การกักขังวัตถุทรงหลายเหลี่ยมแข็งเกร็งด้วยการควบคุมการกระจายตัวของนิ้วจุด

นายพีม พิพัฒนสมพร

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554

CAGING OF RIGID POLYTOPES VIA DISPERSION CONTROL OF POINT FINGERS

Mr. Peam Pipattanasomporn

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program in Computer Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2011

Thesis Title	CAGING OF RIGID POLYTOPES VIA DISPERSION CONTROL OF POINT FINGERS
By	Mr. Peam Pipattanasomporn
Field of Study	Computer Engineering
Thesis Advisor	Assistant Professor Attawith Sudsang, Ph.D.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral Degree

..... Dean of the Faculty of Engineering
(Associate Professor Boonsom Lerdhirunwong, Dr.Ing.)

THESIS COMMITTEE

..... Chairman
(Professor Prabhas Chongstitvattana, Ph.D.)

..... Thesis Advisor
(Assistant Professor Attawith Sudsang, Ph.D.)

..... Examiner
(Assistant Professor Witaya Wannasuphoprasit, Ph.D.)

..... Examiner
(Nattee Niparnan, Ph.D.)

..... External Examiner
(Assistant Professor Jittat Fakcharoenphol, Ph.D.)

พีม พิพัฒนสมพร : การกักขังวัตถุทรงหลายเหลี่ยมแข็งเกร็งด้วยการควบคุมการกระจายตัวของนิ้วจุด (CAGING OF RIGID POLYTOPES VIA DISPERSION CONTROL OF POINT FINGERS) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร. อรรถวิทย์ สุดแสง, 125 หน้า.

วิทยานิพนธ์ฉบับนี้มุ่งเน้นการแก้ปัญหาการกักขังวัตถุ ผู้จัดทำเสนอแนวทางในการออกแบบกรงขังโดยการบังคับควบคุมที่ไม่ซับซ้อนผ่านเครื่องควบคุม การออกแบบกรงขังทำได้โดยการกำหนดจุดวางตำแหน่งนิ้วของเครื่องควบคุมเทียบกับตัววัตถุ และ บอกถึงขอบเขตความผิดพลาดที่ยอมรับได้สำหรับการวางตำแหน่ง โดยที่ขอบเขตความผิดพลาดที่ยอมรับได้นั้นวัดจากมาตรวัดตัวหนึ่ง แนวทางที่นำเสนอใช้การแบ่งปริภูมิเป็นชิ้นส่วนย่อยเว้าเป็นพื้นฐาน เพื่อที่จะสร้างแผนผังการเดินทางสำหรับการค้นหากรงขังที่เป็นไปได้สำหรับระบบที่กำหนดให้ เมื่อแผนผังการเดินทางได้ถูกสร้างขึ้น ผู้ใช้สามารถสอบถามเพื่อที่จะทราบว่าการวางตำแหน่งนิ้วแบบที่กำหนดให้ มีสภาพเป็นกรงขังวัตถุสำหรับระบบที่ให้มาหรือไม่ หรือ ขอรายการกรงขังวัตถุทั้งหมดที่เป็นไปได้ในระบบอย่างมีประสิทธิภาพ แนวทางดังกล่าวสามารถนำไปขยายผลเพื่อจัดการกลุ่มนิ้วที่มีข้อจำกัด และ/หรือ สำหรับจัดการกับความไม่แน่นอนในระบบ

ภาควิชา	...วิศวกรรมคอมพิวเตอร์...	ลายมือชื่อนิสิต .....
สาขาวิชา	...วิศวกรรมคอมพิวเตอร์...	ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก .....
ปีการศึกษา	..... 2554	

##4971822121 : MAJOR COMPUTER ENGINEERING
KEYWORDS : CAGING / GRASPING / MANIPULATION

PEAM PIPATTANASOMPORN : CAGING OF RIGID POLYTOPES VIA DISPERSION CONTROL OF POINT FINGERS. ADVISOR : ASST. PROF. ATTAWITH SUDSANG, Ph.D., 125 pp.

This dissertation addresses the problem of object caging. The author propose approaches for designing "cages" by means of imposing simple constraints to control the manipulator. Each designed cage is represented in the form of a finger placement relative to the object and its error tolerance. The error tolerance determines how much the fingers may deform based on a measure. The proposed approaches are based on decomposing the workspace into convex subsets, then constructing a roadmap to identify possible cages for a given system. After a roadmap is constructed, it is possible to query if a finger placement forms a cage or enumerate all the cages efficiently. The approaches can be extended to handle constrained fingers and/or uncertainty in the system.

Department : . Computer Engineering . Student's Signature . . . . . .
Field of Study : . Computer Engineering . Advisor's Signature . . . . . .
Academic Year : . . . . . 2011 . . . . . .

Acknowledgments

I would like to express my gratitude to my advisor, Asst. Prof. Attawith Sudsang who has supported and helped my personal problems, broaden my vision and trained me in theoretical robotics from the very start of my academic career path. I would like express my thankfulness to my dissertation committee: Prof. Prabhas Chongstitvatana, Asst. Prof. Witaya Wannasuphoprasit, Asst. Prof. Jittat Fakcharoenphol, and Dr. Nattee Niparnan.

Behind my successful academic life, I would like to thank all my family members. Especially, my mother who always support me physically and mentally. For my successful academic research, I could not have been this far without Asst. Prof. Nataphan Kitisin, who insightfully taught me all his lecture courses in mathematics, and mathematicians' disciplines. I must apologize him here that I cannot fullfill his request to get another ph.d. in mathematics. Also, I would like to thank Assoc. Prof. Phichet Chaoha who taught me topology-related courses.

Parts of this works have been inspired from discussions with my friends and juniors. I would like to mention Pawin Vongmasa whom I always discuss with during his annual journey home. I would like to thank Yuttana Suttasupa who help me work on a commercial project and lift a load off me. I would like to mention Pasakorn Tangchanachaianan who inspire me my latest work just added before finishing this disseration. I would like to thank Peerapong Thonnagith and Thanathorn Phoka who keep reminding me of academic schedules and help me on miscellaneous stuff. I also would like to thank all other people, past and present, in the department of computer engineering, the Intelligent System Laboratory 2 (ISL2) at Chulalongkorn University. I appreciate the financial support from Chulalongkorn University graduate scholarship to commemorate the 72^{nd} anniversary of his majesty King Bhumibol Adulyadej, the Thailand Research Fund through the Royal Golden Jubilee Ph.D. Program under Grant No. Ph.D. 1.O.CU/51/D.1, and the 90^{th} Anniversary of Chulalongkorn University Fund through the Ratchadapiseksomphot Fund.

Contents

Chapter Page

List of Tables

List of Figures

Figure Page

Figure Page

Figure Page

CHAPTER I

INTRODUCTION

Caging an object is to confine the object within a bounded region. The problem of object caging was originally posed by Kuperberg (Kuperberg, 1990) as a problem of designing formation of points to prevent an object from escaping to infinity by any continuous rigid motion. A caged object is not necessary immobilized. Studies in caging generally involve attempts to loosely envelope the object by means of simple and robust strategies that tolerate uncertainty and imprecision in measurement and control. In the past few decades, the concept has been applied to various tasks, for example, grasping and in-hand manipulation (Rimon and Blake, 1996), (Davidson and Blake, 1998a), (Davidson and Blake, 1998b), (Gopalakrishnan and Goldberg, 2002), (Sudsang et al., 1997) motion planning (Sudsang et al., 1999), (Sudsang et al., 2002) part feeding (Blind et al., 2001), stable stance computation (Kriegman, 1997), (Rimon et al., 2008). The manipulator to achieve caging may be a single or multiple components that work altogether as a cage. They can be mobile robots, fingers of grippers, arrays of pins, or cylindrical rods.

Grasping an object is to firmly hold the object by the manipulator. Computation of grasps involves finding contact points for the manipulator to effectively counter balance external forces and torques. Stable grasp criteria are usually based on the force closure and the form closure conditions (Bicchi and Kumar, 2000). In the past decades, computed grasps does not give satisfactory result when applied in practice. A primary cause of grasp failure is that the manipulator cannot simultaneously make appropriate contact with the object. Let alone exerting precise forces and torques to balance out external disturbances. Nonsimulataneous contacts usually lead to undesirable forces and/or torques pushing the object away from its initial position. It is difficult for the sensors to accurately keep track of the object pose and for the manipulators to appropriately react to the change. As a result, the fingers may not reach the precomputed contact points, and the object may slip away from the manipulator. A more reliable approach is to form a cage prior to a grasp.

That is to confine the object in a bounded region before an attempt to grasp it. The problem of nonsimultaneous contacts is solved as the manipulator does not need to maintain contact with the object in order to form a cage. After an appropriate cage setup, the knowledge of the object pose is not necessary to maintain the cage. The manipulator can continue to cage the object as long as the shape of the manipulator does not change.

Some cage serves as a waypoint for grasping. As the cage shrinks, the freespace which the object can move also shrinks. Eventually, the manipulator grasps the object. This process is called *error-tolerant grasping* (Davidson and Blake, 1998b). The cage established prior to the grasp is called a *pregrasping cage* (Rodriguez et al., 2011). An error-tolerant grasping is a grasp strategy that provides the required manipulation robustness and precision.

Given a sufficient number of fingers, object caging on a plane can be achieved by evenly placing fingers in a circle formation to surround the object. As long as the distance between any pair of adjacent fingers is kept under an upperbound such as the object's *diameter* (Sudsang, 2002), or *coverage diameter* (Vongmasa and Sudsang, 2006), the object cannot escape. However, this often leads to inefficient utilization of fingers as two fingers are sufficient to cage most concave objects. Rimon and Blake (Rimon and Blake, 1996), have laid fundamental concepts in caging and proposed a numerical solution to determine a *caging set* that contains a given immobilizing grasp of two fingers. A caging set is a maximally connected set of configurations that the object cannot escape by any rigid motion. Caging with two fingers can be classified by the way their separation distance is maintained. One is caging by squeezing fingers. Moving the fingers closer together shrinks the cage. The other is caging by stretching fingers. The cage shrinks as the finger separation distance increases Examples of a squeezing and a stretching cage are shown in Figure 1.1. The dark dots and the shaded regions in the figure represent fingers and objects, respectively.

It has been proven by Rodriguez and Mason (Rodriguez and Mason, 2008) that

a caging set of two finger caging can only be either squeezing or stretching or both, for any compact connected contractible object. Depending on the type of the cage, the object could escape if the distance is either below or above a certain value called the *critical distance*. A critical distance is either *maximal distance* or *minimal distance*. The maximal distance is the greatest distance such that the object cannot escape as long as the fingers' separation distance does not exceed such distance. Conversely, the minimal distance is the smallest distance that the object cannot escape as long as the fingers' separation distance does not fall below the value. The maximal and the minimal distance serve as an upperbound or a lowerbound separation distance to maintain the caging by squeezing and stretching, respectively. Vahedi and van der Stappen (Vahedi and van der Stappen, 2006), Sudsang and Pipattanasomporn (Pipattanasomporn and Sudsang, 2006) independently proposed algorithms that report all two-fingered caging sets for a given polygonal object. As the number of fingers increases, the problem of reporting all caging sets becomes more complex. One reason is that the caging set associated with a caging configuration can no longer be parameterized by just a critical distance. Erickson et al. (Erickson et al., 2003) studied the problem of caging convex object with three fingers and proposed both exact and approximate algorithm to render capture region assuming that two robots are fixed on the boundary of the convex object. Their work was extended to non-convex polygon by Vahedi and van der Stappen (Vahedi and van der Stappen, 2008).

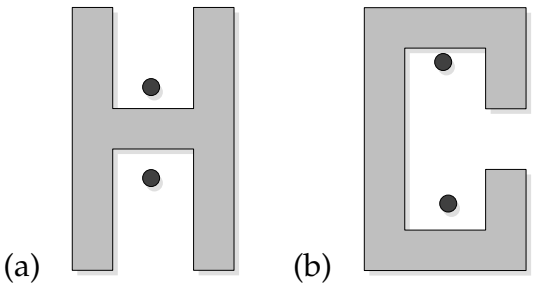

Figure 1.1: (a) caging by squeezing fingers. (b) caging by stretching fingers.

To the best of our knowledge, most caging works are restricted to two-dimensional workspace and a few number of fingers. Caging with more than three fingers mostly remains unexplored. Even for the three-finger caging problem, none

of the published works have proposed an efficient method to create a complete catalog of all caging sets. Though caging with diameter or coverage diameter is an easy and intuitive way to cage objects when many fingers are available, it does not provide significant associations with possible error-tolerant grasps the way caging set based approaches can.

The goal of this research is solve the caging problem for a system embeded in two or three-dimensional space with an arbitrary number of fingers. Constructing an algorithm that determines all caging sets for a given system seems to be the ultimate goal to pursue at first. However, it is too computationally expensive to compute the caging sets for a complex system. More fingers leads to a more complex caging set. A more complex caging set leads to a more complex cage bound that cannot represent by a critical value. Processing potrayed caging sets formed by high-dimensional arrangements or algebraic hypersurfaces can be very resource consuming as well. Regardless of the complexity, the critical information is not the caging set as a whole. It is just a robust strategy to form a cage suitable for a specific task.

1.1 Problem Formulation

Essentially, object caging is all about mutual geometrical obstruction created by an object and a manipulator. An object can be a single or multiply connected components. A cage is to be formed by the manipulator, here, a set of fingers. The fingers are assumed to be free-moving point fingers at first. Extension to disc-shaped fingers and constrained fingers are presented in later chapters. In the classical definition (Kuperberg, 1990), the object is said to be caged as long as the object cannot travel arbitrarily far from the fingers, see Figure 1.2(a). If the object can travel far from the fingers, it is said to escape from the cage, see Figure 1.2(b). On the other hand, if the same phenomenon is observed from the object frame of reference instead of a static one, the formation of fingers will be observed as rigidly moving together towards infinity, escaping from the obstructing object, see Figure 1.2(c). Whether an object is caged or not can be observed from any frame of reference.

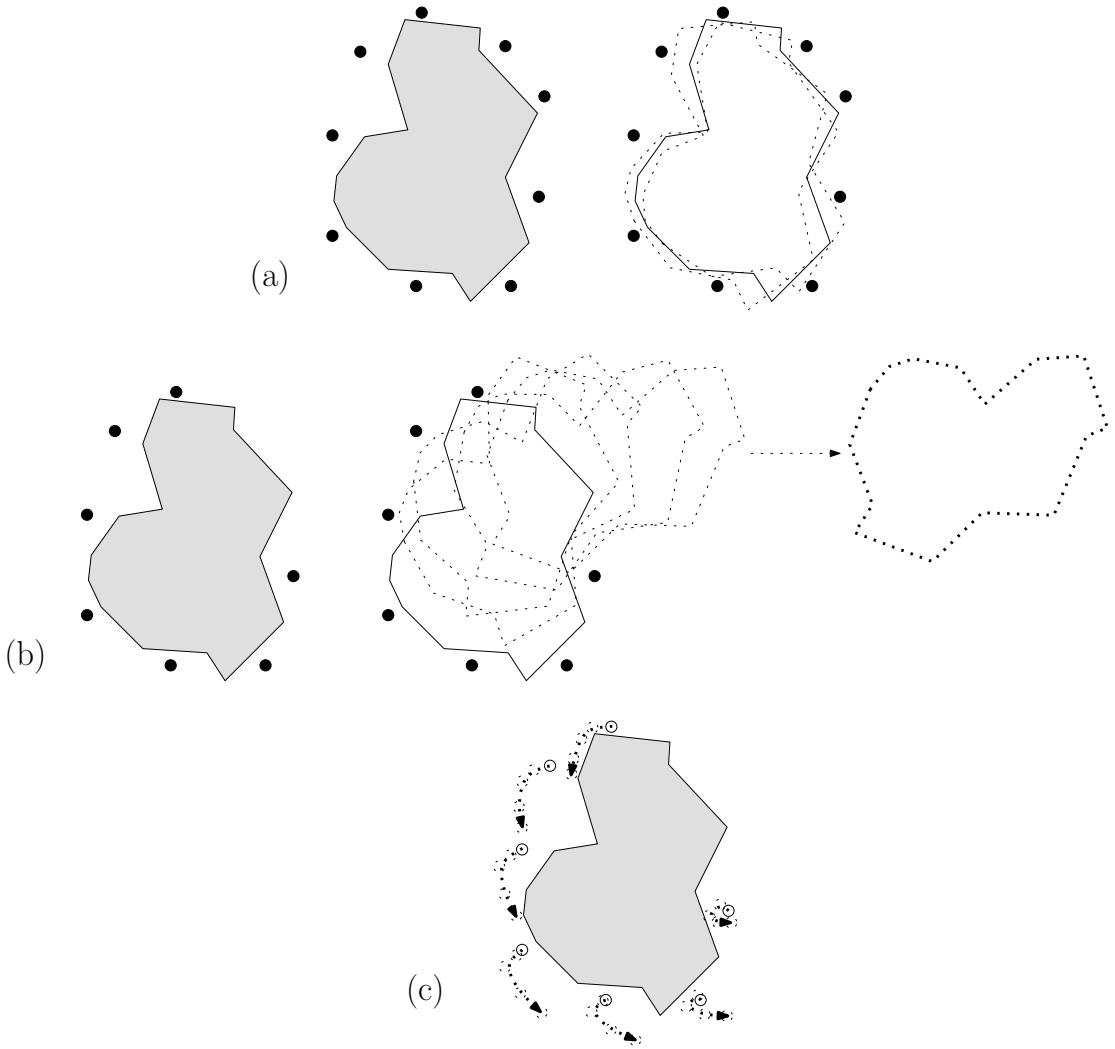

Figure 1.2: (a) the object cannot escape without penetrating the obstructing fingers. (b) by rotating and translating, the object can escape without colliding with the fingers. (c) consider the same situation as (b) but the object is seen as a static obstacle, the finger placement can be rotated and translated to escape from the object.

We are interested in the following caging problem: "Given an object and a number of point fingers, determine all finger placements that cage the object along with their error tolerance." A finger placement consists of positions of the fingers relative to the object. A finger placement that can prevent the object from escaping by any rigid motion is said to be a *caging placement*. A caging placement is a finger placement in a caging set. The other finger placements are referred to as *non-caging placements*. A good caging placement should be able to tolerate error to some degree. That is, each finger may displace from its position in the finger placement to some extent without letting the object escape.

