_1 - v_4 produce no vertex-chords at all.

the boundary of the polygon. This implies that there exist parts of the boundary of the polygon where a single robot can be placed, but a second robot placement is not possible while keeping the polygon in equilibrium. Any polygon where the centre of mass exists outside of the polygon potentially has this property.

The potential for the centre of mass to exist outside of the polygon presents the additional possibility that there are no placements that are in equilibrium *and* equally distribute the mass among both lifters (see Figure 4.3).

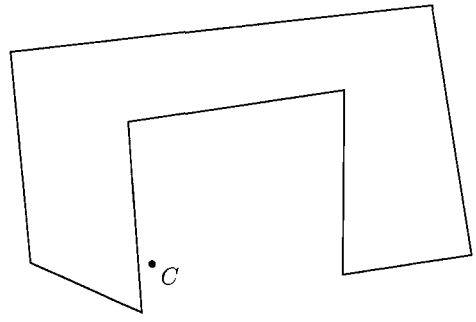


Figure 4.3: A polygon P with robot placements with poor weight distribution.

The expansion to the class of simple polygons affects the three metrics used to determine placement as well, albeit in a more minor way. The most significant effect is on that of the area metric. Recall that the premise behind this metric is that all points on the surface of the polygon serve as potential application points for disturbing forces. When P is convex, all points in the triangles generated in Function 3.3.3 are inside the boundary of P. Such is not necessarily the case if P is non-convex, however. The area metric must therefore identify not only the triangles generated by the candidate chord and neighbouring selected chords, but the intersection of those triangles and the polygon. This intersection represents the increase in polygon coverage created by using the candidate lifting points.

Additionally, all three metrics must now differentiate between lifting points which lie inside and outside of H. Figure 4.4 shows a case where both endpoints of a candidate chord lie inside of the hull of the established lifting points. While placing robots at these new lifting points would serve to reduce the weight lifted by each robot, they do not contribute to the stability of the lifting base, and may not therefore be the ideal points to choose.

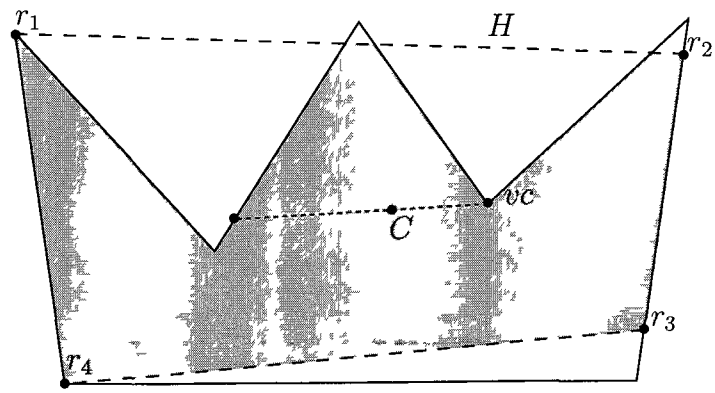


Figure 4.4: The vertex-chord vc lies entirely within H.

The next section will discuss how the algorithm can be adapted to account for these new requirements.

4.2 Algorithmic Alterations

A number of algorithmic adjustments are necessary to adapt the algorithm to work for all simple-polygons. Our underlying ideas for robot placement remain valid, such as placing robots in teams of two for good weight-distribution and maximizing how much of the polygon's area lies inside of the lifting base, though some properties of convex polygons which were used to optimize our algorithm can no longer be applied.

The first step, and the first portion of the algorithm that needs to be addressed is in finding the vertex-chords. As discussed in the previous section, each vertex in a convex polygon generates exactly one vertex-chord, whereas each vertex in a simple polygon can generate anywhere between zero and $n-2$ vertex chords (Figure 4.1). Recall that for convex polygons our algorithm iterates over the vertices of the polygon in radial order while keeping track of the edge on which vertex-chords terminate. The knowledge that each vertex produces exactly one vertex-chord allowed us to generate all vertex-chords in linear time. For non-convex polygons, however, each vertex can generate a number of vertex-chords. Our algorithm, therefore, must check all edges for each vertex to determine whether valid vertex-chords terminate there.

A vertex-chord vc_i is defined by both a vertex v_i (its originating point) and an edge e_j (the edge on which it ends). The vertex-chord vc_k, which, together with vc_i defines an edge-pair, is that which either originates at a vertex neighbouring v_i and terminates on e_j, or originates on a vertex of e_j and terminates on an edge incident to v_i (Figure 4.5).

Function 4.2.1 outlines the algorithm to find all edge-pairs of a simple polygon. Two passes of the set of vertex-chords are performed for each vertex-chord; one for edge e_i (the edge preceding the vertex-chord's originating vertex v_i), and one for edge e_{i+1} (the edge following it). The first pass identifies the vertex-chord which has endpoints on e_j and e_i, while the second identifies the vertex-chord which has endpoints on e_j and e_{i+1}. Once edge pairs have been identified, the process for finding candidate chords remains the same as in the case of convex polygons.

Every convex polygon has at least one chord which indicates placement that would equally distribute weight. Our algorithm uses this knowledge to choose the longest such chord before applying any of the three metrics for further chord selection. Not

Function 4.2.1 *findEdgePairs()*

Input: a list of vertex chords, $vChords$;

Output: a list of valid edge-pairs, $pairs$;

```
1: doBefore = true;
2: for each chord c_i in vChords do
3:    Vertex v_i = getOriginatingVertex(c_i);
4:    if doBefore then
5:       Edge e_s = getPreviousEdge(v_i);
6:    else
7:       Edge e_s = getNextEdge(v_i);
8:    end if
9:    Edge e_t = getTerminatingEdge(c_i);
10:   for each chord c_j in vChords do
11:      if c_i ≠ c_j then
12:         if (e_s contains startingPoint(c_j) AND e_t contains endingPoint(c_j)) OR
            (e_s contains endingPoint(c_j) AND e_t contains startingPoint(c_j)) then
13:            add the edge pair, e_s and e_t to pairs;
14:            doBefore = !doBefore;
15:         end if
16:      end if
17:   end for
18: end for
19: remove duplicates from pairs;
20: return pairs;
```

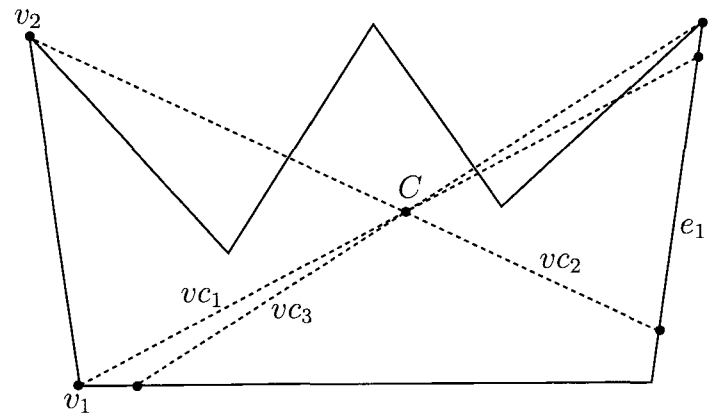


Figure 4.5: The vertex-chord vc_1 is defined by the vertex v_1 and the edge at which it terminates, e_1. Its edge-pairs are defined by vc_2, which originates at a neighbouring vertex and ends at e_1, and by vc_3, which originates at a vertex of e_1 and terminates at an edge incident to v_1.

all simple polygons produce such chords, however. The choice for which chord to choose before applying any metrics must therefore also be altered. As the primary goal of the algorithm is to achieve a favourable weight distribution, if equal weight distributing chords exist, the algorithm chooses the longest such chord. In the event that no such chords exist, the one with the best weight distribution is chosen instead.

All three of our metrics make use of the selected chords which neighbour the candidate. Recall that in all three metrics, the purpose of using these neighbours is to determine how the addition of the candidate affects H. It is only logical, therefore, that the "neighbouring" chords be selected from those that make up H. For convex polygons, these are the two selected chords which are rotationally closest to the candidate in each direction. It is possible for non-convex polygons, however, that one or both of these selected chords have one or both of their endpoints lying inside of H, rather than acting as a vertex of it (Figure 4.6).

Instead of using the neighbouring chords, therefore, the metrics must find the

nearest two points on the boundary of H to each of the candidate's endpoints (Figure 4.6).

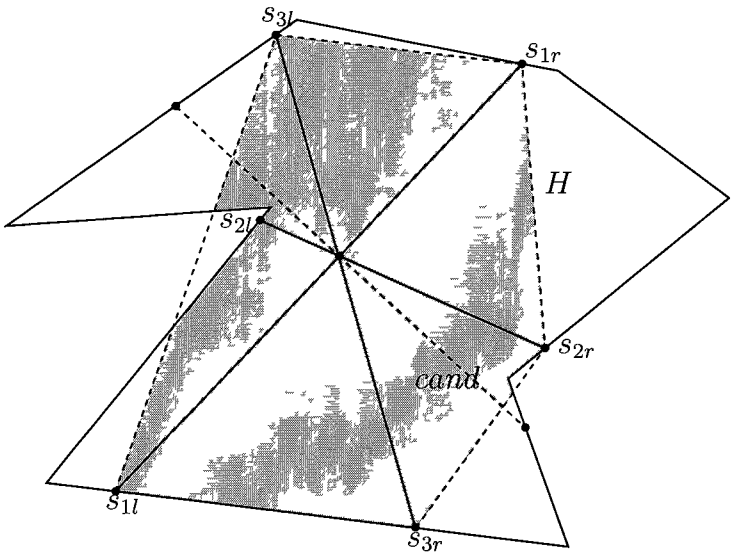


Figure 4.6: One of the chords, s_2, which directly neighbours the candidate chord, *cand*, has an endpoint not on the boundary of H. The points on H's boundary which neighbour *cand*'s endpoints are used instead of the neighbouring selected chords. For the left endpoint of *cand*, the neighbours are s_{3l} and s_{1l}, and the neighbours for the right endpoint of *cand* are s_{2r} and s_{3r}.

Additionally, any point on the boundary of a non-convex polygon may lie inside of H. This implies that candidate lifting points may also lie inside of H. The algorithm must therefore make a distinction between those lifting points which lie outside of H and therefore contribute to stability and those that lie inside of H.

Both the area metric and circle metric handle this issue in the same way. When a candidate chord is examined, if one or both of its endpoints lie inside the convex hull of the lifting points, then placing a robot at that endpoint will make no contribution to stability. In this case, the area of the triangle made up of the candidate's endpoints and the nearest points on H (for the area metric) or the size of the inscribed circle (for the circle metric) is deemed to be zero. Thus, candidate lifting chords where both endpoints lie inside of the convex hull of the lifting points already selected are

not deemed to be contributing to stability and are less likely to be chosen.

A similar approach is applied to the distance metric. However, in addition to determining whether the candidate chord's endpoints lie inside or outside of H, the position of each vertex must be assessed. Recall that the aim of the distance metric is to minimize the potential torque that can be applied by disturbing forces which act outside the boundaries of H. Recall also that the maximum potential torques for a given force value occur at the points most distant from the edge of H, which are the vertices of the polygon. Any vertices which lie inside of H, therefore, must be attributed a distance value of zero (as a force applied there would produce zero disturbing torque).

Additionally, since the area metric aims to maximize the amount of the polygon which lies inside of H, only portions of the polygon which fall inside of this area should be counted toward determining good placements. It is possible that large portions of the triangles given by the candidate chord's endpoints and neighbouring vertices of H are made up of areas outside the polygon's boundaries (Figure 4.7). The area we are concerned with is the intersection of this triangle and the polygon. For each candidate, therefore, the intersection of the generated triangle and the polygon must be calculated and used as the area, rather than the area of the triangle.

4.3 Runtime Analysis

Lemma 4.3.1 *Given a non-convex polygon P with n vertices and a centre of mass C, all candidate placements of robot teams can be found in $O(n^4)$ time.*

Proof Recall that there is the potential now for a line passing through C and a vertex v to intersect as many as $n-2$ edges of P, resulting in as many as $n-2$ vertex-chords. Checking each edge for intersections for each vertex results in a time complexity of $O(n^2)$ to find all vertex-chords.

Once all vertex-chords have been identified, identifying all edge-pairs is an operation that is quadratic in the number of vertex-chords (Function 4.2.1), of which there are $O(n^2)$, resulting in a runtime of $O(n^4)$ to identify edge-pairs.

After calculating all edge-pairs, generating all candidate chords remains unchanged

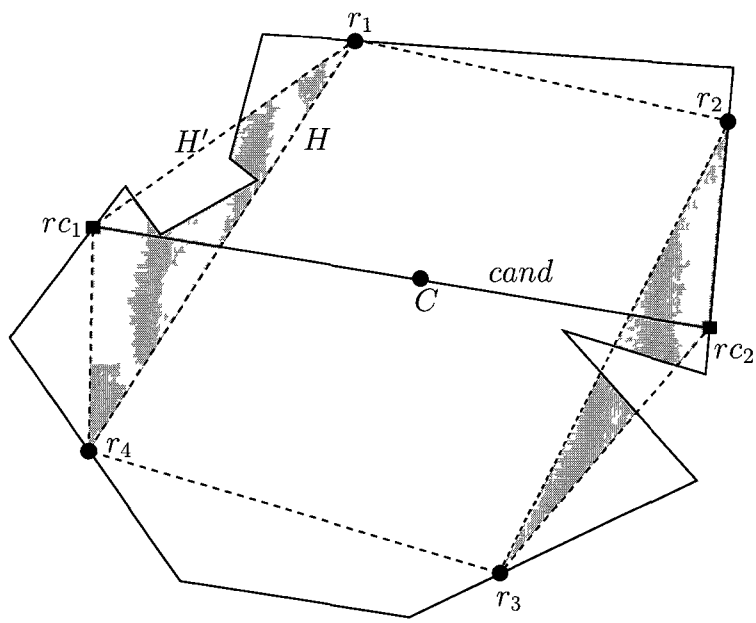


Figure 4.7: The area metric must now consider the generated triangles' intersections with the polygon, rather than simply their area.

from the case of convex polygons, and thus remains constant in the number of edge-pairs, of which there are $O(n^2)$. The time complexity of finding the vertex-chords, edge-pairs and candidate chords is therefore $O(n^2) + O(n^4) + O(n^2) = O(n^4)$. ∎

The complexity of identifying all edge-pairs of the polygon is the most expensive part of our algorithm. While the metrics also suffer increases in complexity due to the extension to non-convex polygons, their increase is less than that of the first part of the algorithm.

Recall also that each metric must identify whether the endpoints of the candidate chord lie inside or outside of H, as well as which points of H are radially closest to the endpoints (the endpoints' neighbours). If H is maintained as each lifting point is selected, then it need not be calculated each time.

Lemma 4.3.2 *Given a convex hull H with m vertices and a candidate chord c_i, identification of the points of H which neighbour c_i, as well as whether the endpoints*

of c_i lie inside or outside of H, can be completed in $O(m)$ time.