Representation of a caging set is simple for a system with two point fingers. In two-finger setting, the fingers' separation distance, a one-dimensional measure, is necessary and sufficient to describe the formation shape of the the two fingers. Decreasing (or increasing) the separation distance is therefore the only approach to enter (or leave) a caging set. This implies that a caging placement along with the critical distance to be maintained (above or below) implicitly represents a caging set. However, the same representation does not apply to a system with three or more fingers. Let us consider the case of three fingers in a two-dimensional workspace. Observe that we need additional parameters (for example, the position of the third finger relative to the previous two fingers) to represent the shape of the finger formation instead of the separation distance alone. Varying any of these parameters changes the formation shape. A *critical boundary* for the parameters is needed to describe a caging set. It should be noted that a critical distance is a kind of critical boundary for a one-dimensional parameter. Keeping fingers within a caging set is no longer as simple as keeping their separation distance below a value, but to keep the fingers' formation shape parameters within a volume wrapped by the critical boundary. This volume depend on the geometry of the object and the parametrization of the configuration space.

Higher number of fingers lead to even more control parameters and more complex critical boundaries. Though this leads to larger caging sets and permits us to cage wider variety of objects, it is much more difficult and resource consuming to

portray the critical boundaries of such caging sets. In practice, we may also face problems of controlling multiple parameters in order to keep the manipulator inside a caging set. Especially when the critical boundary consists of multiple curves and turns in the higher dimensional space. Our solution is to sacrifice some caging placements for simplicity. We aim to identify a portion of a caging set that can be efficiently recognized and robustly deployed instead of the whole caging set.

Consider the object that resembles a section of a double-sided saw blade in Figure 1.3. Suppose that the number of the saw's teeth is k on each side, and l_0, l_1, l_2 are much longer than l_3 so that placing two fingers between any two teeth, one on the left and the other on the right side, always result in caging the saw. This way, each finger can be placed any where in $2k-2$ gaps between the teeth but not all the fingers are on the same side. Therefore, the number of all distinct cages is at least $(2k-2)^n - 2(k-1)^n$ where n is the number of fingers. Given that v is the number of vertices representing the saw, it can be observed that $2k-2 = v$. This means that the worst case running time of an algorithm for identifying all possible cages must be $\Omega(v^n)$. The running time will be exponential with respect to the number of fingers, but will be polynomial to the number of features for a constant number of fingers.

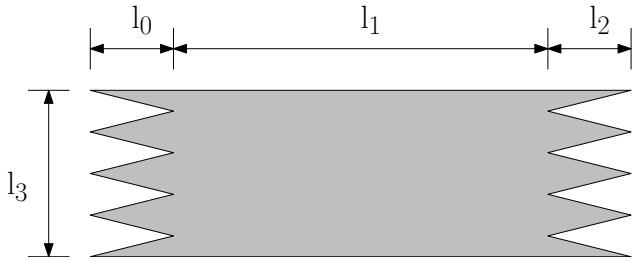

Figure 1.3: An object which has the number of all possible distinct cages exponentially proportional to the number of fingers, see text.

Apart from caging the object, we are also interested in error-tolerant grasping. An error tolerant strategy to grasp an object is to first cage the object appropriately, then "shrink" the cage. Here, to shrink is to control the manipulator to reduce the range of possible object motions. Eventually, the object is grasped. Both of them

move together as a single rigid body. Every possible grasp reached by the aforesaid process is necessarily a cage, see an abstracted diagram in Figure 1.4. The arrow in the diagram shows the state possibly reached after shrinking the cage. The diagram can be created with a morse decomposition of the free configuration space (Choset et al., 2005b). This is not surprising since a solution to a caging problem provided by (Rimon and Blake, 1996) also relies on the stratified morse theory. However, the current practical implementation of general morse decomposition is very limited to small dimensional space. Though the configuration space in our problem is embeded in higher dimension, it possesses many regular structures for example, the space to be decomposed is constructed from cartesian products of multiple copies of the workspace. We will create an algorithm that will take advantages of assumptions stated earlier to generate a similar diagram, called a caging roadmap, efficiently.

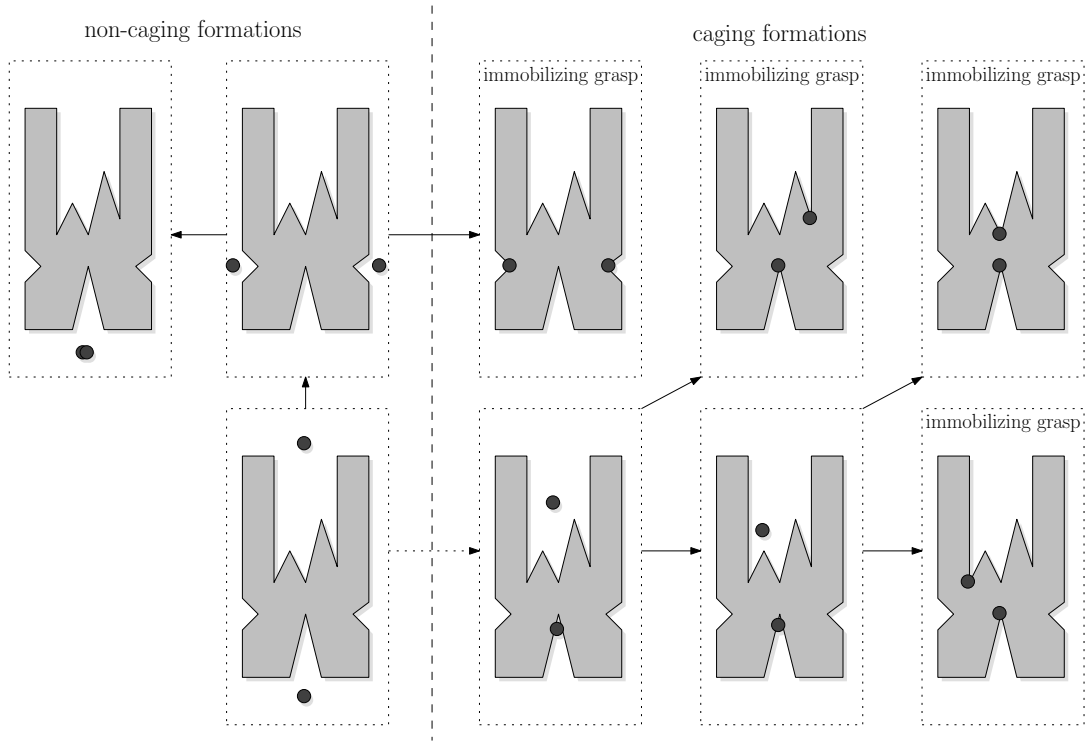

Figure 1.4: An abstracted diagram of cages and grasps.

1.2 Scope and Objectives

Our goal is to study object caging via a simplified finger formation control. We assume that each finger is a point, and the object is a bounded rigid polytope embedded in a compact and contractible subspace of $\mathbb{R}^w$. We propose an algorithm capable of constructing a caging roadmap to assist planning and controlling tasks related to caging and error-tolerant grasping. The caging roadmap is constructed from a given rigid polytope and a simplified control capability. The caging roadmap contains the information of all the caging sets up to a simplified control capability in a simplified representation, including associations to immobilizing grasps. Distinguish sites to cage an object along with their error tolerance can be easily extracted from the roadmap. The roadmap also supports a query: whether a finger placement may function as a cage with respect to the simplified control.

The following is a checklist for the summarized objectives.

1. Study and simplify the problem of caging and error-tolerant grasping of polyhedral objects with multiple fingers.
2. Find a practical control strategy for the fingers to cage and grasp the object such that the strategy will lead to an efficient computation of caging set and provide a robust approach to cage and grasp the object.
3. Design an efficient algorithm to identify all possible solution sets under the simplified control.
4. Design an efficient algorithm to compute the caging roadmap for error-tolerant grasping with simplified control.
5. Provide an efficient algorithm to query whether a finger placement may function as a cage with respect to the simplified control.

Note that all the algorithms running time will be polynomial to the number of features describing the caged polytope, the number of fingers are given as considered as a constant.

1.3 Research Direction

Caging and grasping object by squeezing fingers can be seen as an attempt to envelope object with the fingers. The cage is shrinked by reducing the "size" of the finger formation. Caging by squeezing fingers is undoubtedly one of the most common strategy used by humans and, for this reason, it is supported by most gripper systems. So far, squeezing caging with two fingers is said to be the act of keeping the two fingers' separation distance below a value. Multi-finger squeezing is the act of maintaining the *size of the finger formation* smaller than a value. Possible measures for the size are, for example:

1. **Ring circumference**. The circumference of ring formation is the sum of distance between pairwise *adjacent* fingers:

$$\delta^1_{\text{ring}}(\mathbf{x}_1, ..., \mathbf{x}_n) \equiv \|\mathbf{x}_1 - \mathbf{x}_2\|_2 + \|\mathbf{x}_2 - \mathbf{x}_3\|_2 + ... + \|\mathbf{x}_{n-1} - \mathbf{x}_n\|_2 + \|\mathbf{x}_n - \mathbf{x}_1\|_2,$$

   each $\mathbf{x}_i$ represents a finger position in $\mathbb{R}^w$. Caging by maintaining this below a value is resemble to surrounding the object with fingers. The object cannot move outside unless some fingers move away from the other, increasing the circumference. When $n = 2$ (two fingers), $\frac{1}{2}\delta^1_{\text{ring}}$ is exactly the separation distance between the fingers.

2. **Maximum separation distance among each pair of adjacent fingers in a ring**:

$$\delta^\infty_{\text{ring}}(\mathbf{x}_1, ..., \mathbf{x}_n) \equiv \max\{\|\mathbf{x}_1 - \mathbf{x}_2\|_2, \|\mathbf{x}_2 - \mathbf{x}_3\|_2, ..., \|\mathbf{x}_{n-1} - \mathbf{x}_n\|_2, \|\mathbf{x}_n - \mathbf{x}_1\|_2\}.$$

   This induces caging by maintaining the maximum gap between each pair of adjacent fingers in the ring.

3. **Weighted sum of lattice edge p-lengths**. Let w_{ij} be a non-negative weight for the lattice edge linking between x_i and x_j. Also let $L(i)$ be the set of finger indices such that, for any $j \in L(i)$, the lattice's edge connects $\mathbf{x}_i$ and $\mathbf{x}_j$. The

formation's size is defined as:

$$\delta_L^p(\mathbf{x}_1, ..., \mathbf{x}_n) \equiv \frac{1}{2}\Sigma_{i=1}^{n}\Sigma_{j\in L(i)} w_{ij}\|\mathbf{x}_i - \mathbf{x}_j\|_2^p;$$

when $p \in \{1, 2, ..., \infty\}$. Since $\|\mathbf{x}_i - \mathbf{x}_j\|_2 \in \mathbb{R}_+$ (the set of all nonnegative real values) and $x^p : \mathbb{R}_+ \to \mathbb{R}$ is convex and nondecreasing (Boyd and Vandenberghe, 2004a), $\|\mathbf{x}_i - \mathbf{x}_j\|_2^p$ is convex for any i, j and so does its non-negative weighted sum: $\delta_L^p(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$. This δ_L^p generalizes the previously exemplified formation's size. For example, observe that $\delta_L^1 = \delta_{\text{ring}}^1$, and $\delta_L^\infty = \delta_{\text{ring}}^\infty$ when:

$$L(i) = \begin{cases} 1, & i = n; \\ i+1, & \text{otherwise.} \end{cases}$$

The previously stated functions: δ_{ring}^1, $\delta_{\text{ring}}^\infty$, δ_L^∞, including the separation distance between two fingers; shares one common property in that they are all convex. These functions are non-negative weighted summation or maximum among affine compositions of norms (Boyd and Vandenberghe, 2004a). All of them also attain their minimal values when all the fingers are at the same point and increase in value when the fingers are scattered. The choice of the formation's size depends on user preferences, applications, and mechanics of the manipulator.

In summary, we aim to present algorithms to design finger placements of arbitrary number of fingers that cage a given object. The algorithms are based on controlling a size measure of finger formations which we believe that this will lead to simplified solutions and practical applications.

CHAPTER II

OVERVIEW

2.1 Approaches

This dissertation houses a collection of approaches to caging. The approaches are inspired from the simplicity of two-finger caging. Each approach relies on different constraints enforced on the fingers' positions; therefore, leads to a different kind of cages. Nevertheless, they all share a common framework. This chapter serves as an overview for the framework and introduces the variations among the approaches.

- *Squeezing Caging* is a natural and intuitive way to cage an object. To squeezing cage is to surround the object with the fingers and squeeze them. In the two finger case, this is acheived by controlling the separation distance between the two fingers to stay below a value, i.e., enfocing constraints on the fingers' positions. When more fingers involve, a measure that represents the size of the finger formation replaces the separation distance. With a sufficient number of fingers, it is always possible to cage by squeezing the fingers. To cage by keeping the distance between fingers below the object's diameter (Sudsang, 2002) or the coverage diameter (Vongmasa and Sudsang, 2006) is also a squeezing caging. We present an algorithm for identifying all two-finger squeezing cages in the first half of Chapter III. Its extensions to multiple fingers and additional constraints follow in Chapter IV. Here, additional constraints are constraints on the finger positions induced by physical limitations of the manipulator such as limited communication range and joint length.

- *Stretching Caging* is to cage by keeping the fingers stretched away from each other. Like squeezing, the concept of stretching can be extended to multiple fingers. In contrast to squeezing caging, the size of finger formation is constrained to be above a value. Utilization of stretching caging is less common compared to that of squeezing caging. A convex object cannot be stretching

caged for any number of fingers. Stretching caging is only possible when the object has a sack-like concave section. The second half of Chapter III presents the algorithm for computing two-finger stretching cages. The extension to multiple fingers is similar to that of squeezing caging, which is summarized in Chapter VII. The presented stretching caging algorithm is less feature-rich and less developed compared to the squeezing caging one. Additional constraints appear in Chapter IV are not available for stretching caging. Only approximated constraints introduced in Chapter VI are supported.

- *Combined Squeezing and Stretching Caging* is as the name suggests. Instead of caging by controlling a formation size measure below or above a value. The measure value is controlled to be in a certain interval to cage a given object. This approach to caging can be considered as a constrained squeezing caging, an extension to multi-finger squeezing caging.

- *Robust Caging* is to cage the object by attempting to fix a given finger formation. In practice, perfectly fixing the fingers to a specific finger placement or a specfic finger formation may not be possible due to external disturbances. The fingers may displace from its desired position in the formation to some extent during caging. Robust caging is to cage the object by limiting a norm of such displacements below a value. This caging method is designed to compliment the squeezing and the stretching caging. In a system with more fingers, solution sets for squeezing and stretching caging reported by the algorithms tend to be smaller. This contradicts the fact that it should be easier to cage with more fingers – solution sets only occupy a small subset of a caging set. If the finger formation to fix is a grasp, the robust cage functions as a pregrasping cage (Rodriguez et al., 2011), a waypoint to the grasp. We present a method to compute robust cages in Chapter VIII.

Regardless of the approach, caging in nature is robust to measurement and control noise. Successful caging is usually possible even when the fingers and/or the object is slightly deformed, inaccurately and/or partially observed. Chapter V

presents a failsafe strategy to handle uncertainty of the object. The method that handles object uncertainty is also generalized to deals with control uncertainty. Chapter VI deals with control uncertainty with a focus on a specific application: constraint approximation.

2.2 Basics

Let us begin by introducing the main actors: the fingers (the manipulators) and the object. At first, we assume that the fingers are just points and the object is a rigid body represented by polygons or polyhedra. As the discussion progresses, the assumptions will be more relaxed. The fingers and the object are contained in a workspace which is a two-dimensional plane $\mathbb{R}^2$ or a three-dimensional space $\mathbb{R}^3$. Though the caging concepts and our approaches may extend to greater dimensions, the problems in practice do not. We only focus on two and three-dimensional workspace denoted by $\mathbb{R}^w$ where w is the workspace dimension. For the reference coordinates, we choose the object's coordinates. Consequently, the fingers are the only moving entities. The fingers cannot cage the object if they can move arbitrarily far from the object. Without any constraints imposed, each finger are free to move around the object. That is, for an object occupying a set of points $\mathcal{P}$ without any holes, the set of points reachable by a free finger is $\mathcal{F} \equiv \mathbb{R}^w \backslash \mathcal{P}$. We can safely assume that the object does not contain any inaccessible holes. If it has some, they are either inaccessible or/the object can be trivially caged by placing a finger in the hole. Additionally, the set $\mathcal{P}$ must be open and bounded. This means that the fingers can slide along the object's boundary and the object does not extend to infinity.

With finite number of fingers and bounded object, the object is no longer caged if all the fingers are sufficiently far from the object, i.e., all are outside a sufficiently large box centered at the object. To avoid dealing with infinity, this condition will be used instead of the usual caging condition. A method to compute a sufficiently large box $\mathcal{B}'$ is illustrated in Figure 2.1. The size of $\mathcal{B}'$ is chosen to be a multiple of a smallest object bounding box $\mathcal{O}$ containing the object in its interior. As shown in the illustration, this choice of $\mathcal{B}'$ is sufficient for $n < 12$. Any finger formation that

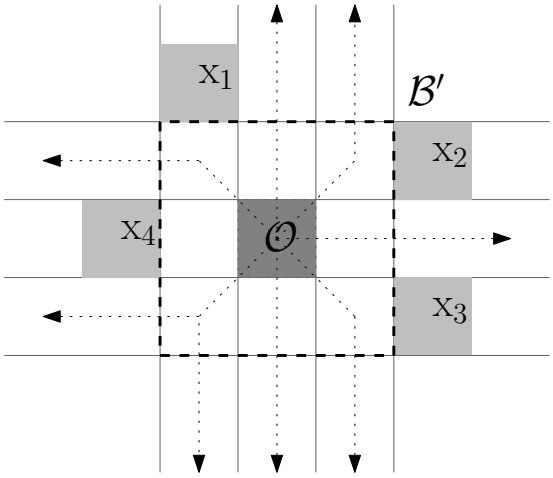

Figure 2.1: A sufficiently large box $\mathcal{B}'$.

all the fingers are outside $\mathcal{B}'$ cannot prevent the object from escaping. This sufficient condition also applies to the three-dimensional workspace.

The configuration space for a system with n free fingers is given by $\Omega \equiv \mathcal{F}^n$. Each caging approach relies on a set of constraints to cage the object. These constraints are either enforced by control policies or physical limitations of the manipulator. The constraints define an infeasible set, a set of all finger placements that violate at least one of the constraints. Finger placements in an infeasible set cannot be reached from the others as long as the constraints are enforced. If the constraint enforcement results in preventing the fingers from escaping, going arbitrarily far from the object, i.e., the object is caged. Each constraint is in the implicit form of $f < l$ where f is a function that maps the positions of the fingers to a real value and l is a real constant. Increasing or decreasing a constraint's parameter either results in a more relax or a more strict constraint, respectively. The more relax the constraint, the more freely the fingers can move. A constraint can be classified as:

1. *Always-active constraint*. This class of constraint is always enforced, for example, the constraint induced by physical limitations of the manipulator.

2. *Toggleable constraint*. A toggleable constraint is enforced after an initial finger placement setup so as to cage the object. A toggleable constraint's parameter

is either fixed or adaptive.

- *Fixed parameter*. The parameter is just a constant real value parameter of the system.
- *Adaptive parameter*. The parameter is a parameter to optimize so as to obtain the largest possible cage. The algorithm for identifying all solution sets is responsible for optimizing this parameter.

All of the caging approaches in this dissertation have only one toggleable constraint with an adaptive parameter. Our problem involves solving the optimal adaptive parameter. The optimal parameter maximizes the fingers' freedom to move around yet cannot escape. Formally, let $f < l$ be the only toggleable constraint with the adaptive parameter l, and the other constraints: $f_1 < l_1, f_2 < l_2, ..., f_c < l_c$. Also let $\Gamma(S, T, U)$ be the set of all paths that starts from a point S, terminates at a point in T, and are contained in U. A path is defined as a continuous map from an interval to the configuration space Ω. A path is said to be contained in a set if its image is contained in the set. At the fundamental level, we want to query if it is possible to cage the object by enforcing the constraints after setup the fingers at an initial finger placement $\mathbf{z}$. In other words, we are to check whether the initial placement $\mathbf{z}$ forms a cage with respect to this set of constraints. This can be formulated as the following optimization problem:

$$\begin{aligned} &\text{minimize} && l \\ &\text{subject to} && \alpha \in \Gamma(\{\mathbf{z}\}, \mathcal{T}, \Omega) \\ & && \forall \mathbf{x} \in \text{img}(\alpha), f(\mathbf{x}) \leq l, f_1(\mathbf{x}) \leq l_1, f_2(\mathbf{x}) \leq l_2, ..., f_c(\mathbf{x}) \leq l_c \end{aligned}$$

Where the set $\mathcal{T}$ denotes a set of all points that all the fingers are outside a sufficiently large box as discussed earlier. The minimal value for the optimization problem will be the adaptive parameter l for the constraint $f < l$. If $f_i(\mathbf{z}) \geq l_i$ for some i, the optimization problem is infeasible, indicating that the finger placement $\mathbf{z}$ is not in any solution set with respect to this set of constraints. Solving the optimization problem is not a trivial task.

Let Ω' be a subset of Ω that does not include finger placements violating any of the constraints $f_1 < l_1, f_2 < l_2, ..., f_c < l_c$. We can write a single-line formula for the adaptive parameter l for an initial finger placement $\mathbf{z} \in \Omega'$ as:

$$l \equiv f^*(\mathbf{z}) = \inf_{\alpha \in \Gamma(\{\mathbf{z}\}, \mathcal{T}, \Omega')} \sup_{\mathbf{x} \in \text{img}(\alpha)} f(\mathbf{x}).$$

Different caging approaches, workarounds, manipulators, and some other factors lead to different f and other functions comprising the constraints. Prior to Chapter VI the constraints must be invariant to the change of the coordinate frame of the fingers. The value of f is the same for the finger positions in the world coordinates, the object coordinates, or any other coordinates obtained from rotating and/or translating all of the finger positions. The coordinate invariant property allows us to freely change the coordinate frame from the world coordinates to the object coordinates without affecting the value of the function. That is: whether a finger placement forms a cage does not depends on the coordinate frame of reference. In Chapter VI, the constraints are approximated with simplier functions and may not satisfy the coordinate-frame invariant property. As a result, solutions may depend on the coordinate frame. Even so, the generated solutions are guaranteed to work, i.e., capable of caging the object. The readers are referred to Chapter VI for more details.

Remaining chapters in this dissertation present solutions to the stated problem. The solutions varies based on different properties of the functions $f, f_1, f_2, ..., f_c$ comprising the constraints. Conceptually, each function can be interpreted as a kind of energy function. Higher function value indicates higher energy state. Higher maximum energy bound (more relaxed constraint) implies that the fingers have more freedom to change its formation. The following is a brief list of functions that portray the behavior of each caging approach.

- *Squeezing Caging*: The function f is convex (or quasi-convex) and invariant to the change of the coordinate frames. A quasi-convex or a convex func-

tion has a property similar to squeezing. The function reduces its value if a point straightforwardly travels to a minimal point of f. The more the fingers are squeezed together, the lower the function value and the less freedom to change its formation. The other functions $f_1, f_2, ..., f_c$ must be convex and coordinate-frame invariant as well. However, the coordinate-change invariant assumption is dropped after Chapter VI.

- *Stretching Caging*: The function f is the opposite of squeezing. It is concave (or quasi-concave) and invariant to the change of coordinate frames. The higher the maximum energy bound, the more the finger can be squeezed together. Unlike squeezing caging, the other functions $f_1, f_2, ..., f_c$ must be linear. They are introduced in Chapter VI.

- *Robust Caging*: The actual function f is an infimum of infinitely-many convex functions, but it is simplified to a minimization among finitely-many convex functions. The strategy for approximating actual f is a variation of the approximation method in Chapter VI. Conditions on the other functions $f_1, f_2, ..., f_c$ are the same as those in squeezing caging.

2.3 Roadmap Construction

Proposed approaches to solve the optimization problem vary and heavily depend on properties of the functions. Nevertheless, it turns out that all algorithms to generate the solutions rely on many common steps. One is to collapse most of the configuration space into a one dimensional skeleton, a *roadmap* (Choset et al., 2005a). In a view point, this is to prune the configuration space out, reducing the search space for the optimal path. The optimal path between any two points can be easily obtained by moving along the roadmap except when *accessing* it. In each chapter, we will construct a roadmap and show that an optimal path between any two points in the roadmap is contained in the roadmap itself. The roadmap is then shown to be accessible from any point in the configuration space. An optimal path from any two points in the configuration space consists of three concatenated paths. The first path is an access path, from a starting point in the configuration space to

an access point in the roadmap. The last path is also an access path, from a terminal point in the configuration space to an access point in the roadmap. The middle part is entirely in the roadmap. It optimally connects the two access points in the roadmap.

The roadmap is represented as a graph. A vertex of the graph is a point, a finger placement in the configuration space. An edge of the graph is a path connecting two vertices. A sequence of adjacent edges represents a path in the graph and a path in the configuration space. The graph is where to search for the optimal path with a modified shortest path algorithm. The cost assigned to each graph edge is the cost for travelling between two adjacent vertices. The cost is the maximal value of the parameter l, the toggleable constraint's adaptive parameter, restricted to the path. The cost for a path in the graph is of course the maximum cost of an edge in the sequence that constitutes the path. An optimal path (in the graph) is a sequence of edge that has the least cost among other paths starting and terminating at appropriate initial and terminal points.

Access points are minimal points of f restricted on the configuration space. The function f may have infinitely many minimal points so we choose only one representative minimal point per connected component of minimal points as an access point. The reason to choose a minimal point as an access point is that any point in the configuration space can always *optimally access* a minimal point. A gradient descent trajectory to a minimal point serves as an optimal path connecting the two points. The (optimal) cost is exactly the value of f at the starting point. Consequently, a roadmap that contains all of the access points satisfies the accessbility and the departability properties. The problem that remains is how to construct a roadmap structure that contains an optimal path connecting any two minimal points. To solve this problem, we rely on the convex decomposition of the configuration space. It is much easier to construct an optimal path connecting points in a decomposed convex subset. Access points are chosen from minimal points restricted in each decomposed convex subset. They are assigned as graph vertices. Within each convex subset, a number of optimal paths connecting vertices are as-

signed as graph edges, depending on caging approaches. To link between two overlapping convex subsets, consider the intersection between overlapping convex subsets which is also convex. The minimal points that lie in the intersection are also assigned as graph vertices. Since the vertices are in both overlapping convex subsets, they are connected to other vertices in each subset in a similar manner. In a point of view, the construction of the roadmap and the search for optimal points are comparable to the *divide* and the *conquer phase* of the divide and conquer paradigm.

We identify a vertex (as a finger placement) that cannot cage the object as *exit vertices* based on simple criteria, for example, all the fingers are sufficiently far from the object. Then, for every vertex in the graph, we determine the cost of an optimal path from it to an *exit vertex*. Such cost is referred to as the optimal cost for the vertex. Optimal cost of an exit vertex is known to be the value of f at the vertex as the fingers can escape without changing any formation starting from that point. Other vertices optimal costs are computed by *propagating* from exit vertices' known optimal costs. The propagation works similar to the propagation of shortest distances in Dijkstra's shortest path algorithm. This is achieved by propagating the known optimal cost starting from exit vertices. To determine whether a finger placement forms a cage with respect to the constraints, just access its vertex for the adaptive parameter l and check if the finger placement is feasible under the constraints.

In a point of view, each vertex that forms a cage with respect to the constraints is a representative of finger placements that can access the vertex with a gradient descent trajectory. A "solution set" is a maximally connected set of finger placements satisfying the constraints. Such a solution set is identified by the set of maximally connected vertices in the same solution set. We apply a simple connected component labelling graph algorithm to keep track of the solution sets.

2.4 Error-Tolerant Grasping

It has been proven by (Rodriguez et al., 2011) that squeezing or stretching fingers while maintaining a cage of an object will always lead to a grasp. However, the grasp may not a desirable one. Reaching a specific grasp while caging the ob-

ject usually requires the knowledge of the object pose. The roadmap is designed for identifying the states of the fingers in detail (more than just indicating that it is in a cage or not). A state of the fingers is a set of the possible convergences (some are grasps) after reducing f (squeezing or stretching the fingers, for example). It is possible to construct a simplified cage-state diagram based on the following rule: "Let $M_1, M_2, ...$ be sets of maximally connected minimal points of f. Vertices belong to the same state if the set of all M_i reachable by gradient descending from every vertex is equivalent (set equal)." Construction of the diagram is, however, optional for typical caging tasks. Sometimes, it is more than necessary. Identifying only the states that lead to a unique M_i usually suffices. Such states can be obtained by connected component labelling on the roadmap as well.

2.5 Summary

This chapter serves as an overview and a guide to the subsequent chapters. The key steps for identifying solution sets always begin with decomposing the configuration space into convex subsets. Then, construct the roadmap and search for optimal escape paths. What varies among chapters are the caging constraints which lead to different roadmap and optimal escape paths.

CHAPTER III

TWO-FINGER CAGING

3.1 Introduction

Our prelimary solution for two-finger caging, i.e., two-finger squeezing and stretching (Figure 1.1), was first published in Pipattanasomporn and Sudsang (2006). The solution is based on rayshooting the two fingers along the line between them to identify "state" of the fingers and formulate the problem into a state-space search one. However, this idea does not extend to a setting with more than two fingers. With more fingers, the rayshooting along the line between the fingers is no longer defined. In this chapter, we reformulate our preliminary approaches to two-finger squeezing and stretching caging. Based on this formulation, we can extend our approaches to multi-finger squeezing and stretching caging to be presented the subsequent chapters.

Let us assume that the two fingers are free moving points. The object is open, bounded, rigid, and embedded in $\mathbb{R}^w$. It is to be caged by either squeezing or stretching the fingers. That is, to impose a constraint on the separation distance, maintaining it below or above a value. Our goal is to identify all possible solution sets formed by imposing a squeezing or a stretching constraint. Our algorithm distinguishes from the one presented by Rimon and Blake (Rimon and Blake, 1996) that their algorithm numerically computes a critical distance of a single caging set given an immobilizing grasp. The algorithm proposed by Vahedi and Stappen in (Vahedi and van der Stappen, 2006), (Vahedi and van der Stappen, 2008) and our algorithm report solution sets in $O(v^2 \log v)$ where v is the number of the object's vertices. Vahedi and Stappen's algorithm operates on pseudo-trapezoids, and is designed to report all solution sets for disc-shaped fingers and a two-dimensional object. Our approach, on the other hand, operates on decomposed convex polytopes so that two and three-dimensional versions of the problem are handled in a generalized manner. Our algorithm report exact solutions for point fingers and approximate

solutions for disc-shaped or spherical fingers.

This chapter is organized as follows. In the following section, we introduce basic notations and properties for two-finger squeezing and stretching caging. Their properties and relationships are shown in Section 3.3. An approach to compute an optimal path between any two finger placements for squeezing caging is presented in Section 3.4. For stretching caging, a different approach is needed and presented in Section 3.5.. The implementation details and results of the algorithm are presented in Section 3.7.

3.2 Assumptions and Definitions

The system consists of an object and two point fingers. The object and the fingers are contained in $\mathbb{R}^w$, where $w \geq 2$ is the dimension of the workspace. The set of all possible finger placements (configurations) is $\mathbb{R}^{2w}$. The separation distance of a finger placement is a function δ that maps a finger placement $\mathbf{x} \equiv (x_1, x_2) \in \mathbb{R}^{2w}$ to $\delta(\mathbf{x}) \equiv \|x_1 - x_2\|_2$, the Euclidean distance (separation distance) between the two fingers. The sets $\left\{\mathbf{x} \in \mathbb{R}^{2w} \mid \delta(\mathbf{x}) \leq d\right\}$ and $\left\{\mathbf{x} \in \mathbb{R}^{2w} \mid \delta(\mathbf{x}) \geq d\right\}$ are referred to as *d-sublevel set of* δ and *d-superlevel set of* δ, respectively. Maintaining the fingers' separation distance less (greater) than a value d is equivalent to squeezing (stretching) the fingers, restricting possible finger placements within d-sublevel (resp. d-superlevel) set of the separation distance function.

The object, denoted by $\mathcal{P} \subset \mathbb{R}^w$, is assumed to be open, rigid and is represented by a bounded polytope without holes (the region not occupied by the polytope is contractible). To avoid dealing with the object's rigid motion directly, we choose the object frame as the frame of reference. The object is considered a static obstacle placed at the origin o and assumed to lie inside $B_1(o)$, a w-dimensional ball of radius 1 centered at o. If the object is caged, it has to block the fingers from escaping to infinity (of course, given that the fingers are constrained either by squeezing or stretching motion). Assuming that the fingers cannot penetrate the polytope, the free workspace is given by $\mathcal{F} \equiv \mathbb{R}^w \backslash \mathcal{P}$. The set of all valid placements for two fingers (free configuration space) is therefore $\mathcal{F}^2$ denoted by Ω. That is, $\mathbb{R}^{2w} \backslash \Omega$

constitutes our configuration space obstacle.

A synchronized motion of two fingers is represented by a path, a continuous map from an interval to Ω. The paths α and β can be concatenated when they share a common end points. The concatenated path is written as $\alpha \cdot \beta$. The image of a path α is the range of α. For brevity, when no confusion may arise, we refer to the image of a path that is contained in a set A as a path in A. A path is bounded if its image is bounded. For a bounded path α, its reverse path is denoted by $\alpha^{-1}(\theta)$. It should be noted that: i) concatenation of any two paths in the d_1-sublevel and the d_2-sublevel of δ is in the max $\{d_1, d_2\}$-sublevel set of δ, and ii) concatenation of any two paths in the d_1-superlevel and the d_2-superlevel of δ is in the min $\{d_1, d_2\}$-superlevel set of δ. If an image of a path contains a finger placement at which both fingers are outside the sufficiently large box bounding the object, the path is an *escape path*.

Squeezing (resp. stretching) is defined as the act of keeping the fingers' separation distance below (resp. above) a given value. A maximally connected set of finger placements that the object cannot escape by squeezing (stretching) the fingers is said to be a solution set of the squeezing (stretching) constraint. For brevity, we refer to a solution set of the squeezing (stretching) constraint as a squeezing (stretching) solution set. When the object cannot escape from a squeezing or a stretching cage, the fingers can be visualized as a point in the configuration space Ω being restricted within a cage. Figure 3.1 shows an abstracted illustration of the configuration space obstacle and some solution sets. The dotted lines marked by $\delta = d_1$ (resp. $\delta = d_2$) represent finger placements with separation distance equal to d_1 (resp. d_2). Each finger placement in region A forms a squeezing cage. Each finger placement in region B forms a stretching cage.

In this setting of two-finger caging, it is shown in (Rodriguez and Mason, 2008) that a caging placement forms a squeezing cage or a stretching cage or both. A finger placement that neither form squeezing nor stretching cages is a non-caging placement.

Let $\mathbf{x}$ be a finger placement and d be a real value. The finger placement $\mathbf{x}$ forms

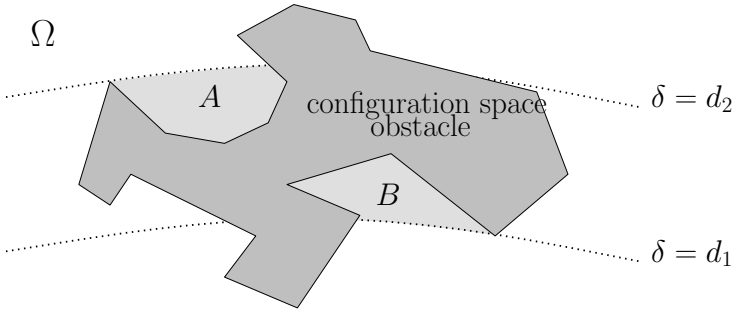

Figure 3.1: Abstracted illustration of the configuration space obstacle and some solution sets.

a squeezing cage if, for some $d > \delta(\mathbf{x})$, every escape path from $\mathbf{x}$ contains some finger placement with a separation distance greater than or equal to d. The *maximal distance* of $\mathbf{x}$ is denoted by $\delta^*(\mathbf{x})$ and is defined as the maximal value of d subject to: every escape path from $\mathbf{x}$ contains some finger placement with a separation distance greater than or equal to d. The most relaxed squeezing constraint containing $\mathbf{x}$ capable of preventing the object from escaping is the constraint $\delta < \delta^*(\mathbf{x})$.

Similarly, the finger placement $\mathbf{x}$ forms a stretching cage if, for some $d > \delta(\mathbf{x})$, every escape path from $\mathbf{x}$ contains some finger placement with a separation distance less than or equal to d. The *minimal distance* of $\mathbf{x}$ is denoted by $\delta_*(\mathbf{x})$ and is defined as the minimal value of d subject to: every escape path from $\mathbf{x}$ contains some finger placement with a separation distance less than or equal to d. The most relaxed stretching constraint containing $\mathbf{x}$ capable of preventing the object from escaping is the constraint $\delta > \delta_*(\mathbf{x})$.

3.3 Fundamental Properties

In this section, some properties related to the previously defined terms are presented. We show that every finger placement in a squeezing solution set has the same maximal distance. A similar fact holds for a finger placement in a stretching solution set and its minimal distance. Additionally, we show that all solution sets are contained in a compact subset of Ω.

Consider finger placements $\mathbf{x}$ and $\mathbf{y}$ connected by a path in d-sublevel set of δ such that d is less than $\delta^*(\mathbf{y})$, the maximal distance of $\mathbf{y}$. Under this condition, the maximal distance of $\mathbf{x}$ and $\mathbf{y}$ are equal.