Proof Determining whether or not the endpoints p_1 and p_2 of c_i lie inside of H is linear in the size of H. If p_1 or p_2 lies to one side (left or right) of all edges of H, then it is inside of H. Furthermore, determining which points of H are the neighbours of p_1 and p_2 can be done simultaneously at no extra cost. If a ray originating at C and passing through p_1 intersects an edge of H, then the edge's endpoints are p_1's neighbours. The cost to find whether c_i's endpoints lie inside or outside of H, as well as which points of H neighbour c_i's endpoints is therefore $O(m)$. ∎

The area metric suffers the largest increase in complexity and runtime due to the the increased number of candidate chords as well as the difficulty in calculating the intersection of the triangles (defined by the endpoints of the candidate chords and their neighbours) and the polygon.

Lemma 4.3.3 *Given a non-convex polygon P with n vertices, a list of candidate robot team placements, and the convex hull of selected placements, the area metric can select a team placement in $O(n^3 logn)$ time.*

Proof Recall that the area metric must now compute the intersection of the polygon and a triangle (whose vertices are the candidate's endpoint and its neighbours). Assuming each candidate's endpoints are outside of H and the neighbouring points on H are known ($O(m)$ for each candidate from Lemma 4.3.2) the operation of calculating the affected area for each candidate chord increases from a constant value to $O(nlogn + klogn)$ (from an algorithm presented in [13] to compute the intersection of two planar subdivisions), where k is the complexity of the intersection. However, k is, at most, a linear factor of n. The resultant cost to compute the intersection for each triangle is therefore $O(nlogn)$. Since the number of candidate chords has also increased, along with the number of vertex-chords and edge-pairs, to $O(n^2)$, using the area metric to identify the best candidate chord among a list has jumped from $O(n)$ in the case of convex polygons to $O(n^3 logn)$. ∎

Theorem 4.3.4 *Given a non-convex polygon P with n vertices and m robot lifting teams, the time complexity of our algorithm using the area metric is $O(n^4)$.*

Proof From Lemma 4.3.1, we can find all $O(n^2)$ candidates in $O(n^4)$ time. Lemma 4.3.2 shows that for a given candidate, its endpoints' neighbours and positions relative to H can be found in $O(m)$ time, resulting in $O(mn^2)$ time to perform the calculation for all candidate chords. Finally, Lemma 4.3.3 shows that the triangle made up of the candidate's endpoints and their neighbours intersected with P can be calculated in $O(nlogn)$ time for each candidate endpoint, resulting in an overall time of $O(n^3logn)$ time for all candidates. Since our algorithm is iterative, and therefore each operation after finding all candidates must be performed for each of the m teams to be placed, the overall runtime for our algorithm using the area metric is $O(n^4) + O(m^2n^2) + O(mn^3logn) = O(n^4)$. ∎

Since the circle metric remains largely unchanged from convex polygons to non-convex polygons, its runtime changes very little, though it is impacted by the increase in the number of candidate chords. Once it has been determined that the endpoints lie outside H and the neighbours have been identified, calculation of the inscribed circle still requires constant time.

Lemma 4.3.5 *Given a non-convex polygon P with n vertices, a list of candidate robot team placements, and the convex hull of selected placements, the circle metric can select a team placement in $O(mn^2)$ time.*

Proof For a given candidate, determining its endpoints' neighbours and positions relative to H requires $O(m)$ time (Lemma 4.3.2). Calculating the size of the largest inscribed circle within the triangle defined by the candidate endpoint and neighbours remains a constant time operation [9]. There are $O(n^2)$ possible candidate chords, resulting in $O(mn^2)$ runtime to select the best candidate. ∎

Theorem 4.3.6 *Given a non-convex polygon P with n vertices and m robot lifting teams, the time complexity of our algorithm using the circle metric is $O(n^4)$.*

Proof From Lemma 4.3.1, candidate chords can be identified in $O(n^4)$ time. From Lemma 4.3.5, a candidate can be chosen by the circle metric in $O(mn^2)$ time. Since the circle metric is invoked for each team to be selected, the overall runtime for the circle metric is $O(n^4) + O(m^2n^2) = O(n^4)$ ∎

The distance metric sees a small overall increase in its runtime, as shown in the following lemma.

Lemma 4.3.7 *Given a non-convex polygon P with n vertices, a list of candidate robot team placements, and the convex hull of selected placements, the distance metric can select a team placement in $O(n^3)$ time.*

Proof From Lemma 4.3.2, all candidates' neighbours can be found in $O(mn^2)$. Let n_1 and n_2 be the neighbours of one endpoint of one candidate chord, and n_3 and n_4 be the neighbours of the candidate chord's other endpoint. The vertices that may be affected by the selection of this particular candidate chord are those which lie inside of the angles $\angle n_1 C n_2$ or $\angle n_3 C n_4$. Determining which vertices are affected for a given candidate, therefore, requires $O(n)$ time once the candidate's neighbours are identified.

Each affected vertex of the polygon must be checked to determine which of three different regions it lies in: inside of H, between H and H', or outside of H'. In the first case, the vertex remains unaffected by the inclusion of the candidate points. In the second case, it is affected, but only by the distance it lies from H. Finally, in the latter case, its distance decreases by the difference between its distance to H and its distance to H'. Since both neighbours have already been identified, these operations require no more time than they do for the convex case. Determining which region a vertex lies in can be completed in constant time for each candidate chord, as can the calculation of the impact made by the inclusion of the new candidate chord. The result, therefore, is that checking the distances of all affected vertices is linear in the number of candidate chords.

The cost of identifying a single candidate's neighbours and affected vertices, then calculating the impact of including that candidate is therefore:

$$O(m) + O(n) + O(n) = O(n)$$

Since there are $O(n^2)$ candidates, the total cost is $O(n^3)$. ∎

Theorem 4.3.8 *Given a non-convex polygon P with n vertices and m robot lifting teams, the time complexity of our algorithm using the distance metric is $O(n^4)$.*

Proof From Lemma 4.3.1, all candidate chords can be found in $O(n^4)$ time. Lemma 4.3.7 shows that for each team to be placed, the distance metric can select a placement in $O(n^3)$ time. Since the distance metric is invoked for each team to be placed, the overall runtime of our algorithm using the distance metric for a non-convex polygon is $O(n^4) + O(mn^3) = O(n^4)$. ∎

While all three metrics have different time requirements, it is the identification of candidate lifting points which serves as the most expensive part of our algorithm when working with non-convex polygons. Regardless of the metric used, the overall runtime of our algorithm for non-convex polygons is $O(n^4)$.

4.4 Summary

This chapter has shown how the algorithm presented in Chapter 3 for obtaining lifting points for convex polygonal objects can be extended to work for the more general class of simple polygons as well. The problems with extending the algorithm were identified, alterations to the algorithm outlined, and the runtime analysis presented. Despite a number of issues which arise from this extension, the increased runtime requirements of $O(n^4)$ (up from as much as $O(mn^2)$ for convex polygons) are a small price to pay for the increased robustness of the extended algorithm.

Chapter 5

Experiments and Results

This chapter will address the implementation of the algorithm created as well as the results of the experiments conducted. The implementation will be discussed in Section 5.1, followed by an explanation of the data set used for testing. An analysis of how our algorithm performs for the three desired lift properties introduced in Section 3.1 follows. Weight distribution, as the primary goal of this work, will be addressed first in Section 5.3, followed by the secondary goal of stability in Section 5.4. Finally, despite the algorithm not specifically aiming to achieve form-closure, a discussion of how well our algorithm performs in doing so will follow in Section 5.5.

5.1 Implementation

Our algorithm was implemented and run in order to test its efficacy in identifying lifting points on a variety of different polygons. The user interface was written in Java, and presents several ways of generating and visualizing test data. The application provides a simple interface (Figure 5.1) which allows the user to specify a polygon either by drawing it (sequentially clicking the location of its vertices) or by importing one from a file (see Figure A.1.1 in Appendix B). As the user adds points to the polygon by clicking vertices, the centre of mass (indicated by the black dot) of the polygon is recalculated. The centre of mass is calculated using the assumption of an even mass distribution throughout the polygon. It can, however, be redefined by the user at any point by clicking on *Move CoM* in order to represent a non-uniform mass distribution.

Once a polygon as been defined, the user may specify the number of robot teams to place by indicating in the box labeled *# of Robot Pairs* or by incrementally adding/removing teams by clicking the appropriate buttons. The *Delta* field allows the user to specify the allowable distribution of weight within each pair of robots.

This value represents the ratio of weight discussed in Section 3.2.

Grasp Type allows the user to choose from one of five values: Area, Circle, Distance, Distributed and Random. Area, Circle and Distance indicate that the lift points should be chosen based on one of the three respective metrics. Distributed and Random are placements that were created in order to provide a benchmark against which to test our algorithm. The distributed method distributes the lifters around the object such that the angle between teams is equal, where possible. The random method places teams of robots randomly, such that each team is distributed across the centre of mass. These two additional selection methods are explained in further detail in Section 5.2.

Clicking on the *Generate*, *Add Team* or *Remove Team* buttons re-runs the algorithm against the specified polygon, generating the requested number of lift points (blue dots on the polygon's boundary in Figure 5.1). Additionally, information pertaining to the object's form-closure state and stability are relayed using different coloured regions. The green region indicates what portion of the polygon's surface lies inside of the convex hull of the lifting points. Recall from Section 3.1.2 that it is only forces applied to the polygon's surface outside of this region which could upset the stability of the object. The proportion of the polygon which is contained within this region is indicated as a percentage in the upper-left corner of the drawing area (*49.06% covered* in Figure 5.1).

The two remaining coloured regions (blue and red) are both related to the determination of whether the indicated lifting points immobilize the object. The red region shows the set of points about which the object can be rotated clockwise without the lifting points penetrating the boundary of the object. The blue region shows the set of points about which the polygon can be rotated counterclockwise. Generation of these regions is based on the work of Van der Stappen in [44], and is explained in Section 5.5.

When the size of both of these regions is zero, the object has been placed in form-closure. Since it is possible that a red or blue region exists off-screen, the polygon's form-closure status is also indicated in the upper-left corner of the drawing area (*closed: false* in Figure 5.1).

In addition, our application allows the user to generate and output a variety of data for either the displayed polygon or for a large set of polygon files. The number of possible lifting teams is determined for each polygon, then for each team of lifters to be placed, a variety of data is calculated then output to a file to simplify analysis. The following section explains this data in further detail.

Some examples of output from the implementation comparing the effects of various centres of mass are shown in Appendix A.

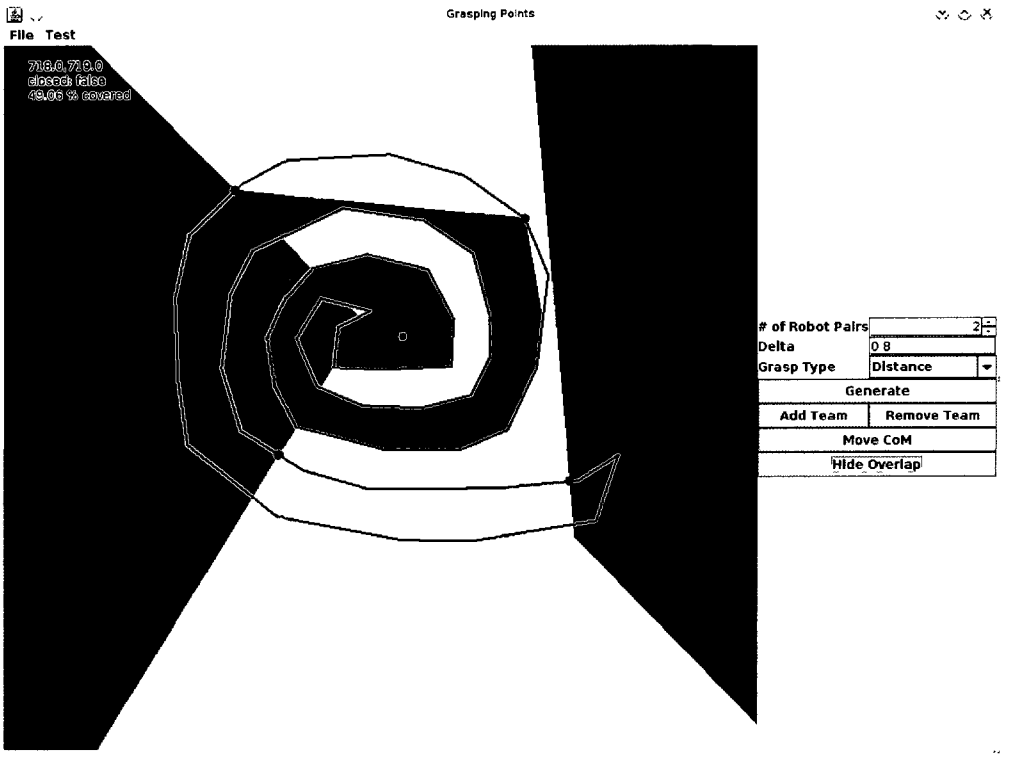


Figure 5.1: Screen shot of our application.

5.2 Data Set

Sections 5.3, 5.4 and 5.5 discuss the results of our algorithm as run against the sample set of polygons found in Appendix B. We believe this set to be a fair representation of the set of polygons expected to be encountered. Both regular and irregular convex polygons were included, with a varying number of vertices. A number of non-convex polygons were also included, also with varying numbers of vertices. In addition to

some very simple polygons, several more complex polygons with a large number of vertices were tested.

The location of the centre of mass plays an integral part in our algorithm's ability to generate good lifting points. In a real-world application, there is no guarantee that an object being lifted would be of a constant density and have an equal weight distribution throughout the object. As such, many of our testing polygons were included multiple times, though with differing centres of mass. This effectively models a non-uniform weight distribution throughout the polygon, and allows us to test the efficacy of our algorithm on such objects.

Our algorithm was run for each of the three selection metrics (area, circle and distance) against every polygon in the set. Two alternative selection methods, random and distributed placements, were also tested. Random placements were selected by first randomly placing one member of each two robot team on the boundary of the polygon, then placing the second team member across the centre of mass from the first.

The distributed selection method distributed the lifting points equally about the object, such that the angle between lifting teams was kept equal (Figure 5.2a). In instances where this results in multiple possible placements, the point furthest from the centre of mass was chosen. Note that on some polygons, when the centre of mass lies outside of the polygon, there are areas where a team cannot be placed. In these instances, the teams were evenly distributed throughout the valid placement areas of the polygon (Figure 5.2b).

As each successive pair of lifting points was selected for each of the five selection methods, several pieces of data were recorded, including

- The percentage of the polygon covered by H
- Whether or not the polygon is in form-closure
- The minimum, average, and maximum weight at each lifting point

Given the nature of random placements, 200 samples were generated for each polygon. The above data values are computed, but then averaged over all 200 samples

(in the case of form-closure, this results in a value between 0 and 1 indicating the portion of the 200 samples that were in form-closure).