Proposition 1. *Let $\mathbf{x}, \mathbf{y} \in \Omega$. If $\mathbf{x}$ and $\mathbf{y}$ are connected by a path α in d-sublevel set of δ where $d < \delta^*(\mathbf{y})$, $\delta^*(\mathbf{x})$ is equal to $\delta^*(\mathbf{y})$.*

Proof. Suppose that β, and γ, be an escape path from $\mathbf{x}$ and an escape path from $\mathbf{y}$, respectively. Also suppose that β and γ lie in the $\delta^*(\mathbf{x})$-sublevel set of δ and the $\delta^*(\mathbf{y})$-sublevel set of δ, respectively. The concatenated path $\alpha \cdot \gamma$ forms an escape path in $\delta^*(\mathbf{y})$-sublevel set of δ. This implies that $\delta^*(\mathbf{x}) \leq \delta^*(\mathbf{y})$. However, it is not possible for $\delta^*(\mathbf{x})$ to be less than $\delta^*(\mathbf{y})$; otherwise, $\alpha^{-1} \cdot \beta$ forms an escape path that starts from $\mathbf{y}$ and resides in d-sublevel set of δ where $d < \delta^*(\mathbf{y})$. This contradicts the definition of $\delta^*(\mathbf{y})$. ■

Proposition 2. *All finger placements in a squeezing solution set have the same maximal distance d^* and are pairwise connected by a path in the interior of the d^*-sublevel set of δ.*

Proof. Let $\mathbf{x}$ and $\mathbf{y}$ be in the same solution set. Consequently, there exists a path from $\mathbf{x}$ to $\mathbf{y}$, say α, such that, any point belongs to the image of α is in the solution set. Let $\mathbf{z}^*$ be a point in the image of α such that $\delta(\mathbf{z}^*) \geq \delta(\mathbf{z})$ for any $\mathbf{z}$ in the image of α. Since $\mathbf{z}^*$ is in the solution set, $\delta(\mathbf{z}^*) < \delta^*(\mathbf{z}^*)$. This means that $\mathbf{z}$ and $\mathbf{z}^*$ are connected by a path in d-sublevel of δ such that $d = \delta(\mathbf{z}^*) < \delta^*(\mathbf{z}^*)$. By Proposition 1, $\delta^*(\mathbf{z}) = \delta^*(\mathbf{z}^*)$. ■

To certify that a configuration $\mathbf{x}$ lies in a squeezing solution set containing $\mathbf{y}$, it is sufficient to show that $\mathbf{x}$ and $\mathbf{y}$ are connected by a path in the d-sublevel of δ where $d < \delta^*(\mathbf{y})$. A problem to be solved is how to compute the maximal distance of the finger placement $\mathbf{y}$. That is to determine the optimal escape path α from $\mathbf{y}$ (with respect to the squeezing constraint). Among other escape paths from $\mathbf{y}$, the optimal one has the least upperbound separation distance. The least upperbound separation distance is the maximal distance of $\mathbf{y}$. Formally, the optimal escape path

from $\mathbf{y}$ is the minimizer for following optimization problem:

$$\begin{aligned} \text{minimize} \quad & d \\ \text{subject to} \quad & \alpha \in \Gamma(\{\mathbf{y}\}, \mathcal{T}, \Omega) \\ & \forall \mathbf{x} \in \text{img}(\alpha), \delta(\mathbf{x}) \leq d. \end{aligned}$$

The minimal value of the optimization problem is the maximal distance of $\mathbf{y}$. Alternatively, it can be written as:

$$\delta^*(\mathbf{y}) = \inf_{\alpha \in \Gamma(\{\mathbf{y}\}, \mathcal{T}, \Omega)} \sup_{\mathbf{x} \in \text{img}(\alpha)} \delta(\mathbf{x}).$$

An approach to compute an optimal escape path with respect to the adaptive squeezing constraint is presented in Section 3.4.

Similar property for the stretching caging holds and can be proven in the same manner.

Proposition 3. *All configurations in a stretching solution set have the same minimal distance d_* and are pairwise connected by a path in the interior of d_*-superlevel set of δ.*

To identify a stretching cage, we need its representative finger placement and its minimal distance. The minimal distance of a finger placement $\mathbf{y}$ is obtained by considering optimal escape paths from $\mathbf{y}$ and finding its greatest lowerbound separation distance. Formally, the minimal distance is the maximal value of the following optimization problem:

$$\begin{aligned} \text{maximize} \quad & d \\ \text{subject to} \quad & \alpha \in \Gamma(\{\mathbf{y}\}, \mathcal{T}, \Omega) \\ & \forall \mathbf{x} \in \text{img}(\alpha), \delta(\mathbf{x}) \geq d. \end{aligned}$$

Equivalently, the minimal distance of $\mathbf{y}$ is:

$$\delta_*(\mathbf{y}) = \sup_{\alpha \in \Gamma(\{\mathbf{y}\}, \mathcal{T}, \Omega)} \inf_{\mathbf{x} \in \text{img}(\alpha)} \delta(\mathbf{x}).$$

Determining the optimal escape path (with respect to the stretching constraint) is to be explained in Section 3.5.

From a non-caging placement, it is possible to rigidly move the fingers together (without changing its separation distance) arbitrarily far from the object. This implies that:

Proposition 4. *Any two non-caging placements* $\mathbf{x}, \mathbf{y}$ *are connected by a path which is in* $\max\{\delta(\mathbf{x}), \delta(\mathbf{y})\}$*-sublevel and* $\min\{\delta(\mathbf{x}), \delta(\mathbf{y})\}$*-superlevel.*

After moving sufficiently far from the object, the separation distance between the fingers may change to a desired value without being obstructed.

At a non-caging placement with separation distance d, say $\mathbf{x}$ with $\delta(\mathbf{x}) = d$, the object can escape even if a squeezing or a stretching constraint is imposed. This implies that a path from an arbitrary finger placement $\mathbf{z}$ to such non-caging placement $\mathbf{x}$ can be recognized as an escape path from $\mathbf{z}$.

Proposition 5. *A finger placement* $\mathbf{x} \equiv (x_1, x_2) \in \mathbb{R}^{2w}$ *is non-caging if at least one of the following conditions holds:*

1. $\delta(\mathbf{x}) = 0$,
2. *both* x_1 *and* x_2 *are outside* $B_1(o)$,
3. $\delta(\mathbf{x}) \geq 2$,
4. *either* x_1 *or* x_2 *is not in* $B_3(o)$.

Proof. 1) The fingers are a single point so they can travel anywhere by the assumption on the object does not contain any hole and bounded.

2) The object can translate towards a direction without being blocked by any finger. The direction to translate can be any direction perpendicular to the line connecting the fingers.

3) The fingers cannot block the object because their separation is greater than the object's diameter.

4) Given that a finger placement $\mathbf{x} \equiv (x_1, x_2) \notin B_3(o)^2$, x_1 (inclusive) or x_2 is outside $B_3(o)$. If both fingers are outside $B_1(o)$, $\mathbf{x}$ is non-caging by 2). If one finger is inside $B_1(o)$ and the other is outside $B_3(o)$, the separation distance must be greater than two; therefore, $\mathbf{x}$ is non-caging by 3). ■

Proposition 5 is a sufficient condition for non-caging placements. It allows us to work entirely in a compact domain by guaranteeing that all finger placements outside $B_3(o)^2$ are non-caging. Let $\mathcal{B}$ be a compact w-dimensional cube containing $B_3(o)$. The compact version of free workspace and free configuration space are denoted by $\bar{\mathcal{F}} \equiv \mathcal{F} \cap \mathcal{B}$ and $\bar{\Omega} \equiv \bar{\mathcal{F}}^2$, respectively. Note that $\bar{\mathcal{F}}$ and $\bar{\Omega}$ are polytopes by construction.

3.4 Squeezing Cage

In this section, we present a strategy for constructing an optimal escape path with respect to the squeezing constraint. Let us first consider the situation when the current configuration $\mathbf{x} \in \bar{\Omega}$ is inside a convex set $K \subseteq \bar{\Omega}$. Suppose that the fingers' configuration is to change from $\mathbf{x}$ to $\mathbf{y} \in \bar{\Omega}$ inside another convex set $K' \subseteq \bar{\Omega}$ that overlaps with K ($K \cap K' \neq \varnothing$) under the condition that the fingers must lie in $K \cup K'$. An optimal path connecting $\mathbf{x}$ and $\mathbf{y}$ satisfying the condition is a path α that minimizes the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & d \\ \text{subject to} \quad & \alpha \in \Gamma(\{\mathbf{x}\}, \{\mathbf{y}\}, K \cup K') \\ & \forall \mathbf{z} \in \text{img}(\alpha), \delta(\mathbf{z}) \leq d. \end{aligned}$$

Unavoidably, any path from $\mathbf{x}$ must enter $K \cap K'$ in order to reach $\mathbf{y}$. An optimal path is a concatenation of two straight line paths. One from $\mathbf{x}$ to a finger placement $\mathbf{z} \in K \cap K'$ where δ restricted in $K \cap K'$ attains its minimal value, i.e.,

$\delta(\mathbf{z}) \leq \delta(\mathbf{z}')$ for any $\mathbf{z}' \in K \cap K'$. The other is a straight line path from $\mathbf{z}$ to $\mathbf{y}$. Because K and K' are convex, the optimal path is in $K \cup K'$. Moreover, it is contained in $\max\{\delta(\mathbf{x}), \delta(\mathbf{y}), \delta(\mathbf{z})\}$-sublevel set of δ (see an example in Figure 3.2). The straight line path from $\mathbf{x}$ to $\mathbf{z}$ is $\alpha(\theta) = (1-\theta)\mathbf{x} + \theta\mathbf{z}$. Since δ is a convex function:

$$\delta(\alpha(\theta)) \quad \leq \quad (1-\theta)\delta(\mathbf{x}) + \theta\delta(\mathbf{z}); \tag{3.1}$$

for any $\theta \in [0,1]$. Taking supremum over $\theta \in [0,1]$ on both sides yields:

$$\sup_{\theta \in [0,1]} \delta(\alpha(\theta)) \quad \leq \quad \max\{\delta(\mathbf{x}), \delta(\mathbf{z})\}.$$

This implies that α lies in the $\max\{\delta(\mathbf{x}), \delta(\mathbf{z})\}$-sublevel set of δ. The same goes for the straight line path β from $\mathbf{z}$ to $\mathbf{y}$. It lies in the $\max\{\delta(\mathbf{z}), \delta(\mathbf{y})\}$-sublevel set of δ. Let $d^* \equiv \max\{\delta(\mathbf{x}), \delta(\mathbf{y}), \delta(\mathbf{z})\}$. By concatenating α and β, we obtain $\alpha \cdot \beta$, a path that starts from $\mathbf{x}$, terminates at $\mathbf{y}$, and lies in the d^*-sublevel of δ. Such d^* is the optimal value because $\mathbf{x}$ and $\mathbf{y}$ must be end points. Observe that when $K = K'$:

Proposition 6. *The straight line path connecting any two points* $\mathbf{x}, \mathbf{y}$ *in a convex subset lies in the* $\max\{\delta(\mathbf{x}), \delta(\mathbf{y})\}$*-sublevel set of* δ*.*

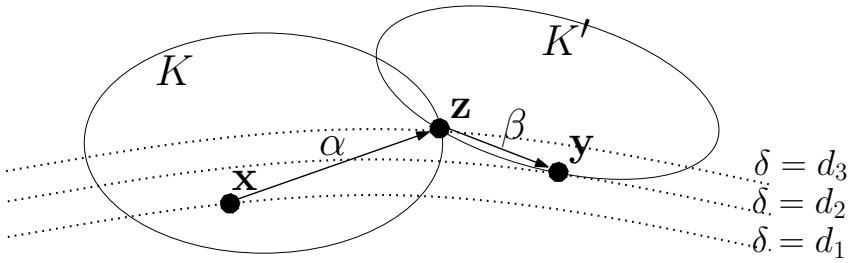

Figure 3.2: An example of K and K'. For this example, $d_1 < d_2 < d_3$. Among paths in $\Gamma(\{\mathbf{x}\}, \{\mathbf{y}\}, K \cup K')$, the optimal path lies in the $\max\{d_1, d_2, d_3\} = d_3$-sublevel of δ.

3.4.1 Optimal Path

Now let us consider a sequence of overlapping convex subsets of $\bar{\Omega}$ denoted by: $K_1, K_2, ..., K_k \subseteq \bar{\Omega}$; for any $i \in \{1, 2, ..., k-1\}$, $K_i \cap K_{i+1} \neq \varnothing$. The fingers are to

move from $\mathbf{z}_0 \in K_1$ to some finger placement in K_2, then to some finger placement in K_3, ..., finally to $\mathbf{z}_k \in K_k$ in such a way that the fingers must remain in K_i while moving from $\mathbf{z}_{i-1}$ to $\mathbf{z}_i$ for any $i \in \{1, 2, ..., k\}$. Under this condition, our task is to find an optimal path that starts at $\mathbf{z}_0$, terminates at $\mathbf{z}_k$, and travels through the sequence of convex subsets. Let $\mathbf{z}_i$ be a finger placement in $K_i \cap K_{i+1}$ such that δ restricted in $K_i \cap K_{i+1}$ attains its minimal value at $\mathbf{z}_i$, for $i \in \{1, 2, ..., k-1\}$. As discussed previously, a straight line path connecting each pair of $\mathbf{z}_{i-1}$ and $\mathbf{z}_i$ is optimal. Since the path has to travel in the given sequence of overlapping convex subsets, concatenation among the straight line paths connecting $\mathbf{z}_{i-1}$ and $\mathbf{z}_i$ forms an optimal path connecting $\mathbf{z}_0$ and $\mathbf{z}_k$. The optimal path is in $d^* \equiv \max\{\delta(\mathbf{z}_i) \mid i \in \{0, 1, ..., k\}\}$-sublevel set of δ.

To determine optimal escape path from a finger placement, we consider all possible sequences of overlapping convex subsets. We decompose $\bar{\Omega}$ into finite overlapping convex subsets and construct a roadmap, a graph $(\mathcal{V}, \mathcal{E})$. Let $\mathcal{K}$ be the set of all decomposed convex subsets of $\bar{\Omega}$ and $\mathcal{I}$ be the set of all intersections between two members in $\mathcal{K}$. Each member $K \in \mathcal{K} \cup \mathcal{I}$ induces a graph vertex $\mathbf{v}$, a member of $\mathcal{V}$. The graph vertex $\mathbf{v}$ is a point where δ restricted in K attains its minimal value. Each member $I = K \cap K' \in \mathcal{I}$ ($K \cap K' \neq \varnothing$ and $K, K' \in \mathcal{K}$) induces two graph edges connecting vertices induced by K, K', and I. For simplicity, we assume that a member of $\mathcal{K}$ is contained in $\partial\mathcal{B}^2$. Recall that all configurations in $\partial\mathcal{B}^2$ are non-caging so any graph vertices induced by any K contained in $\partial\mathcal{B}^2$ are non-caging placements. The set of all these graph vertices are denoted by $\mathcal{V}_{\text{exit}}$.

3.4.2 Maximal Distance Propagation

To determine the maximal distance of a configuration $\mathbf{x} \in K$ where $K \in \mathcal{K}$, previously discussed strategy is translated to computing an upperbound separation distance for each sequence of vertices that starts with $\mathbf{v}$ induced by K and ends with one contained in $\mathcal{V}_{\text{exit}}$. The least upperbound separation distance among such sequences is $\delta^*(\mathbf{v})$. By Proposition 6, the maximal distance of a configuration $\mathbf{x}$ is:

$$\delta^*(\mathbf{x}) = \max\{\delta(\mathbf{x}), \delta^*(\mathbf{v})\}. \tag{3.2}$$

In case that $\mathbf{v}$ is not in $\mathcal{V}_{\text{exit}}$, the escape sequence is not terminated. Let $N_{\mathbf{v}}$ be the set of all vertices adjacent to $\mathbf{v}$. The maximal distance of $\mathbf{v}$ is:

$$\begin{aligned} \delta^*(\mathbf{v}) &= \min_{\mathbf{v}' \in N_{\mathbf{v}}} \left\{ \max \left\{ \delta(\mathbf{v}), \delta^*(\mathbf{v}') \right\} \right\} \\ &= \max \left\{ \delta(\mathbf{v}), \min_{\mathbf{v}' \in N_{\mathbf{v}}} \left\{ \delta^*(\mathbf{v}') \right\} \right\}. \end{aligned} \tag{3.3}$$

The propagation ends at $\mathbf{v}_{\text{exit}} \in \mathcal{V}_{\text{exit}}$ because at such a vertex there exists a path that brings the fingers to the same point, therefore:

$$\delta^*(\mathbf{v}_{\text{exit}}) = \delta(\mathbf{v}_{\text{exit}})$$

We have presented the approach to compute the maximal distance of any given finger placement. For the minimal distance, however, we rely on a different approach to be presented in the following section.

3.5 Stretching Cage

Consider decomposing M-dimensional $\bar{\Omega}$ into convex subsets satisfying the following properties:

1. Each decomposed convex subset must be a compact M-dimensional convex polytope.
2. Any two overlapping convex subsets, say K and K', must share a common $(M-1)$-dimensional face. $K \cap K'$ is a convex polytope on the boundary of K and the boundary of K'. Every vertex of $K \cap K'$ is a vertex of K and a vertex of K'.

We claim that: *an optimal path that starts and terminates at vertices of the decomposed convex polytopes lies on edges of the decomposed convex polytopes.* This is a product of an observation that projecting (pushing) an optimal path in a convex polytope onto the boundary of the convex polytope preserves path optimality. The projection

function onto the relative boundary of K given a source point $\mathbf{s}$ is denoted by $\pi_{K,\mathbf{s}}$. The projected point $\pi_{K,\mathbf{s}}(\mathbf{x})$ satisfies the following properties:

- $\pi_{K,\mathbf{s}}(\mathbf{x})$ lies on the relative boundary of K,
- $\pi_{K,\mathbf{s}}(\mathbf{x})$, $\mathbf{s}$, and $\mathbf{x}$ are collinear, and
- $\mathbf{x}$ is between $\mathbf{s}$ and $\pi_{K,\mathbf{s}}(\mathbf{x})$.

An example of a projection is illustrated in Figure 7.3. When K is a compact convex set and $\mathbf{s}$ is in the relative interior of K, the projection $\pi_{K,\mathbf{s}}$ is well-defined and continuous.

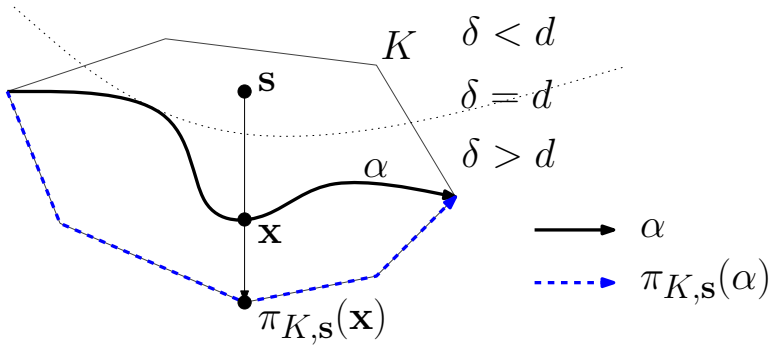

Figure 3.3: Projecting a path α (solid line) to another path $\pi_{K,\mathbf{s}}(\alpha)$ (dashed line) on the relative boundary of K. If α is optimal, the projected path $\pi_{K,\mathbf{s}}(\alpha)$ is optimal as well.

Proposition 7. *Let K be a convex subset of $\bar{\Omega}$, and α be a path in the d-superlevel set of $\delta|_K$. If some point in the relative interior of K are not in the image of α, there exists a source point $\mathbf{s}$ in the relative interior of K such that the projected path $\pi_{K,\mathbf{s}}(\alpha)$ is in the d-superlevel set of $\delta|_{\partial K}$.*

Proof. Let D be the complement of d-superlevel set of δ. We have that D is open and convex since D is the interior of d-sublevel set of a convex function δ, see (Boyd and Vandenberghe, 2004a). If the intersection between D and K is not empty, there exists a source point $\mathbf{s}$ that lies in D and the relative interior of K. Otherwise, any point $\mathbf{s}$ in the relative interior of K that is not in the image of α can be chosen as the

source point. To obtain a path on the relative boundary of K, we project α to the relative boundary of K. The projection $\pi_{K,\mathbf{s}}(\alpha)$ is a path since $\pi_{K,\mathbf{s}}$ is continuous. To show that $\pi_{K,\mathbf{s}}(\alpha)$ is in the d-superlevel set of δ, let us show that it does not overlap with D. If the intersection between D and K is empty, $\pi_{K,\mathbf{s}}(\alpha)$ is always in the d-superlevel set. Otherwise, suppose that $\pi_{K,\mathbf{s}}(\mathbf{z})$ is in D, for some $\mathbf{z} \in \text{img}(\alpha)$. By definition of $\pi_{K,\mathbf{s}}$, $\mathbf{z}$ must lie inbetween $\mathbf{s}$ and $\pi_{K,\mathbf{s}}(\mathbf{z})$; therefore, $\mathbf{z}$ must be in D. This is a contradiction since α is in the d-superlevel set. ■

The following corollary is deduced from the proof of Proposition 7.

Corollary 8. *Let K be a convex subset of $\bar{\Omega}$, and $\mathbf{p}$ be a finger placement in K. If K is not a singleton, there exists a point $\mathbf{s}$ in the relative interior of K that a straight line from $\mathbf{p}$ to $\pi_{K,\mathbf{s}}(\mathbf{p})$ is in the $\delta(\mathbf{p})$-superlevel set of $\delta|_K$.*

Consider the situation when K is a convex M-polytope, $M \geq 2$. By Corollary 8, for a given finger placement $\mathbf{p}$ in K, there exists a source point $\mathbf{s}$ and a straight line path from $\mathbf{p}$ to its projection $\pi_{K,\mathbf{s}}(\mathbf{p})$ in the $\delta(\mathbf{p})$-superlevel set of $\delta|_K$. If $\pi_{K,\mathbf{s}}(\mathbf{p})$ is not a vertex of K, it must lie in the relative interior of a lower dimensional face of K. The lower dimensional face is convex because it is a face of a convex polytope. Consequently, Corollary 8 can be repeatedly applied to the projected point contained in the convex face until the projected point is a vertex of K. This implies that there exists a path that starts from $\mathbf{p}$, terminates at a vertex of K and lies in the $\delta(\mathbf{p})$-superlevel set of $\delta|_K$. The path depends on source points for projection. However, any $\delta(\mathbf{p})$-superlevel path from $\mathbf{p}$ to a vertex of K later proves to be equally useful. We ignore the choice of source points and define β_K as a function that maps a finger placement $\mathbf{p} \in K$ to an optimal path from $\mathbf{p}$ to a vertex of K.

3.5.1 Optimal Path

With necessary background presented, let us prove the claim. Consider an optimal escape path γ that traverses through convex polytopes $K_1, K_2, ..., K_k$. Without loss of generality, we assume that $\gamma = \gamma_1 \cdot \gamma_2 \cdot ... \cdot \gamma_k$ and each path γ_i lies in K_i. Let $\mathbf{a}_j$ and $\mathbf{b}_j$ be the initial and the terminal points of γ_j. For $1 < j < k$, observe

that $\mathbf{a}_j \in K_{j-1} \cap K_j$, $\mathbf{b}_j \in K_j \cap K_{j+1}$, $\mathbf{a}_j = \mathbf{b}_{j-1}$ and $\mathbf{b}_j = \mathbf{a}_{j+1}$. The path γ_j is in the d_j-superlevel set of δ for some value d_j so $\delta(\mathbf{a}_j) \geq d_j$, and $\delta(\mathbf{b}_j) \geq d_j$. By Corollary 8, $\beta_{K_{j-1}\cap K_j}(\mathbf{a}_j)$ and $\beta_{K_j \cap K_{j+1}}(\mathbf{b}_j)$ lies in the d_j-superlevel set of δ. Hence, the path $\gamma'_j \equiv (\beta_{K_{j-1}\cap K_j}(\mathbf{a}_j))^{-1} \cdot \gamma_j \cdot \beta_{K_j \cap K_{j+1}}(\mathbf{b}_j)$ is in the d_j-superlevel of δ. Also observe that γ'_j begins where γ'_{j-1} ends and γ'_j ends where γ'_{j+1} begins.

For any path γ_j, we can always construct another γ'_j which begins at a vertex and ends at another vertex of K_j, denoted by $\mathbf{a}'_j$ and $\mathbf{b}'_j$ respectively. Both paths lie in the d_j superlevel set of δ. Now let us apply Proposition 7 to γ'_j. If γ'_j does not occupy the entire K_j, we can apply the projection described in the proposition to construct another path γ''_j in the d_j-superlevel set of δ. Clearly, γ''_j is a path from $\mathbf{a}'_j$ to $\mathbf{b}'_j$ that lies on convex $(M-1)$-dimensional faces of K_j. Suppose that γ''_j crawls through N convex faces of K_j. We can correspondingly divide γ''_j into N smaller paths each of which lies in each convex $(M-1)$-dimensional face, i.e., $\gamma''_j = \gamma''_{j,1} \cdot \gamma''_{j,2} \cdot ... \cdot \gamma''_{j,N}$. Recall that this is exactly the same as when γ is divided into $\gamma_1 \cdot \gamma_2 \cdot ... \cdot \gamma_k$. The same process is repeated on γ''_j until Proposition 7 can no longer apply. That is, when every divided path occupies the entire face (1-dimensional face) of K_j. This means that for a given optimal path from $\mathbf{a}'_j$ to $\mathbf{b}'_j$, γ'_j, we can construct another optimal path that joins the same pair of vertices and lies on the edges of K_j.

An example of repeated path decomposition and projection is illustrated in Figure 3.4. The path γ_j is in d_j-superlevel set of δ restricted to K_j. End points of γ_j: $\mathbf{a}_j$ and $\mathbf{b}_j$; lies on $K_{j-1} \cap K_j$ (the bottom face of K_j) and on $K_j \cap K_{j+1}$ (the top face of K_j), respectively. The path γ''_j is, by construction, a path that starts from $\mathbf{a}'_j$, terminates at $\mathbf{b}'_j$, and lies in the d_j-superlevel set of δ restricted on the relative boundary of K_j. The end points $\mathbf{a}'_j$ and $\mathbf{b}'_j$ are vertices of K_j. We further decompose γ''_j into three paths each lies on a face of K_j, then repeat the projection process on each of them. Concatenation of the projected paths gives β, a path that starts from $\mathbf{a}'_j$, terminates at $\mathbf{b}'_j$, and lies in the d_j-superlevel set of δ restricted to the edges of K_j.

For $j = 1$, the path $\gamma'_1 = \gamma_1 \cdot \beta_{K_1 \cap K_2}(\mathbf{b}_1)$ since it is assumed that γ_1 begins at a vertex of K_1. Similarly, γ is assumed to terminate at a vertex of K_k so $\gamma'_k =$

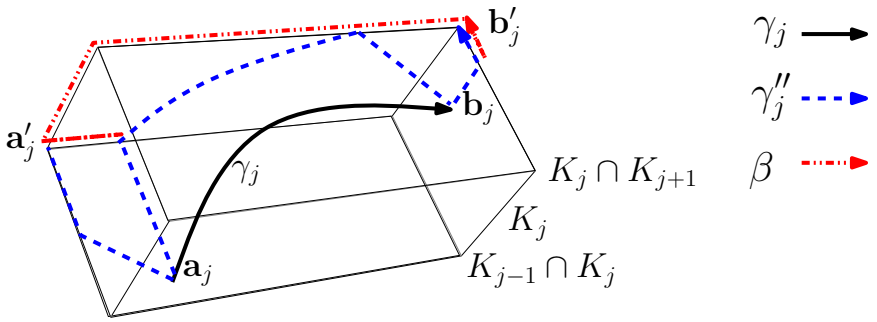

Figure 3.4: An example of repeated path decomposition and projection.

$(\beta_{K_{k-1}\cap K_k}(\mathbf{a}_k))^{-1} \cdot \gamma_k$. Same as other γ'_j, γ'_1 and γ'_k can be successively projected to the edges of their containing convex polytopes. At this point, we successfully prove the claim that an optimal path between vertices of the decomposed convex polytopes exists on the edges of the decomposed convex polytopes.

Let $\mathcal{V}_0$ and $\mathcal{E}_0$ be, respectively, the set of all vertices and the set of all edges of all decomposed convex polytopes. For each $\mathbf{e}$ in $\mathcal{E}_0$, we pick a point that δ restricted to the edge attains its minimal value. Let $\mathcal{M}$ be the set containing all picked points. The graph, the roadmap, for stretching caging $(\mathcal{V}, \mathcal{E})$ is defined as follows. The set of vertices $\mathcal{V}$ the union of $\mathcal{V}_0$ and $\mathcal{M}$. Each vertex $\mathbf{m} \in \mathcal{M}$ connects two vertices in $\mathcal{V}$ which are end points of the edge $\mathbf{e}$ that contains $\mathbf{m}$.

3.5.2 Minimal Distance Propagation

Since each vertex on $\partial\mathcal{B}^2$ is a non-caging configuration, it has the minimal distance equal to its separation distance. Let the set containing these vertices be denoted by $\mathcal{V}_{\text{exit}}$ That is, for each $\mathbf{v}_{\text{exit}} \in \mathcal{V}_{\text{exit}}$,

$$\delta_*(\mathbf{v}_{\text{exit}}) \quad = \quad \delta(\mathbf{v}_{\text{exit}}).$$

From these vertices, we propagate the minimal distance to the rest in $\mathcal{V}$. The propagation channels are the edges in $\mathcal{E}$ which are straight lines connecting adjacent

vertices.

$$\begin{aligned} \delta_*(\mathbf{v}) &= \max_{\mathbf{v}' \in N_\mathbf{v}} \left\{ \min \left\{ \delta(\mathbf{v}), \delta_*(\mathbf{v}') \right\} \right\} \\ &= \min \left\{ \delta(\mathbf{v}), \max_{\mathbf{v}' \in N_\mathbf{v}} \left\{ \delta_*(\mathbf{v}') \right\} \right\} \end{aligned} \tag{3.4}$$

where $N_\mathbf{v}$ is the set of all vertices adjacent to $\mathbf{v}$. Alternatively, it is possible to propagate negative minimal distance:

$$-\delta_*(\mathbf{v}) = \max \left\{ -\delta(\mathbf{v}), \min_{\mathbf{v}' \in N_\mathbf{v}} \left\{ -\delta_*(\mathbf{v}') \right\} \right\}. \tag{3.5}$$

To compute minimal distance for $\mathbf{x}$ in a decomposed convex polytope K and $\mathbf{x}$ is not a vertex in $\mathcal{V}$, recall the optimal path $\beta_K(\mathbf{x})$ that connects $\mathbf{x}$ and some vertex $\mathbf{v}$ of K. Since $\mathbf{v}$ is a vertex in $\mathcal{V}$, the minimal distance $\delta_*(\mathbf{v})$ can be computed from the propagation described above. If $\delta_*(\mathbf{v}) < \delta(\mathbf{x})$ then $\beta_K(\mathbf{x})$ is in $\delta_*(\mathbf{v})$-superlevel set of δ. Therefore, from Proposition 3, $\mathbf{x}$ is in a stretching solution set and $\delta_*(\mathbf{x}) = \delta_*(\mathbf{v})$. Otherwise, $\mathbf{x}$ does not form a stretching cage.

3.6 Spherical Fingers

Assigning the input object $\mathcal{P}$ to be the point-based Minkowski sum of the finger shape and the polytope representing the object reduces the problem of caging with a pair of discs or spheres to the point-finger caging problem, see Figure 3.5. However, the input object may contain curved parts and cannot be represented by a polytope. Running the algorithm with a simplified polytope $\mathcal{P}'$ representing the object as input may generate false solutions, i.e., some reported solutions may not be a caging placement.

3.6.1 Squeezing Caging with Spherical Fingers

If an object $\mathcal{P}'$ is a subset of $\mathcal{P}$, any valid two-finger placement for $\mathcal{P}$ will be valid for $\mathcal{P}'$. Therefore, an escape path for $\mathcal{P}$ is also an escape path for $\mathcal{P}'$. Consequently, the maximal distance of any valid finger placement for $\mathcal{P}'$ is not greater

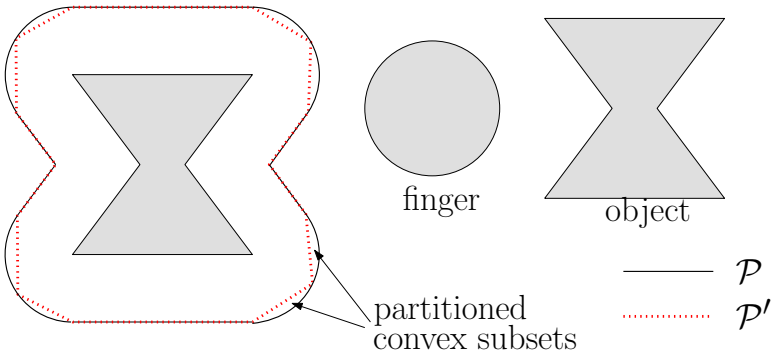

Figure 3.5: The solid outline represents the point-based Minkowski sum of a disc-shaped finger and an object, $\mathcal{P}$. The dotted-line outline represents the simplified geometry $\mathcal{P}'$.

than that for $\mathcal{P}$. That is, if the simplified object is $\mathcal{P}'$ being a subset of the actual object $\mathcal{P}$, any maximal distance for the simplified object will be the lowerbound of that for the actual object. In other words, a cage for $\mathcal{P}'$ is also a cage for the actual object $\mathcal{P}$. It should be noted that some of the reported cages may not be a valid cage for $\mathcal{P}$. Such cages are discarded. If the object is not oversimplified, only small solution sets will be dropped.

We suggest a simplification that preserves straight lines and flat faces of the actual object while curved parts are simplified to edges and faces such that the newly introduced vertices must lie on the boundary of $\mathcal{P}$. The regions occupied by $\mathcal{P}$ but not $\mathcal{P}'$ are partitioned into convex subsets, as in Figure 3.5. This simplification helps us in determining whether a squeezing solution set for $\mathcal{P}'$ should be discarded or not. Given a solution set for $\mathcal{P}'$, let $\mathbf{x} \equiv (x_1, x_2)$ be a finger placement satisfying the following properties:

- δ restricted in the solution set attains its minimal value at $\mathbf{x}$, and
- x_1 and x_2 are in partitioned convex subsets of close$(\mathcal{P})\backslash\mathcal{P}'$ denoted by A_1 and A_2, respectively.

Since A_1 and A_2 are convex, $A_1 \times A_2$ is convex. By Proposition 6, we only need to find where δ restricted in $(A_1 \cap \partial\mathcal{P}) \times (A_2 \cap \partial\mathcal{P})$ attains its minimal value. That is,

computing the least distance between two curves (in two-dimension) or two surfaces (in three-dimension). The least distance is then checked against the maximal distance associated to the solution set for $\mathcal{P}'$ containing $\mathbf{x}$. If it is greater, the solution set is discarded.

3.6.2 Stretching Caging with Spherical Fingers

Applying the same concept, the simplified object $\mathcal{P}'$ must be a subset of the actual object $\mathcal{P}$ so that the minimal distance of any valid placement for $\mathcal{P}'$ is not less than that for $\mathcal{P}$.

The previously introduced simplification that preserves straight lines and flat faces works even better for stretching caging. Let $\mathbf{x}$ be a finger placement at which δ attains its maximal value in a stretching solution set for $\mathcal{P}'$. Without loss of generality, we assume that each finger is at a vertex of $\mathcal{P}'$ at $\mathbf{x}$ (if not we can "push" them to vertices using the β_K function defined in Section 3.5). By the proposed simplification rule, $\mathbf{x}$ must be on the boundary of $\mathcal{P}$. This implies that a finger placement that forms a stretching cage for $\mathcal{P}'$ also forms a stretching cage for $\mathcal{P}$.

3.7 Implementation and Results

The recurrence relation (3.3) is resemble to that of Dijkstra's shortest path algorithm (Cormen et al., 2001). In our case, the maximal distance serves as the "shortest distance" to a non-caging placement. The only difference is that addition is replaced by maximization during updating each adjacent vertex's shortest distance. Since maximization does not decrease the value propagated across edges, it is possible to construct a graph with non-negative cost on each edge for the shortest path algorithm, reducing our problem to the shortest path problem. This also applies for (3.5), the recurrence of negative minimal distances. By solving shortest path problems, we obtain the maximal and minimal distances of graph vertices. We employ a disjoint set structure to keep track of vertices contained in the same solution sets. Initially, a representative set $\zeta_{\mathbf{v}}$ is initialized for each vertex $\mathbf{v}$ in $\mathcal{V}$. By Proposition 1 and 2, any two sets $\zeta_{\mathbf{v}}$ and $\zeta_{\mathbf{v}'}$ are merged if they satisfy:

- $\mathbf{v}'$ is adjacent to $\mathbf{v}$, and
- $\mathbf{v}$ and $\mathbf{v}'$ are connected by a path in the interior of $\delta^*(\mathbf{v})$-sublevel set of δ.

Similar set merging scheme applies to stretching solution sets.