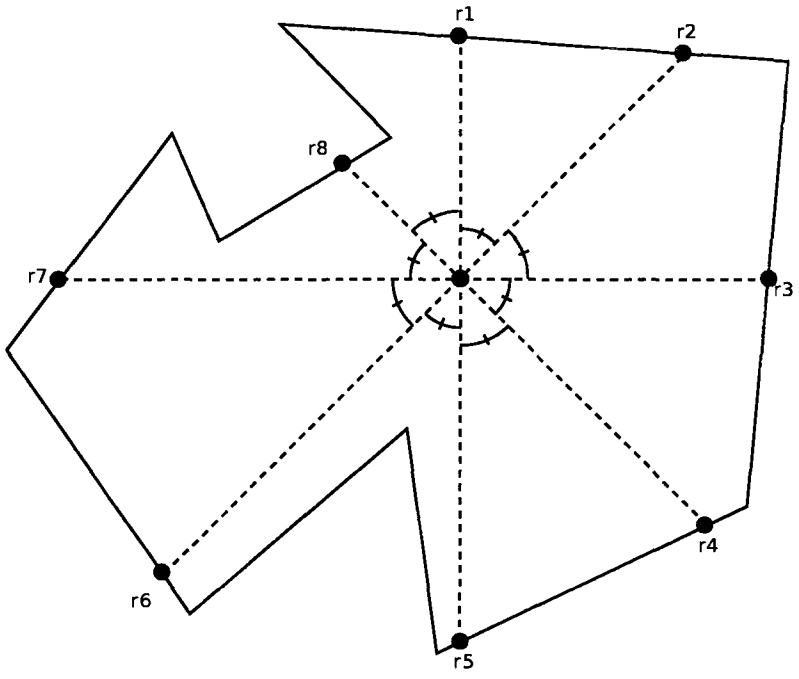
In all tests performed, the Δ was set to 0.8, ensuring that all robot teams maintained, at worst, a 0.8:1 ratio of weight for each robot pair when using our algorithm. The distributed and random selection methods, however, were not subject to this restriction.

5.3 Weight Distribution

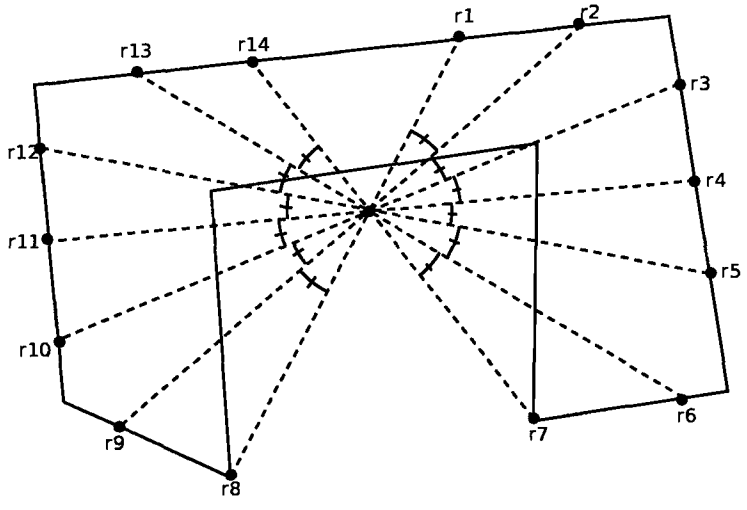
Recall that the primary goal of our algorithm is to distribute the weight of the object near-evenly among the lifting robots. In order to best measure how effectively our algorithm achieves this property, the average weight difference between robots is examined. It is important to note that our algorithm makes no considerations for the weight of the object, nor for the lifting capabilities of the robots. If, however, both of these parameters are known, the minimum number of robots required (assuming all robots lift at their maximum capability) is easily calculated. Our algorithm, by equalizing the weight distribution, can be used to minimize the number of robots. The weight at each robot can then iteratively be reduced as required.

For each polygon, team placements were repeatedly selected until there were no placements remaining which distributed weight within the acceptable range $(0.8 : 1)$. Each team was assigned an equal portion of the object's weight. The weight at each robot was then calculated, and the difference between the robot with the highest weight and the robot with the lowest weight was recorded. If the robot lifting the most weight lifts nearly the same as the robot lifting the least weight, then it logically follows that all robots are lifting nearly the same amount. For the sake of clarity, our tests assume that the object's weight is always 100kg.

Figure 5.3 shows the difference in the greatest weight lifted among the robots and the least weight lifted, averaged over the polygon set, and Figure 5.4 shows the standard deviation. For example, assume four robots (2 teams) were to be placed and the distance metric were used for placement selection. The average weight lifted by each robot would be 25kg (i.e., 100kg/4 robots). Figure 5.3 indicates that our algorithm would ensure that each robot is, in fact, lifting within 1.5kg (with a standard



(a)



(b)

Figure 5.2: a) The angle between neighbouring robot teams is equal. b) The teams are distributed throughout the available placement space of the polygon. Notice how no robots are able to be placed in the upper and lower sections.

deviation of 1.9kg) of all other robots. This implies that all robots are lifting between approximately 24.25kg and 25.75kg of the object's mass (0.75kg above, and 0.75kg below the 25kg average).

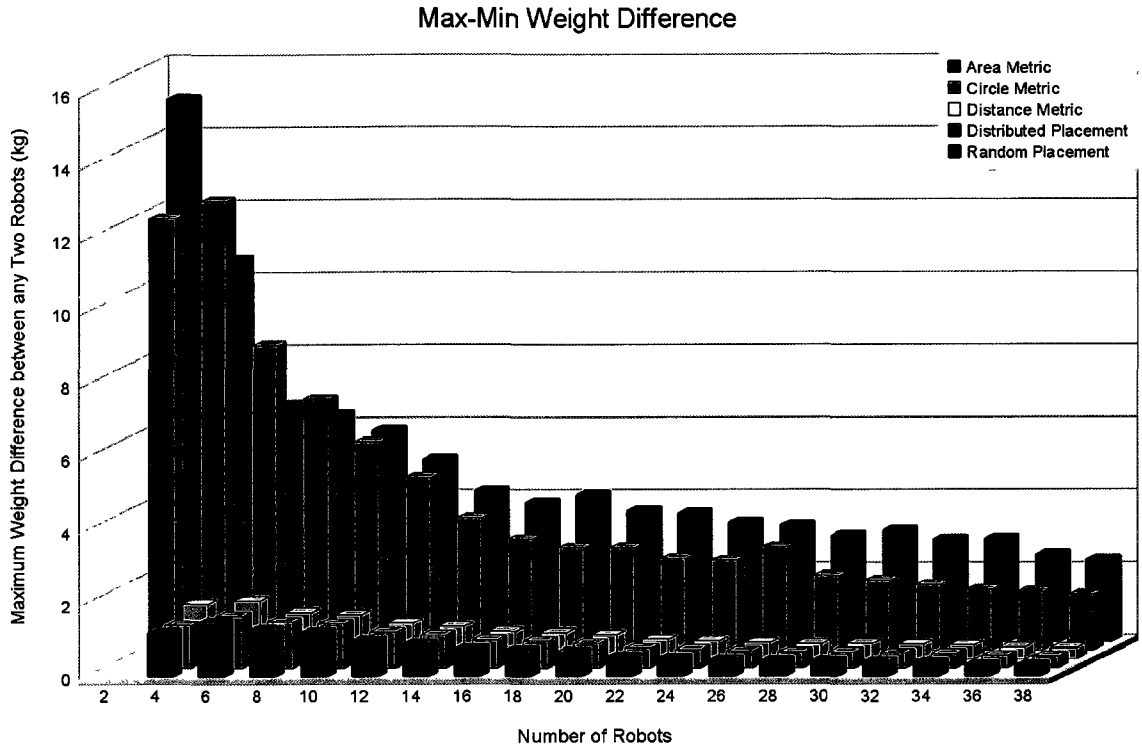


Figure 5.3: Differences between the lifting point with the most weight and the lifting point with the least weight.

If, on the other hand, the robots were to be distributed evenly about the object, the robots will each be lifting within approximately 12.5kg (with a standard deviation of 14.5kg) of all other robots, which implies that they are all lifting between approximately 18.75 and 31.25 of the object's mass.

If 30 robots (15 teams) were used to lift the object, the average weight lifted by each robot would be approximately 3.3kg (i.e., 100kg/30 robots). Using our algorithm with the area metric would have all robots lift within 0.4kg of one another (with a standard deviation of approximately 0.3kg), while randomly distributing the robots about the object would result in them lifting within 3.1kg (standard deviation of 2.5kg) of the rest. Therefore, while our algorithm has all robots lifting between 3.1kg

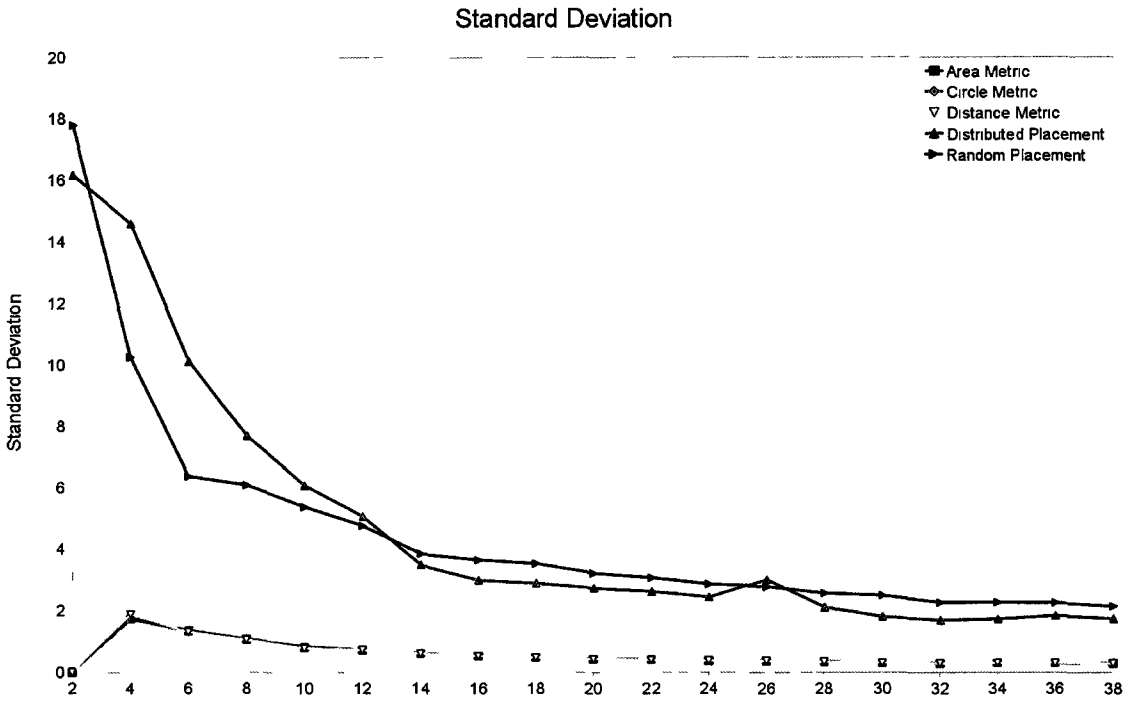


Figure 5.4: Standard deviations of the the values in Figure 5.3.

and 3.5kg of the object's mass, random placements would result in all robots lifting between 1.75kg and 4.85kg of the object's mass.

All placement methods show a generally decreasing difference in weights as more robots are placed. This is to be expected; since each team is assigned equal weight, the maximum difference between any two robots will be limited to this amount. For example, if 20 robots were used to lift the object, each team would be assigned 10% of the object's weight. In the worst case, one robot would be within a small epsilon of the object's centre of mass, while its teammate would be across the centre of mass. In this case, the robot nearest to the centre of mass would be lifting nearly the entire 10% assigned to the team, while its teammate would be lifting nearly none.

Another method of looking at this data is to examine how these differences compare to the weight lifted by each team. Figure 5.5 shows the maximum portion of a team's weight any one robot would lift, with Figure 5.6 showing the relevant standard deviations. Returning to the previous example with 10 teams, our algorithm (regardless of which metric is chosen) has selected placements such that all robots lift

between 46.6% and 53.4% of their teams' respective weight (with a standard deviation of approximately 2.2%), whereas using a distributed placement would have them lifting between 35.5% and 64.5% (standard deviation of 14.0%). Figure 5.5 indicates that, regardless of the number of robots used, our algorithm ensures that every robot lifts approximately half of their team's weight.

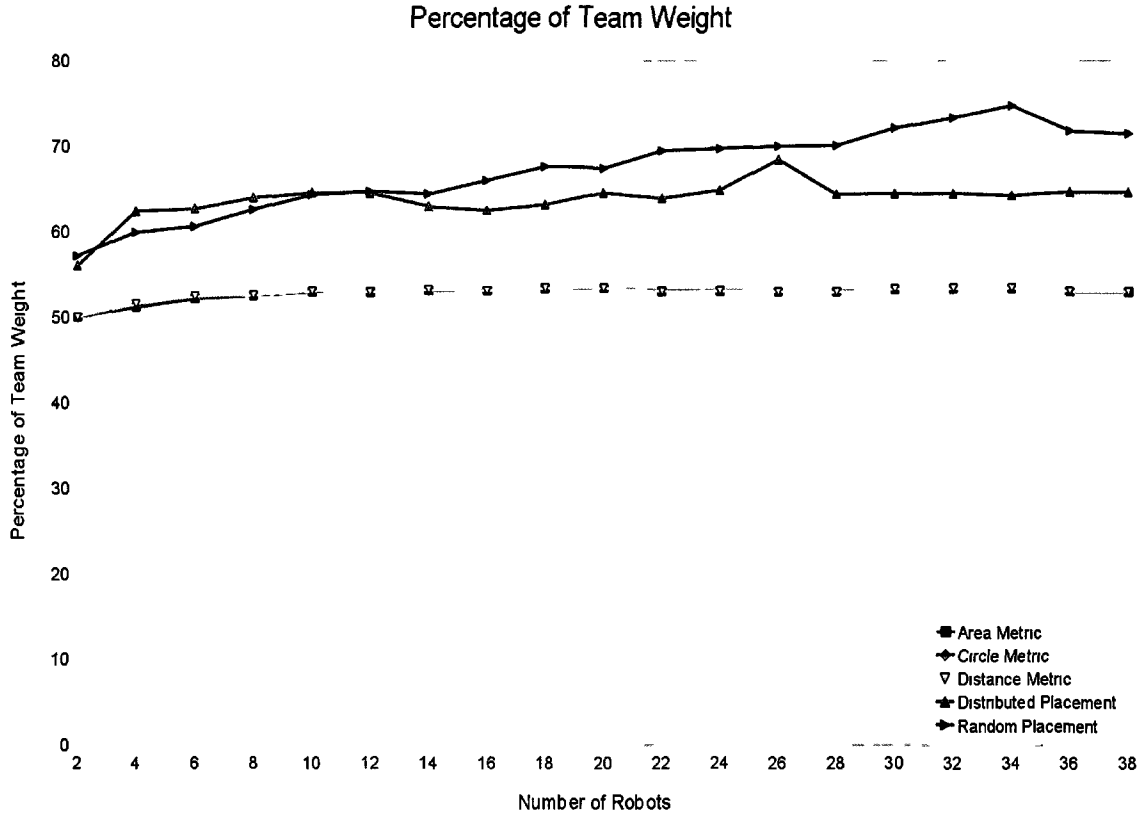


Figure 5.5: The maximum weight lifted by any one robot as a percentage of each team's overall lifted weight. Recall that each team consists of two robots, and that each team has been assigned an equal portion of the object's weight.