The process to obtain all solution sets is as follows:

1. Decompose $\bar{\mathcal{F}}$ into convex polytopes. We assume that the decomposition does not introduce new vertices for simplicity of analysis. Our implementation used Geompack++ (Joe, 2010) to triangulate polygon and polyhedra, taking $O(v^2)$ time in worst case where v is the number of vertices representing the object.
2. Decompose $\bar{\Omega}$ into convex polytopes. Each decomposed convex polytope is a cartesian product of two decomposed convex subsets of $\bar{\mathcal{F}}$.
3. Initialize the graph $(\mathcal{V}, \mathcal{E})$. This involves computation of optimal points of δ restricted in convex polytopes. For squeezing caging, GJK algorithm (Van den Bergen, 1999) is applied to the shortest distance between two convex polytopes. If the decomposition in the first step is a triangulation, the number of faces, edges are linear to the number of vertices representing the object. Consequently, the total number of graph vertices is $O(v^2)$.
4. Identify $\mathcal{V}_{\text{exit}}$,. Initialize critical distance of such a vertex to its separation distance.
5. Propagate critical distances over the graph $(\mathcal{V}, \mathcal{E})$ according to (3.3) or (3.4). Using the shortest path algorithm, the propagation over the graphs requires $O(v^2 \log v)$
6. Report the solution sets by iterating through every set in the disjoint set structure.

The time complexity in the fifth step dominates that of the others. Our two-finger caging algorithm has $O(v^2 \log v)$ time complexity. Some results produced by

the algorithm are in Figure 3.7, for planar objects, and Figure 3.8, 3.9, 3.10 for three-dimensional objects. The code was written in C++ and tested on Intel Core 2 6300, 1.86 GHz with 1 GB of RAM using only one core. The accumulative running time of steps 2) through 6) is reported in Table 4.1. Time spent for computing stretching solution sets is significantly less than that for squeezing ones. This is because the computation of minimal distance between two features in step 3), computing distance between features used in stretching caging is simpler and less time consuming. It should be noted that the number of solution sets does not significantly affect the running time because the algorithm has to operate on every graph vertex regardless of the number of solution sets.

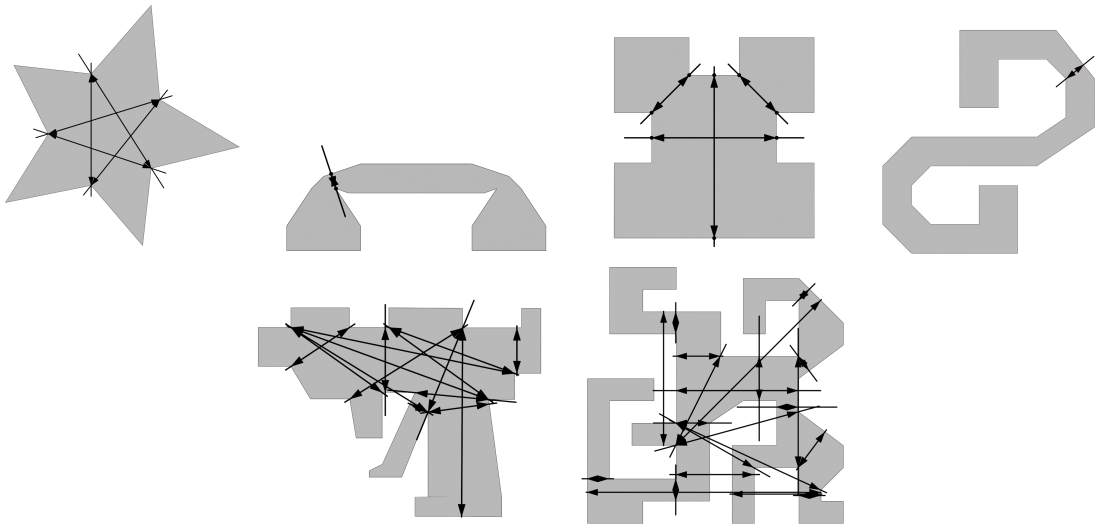

Figure 3.6: The objects are: *star*, *phone*, *jigsaw*, *snake*, *gun*, *maze*. Each line segment's length is equal to a maximal distance of a squeezing cage and the arrows' ends corresponds to a configuration with least separation distance in its containing solution set.

To find in which solution set a finger placement $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^{2w}$ is, we first locate its containing $K \equiv F_1 \times F_2$ where F_1, F_2 are decomposed convex subsets of $\bar{\mathcal{F}}$. The convex sets f_1 and f_2 can be found by solving a point location problem (Snoeyink, 2004), which takes $O(\log v)$ time. For squeezing caging, compute the maximal distance with (3.2). If it is greater than $\delta(\mathbf{x})$, the finger placement $\mathbf{x}$ is in a squeezing solution set $\zeta_{\mathbf{v}}$ where $\mathbf{v}$ is the vertex associated to K. For stretching caging, we proceed to check whether $\mathbf{x}$ lies in a stretching solution set by successively projecting $\mathbf{x}$ to a vertex $\mathbf{v}$ of K. By Corollary 8, the minimal distance is $\min\{\delta(\mathbf{x}), \delta_*(\mathbf{v})\}$. If $\delta(\mathbf{x})$ is greater than the minimal distance, $\mathbf{x}$ is in a stretching solution set $\zeta_{\mathbf{v}}$.

3.8 Summary

In this chapter, we have presented an algorithm for reporting all solution sets for two point fingers and a given two or three-dimensional polyhedra representing a rigid object. The algorithm also generates graph structures from which maximal and minimal distance at any finger placement can be queried. For spherical finger case, we have presented a simple extension to report approximate solution sets

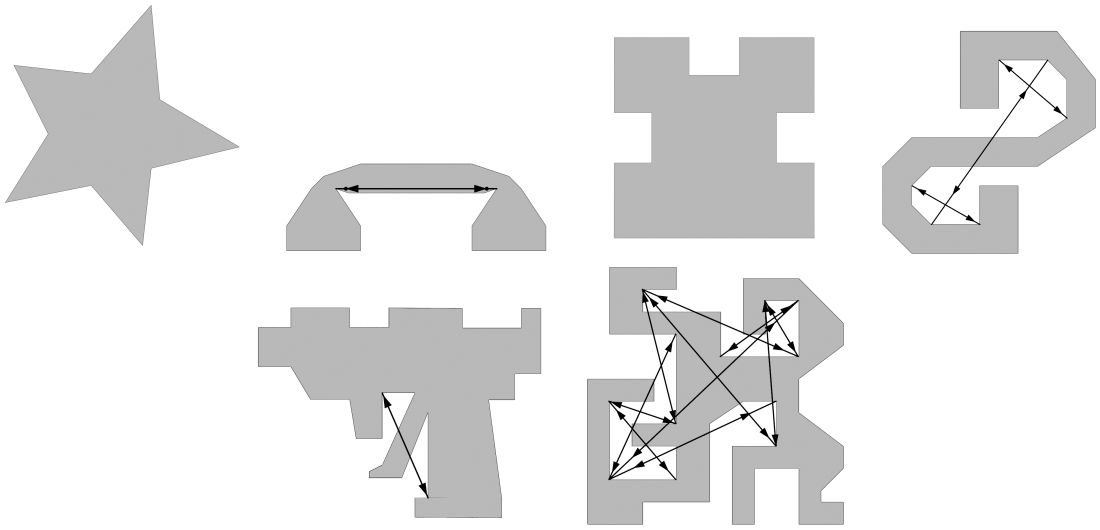

Figure 3.7: The distance between each pair of arrows' ends is equal to a minimal distance of a stretching cage and the line segment length corresponds to a configuration with greatest separation distance in its containing solution set.

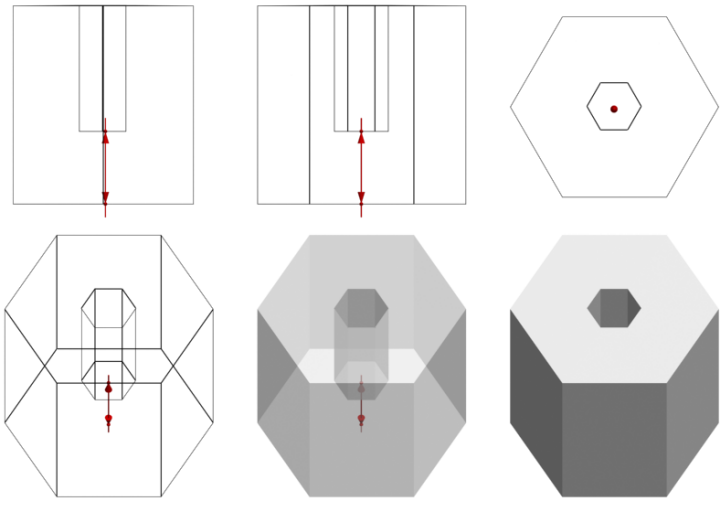

Figure 3.8: Extruded *hexagon* with a hole. Only one solution set of the squeezing constraint is caging at the bottom of the hole. Top row: left, front, top wireframe views of the object. Bottom row: wireframe, transparent, shaded views of the object.

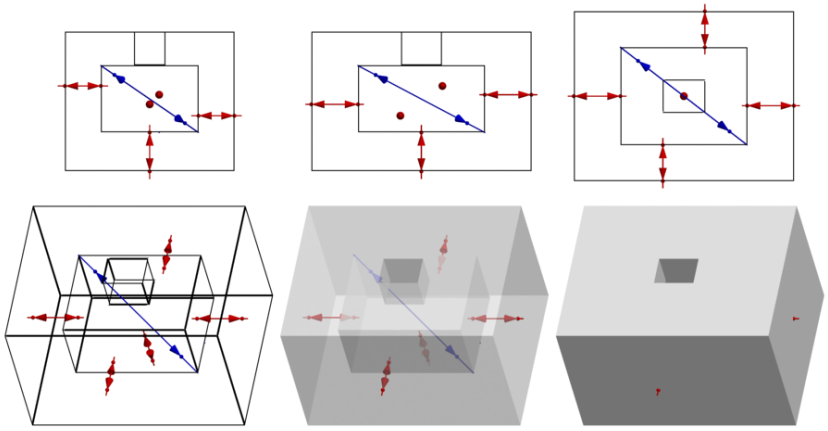

Figure 3.9: *Culled* box. For each solution set, the fingers must be on each side of the box except the top side (five squeezing solution sets in total). The only one stretching solution set is caging by stretching fingers inside the box.

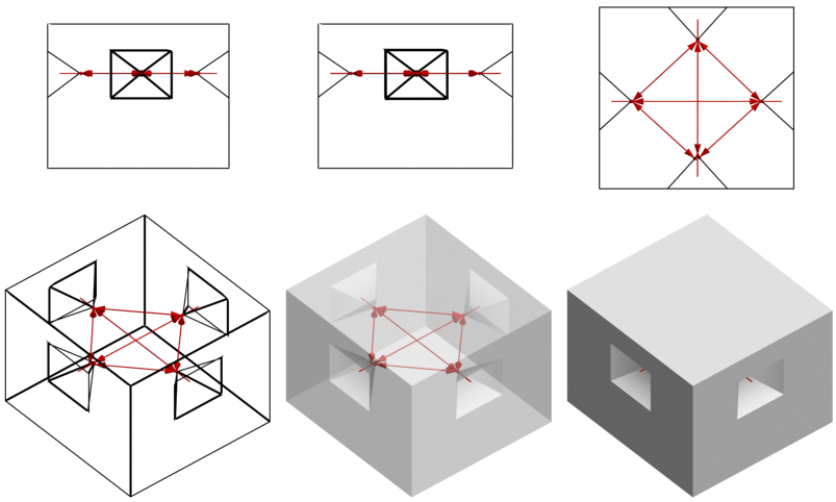

Figure 3.10: *Crate*. Two out of six squeezing solution sets require the fingers to be at opposite concave sections. The rest requires the fingers to be concave sections of adjacent sides. The algorithm does not report any stretching solution sets.

Table 3.1: Algorithm Execution Time (Two-Finger Caging)

object	v	w	squeeze		stretch	
			#	ms	#	ms
star	14	2	5	5	0	2
phone	22	2	1	10	1	4
jigsaw	20	2	4	8	0	3
snake	28	2	1	29	3	18
gun	38	2	15	75	1	53
maze	54	2	22	246	11	200
hexagon	36	3	1	1214	0	22
culled	33	3	5	528	1	17
crate	41	3	6	1821	0	31

guaranteed to be subsets of the actual ones. The extension to the algorithm to support more fingers is presented in the next chapter.

CHAPTER IV

MULTI-FINGER SQUEEZING CAGING

4.1 Introduction

In case of two-finger caging, squeezing and stretching are primary operations of the manipulator which control the separation distance between the fingers. To control the separation distance from going above (below) an upperbound (lower-bound) is to impose a constraint. At an appropriate finger placemen, imposing an appropriate constraint on the fingers results in caging the object, see Figure 1.1(a) and 1.1(b), for example. The bound cannot be arbitrarily large or small. At a critical distance, the cage is broken. The critical distance and a representative caging placement are sufficient to describe a maximally connected set of caging placements for two-finger caging.

Erickson et al. (Erickson et al., 2003) address the problem of three finger caging of a convex polygon on a plane. Two out of the three fingers are called base fingers. They are constrained in such a way that their separation distance is set to a given constant. Erickson et al. propose an exact algorithm that, given positions of the base fingers, computes a *capture region*, a region to place the other finger that together with the fixed fingers cages the object. The time complexity of the algorithm is $O(v^6)$ where v is the number of the polygon's vertices. Vahedi and Stappen (Vahedi and van der Stappen, 2008) extend the results to non-convex polygons in $O(v^6 \log^2 v)$. It should be noted that the capture region is actually a 2D slice of the 6D caging set. Each slice is identified by the positions of the fixed fingers.

An approach to multi-finger caging relies on controlling a 1-DOF function that depends on the positions of the fingers. This generalizes the two-finger caging with the separation distance as the 1-DOF function. Our preliminary work (Pipattanasomporn et al., 2008) extends the two-finger caging algorithm to a multi-finger caging algorithm under the assumption that the 1-DOF function is convex and invariant to the manipulators' coordinate frame. A few years later, Rodriguez et al.

(Rodriguez et al., 2011) study the condition for a multi-finger cage to constitute waypoint to grasping, a pregrasping cage. They show that caging by preventing a value of a function F below or above a value is a pregrasping cage if caging in such a manner is not possible as the value of F gets arbitrary large or small.

Inspired by 1-DOF constraint-based approaches to caging, we extend the two-finger squeezing caging to n-finger squeezing caging with user-provided constraints. The object is caged by controlling the fingers in a sufficiently tight formation satisfying the constraints. Our goal is to develop an algorithm for identifying maximally connected sets of finger placements that cage an object in the aforesaid manner. The maximally connected sets will be referred to as solution sets to be represented by their representative finger placement and the constraints. Caging is a two-step process. The first step is to place the fingers inside a solution set. Then, impose the constraints. The constraints are induced by physical limitations, kinematics of the manipulator, and/or control policies. Some constraints will only be active after the initial finger placement setup, e.g., the ones that keeps the fingers squeezed together. The others are always active, for example, those induced by physical limitation of the manipulator. In practice, the manipulator can be free moving robots on a plane constrained by communication distance among the robots, or an articulated hand with constraints among links. The ability to support user-provided constraints permits our algorithm to report solutions, caging strategies, that take advantage of specific environments. Theoretically, the constraints invalidate some caging placements but also restrict possible motion of the fingers. They both hinder and help in caging. To the best of our knowledge, none of previously published caging algorithms have neither taken constraints from the manipulator into account nor supported custom constraints. Given a number of fingers and the constraints, our algorithms generate solution sets from a given 2D or 3D object. They are capable of reporting exact solutions if all constraints are convex; otherwise, approximate solutions.

4.2 Formulation

This chapter addresses the problem of caging a rigid object $\mathcal{P}$ with n point fingers. Same as in the previous chapter, the object $\mathcal{P}$ is represented by an open but bounded w-polyhedra. It may consist of multiple components but all of them must move together as a single rigid body. The compliment of $\mathcal{P}$ is assumed to be connected, i.e., the object does not contain any inaccessible holes. A state of the fingers is called a finger placement represented by $\mathbf{x} = (\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_n)$ in $\mathbb{R}^{w \times n}$. Each $\mathrm{x}_i \in \mathbb{R}^w$ is the position of i-th finger in the object's coordinates. A finger placement is valid if none of the fingers are in the interior of the object. The free workspace, i.e., the set of all valid positions of a finger, is given by $\mathcal{F} \equiv \mathbb{R}^w \backslash \mathcal{P}$. The free configuration space, or c-free, of n point fingers, when no constraints are enforced, is $\Omega \equiv \mathcal{F}^n$; while $\mathbb{R}^{w \times n} \backslash \Omega$ is the c-obstacle.

In this section, the constraints imposed on the fingers have to be in the following form: $\delta(\mathbf{x}) < d, d \in \mathbb{R}$ where $\delta : \mathbb{R}^{w \times n} \to \mathbb{R}$ is a function that is convex, and does not change its value after rotating or translating the whole finger formation. The function δ and its corresponding constraint is called a *dispersion function* and a *dispersion constraint*, respectively. The second property of δ ensures that $\delta(\mathbf{x}) = \delta(\mathbf{x}')$ if there exists a rigid transformation $\mathbf{T}$ that transforms each i^{th}-finger x_i to $\mathbf{T}(\mathrm{x}_i) = \mathrm{x}'_i$. A dispersion function depends only on the formation shape of the finger placement. Changes of coordinate frame or the object's pose (rotation and translation) do not affect the function nor the constraint.

The following are example dispersion functions:

- the greatest separation distance between a pair of fingers:

$$\delta^{\infty}(\mathbf{x}) \equiv \max \left\{ \|\mathrm{x}_i - \mathrm{x}_j\| \mid i, j \in \{0, 1, ..., n\} \right\},$$

- the maximal distance of any two adjacent fingers:

$$\delta^{\infty}_{\text{adj}}(\mathbf{x}) \equiv \max \left\{ \|\mathrm{x}_n - \mathrm{x}_1\|, \|\mathrm{x}_1 - \mathrm{x}_2\|, ..., \|\mathrm{x}_{n-1} - \mathrm{x}_n\| \right\},$$

- the sum of square distance of any two adjacent fingers:

$$\delta^2_{\text{adj}}(\mathbf{x}) \equiv \|\mathrm{x}_n - \mathrm{x}_1\|^2 + \|\mathrm{x}_1 - \mathrm{x}_2\|^2 + ... + \|\mathrm{x}_{n-1} - \mathrm{x}_n\|^2.$$

All of these functions are convex since they are addition or supremum of norms (Boyd and Vandenberghe, 2004b). In addition, they are invariant to rigid transformations of the finger formation because they depend only on distances between the fingers.

We claim that a dispersion function attains its minimal value at any placement that all fingers collapse to a single point.

Proposition 9. *For any* $\mathbf{x}$ *and* $\mathbf{x}'$ *having the same formation shape, the dispersion function values at any finger placements that lie between* $\mathbf{x}$ *and* $\mathbf{x}'$ *are less than those at* $\mathbf{x}$ *and* $\mathbf{x}'$.