By ensuring that all robots bear near equal weight, an implementation of our algorithm on physical robots which have low load-bearing restrictions would help to minimize the number of robots required to lift heavy objects. For instance, if the robots have a capacity to lift only 5kg each and the object to be lifted weighs 100kg, it is clear that at least 20 robots would be required, assuming every robot was lifting exactly their full capacity. Using our algorithm (with a delta value of 0.8) to select

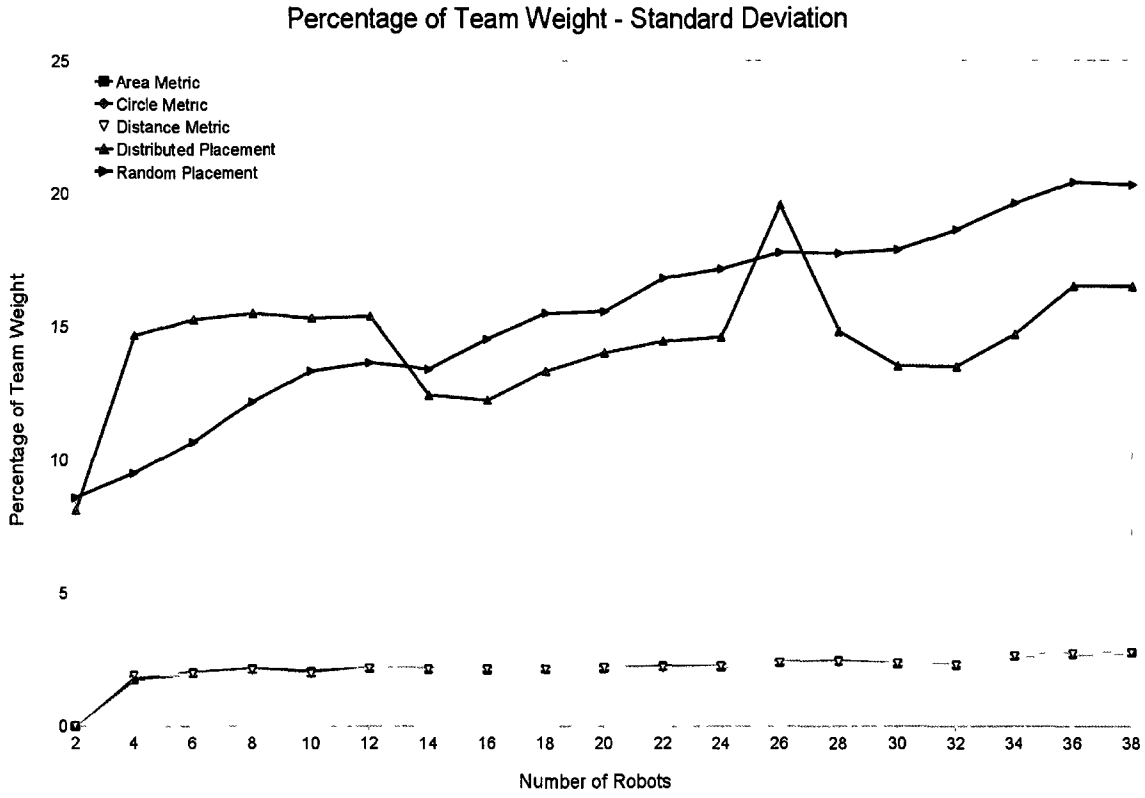


Figure 5.6: The standard deviation for the maximum weight lifted by any one robot as a percentage of each team's overall lifted weight.

lifting points would allow the use of 22 robots while ensuring that no single robot is bearing more than their 5kg maximum load. By comparison, using a distributed robot placement or random placement would, on average, require 28 and 30 robots, respectively.

5.4 Stability

A secondary goal of our algorithm is to ensure that the lifting points selected result in the lift remaining relatively stable. Recall that we defined this stability as preventing a lifted object from being upset by small disturbances, which are modeled as forces applied at different spots on the polygon. Our algorithm aims to prevent such disturbances by maximizing the amount of the polygon's surface that lies within the convex hull of the lifting robots, using one of three different metrics.

In order to examine how well our algorithm achieves stability, we look at how much of the polygon is contained within the convex hull of the lifting points. The intersection of this convex hull and the polygon is obtained, and its area compared with that of the polygon alone. Ideally, the intersection's area is equal to that of the polygon, indicating that there are no points on the polygon which lie outside of the lifters. Any force applied to the surface of the polygon, therefore, would increase the weight on the lifters but would not tip the object.

Figure 5.7 shows the percentage of the polygon's area that is inside the convex hull of the lifting points. The distributed method achieves a slightly higher coverage than our algorithm. This is unsurprising, given the way it spaces out the lifting points. By spreading all the lifters out at equal angles, the distributed placement is far more likely to cover "fat" polygons and areas of polygons.

Interestingly, all three metrics produce very similar coverage. While there is some difference in the coverage values for lower robot numbers (particularly for four and six robots), the difference is very small. This is indicative of the fact that for the majority of polygons, all three metrics select the same placements, or placements that yield very similar coverages. All three metrics chose the same second team placement for approximately 79% of the polygons in our test set (recall that the first team is chosen before the metrics are applied). All three metrics chose the same third placement for approximately 95% of those polygons where the same second placement was selected. In most cases where the metrics choose differently, the coverage difference is small (less than 10%). All this suggests that while there are some instances where there is some benefit to using different metrics, in the majority of cases the difference is negligible or non-existent.

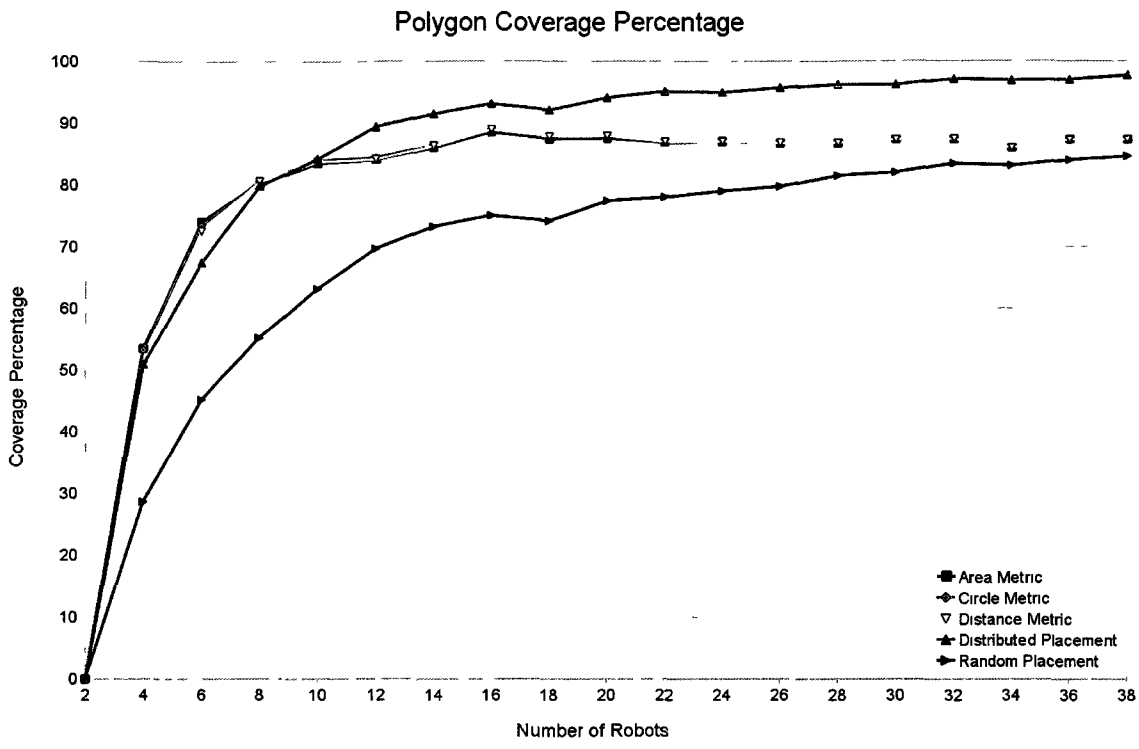


Figure 5.7: The average polygon coverage using the various placement schemes.

Note the relatively high standard deviations in Figure 5.8. Due to the weight distributing nature of our algorithm, at times it is impossible to get high coverage values for a polygon because our primary objective was to distribute weight, not achieve stability. Figure 5.9 shows such a case. By moving the centre of mass slightly, in certain circumstances our algorithm is unable to achieve a high coverage for the same number of robot placements. The difference in coverage for the polygon shown in Figure 5.9 is over 71%. Without the restriction of attempting equal weight distribution, the distributed and random placements are less susceptible this problem.

Despite this, on average, our algorithm achieves comparable coverage with both the random and distributed placements. Our algorithm averages only about 10% less (with 10% standard deviation) polygon coverage than the distributed placement, and covers slightly more than does the random placements.

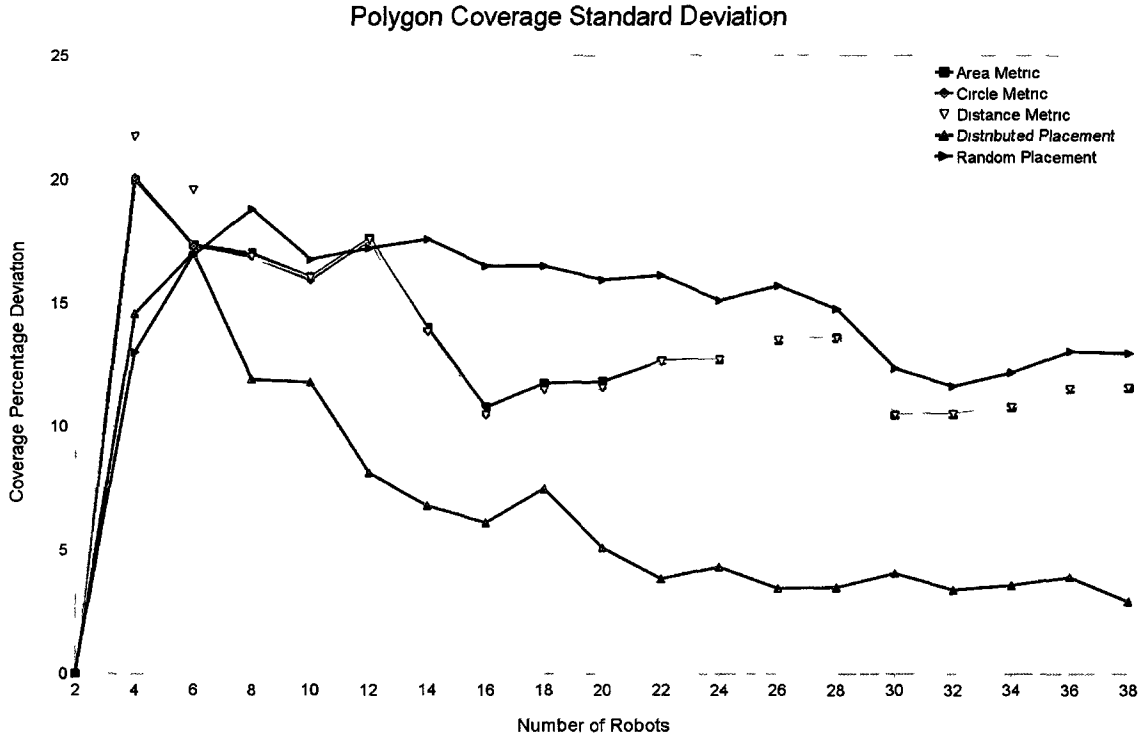
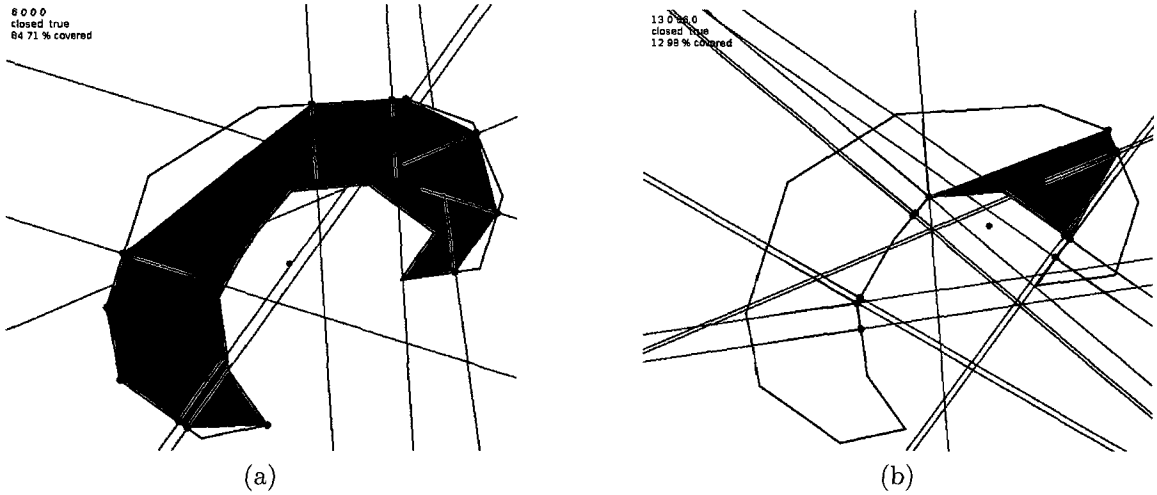


Figure 5.8: The standard deviation of polygon coverage.



(a) (b)

Figure 5.9: Moving the centre of mass can greatly affect the attainable polygon coverage. a) 84.71% coverage with 12 lifting points b) 12.98% coverage with 12 lifting points on the same polygon, after moving the centre of mass.

5.5 Form Closure

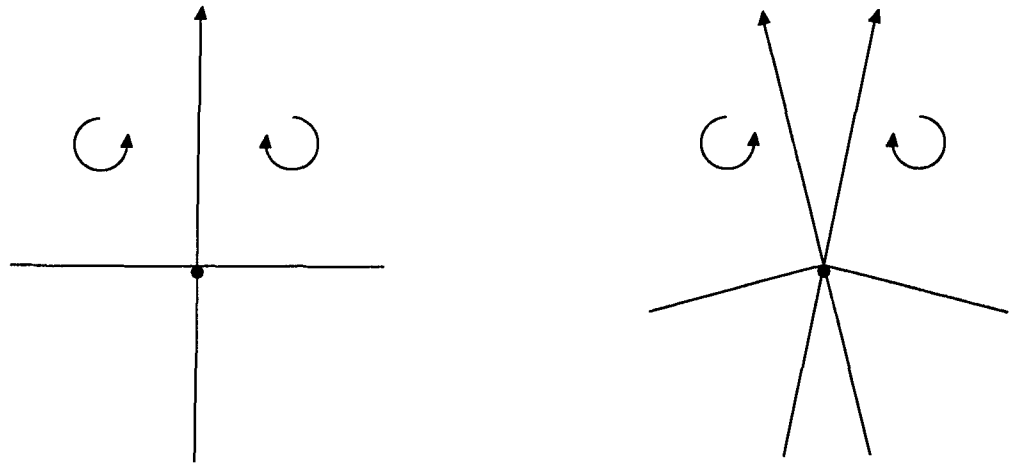
As discussed in Section 3.2, our algorithm does not attempt to tailor its selection of lifting points to facilitate placing the polygon in form-closure. Given the significant portion of literature dedicated to immobilization of objects, however, it is certainly a side-effect worth examining. In order to test whether a polygon is in form-closure, we use the method discussed by [44], outlined below.