Proof. The invariance property of a dispersion function's implies that $\delta(\mathbf{x}) = \delta(\mathbf{x}')$. A dispersion function is a convex function so it satisfies Jensen's inequality (Boyd and Vandenberghe, 2004a). Hence, dispersion of every point inbetween $\mathbf{x}$ and $\mathbf{x}'$ is less than $\delta(\mathbf{x}) = \delta(\mathbf{x}')$. ■

Proposition 10. *Every dispersion function* δ *attains its minimal value when all the fingers collapse to a point, i.e., at* $(\mathrm{x}, \mathrm{x}, ..., \mathrm{x}) \in \mathbb{R}^{w \times n}$ *for any* $\mathrm{x} \in \mathbb{R}^w$.

Proof. Since any dispersion function is rigid-transformation invariant, it suffices to show that any dispersion function attains its minimal value at the origin: $\mathbf{0}_{w \times n} \equiv (0_m, 0_m, ..., 0_m) \in \mathbb{R}^{w \times n}$, where 0_m is a w-vector with all zero elements.

When w is even, $-\mathbf{I}_w \in SO(w)$ where $\mathbf{I}_w$ is a w-by-w identity matrix. It is possible to rigidly transform an arbitrary finger placement $\mathbf{x}$ to $-\mathbf{x}$ with the rotation $\mathbf{R} = -\mathbf{I}_w$, and the translation $\mathrm{t} = 0_w$. Since the origin lies at the midpoint between $\mathbf{x}$ and $\mathbf{x}' = -\mathbf{x}$, the dispersion function attains its minimal value at the origin, by Proposition 9.

When w is odd, it can be verified that:

$$\mathbf{A} \equiv \begin{pmatrix} 1 & 0_{(w-1)}^{\mathrm{T}} \\ 0_{(w-1)} & -\mathbf{I}_{(w-1)} \end{pmatrix}, \bar{\mathbf{A}} \equiv \begin{pmatrix} -\mathbf{I}_{(w-1)} & 0_{(w-1)} \\ 0_{(w-1)}^{\mathrm{T}} & 1 \end{pmatrix};$$

are both in $SO(w)$. Let $\mathbf{x}$ be an arbitrary finger placement in the form:

$$\mathbf{x} \equiv \begin{pmatrix} \mathbf{t} \\ \mathbf{b} \end{pmatrix} \in \mathbb{R}^{w \times n}, \mathbf{t} \in \mathbb{R}^{1 \times n}, \mathbf{b} \in \mathbb{R}^{(w-1) \times n}.$$

Consider the midpoint $\mathbf{z}$ between $\mathbf{x}$ and $\mathbf{Ax}$:

$$\mathbf{z} \equiv \frac{1}{2}(\mathbf{x} + \mathbf{Ax}) = \frac{1}{2} \begin{pmatrix} \mathbf{t} + \mathbf{t} \\ \mathbf{b} - \mathbf{b} \end{pmatrix} = \begin{pmatrix} \mathbf{t} \\ \mathbf{0}_{(w-1) \times n} \end{pmatrix},$$

and the midpoint between $\mathbf{z}$ and $\bar{\mathbf{A}}\mathbf{z}$:

$$\frac{1}{2}(\mathbf{z} + \bar{\mathbf{A}}\mathbf{z}) = \frac{1}{2} \begin{pmatrix} \mathbf{t} - \mathbf{t} \\ \mathbf{0}_{(w-1) \times n} + \mathbf{0}_{(w-1) \times n} \end{pmatrix} = \mathbf{0}_{w \times n},$$

where $\mathbf{0}_{(w-1) \times n}$ is the $(w-1)$-by-n zero matrix. Apply Proposition 9 twice to obtain that $\delta(\mathbf{z}) \leq \delta(\mathbf{x})$ and $\delta(\mathbf{0}_{w \times n}) \leq \delta(\mathbf{z})$. Hence, $\delta(\mathbf{0}_{w \times n}) \leq \delta(\mathbf{x})$ for any $\mathbf{x} \in \mathbb{R}^{w \times n}$. ■

This implies that if a dispersion constraint is feasible, every placement that all fingers are at the same point will always satisfy the dispersion constraint. Figure 4.1 illustrates a typical relationship among finger placements, dispersion values and caging states. Roughly speaking, a dispersion function measures how loose a formation of fingers is. To control a dispersion function is to control the looseness of the formation, independent of the object's pose or the coordinate frame. A separation distance function in two-finger caging is a dispersion function as $\delta^{\infty}, \delta^{\infty}_{\mathrm{adj}}$ reduces to the separation distance function when $n = 2$.

A dispersion constraint: $\delta < d$; restricts feasible finger placements, only those with δ less than d is feasible. Let us first assume that the constraint is toggleable, and

is not activated until an initial finger placement setup is complete. Prior to the activation, the free configuration space is Ω so the fingers are free to move everywhere by the assumption on $\mathcal{P}$. Imposing constraints restricts the fingers' configuration to be in a connected component of the intersection between Ω and the constraint's feasible region. A finger placement $\mathbf{x}$ is said to be in a solution set (for the constraint) if its containing connected component is bounded in some dimension, i.e., some fingers cannot move arbitrarily far from the object. When fingers cannot cage the object, they can translate anywhere. Therefore, a finger placement in the connected component that is unbounded in any dimension cannot be in a cage formed by the constraint.

The real constant d of the constraint need to be adjusted to fit each caging site and the object's shape. For example, the fingers under δ^2_{adj} need to be more spread out when caging a large section so the constant has to be relatively large compare to another constant applied to caging a smaller section. Similar to the critical distance in two-finger squeezing caging, we are interested in determining the largest possible value that the constraint can still prevent the object from escaping. Such a value is called a maximal dispersion. When the dispersion constraint is δ^2_{adj} and $n = 2$, it reduces to the critical distance. The maximal dispersion at $\mathbf{x}$ is denoted by $\delta^*(\mathbf{x})$. It is derived from an "optimal escape path" that starts at $\mathbf{x}$. A path is a continuous map from an interval to finger placements in the free configuration space. A path is an escape path if it brings all the fingers arbitrarily far from the object. A path is not blocked by the constraint $\delta < d$ if the supremum value of δ restricted on the path's image is less than d. A path α is said to be optimal among other paths satisfying some given feasibility condition if it minimizes:

$$\hat{\delta}(\alpha) \equiv \sup_{\mathbf{y} \in \text{img}(\alpha)} \delta(\mathbf{y}).$$

For an optimal escape path, the feasibility condition is that the path must be an escape path from $\mathbf{x}$. The maximal dispersion at a finger placement $\mathbf{x}$ is:

$$\delta^*(\mathbf{x}) \equiv \inf_{\alpha \in \Gamma(\{\mathbf{x}\}, \mathcal{T}, \Omega)} \hat{\delta}(\alpha),$$

Any path $\alpha \in \Gamma(\{\mathbf{x}\}, \mathcal{T}, \Omega)$ with $\hat{\delta}(\alpha) = \delta^*(\mathbf{x})$ is an optimal escape path from $\mathbf{x}$. If the starting point (initial finger placement), say $\mathbf{x}$, is such that $\delta(\mathbf{x}) = \delta^*(\mathbf{x})$, it is impossible to satisfy the dispersion constraint – keeping the dispersion below $\delta^*(\mathbf{x})$. Every placement that all fingers collapse to a point is not in a cage formed by any dispersion constraint because a point can translate to escape and no dispersion value is changed during the process. In case that $\delta(\mathbf{x}) < \delta^*(\mathbf{x})$, imposing the dispersion constraint $\delta < \delta^*(\mathbf{x})$ will always result in a successful caging.

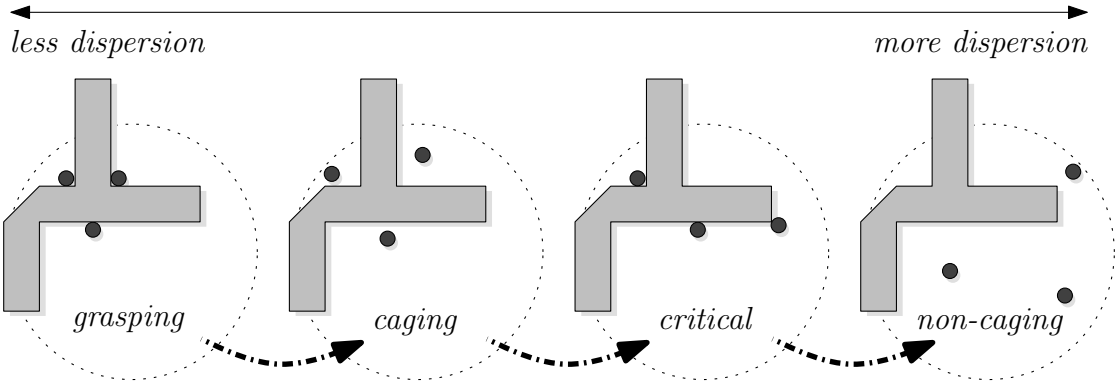

Figure 4.1: Formation dispersion increases from left to right.

An important property of maximal dispersion is that: the maximal dispersion of a placement $\mathbf{x}$ is the same as that of a placement $\mathbf{y}$ if they are connected by any path in a d-sublevel set of δ such that d is less than the maximal dispersion of $\mathbf{y}$. The proof is exactly the same as Proposition 1.

Proposition 11. *The maximal dispersion of any finger placements* $\mathbf{x}$, $\mathbf{y}$ *are equal if there exists a path* α *from* $\mathbf{x}$ *to* $\mathbf{y}$ *such that* α *is contained in d-sublevel set of* δ*, for some* $d < \delta^*(\mathbf{y})$.

This leads to the following fact (similar to Proposition 2) that:

Proposition 12. *All finger placements in a cage formed by a dispersion constraint share the same maximal dispersion.*

Proposition 12 implies that it is reasonable to report maximally connected set

of caging placements with a representative caging placement $\mathbf{x}$ and its maximal dispersion $\delta^*(\mathbf{x})$.

4.3 Optimal Path Construction

To compute a maximal dispersion, we construct an escape path guaranteed to be optimal. The construction process is the same as that described in two-finger squeezing caging with a dispersion function δ. The following is a review of Chapter 3.4. Suppose that the fingers are to move a placement $\mathbf{x}$ inside a convex set $K_1 \subseteq \Omega$. to another placement $\mathbf{y}$ in another convex set $K_2 \subseteq \Omega$ and the intersection between K_1 and K_2 is not empty. Let $\mathbf{z}$ be a finger placement at which δ restricted on $K_1 \cap K_2$ attains its minimal value Applying Jensen's inequality (Boyd and Vandenberghe, 2004a), the concatenation of the two straight line paths from $\mathbf{x}$ to $\mathbf{z}$ and from $\mathbf{z}$ to $\mathbf{y}$ is an optimal path contained in $\max\{\delta(\mathbf{x}), \delta(\mathbf{y}), \delta(\mathbf{z})\}$-sublevel set of δ.

An escape path starting at a given placement possibly passes through any valid sequences of overlapping convex subsets of Ω. To consider all possibilities, we decompose Ω into finite overlapping convex subsets. Let $\mathcal{K}$ be the collection of all decomposed convex subsets and $\mathcal{I}$ contains all non-empty intersections of any two overlapping convex subsets in $\mathcal{K}$. That is:

$$\mathcal{I} \equiv \{K_1 \cap K_2 \mid K_1, K_2 \in \mathcal{K}, K_1 \cap K_2 \neq \varnothing\}.$$

The set $\mathcal{K}$ covers Ω, i.e., $\bigcup_{K \in \mathcal{K}} K = \Omega$, and every set in $\mathcal{I}$ is also a convex subset of Ω.

All optimal paths constructed this way will lie on a one-dimensional skeleton of Ω except when connecting their terminal points. The skeleton consists of:

- distinguished minimizers of δ restricted in each convex subset, and
- straight lines connecting them if their corresponding convex subsets overlap.

This structure forms a graph $(\mathcal{V}, \mathcal{E})$ such that the vertices are the distinguished min-

imizers and the edges are the straight lines connecting adjacent vertices.

4.4 Maximal Dispersion Propagation

The maximal dispersion propagation is the same as maximal distance propagation except how *exit vertices* are chosen. According to Proposition 11 and 6, a placement $\mathbf{x}$ in a decomposed convex subset $K \in \mathcal{K}$ with an associated vertex $\mathbf{v} \in \mathcal{V}$ has:

$$\delta^*(\mathbf{x}) = \max\left\{\delta(\mathbf{x}), \delta^*(\mathbf{v})\right\}. \tag{4.1}$$

That is, it is possible to evaluate the maximal dispersion of $\mathbf{x}$ if that of $\mathbf{v}$ is known. If K contains a placement that all fingers collapse to a point, $\delta^*(\mathbf{v})$ is trivially the global minimal value of δ. The maximal dispersion propagation will begin at such $\mathbf{v}$, an exit vertex. Otherwise, the maximal dispersion of $\mathbf{v}$ will be propagated from the others whose maximal dispersions are already known. Let $\mathcal{N}_\mathbf{v}$ be the set of all vertices adjacent to $\mathbf{v}$. Any member in $\mathcal{N}_\mathbf{v}$ must be in the same convex set as $\mathbf{v}$ by construction, it follows from Proposition 6 that, for any $\mathbf{v}' \in \mathcal{N}_\mathbf{v}$:

$$\delta^*(\mathbf{v}) \;\leq\; \max\left\{\delta(\mathbf{v}), \delta^*(\mathbf{v}')\right\}. \tag{4.2}$$

We claim that if $\mathbf{v}$ is not an exit vertex, an optimal escape path that begins with a straight line from $\mathbf{v}$ to some $\mathbf{v}' \in \mathcal{N}_\mathbf{v}$ exists. Therefore:

$$\begin{aligned} \delta^*(\mathbf{v}) &= \min_{\mathbf{v}' \in \mathcal{N}_\mathbf{v}} \max\left\{\delta(\mathbf{v}), \delta^*(\mathbf{v}')\right\} \\ &= \max\left\{\delta(\mathbf{v}), \min_{\mathbf{v}' \in \mathcal{N}_\mathbf{v}} \delta^*(\mathbf{v}')\right\}. \end{aligned} \tag{4.3}$$

Let us suppose for a contradiction that it is possible to obtain a "better" escape path, say α, contained in K entirely, that is:

$$\delta(\mathbf{v}) \leq \hat{\delta}(\alpha) < \min_{\mathbf{v}' \in \mathcal{N}_\mathbf{v}} \delta^*(\mathbf{v}'). \tag{4.4}$$

Once the fingers can move arbitrarily far from the object along α, they can always translate to somewhere spacious enough and collapse to a single point (by Proposition 10). If the process of collapsing to a point is a path in K, it will contradict with the assumption that $\mathbf{v}$ is not an exit vertex. Otherwise, there exists a path that leaves K yet better than one of the suggested straight lines, also a contradiction.

4.5 Dispersion Constrained Squeezing Caging

The previously discussed method computes maximal dispersion values for a specific δ without additional constraints on the fingers. When the fingers are constrained, some escape paths will be invalid. We denote the free configuration space under constraints by Ω'. The maximal dispersion at $\mathbf{x}$ under constraints is defined similarly but the set of all escape paths from $\mathbf{x}$: $\Gamma(\{\mathbf{x}\}, \mathcal{T}, \Omega')$ is a subset of $\Gamma(\{\mathbf{x}\}, \mathcal{T}, \Omega)$ since Ω' is a subset of Ω. The problem is that it may not possible to decompose Ω' into finite overlapping convex subsets, e.g., Ω' that consists of smooth concave sections. However, when the constraints on the fingers are convex, such a decomposition can be obtained by applying the convex constraints on decomposed convex subsets of Ω. The constraints should be independent of the object's pose as well.

Apart from δ, we assume additional dispersion constraints in the form: $\delta_1 < d_1, \delta_2 < d_2, ..., \delta_c < d_c$. Imposing constraints restricts the fingers' configuration to be in the intersection of Ω and the constraints' feasible regions. The intersection among the free configuration space and the feasible regions of the constraints $1, 2, ..., i$ is denoted by Ω_i. We also define $\Omega_0 \equiv \Omega$ for convenience. Since the constraints are all dispersion constraints, if a finger placement that all fingers are at the same point is infeasible then every finger placement is infeasible for the entire Ω_c. In such a case, no solution is reported, a trivial case.

The constraints are classified into: *always-active constraints* and *toggleable constraints*. The former cannot be turned off and will also restrict the fingers' movement even during the initial finger placement setup phase. They are for example induced by physical limitations of the manipulator. The latter, on the other hand, are not im-

posed during the initial setup phase, e.g., additional constraints from control policy. The first b constraints: $\delta_1 < d_1, \delta_2 < d_2, ..., \delta_b < d_b$, are assumed to be always-active constraints while $\delta_{b+1} < d_{b+1}, \delta_{b+2} < d_{b+2}, ..., \delta_c < d_c$ are toggleable constraints. The intersection between Ω_c and d-sublevel set of δ for some d is the set of all valid finger placements after imposing all of the constraints. A finger placement $\mathbf{x}$ is said to be a in a solution set (for the constraints) if $\mathbf{x}$ is contained in a connected component of the intersection that is bounded in some dimension. We are only interested in such $\mathbf{x}$ that is also contained in a connected component of Ω_b that is unbounded in any dimension because they are "reachable" by the fingers. Otherwise, either the fingers cannot reach it or the fingers always cage the object by the effect of the always-active constraints.

4.6 Fundamental Algorithm

The recurrence relation (4.3) resembles to that of Dijkstra's shortest path algorithm (Cormen et al., 2001). The maximal dispersion of a vertex $\mathbf{v}$, $\delta^*(\mathbf{v})$, corresponds to the "shortest distance to $\mathbf{v}$". The "cost" of an edge connecting $\mathbf{v}$ and $\mathbf{v}'$ is the cost of a straight path from $\mathbf{v}$ to $\mathbf{v}'$, $\max\{\delta(\mathbf{v}), \delta(\mathbf{v}')\}$. The cost of a path is not the sum but the maximum of edge cost.

We allocate a disjoint set structure to maintain sets of vertices contained in the same solution set. Initially, for each vertex $\mathbf{v} \in \mathcal{V}$, its corresponding disjoint set denoted by $\zeta_{\mathbf{v}}$ is initialized as a root. By Proposition 11 and 12, any two sets $\zeta_{\mathbf{v}}$ and $\zeta_{\mathbf{v}'}$ will be merged if:

- $\mathbf{v}$ is adjacent to $\mathbf{v}'$, and
- $\mathbf{v}$ and $\mathbf{v}'$ are connected by a path lying in the interior of $\delta^*(\mathbf{v}')$-sublevel set.

A question that arises is whether this method will report all solution sets for the constraint.

If a vertex $\mathbf{v}$ that corresponds to a decomposed convex subset is not in a solution set, the convex subset will not overlap with any solution set. This is because

a straight line from any placement $\mathbf{x}$ in the convex subset to $\mathbf{v}$ can be concatenated with an escape path from $\mathbf{v}$ to form an escape path from $\mathbf{x}$ in $\delta(\mathbf{x})$-sublevel set of δ.