Each robot placed imposes some restriction on the ability of the polygon to move within the plane without the robot penetrating the boundary of the polygon. The movement restriction is determined by drawing a line through the robot's position, normal to the edge against which it is placed (Figure 5.10a). The points on each side of this line are the points about which the polygon can be rotated; counterclockwise on one side of the line (the left side, if the line is directed toward the inside of the polygon), clockwise on the other. This line therefore divides the plane into two closed half-planes. As each robot is added, new half-planes are generated. The intersection of each type of half-plane (left or right side) is the area in which the appropriate type of rotations are possible.

When a robot is placed at a vertex, one of two possibilities occurs: either two lines, each defining two half-planes, are contributed to the closure of the polygon, or no half-planes are contributed. The former occurs when the robot is placed at a concave vertex of the polygon (Figure 5.10b). A line is generated for each of the incident edges. When the robot is placed at a convex vertex, it is treated as contributing nothing to the closure of the polygon. If the robot is placed at a convex vertex, it is treated as contributing nothing to object closure.

Eventually, as more robots are added, it is possible that the intersection of both types of half-planes is reduced to zero, at which point the polygon is in form-closure.

Not all polygons were able to be placed in form-closure. There were a number of them (12 in our test set) that, based on the location of the centre of mass and the available number of placements, could not be placed in form-closure regardless of the selection method used. Since form-closure wasn't achieved by any of the selection methods (including all 200 random selections) for these polygons, they were removed from the data set for this test.



(a) (b)

Figure 5.10: a) A robot placed on an edge of the polygon. Clockwise rotations are possible about all points to the right of the vertical line, and counterclockwise to the left. b) A robot placed at a concave vertex of the polygon. Clockwise rotations are possible about all points in the region to the right of both lines, and counterclockwise rotations to the left.

Figure 5.11 shows how well the various selection methods achieve form-closure on the remaining polygons. Each point on the graph shows the closure status of a subset of the testing data. The test set is subdivided based on how many robot placements our algorithm reports are possible. For each successive team to be placed, all those polygons which have already placed the maximum number of possible robots are removed from the testing set. If, for instance, our algorithm is able to place up to eight robots for a given polygon, then that polygon is represented only in subsets for which the number of robots placed is less than or equal to eight. In other words, for a given number of robots, the subset of the test polygons represented on the graph are all those polygons for which at least the indicated number of robots can be placed.

For instance, assume eight robots are to be placed. Of all those polygons in the data set for which our algorithm is able to place at least eight robots, approximately 80% are in form-closure when our algorithm, using the distance metric, places eight robots. For all those polygons in the data set for which our algorithm is able to place at least four robots, approximately 41% are in form-closure by using a distributed

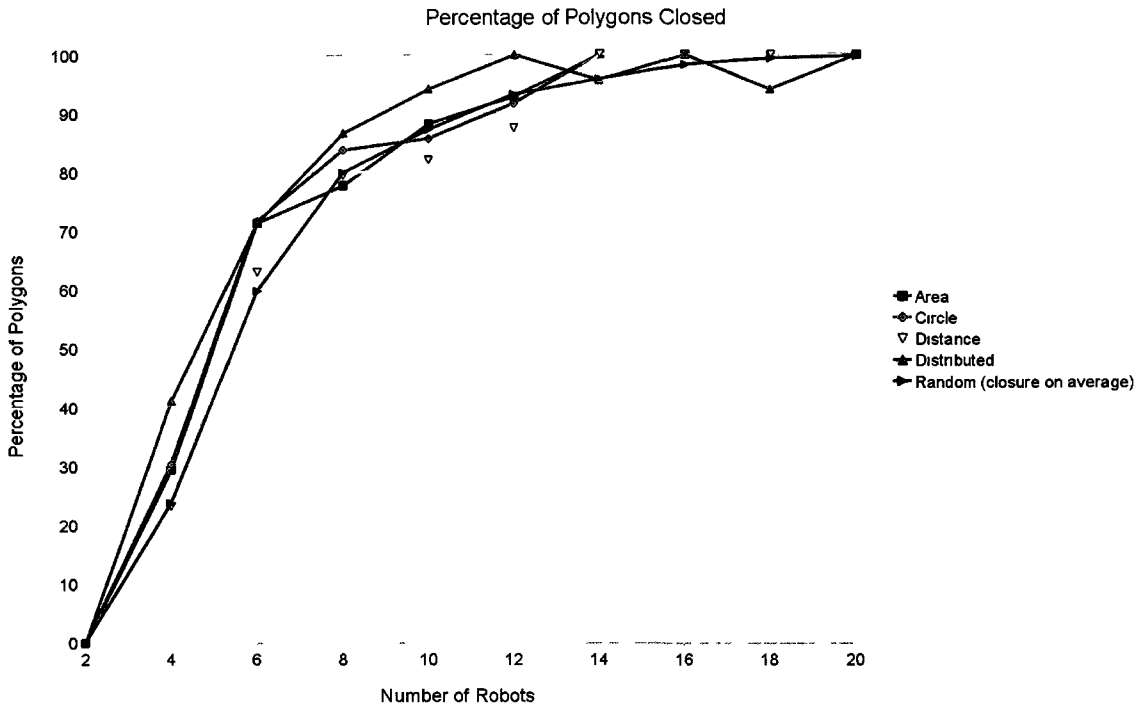


Figure 5.11: The percentage of polygons for which we can place at least as many robots as indicated along the x-axis which are in form-closure by placing the indicated number of robots.

placement with four robots.

All five selection methods perform relatively similarly. There is no single method that clearly achieves this property better than the others for all robot numbers. Our algorithm has the property that once closure is achieved, it is retained for all future teams to be added. This is due to the iterative nature of our algorithm. The addition of another lifting team does not alter the placement of the existing teams, and therefore can only improve closure rather than worsen it. Moreover, our algorithm achieves form closure in over 88% of the polygons in which any one selection method was able to achieve it.

While the distributed selection method has the highest average rate of form-closure when using 12 robots or less, it does not have the property of retaining it when new teams are added. Note how some polygons lose form-closure when going from 12 to 14 robots, and from 16 to 18 robots.

Despite not having been designed to do so, our algorithm achieves remarkably high rates of closure for the polygons of our test set. Those cases where it was unable to achieve closure were largely those polygons in which very few robots were able to be placed due to the restrictions of weight distribution.

5.6 Summary

An implementation of our algorithm was designed and coded, and used to test a representative sample of polygons. We have shown that our algorithm provides good weight distribution while still performing relatively well in achieving lifting stability when compared against two alternative placement selection algorithms. Moreover, our algorithm still manages to place the majority of our testing polygons in form-closure, thus immobilizing them in the horizontal plane.

Chapter 6

Conclusions

6.1 Conclusions

We have presented two variations of an algorithm for placing lifting robots around a polygonal object such that the weight borne by each robot is near equal. The first algorithm, which works for all convex objects, identifies m lifting points for an object with n vertices in $O(mn)$ or $O(mn^2)$ time, depending on the metric used for determining good placements. The second algorithm, an extension of the first, works for non-convex objects and identifies m lifting points in $O(n^4)$ time. The differences in weights borne by the robots, and therefore how near to equal the resulting weight-distribution is, is determined by the user and implementation. The user can then specify whether all robots are to lift exactly the same weight, or whether there is allowed to be some variation.

We have shown how our algorithm effectively distributes weight among the robots, making it simple to ensure that no robots are overburdened or under burdened. In addition, we have shown that our algorithm is generally able to achieve a relatively stable lift, such that external forces are unlikely to cause the robots to drop the object. It allows teams of robots of varying sizes to lift objects, making it much more versatile than existing work.

Our algorithm, to the best of our knowledge, is the first to distribute weight evenly among large teams of lifting robots.

An implementation was created, and a variety of polygons tested, and the results for weight-distribution, stability and form-closure were examined. These results were compared against two alternative placement schemes: a distributed placement in which lifting points were distributed about the object, and a random placement in which lifting points were randomly selected. It was found that our algorithm both alternative robot placement schemes in weight-distribution, and performs similarly in

both stability and closure.

Experiments showed that with the user-defined variability set to 0.8 (allowing the ratio of one robot in a two-robot team to lift 0.8 times as much weight as its partner), our algorithm achieves between 5 and 8 times better weight-distribution than both the distributed and random placements. Even with the restriction on weight-distribution, our algorithm also achieves approximately 87% stability, as measured by polygon coverage. Finally, despite form-closure not being a goal of our algorithm, it was achieved for over 88% of the test polygons in which any one selection method was able to achieve it.

6.2 Future Work

Our work introduces a novel approach to distributing weight among a team of lifting robots. While it serves as a good starting point, there are a variety of places where it can be expanded in future work. The most obvious such extension is to create a physical implementation, using some form of lifting robots. While our algorithm identifies lifting points, it addresses none of the varied challenges involved in a physical implementation, including path-planning, object identification, synchronization or communication. There are a number of unique challenges involved when dealing with physical hardware, and such challenges were outside the scope of this thesis.

In addition to the physical implementation, there are a number of places in which the algorithm can be expanded and improved. First and foremost, while we have shown that the algorithm works for both convex and non-convex polygons, the time complexity for non-convex polygons is significant. There are a number of places where advanced algorithms and data structures would likely be able to improve algorithmic runtime, reducing the current $O(n^4)$ requirements.

As mentioned in Section 3.1, determining unique weight distribution is exceedingly difficult with four or more robots and requires additional information regarding material properties. It is possible, however, to determine the weight distribution with three robots with no additional information. A logical extension, therefore, would be to enable teams of three robots in addition to the teams of two. While this would likely complicate the algorithm, it would also enable larger coverage areas and would

allow any number of robots (greater than one) to be utilized rather than only even numbers of robots, while still maintaining good weight-distribution.

Recall that our algorithm places at most one robot team per edge-pair, and uses exactly one location per edge-pair for team placement. There is, however, potentially a range of places on each edge-pair wherein robots can be placed, depending on the acceptable weight-distribution. An extension to our work could allow robot placements anywhere within this range rather than solely at the spot which most equalizes weight-distribution for the team. This would allow more robot placements, and may allow improvements to the lift stability by increasing polygon coverage while still keeping weight distribution within the acceptable range.

Our algorithm treats all robots as point robots. In any physical system, this would not be the case. In order to place robots around a physical object, the sizes of the robots must be taken into consideration. Another logical extension, therefore, and one which would be required in order to effectively complete a physical implementation, would be to take into account the dimensions of the robots. Not only can no two robots occupy the same physical space, but there may also be some path planning considerations in that there may be sections of the object which are unreachable to the robots.

A final consideration for future implementations regards the grasping mechanism. While we identify lifting points at the boundary of the object, a lifting robot would need to place its "hand" some distance under the object's base. This can require adaptations to the algorithm for certain polygons where two robots are placed across particularly narrow polygon sections. If sections of the polygon are too narrow, it may not be possible to place two robots across that section, and adaptations to the algorithm would need to account for this.

Appendix A

Examples

A.1 Sample Input

Figure A.1.1 A sample input polygon file. Produces the polygon in Figure A.1

```
CoM:(278,227)
(539,220)
(403,115)
(386,219)
(278,306)
(149,187)
(263,124)
(88,101)
(82,320)
(133,437)
(546,446)
(665,426)
```

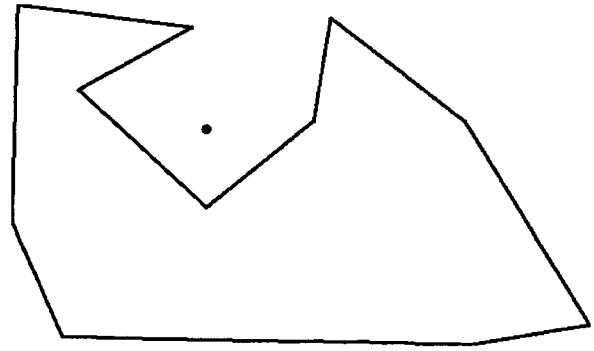


Figure A.1: A sample polygon produce with the input file shown in Figure A.1.1

A.2 Implementation Output Examples

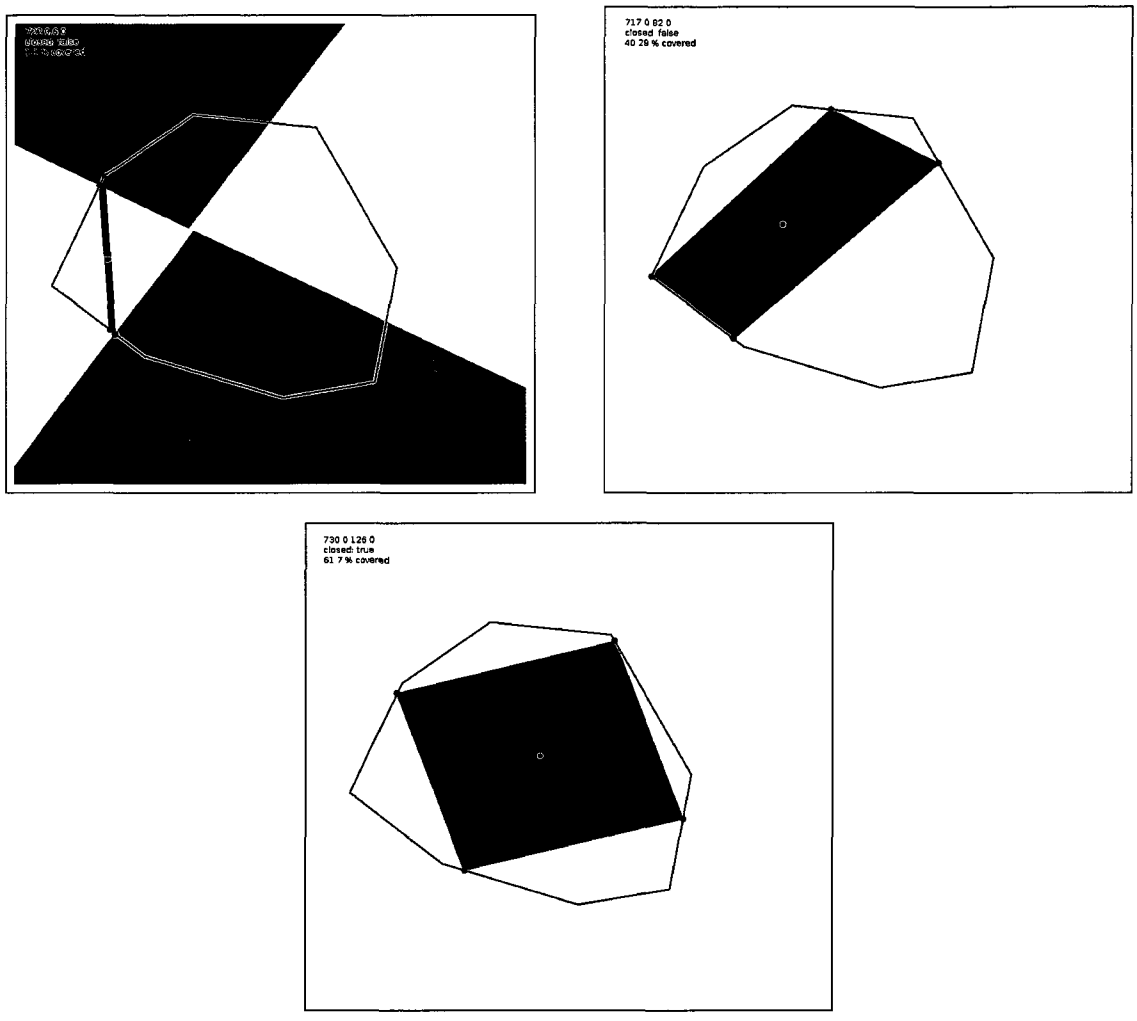
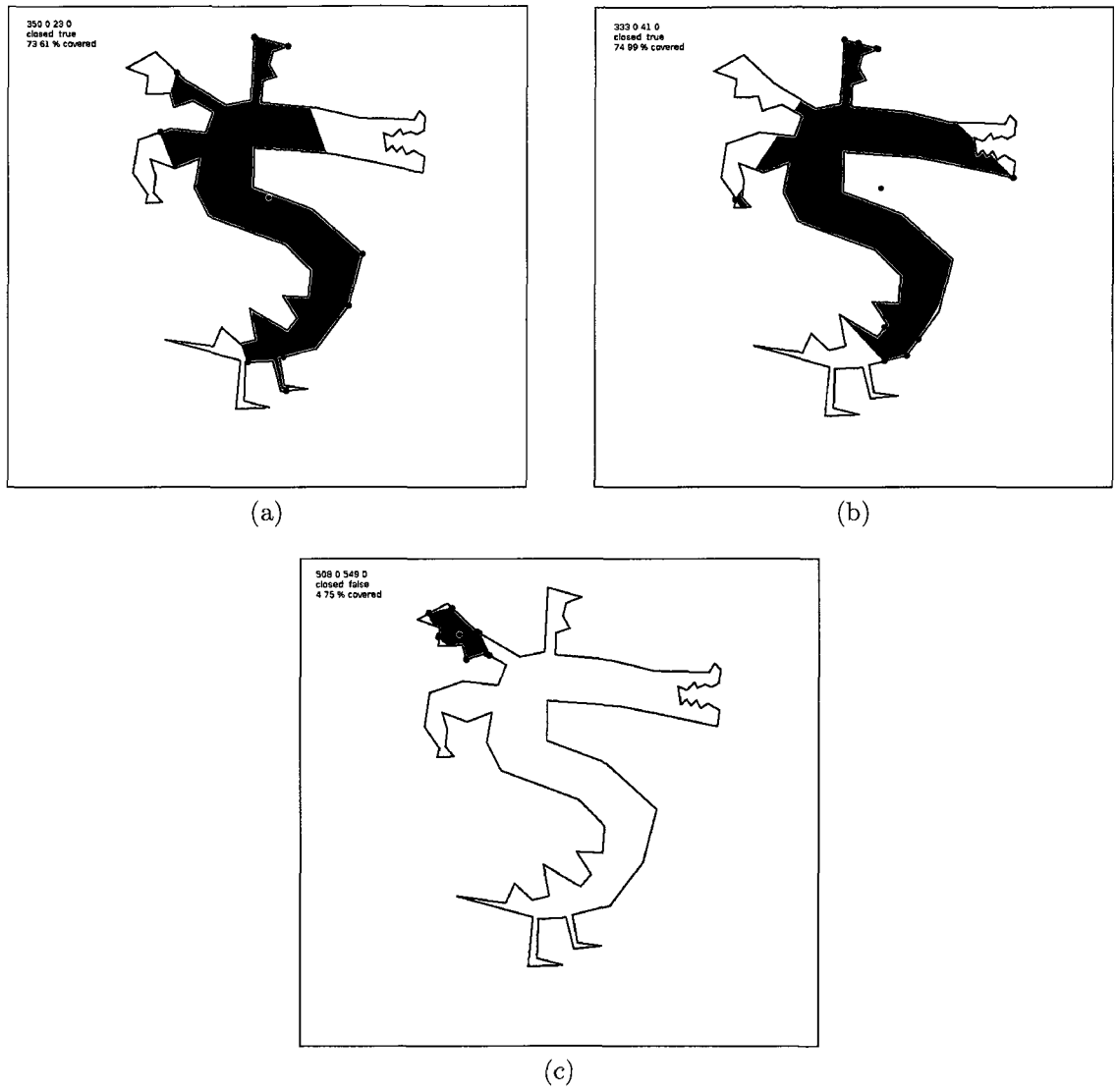


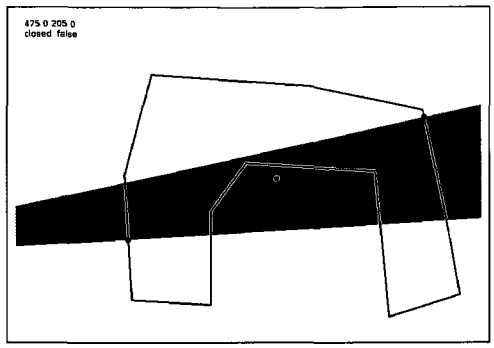
Figure A.2: A convex polygon with two robot teams (4 robots total) placed. The centre of mass in each image is different, showing how different centres of mass can affect robot placement.



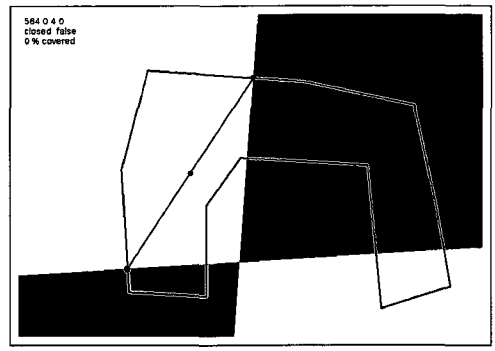
(a) (b)

(c)

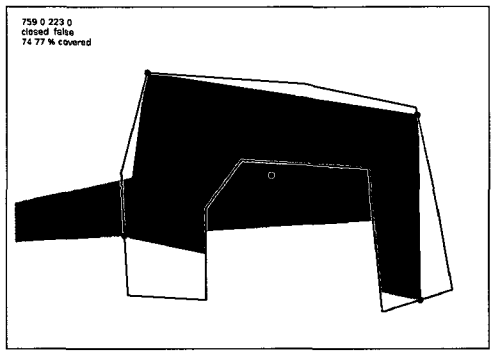
Figure A.3: The maximum coverage is attained in each figure. Additional teams of robots would lessen the weight borne by each robot, but would not increase stability. Figure (a) uses the geometric centre as the object's centre of mass. The centre of mass in (b) and (c) is offset.



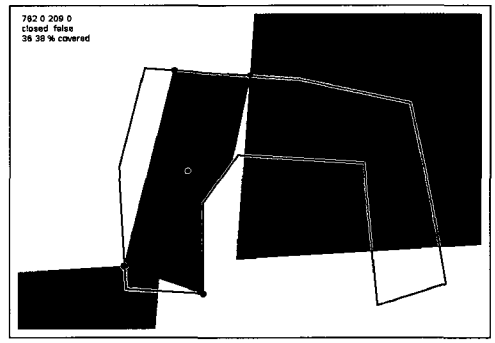
(a) 2 lifters, geometric CoM



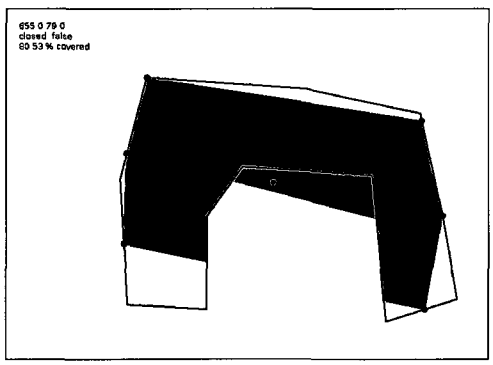
(b) 2 lifters, offset CoM



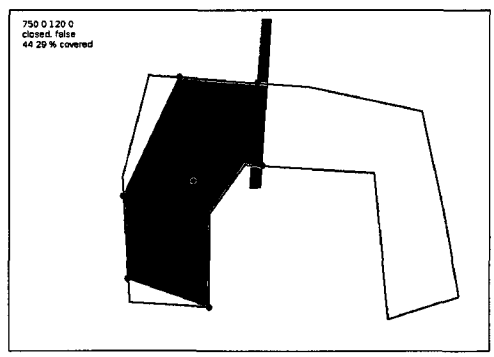
(c) 4 lifters, geometric CoM



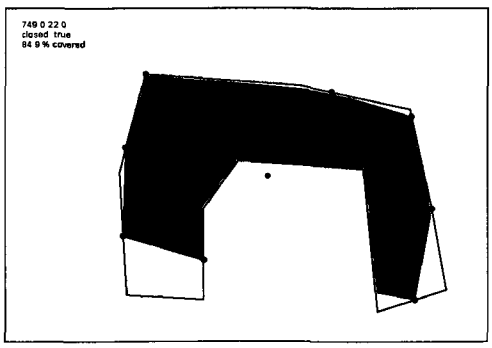
(d) 4 lifters, offset CoM



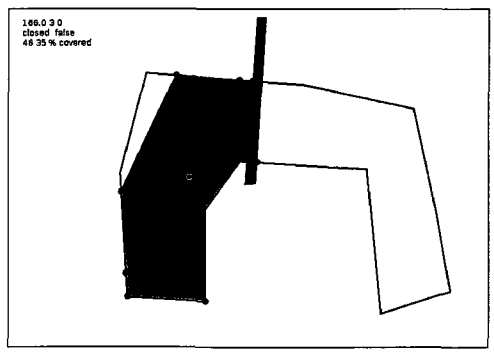
(e) 6 lifters, geometric CoM



(f) 6 lifters, offset CoM



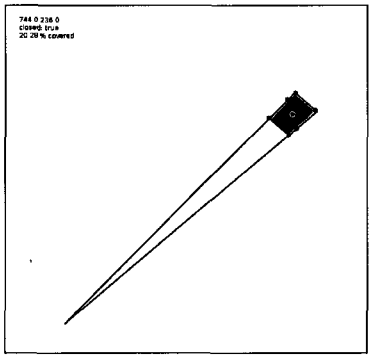
(g) 8 lifters, geometric CoM



(h) 8 lifters, offset CoM

Figure A.4: A horseshoe-shaped polygon whose centre of mass lies at the geometric centre (left images) and offset (right images).

(a) Geometric centre of mass, three lifting teams.



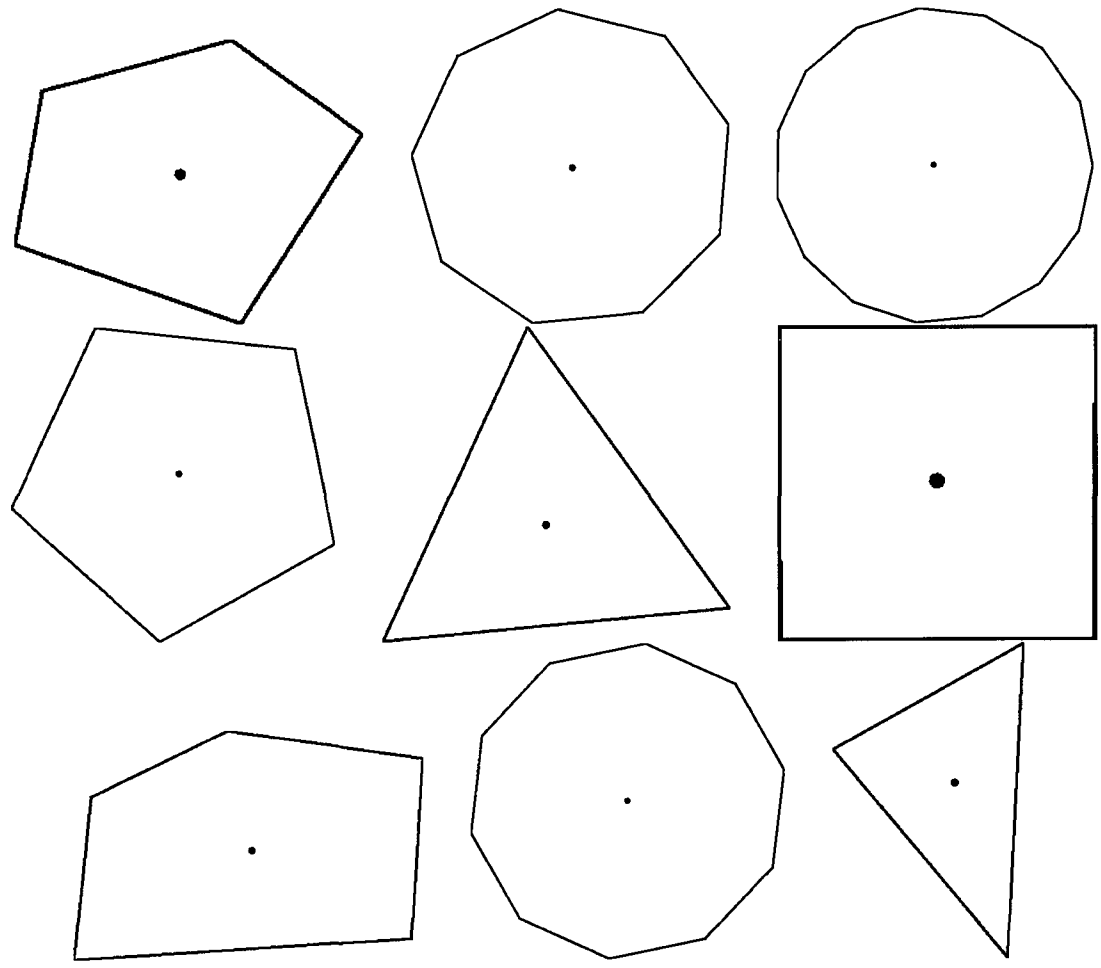
(b) Offset centre of mass, three lifting teams.

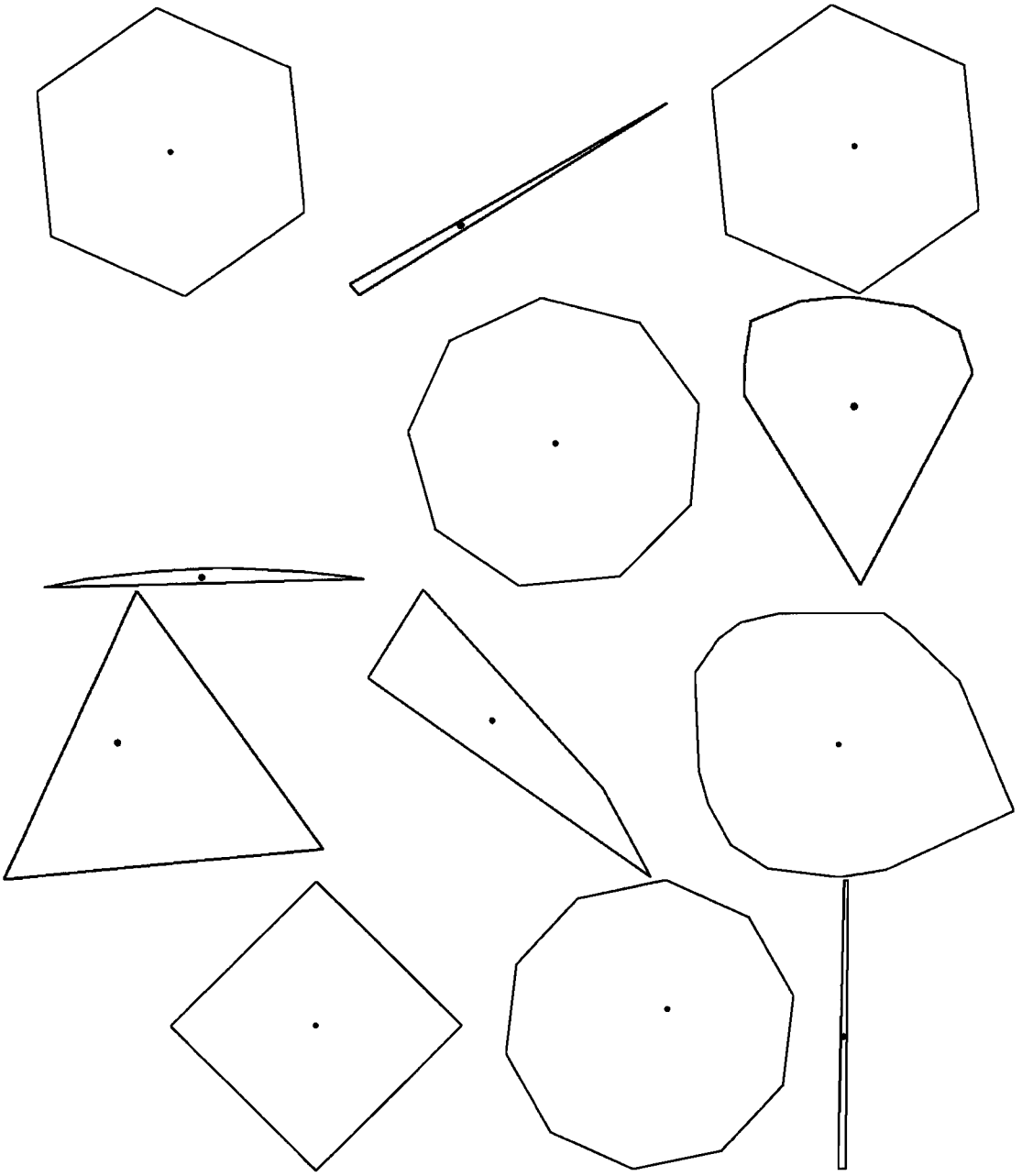
Figure A.5

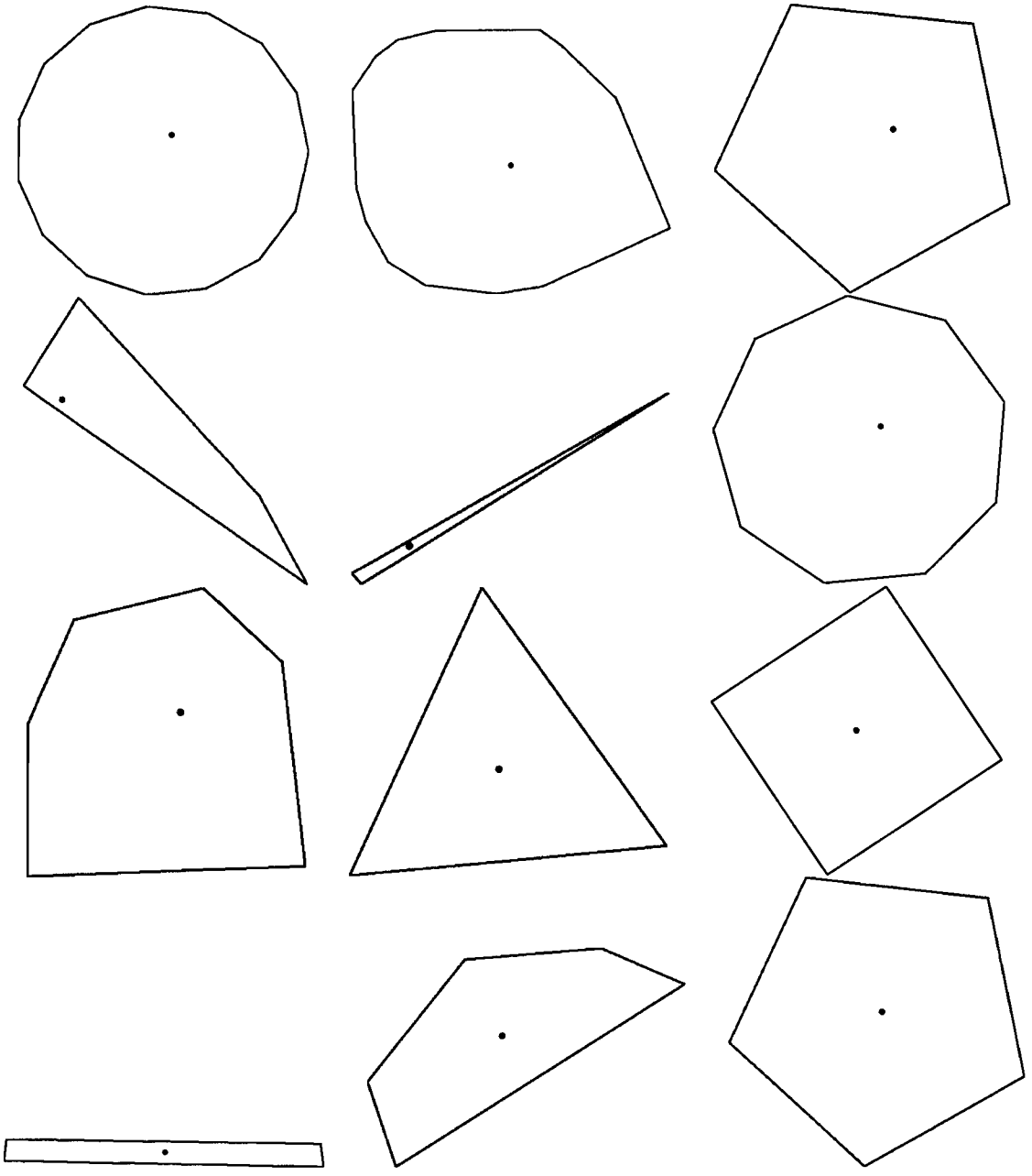
Appendix B

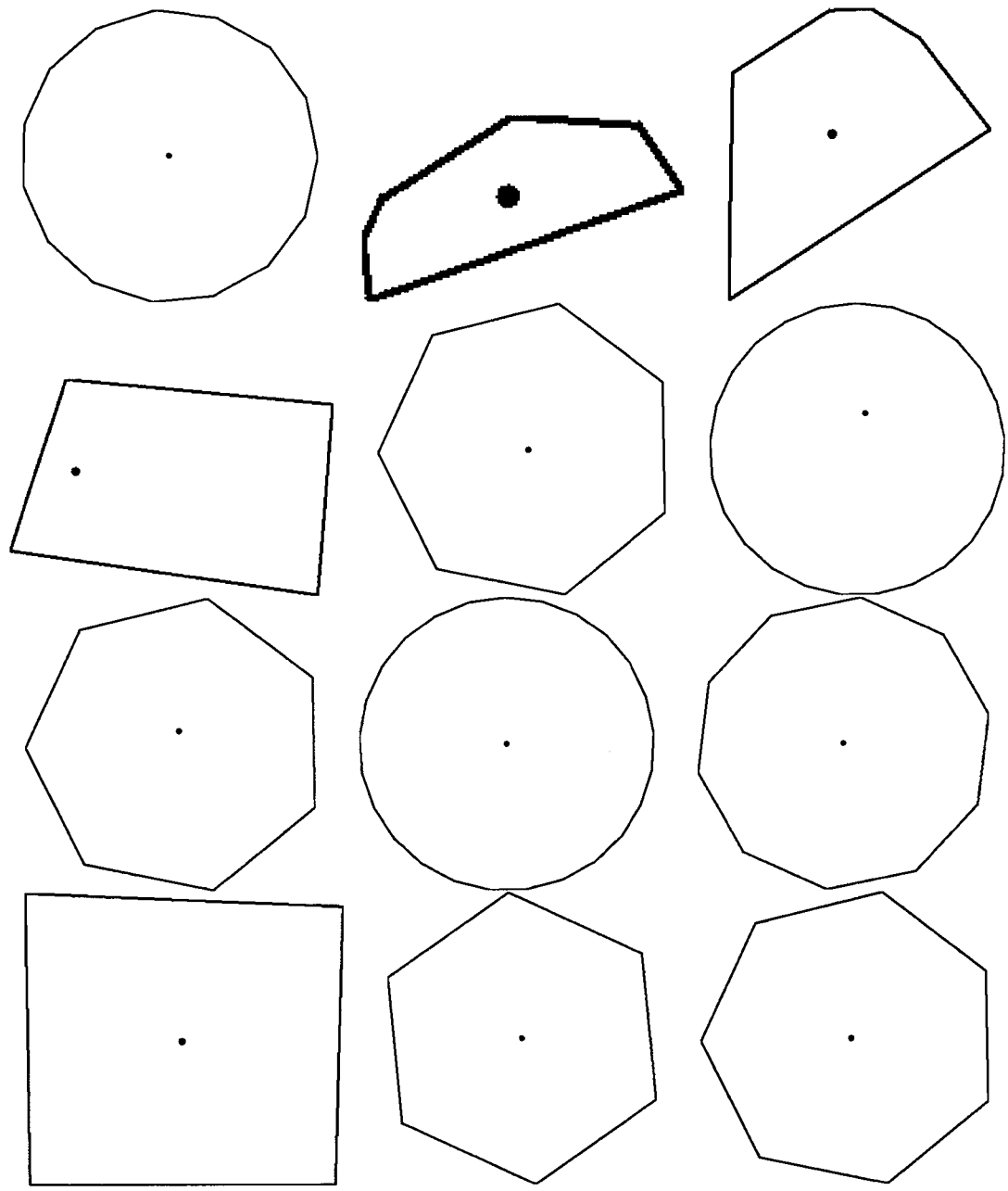
Test Data

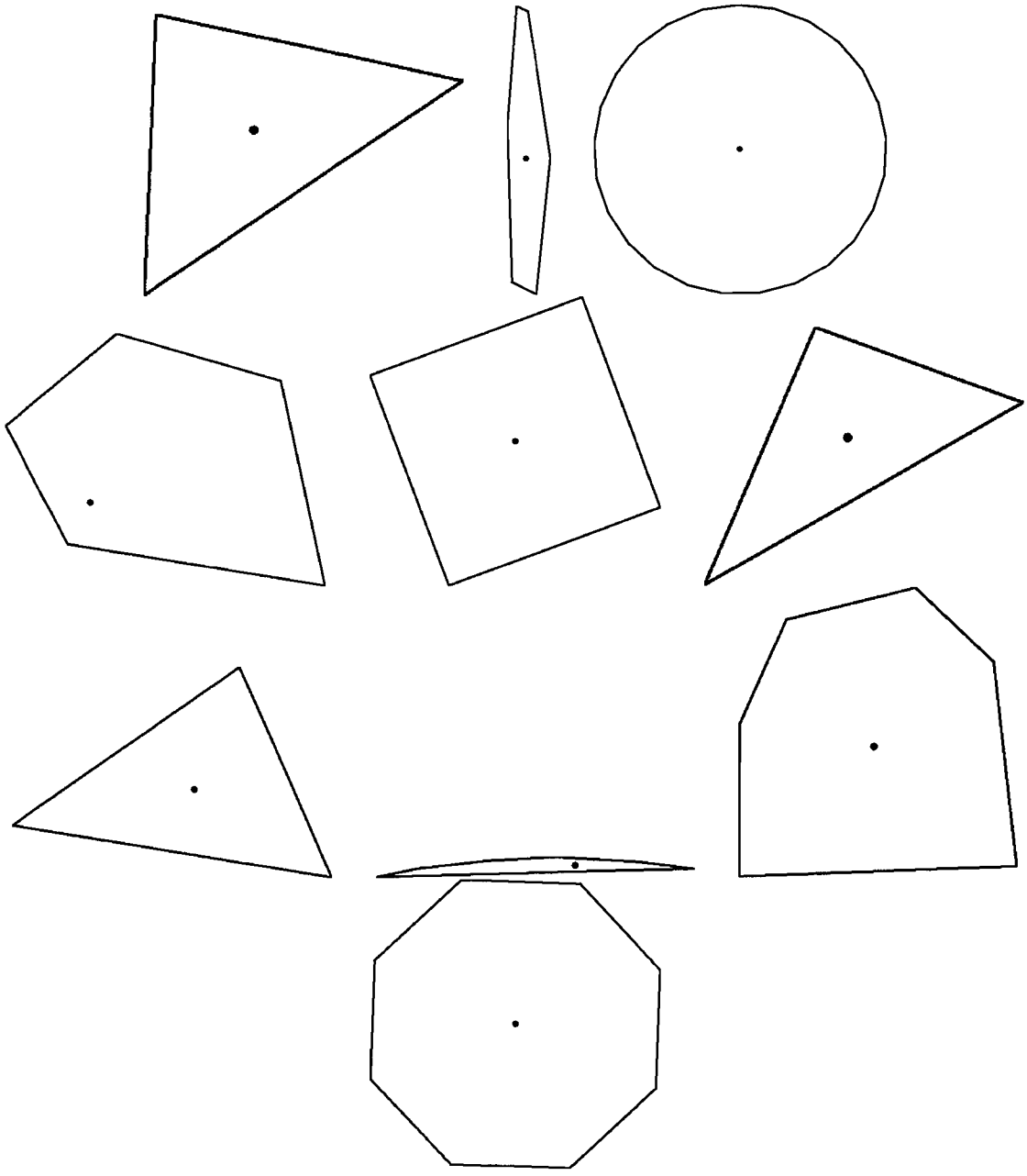
B.1 Convex Test Polygons



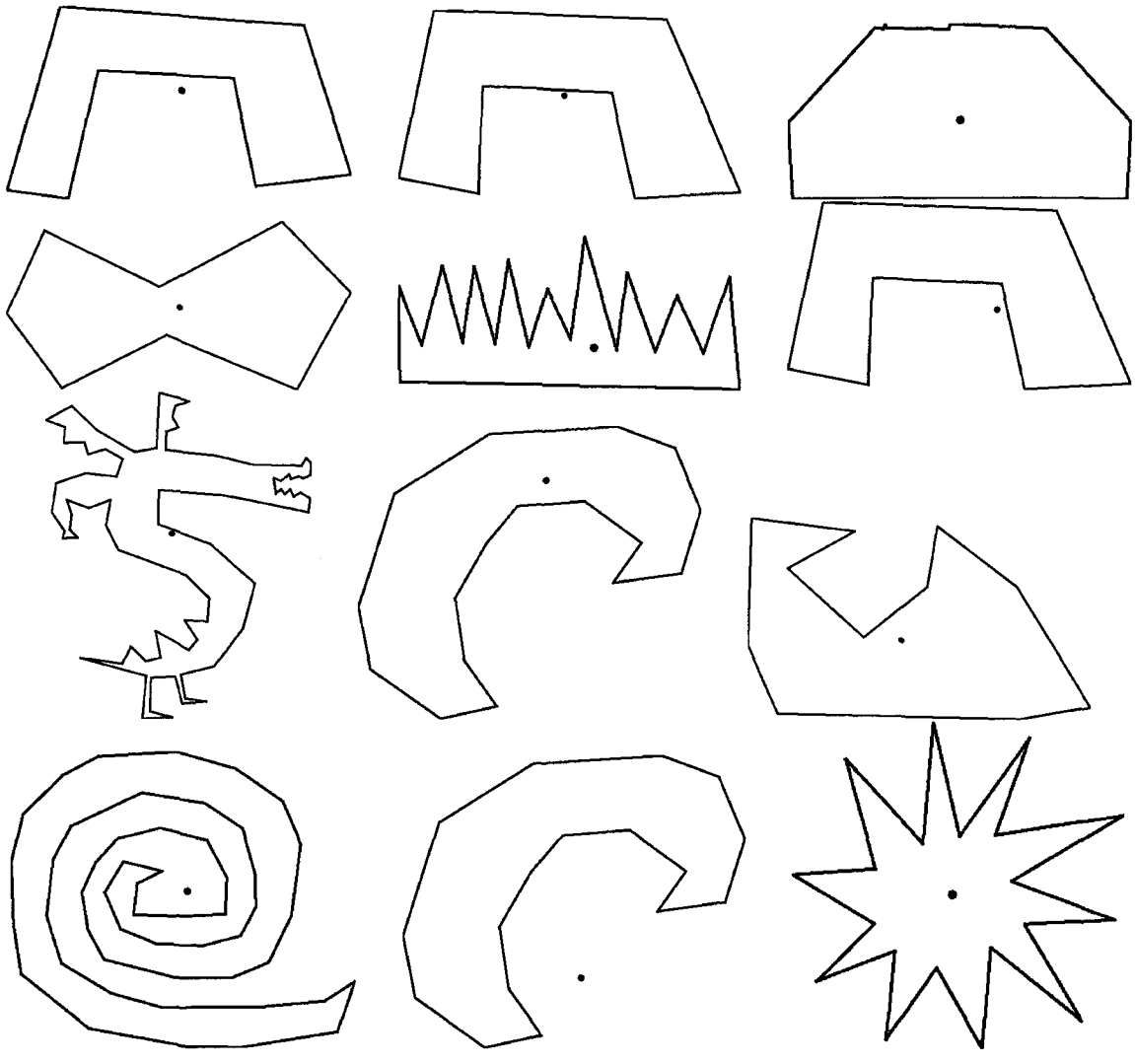


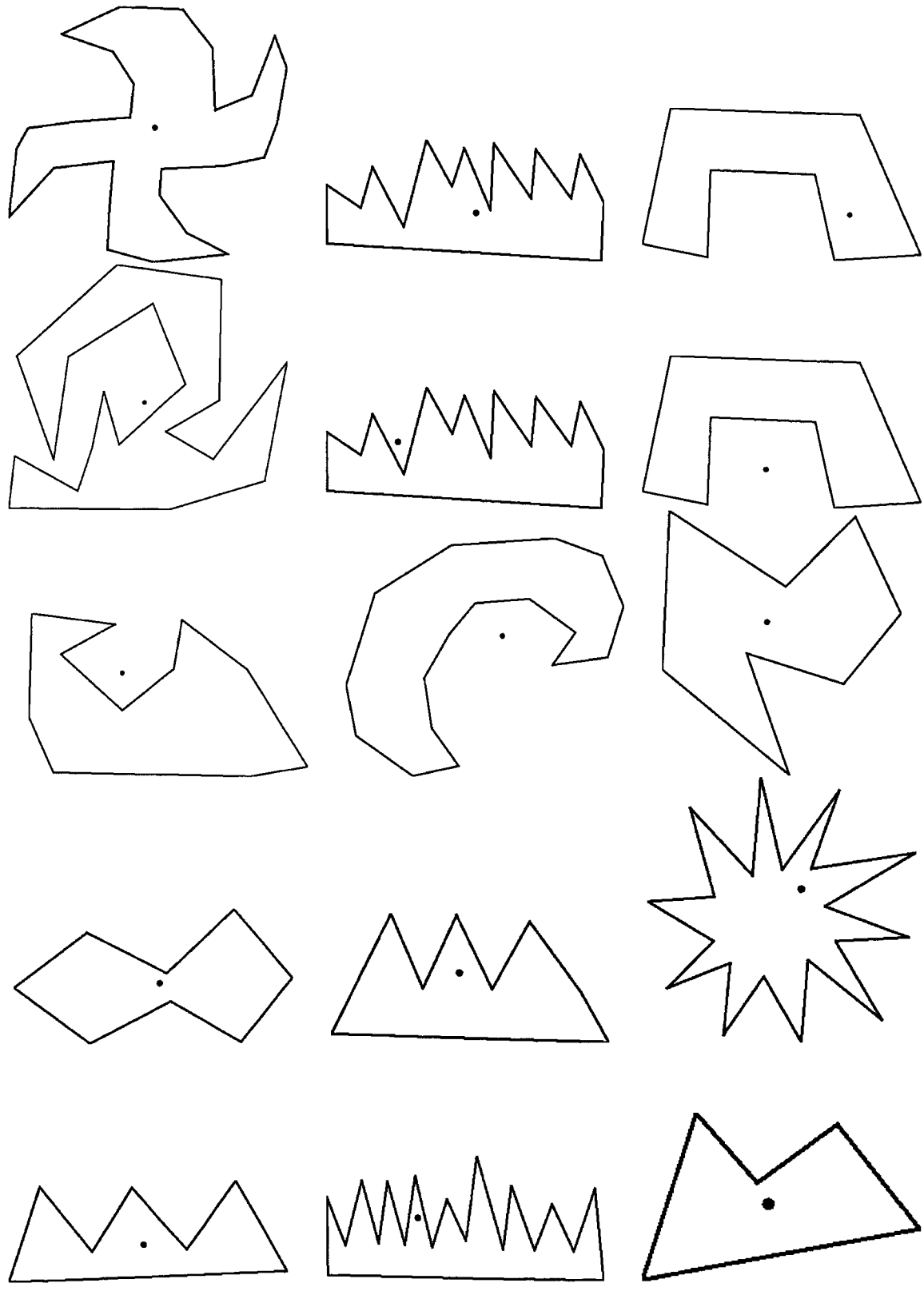






B.2 Non-Convex Test Polygons





Bibliography

[1] M. Ahmadabadi and E. Nakano. A cooperation strategy for a group of object lifting robots. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS '96*, volume 1, pages 125–131. IEEE, 1996.

[2] M. Ahmadabadi and E. Nakano. A constrain and move approach to distributed object manipulation. *IEEE Transactions on Robotics and Automation*, 17(2):157–172, Apr. 2001.

[3] M. Ahmadabadi, S. Rushan, and E. Nakano. A constrain-move based distributed cooperation strategy for four object lifting robots. In *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, pages 2030–2035 IEEE, 2000.

[4] J. Bay. Design of the army-ant cooperative lifting robot. *IEEE Robotics & Automation Magazine*, 2(1):36–43, Mar. 1995.

[5] A. Bendiksen and G. Hager. A vision-based grasping system for unfamiliar planar objects. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 2844–2849. IEEE, 1994.

[6] A. Bicchi. Hands for dexterous manipulation and robust grasping: a difficult road toward simplicity. *IEEE Transactions on Robotics and Automation*, 16(6):652–662, 2000.

[7] G. Bone. Multi-metric comparison of optimal 2D grasp planning algorithms. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, pages 3061–3066. IEEE, 2001.

[8] P. Bose, D. Bremner, and G. Toussaint. All convex polyhedra can be clamped with parallel jaw grippers. In *Sixth Canadian Conference on Computational Geometry*, volume 6, pages 291–302. Elsevier Science B.V., Sept. 1996.

[9] P. Bose, J. Czyzowicz, E. Kranakis, and A. Maheshwari. *Algorithms for packing two circles in a convex polygon*, pages 93–104. Springer Berlin / Heidelberg, 2004.

[10] R. Brown and J. Jennings. A pusher/steerer model for strongly cooperative mobile robot manipulation. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, pages 562–568. IEEE Comput. Soc. Press, 1995.

[11] J.-S. Cheong, H. J. Haverkort, and A. F. Stappen. Computing All Immobilizing Grasps of a Simple Polygon with Few Contacts. *Algorithmica*, 44(2):117–136, Dec. 2005.

[12] J. Cornella and R. Suarez. Efficient Determination of Four-Point Form-Closure Optimal Constraints of Polygonal Objects. *IEEE Transactions on Automation Science and Engineering*, 6(1):121–130, Jan. 2009.

[13] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry - Algorithms and Applications*, chapter 2. Springer-Verlag, second edition, 1998.

[14] B. Donald, L. Gariepy, and D. Rus. Distributed manipulation of multiple objects using ropes. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings*, number April, pages 450–457. IEEE, 2000.

[15] J. Erickson, S. Thite, F. Rothganger, and J. Ponce. Capturing a Convex Object With Three Discs. *IEEE Transactions on Robotics*, 23(6):1133–1140, Dec. 2007.

[16] F. Ghaderi and M. Ahmadabadi. A cooperative fault tolerance strategy for distributed object lifting robots. In *IEEE/RSJ International Conference on Intelligent Robots and System*, pages 2721–2727. IEEE, 2002.

[17] M. Hashimoto, F. Oba, H. Nakahara, K. Imamaki, and T. Eguchi. Trajectory generation and tracking control methods for a multiple transfer robots system. In *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, pages 799–804. IEEE, 1991.

[18] H. Haverkort. Computing All Form-Closure Grasps of a Simple Polygon with Few Fingers. *Algorithmica*, (44):117–136, 2006.

[19] Y. Hirata and K. Kosuge. An Algorithm for Testing Object Caging Condition by Multiple Mobile Robots. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2664–2669. IEEE, 2005.

[20] W. Howard and V. Kumar. On the stability of grasped objects. *IEEE Transactions on Robotics and Automation*, 12(6):904–917, 1996.

[21] I. Kamon, T. Flash, and S. Edelman. Learning to grasp using visual information. In *Proceedings of IEEE International Conference on Robotics and Automation*, number April, pages 2470–2476. IEEE, 1996.

[22] D. Kragic, A. Miller, and P. Allen. Real-time tracking meets online grasp planning. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, pages 2460–2465. IEEE, 2001.

[23] C. Kube and E. Bonabeau. Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, 30(1-2):85–101, Jan 2000.

[24] V. Kumar. Object closure and manipulation by multiple cooperating mobile robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation*, number May, pages 394–399. IEEE, 2002.

[25] X. Markenscoff and C. H. Papadimitriou. Optimum Grip of a Polygon. *The International Journal of Robotics Research*, 8(2):17–29, Apr. 1989.

[26] A. Miller and P. Allen. GraspIt! *IEEE Robotics & Automation Magazine*, 11(4):110–122, Dec. 2004.

[27] B. Mirtich and J. Canny. Easily computable optimum grasps in 2-D and 3-D. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 739–747. IEEE Comput. Soc. Press, 1994.

[28] R. Ozawa, S. Arimo, and S. Nakamura. Control of an object with parallel surfaces by a pair of finger robots without object sensing. *IEEE Transactions on Robotics*, 21(5):965–976, Oct. 2005.

[29] N. Pollard. Synthesizing grasps from generalized prototypes. In *Proceedings of IEEE International Conference on Robotics and Automation*, number April, pages 2124–2130. IEEE, 1996.

[30] J. Ponce and B. Faverjon. On computing three-finger force-closure grasps of polygonal objects. *IEEE Transactions on Robotics and Automation*, 11(6):868–881, 1995.

[31] J. Ponce, S. Sullivan, A. Sudsang, J.-D. Boissonnat, and J.-P. Merlet. On Computing Four-Finger Equilibrium and Force-Closure Grasps of Polyhedral Objects. *The International Journal of Robotics Research*, 16(1):11–35, Feb. 1997.

[32] E. P. Popov. *Engineering Mechanics of Solids*, pages 782–784. Prentice Hall, Upper Saddle River, NJ, second edition, 1998.

[33] D. Rus, B. Donald, and J. Jennings. Moving furniture with teams of autonomous robots. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, pages 235–242. IEEE Comput. Soc. Press, 1995.

[34] J. Sasaki, A. Yamashita, N. Miyata, Y. Aiyama, J. Ota, and T. Arai. Constraint of contacting points in cooperative handling. In *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications*, number October, pages 1425–1430. IEEE, 1998.

[35] R. A. Serway. *Physics for Scientists and Engineers*, chapter 5, 9, 11. Saunders College Publishing, second edition, 1986.

[36] K. Shimoga. Robot Grasp Synthesis Algorithms: A Survey. *The International Journal of Robotics Research*, 15(3):230–266, June 1996.

[37] K. Stanley, A. Jerbi, and W. Gruver. A fast two dimensional image based grasp planner. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients*, pages 266–271. IEEE, 1999.

[38] S. Stansfield. Robotic Grasping of Unknown Objects: A Knowledge-based Approach. *The International Journal of Robotics Research*, 10(4):314–326, Aug. 1991.

[39] D. Stilwell and J. Bay. Toward the development of a material transport system using swarms of ant-like robots. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 766–771. IEEE Comput. Soc. Press, 1993.

[40] A. Sudsang and J. Ponce. On grasping and manipulating polygonal objects with disc-shaped robots in the plane. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation*, volume 3, pages 2740–2746. IEEE, Sept. 1998.

[41] A. Sudsang, J. Ponce, M. Hyman, and D. Kriegman. On manipulating polygonal objects with three 2-DOF robots in the plane. In *Proceedings 1999 IEEE International Conference on Robotics and Automation*, number May, pages 2227–2234. IEEE, 1999.

[42] A. Sudsang, F. Rothganger, and J. Ponce. Motion planning for disc-shaped robots pushing a polygonal object in the plane. *IEEE Transactions on Robotics and Automation*, 18(4):550–562, Aug. 2002.

[43] T. Sugar, J. Desai, V. Kumar, and J. Ostrowski. Coordination of multiple mobile manipulators. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, volume 20, pages 3022–3027. Ieee, 2001.

[44] A. F. Van Der Stappen. Computing Immobilizing Grasps of Polygonal Parts. *The International Journal of Robotics Research*, 19(5):467–479, May 2000.

[45] P. Vongmasa and A. Sudsang. Coverage Diameters of Polygons. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4036–4041. IEEE, Oct. 2006.

[46] Z. Wang, M. Admadabadi, E. Nakano, and T. Takahashi. A multiple robot system for cooperative object transportation with various requirements on task performing. In *Proceedings 1999 IEEE International Conference on Robotics and Automation*, number May, pages 1226–1233. Ieee, 1999.

[47] M. Zhang and K. Goldberg. Gripper point contacts for part alignment. *IEEE Transactions on Robotics and Automation*, 18(6):902–910, Dec. 2002.

[48] X. Zhu, H. Ding, S. Member, and J. Wang. Grasp analysis and synthesis based on a new quantitative measure. *IEEE Transactions on Robotics and Automation*, 19(6):942–953, Dec. 2003.