Proposition 13. *Any convex subset of Ω overlaps with at most one solution set.*

Proof. Suppose for contradiction that $\mathbf{x}$ and $\mathbf{y}$ are in the same convex subset of Ω_c but are from two different solution sets. We assume without loss of generality that $\delta^*(\mathbf{x}) \geq \delta^*(\mathbf{y})$. Since $\mathbf{x}$ and $\mathbf{y}$ are from different solution sets, they are not connected by any path that lies in the interior $\delta^*(\mathbf{x})$-sublevel set of δ. However, Proposition 6 states that there exists a path from $\mathbf{x}$ to $\mathbf{y}$ that is in the $\max\{\delta(\mathbf{x}), \delta(\mathbf{y})\}$-sublevel set of δ. Observe that both $\mathbf{x}$ and $\mathbf{y}$ are from different solution sets so $\delta(\mathbf{x}) < \delta^*(\mathbf{x})$ and $\delta(\mathbf{y}) < \delta^*(\mathbf{y})$. Hence, $\max\{\delta(\mathbf{x}), \delta(\mathbf{y})\} < \max\{\delta^*(\mathbf{x}), \delta^*(\mathbf{y})\} \leq \delta^*(\mathbf{x})$. By Proposition 11, $\mathbf{x}$ and $\mathbf{y}$ are in the same solution set. This is a contradiction. ■

Since we have decomposed the configuration space into convex subsets, each solution set must overlap with some decomposed convex sets. If a decomposed convex set overlap with one of the solution sets, its corresponding vertex has to be in the solution set. Hence, the algorithm will report all solution sets.

For simplicity, we will first consider the case that $\delta_1 < d_1, \delta_2 < d_2, ..., \delta_c < d_c$, are all always-active constraints, i.e., $b = c$. The implementation details and time complexity analysis of the algorithm are as follows:

1. Decompose the free workspace $\mathcal{F}$ into convex subsets by triangulating $\mathcal{F}$ without introducing new vertices. This operation produces a finite subcover, denoted by $\mathcal{W}$, of $\mathcal{F}$ containing convex w-faces (w-dimensional polytope). The size $\mathcal{W}$ is linearly proportional to v, the number of vertices representing the object $\mathcal{P}$. Geompack++ (Joe, 1991) was used in triangulation. The worst-case time complexity is $O(v^2)$.

2. Construct $\mathcal{K}$ and $\mathcal{I}$. Each member in $\mathcal{K}$ is created from Cartesian products among n decomposed convex subsets of $\mathcal{F}$; therefore, is convex and covers

the free configuration space Ω.

$$\mathcal{K} \;=\; \{W_1 \times W_2 \times ... \times W_n \mid W_i \in \mathcal{W}\}\,.$$

Consider two members of $\mathcal{K}$ in the form: $W_1 \times W_2 \times ... \times W_n$ and $W_1' \times W_2' \times ... \times W_n'$. Their intersection will be $(nw-1)$-dimensional if, for an integer i, the face W_i and W_i' are adjacent to each other in the workspace and $W_j = W_j'$ for any $j \neq i$. Every member of $\mathcal{I}$ is also a convex set since each is an intersection between two convex sets. Adding smaller-dimensional intersections will be more than necessary and slow down the algorithm.

3. Construct the graph $(\mathcal{V}, \mathcal{E})$. For every member X in $\mathcal{K}$ and $\mathcal{I}$, we formulate a convex optimization problem: find $\mathbf{v}$ that minimizes δ subject to the dispersion constraints: $\delta_1 < d_1$, $\delta_2 < d_2$, ..., $\delta_c < d_c$; and the linear constraints representing the convex polytope X. The vertex $\mathbf{v}$, if exists, will be added to $\mathcal{V}$. For each member $X, Y \in \mathcal{K}$ such that $X \cap Y \in \mathcal{I}$, let $\mathbf{x}$ and $\mathbf{y}$ be vertices associated with X and Y, respectively. If a vertex $\mathbf{z}$ associated with $X \cap Y$ exists, the line segments $\overline{\mathbf{xz}}$ and $\overline{\mathbf{zy}}$ will be added to $\mathcal{E}$. Assuming that the number of dimension and dispersion constraints are small constants, an interior point method optimizer can solve the problem at a reasonable precision within $O(1)$, treating n as a constant. Hence, solving all of the convex optimization problems requires $O(v^n)$ time. Our implementation relies on *MOSEK optimization software* to perform this task.

4. Initialize maximal dispersion values (cost) of each vertex $\mathbf{v}$ in $\mathcal{V}$. If its associated convex subset is in the form of $W \times W \times ... \times W$ for some $W \in \mathcal{W}$, its maximal dispersion is $\delta(\mathbf{v})$. Otherwise, the maximal dispersion value of the vertex is initially assigned to $+\infty$.

5. Propagate maximal dispersion values over the graph $(\mathcal{V}, \mathcal{E})$ according to (4.3), to obtain all maximal dispersion of every vertex. We apply the shortest path algorithm to extract least cost unvisited vertex, propagate to its adjacent vertices, and merge the disjoint sets if connected. This step takes $O(v^n \log v)$ time.

6. Report all solution sets formed by the constraint. Each solution set consists of vertices in $\mathcal{V}$ representing union of convex subsets of Ω_c. The number of these sets are $O(v^n)$ and can be enumerated in $O(v^n)$ time.

The time complexity spent in propagating the maximal dispersion during step 5 dominates that of the others. Hence, the fundamental algorithm has $O(v^n \log v)$ time complexity.

To find which set of caging placements contains a given placement $\mathbf{x} = (x_1, x_2, ..., x_n)$, one has to identify first its containing a convex subset in the form of $X = W_1 \times W_2 \times ... \times W_n$, $W_i \in \mathcal{W}$. A point location algorithm (Snoeyink, 2004) may apply to identify each K_i containing x_i. If an associated vertex of X is not accessible (not visited during propagation), the object cannot be caged there because it cannot be reached by the fingers. Otherwise, the maximal dispersion of $\mathbf{x}$ is evaluated according to (4.1). If it is greater than $\delta(\mathbf{x})$, $\mathbf{x}$ is a caging placement in the solution set associated with X. In case that $\mathbf{x}$ is "not accessible" in $(\mathcal{V}, \mathcal{E})$, its maximal dispersion remains $+\infty$. A query takes $O(\log v)$ time in total.

In case that $b < c$, the algorithm will run in two passes. The first pass will create a graph $(\mathcal{V}_b, \mathcal{E}_b)$ for Ω_b with the dispersion function δ assigned to a constant. Optimization problems for vertices in step 3 become feasibility problems. The propagation in step 5 will mark accessible vertices. The second pass remains the same as the previous case. The solution sets are those reported in the second pass that contain some vertices accessible in the first pass. The query algorithm follows the first step to identify the containing convex subset X. The input to the query is a caging placement only if the vertex associated with X is accessible in $(\mathcal{V}_b, \mathcal{E}_b)$ but not in $(\mathcal{V}, \mathcal{E})$.

4.7 Results

The algorithm was implemented in C++ and tested with several 2D and 3D objects on Intel Core i5 CPU 650 at 3.20 GHz, using only one core in the execution. The execution time for the experiments are shown in Table 4.1. We use $\delta \equiv \delta^2_{\text{adj}}$

in all experiments. In the bottom row of Figure 4.2, the distance between finger 1 and 2 is also bounded below a value, having an additional always-active dispersion constraint. The objects are shown in gray. Black dots show representative finger placements of solution sets. The function δ restricted on their corresponding solution set attains its minimal value at each representative finger placement. A dashed loop with a numeric label represents a projection of a solution set on the workspace of a specified finger. For unconstrained cases, two solutions are identified as one if one solution can be obtained from permuting or reversing the finger labels of the other. The solutions shown are sorted by their *quality* in a descending manner. Quality of a solution is simply defined as the gap between the maximal dispersion and the dispersion of the representative placement.

Observe that some fingers at representative placements are possibly at the same point indicating that the object can be caged with fewer fingers. Though more solutions are reported in settings with more fingers, the reported solutions can be very small and the fingers need to be very close to the object – the fingers can easily collapse to a point and escape by rising only a small amount of dispersion, starting from the representative caging placement, see Figure 4.3, 4.4. A straightforward method to improve solutions' quality is to prevent the fingers from getting too close together, i.e., by imposing constraints in the form $\|\mathbf{x}_i - \mathbf{x}_j\| \geq d$, where d is a positive constant. Such constraints are concave – not dispersion constraints. We will continue the discussion on this issue later in Chapter VI.

Table 4.1: Algorithm Execution Time (Fundamental)

object	v	n	c	cvx op. (s)	total (s)
Figure 4.2 (top)	13	3	1	1.77	1.82
Figure 4.2 (bottom)	13	3	2	4.00	4.05
Figure 4.3	8	4	1	6.37	6.83
Figure 4.4	6	4	1	6.18	6.60
Figure 4.5	6	5	1	52.61	74.68

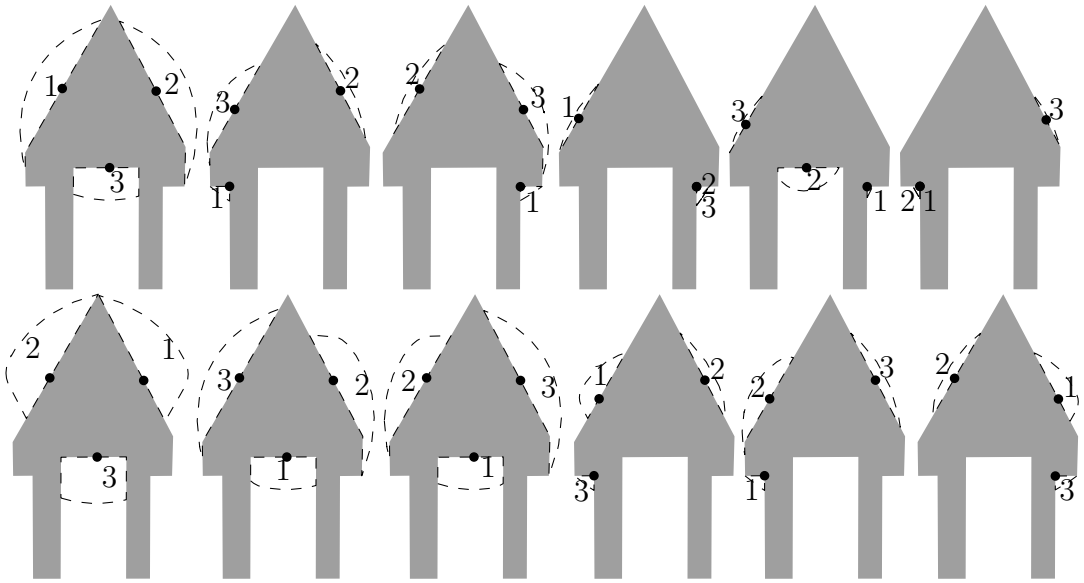

Figure 4.2: Top and bottom row: top six solutions without and with an upperbound dispersion constraint on the finger pair (1, 2), respectively.

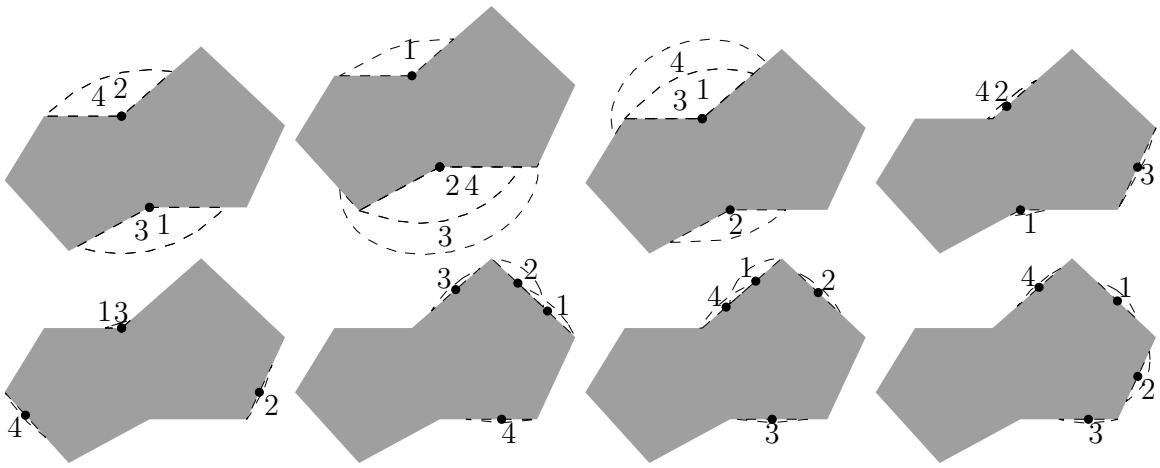

Figure 4.3: Eight best solutions for four fingers unconstrained case.

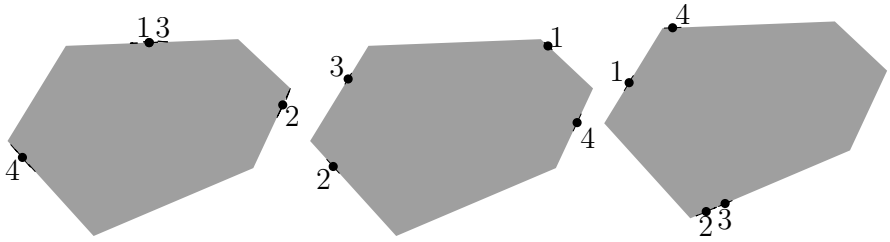

Figure 4.4: All three solution sets for a convex object with four fingers.

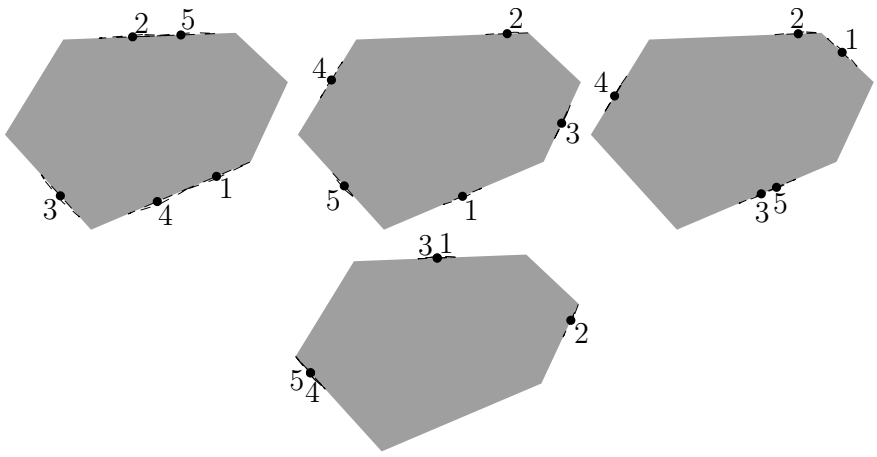

Figure 4.5: Top four solutions for caging a convex polyhedron with five fingers.

CHAPTER V

IMPERFECT OBJECT MEASUREMENT

5.1 Introduction

This chapter presents an extension to the fundamental algorithm to handle uncertainty of the object shape. The exact object shape $\mathcal{P}$ is not known. It is possibly deformable or time-varying but must be inbetween $\mathcal{P}^-$ and $\mathcal{P}^+$, that is: $\mathcal{P}^- \subseteq \mathcal{P} \subseteq \mathcal{P}^+$. The sets $\mathcal{P}^-$ and $\mathcal{P}^+$ serve as a lowerbound and an upperbound of $\mathcal{P}$, guaranteeing that the object occupies points in $\mathcal{P}^-$ and does not occupy points outside $\mathcal{P}^+$, see Figure 5.1(a). To ensure that a finger placement cages the object

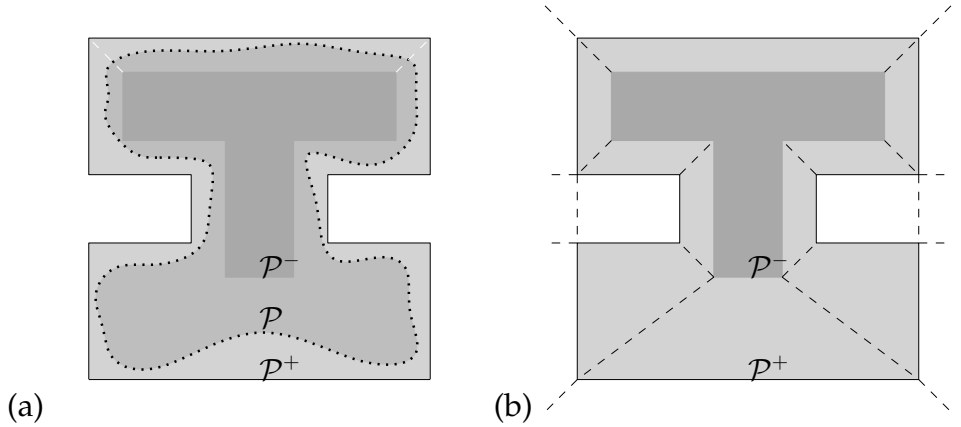

Figure 5.1: (a) the set $\mathcal{P}$ represents the unknown actual object inbetween given bounds $\mathcal{P}^-$ and $\mathcal{P}^+$. (b) decomposing the free workspace into convex polygons.

in the presence of uncertainty, all escape paths from the finger placement must be blocked by imposed constraints regardless of the object shape $\mathcal{P}$ inbetween $\mathcal{P}^-$ and $\mathcal{P}^+$. Consider two object shapes $\mathcal{A}$ and $\mathcal{A}'$ that $\mathcal{A}$ contains $\mathcal{A}'$. A finger trajectory in the free workspace of $\mathcal{A}$ is also a valid trajectory of $\mathcal{A}'$. The same holds for every escape path from a finger placement, i.e., any escape path valid for $\mathcal{A}$ is also valid for $\mathcal{A}'$. Let us consider a finger placement $\mathbf{x}$ valid for $\mathcal{A}$. Since $\mathcal{A}'$ contains $\mathcal{A}$, $\mathbf{x}$ is also valid for $\mathcal{A}'$. If $\mathbf{x}$ is a caging placement for $\mathcal{A}'$, $\mathbf{x}$ must be a caging placement for $\mathcal{A}$ as well. Therefore:

Proposition 14. *A caging placement* $\mathbf{x}$ *for* $\mathcal{P}^-$ *is also a caging placement for any* $\mathcal{P}$ *containing* $\mathcal{P}^-$ *if* $\mathbf{x}$ *is a valid finger placement for* $\mathcal{P}$*.*

A solution set for imperfect object should contain only caging placements that certainly cage the uncertain shape $\mathcal{P}$. Is it a solution set for an exact shape of $\mathcal{P}^-$? What about a solution set for an exact shape of $\mathcal{P}^-$ that contains some finger placements inside $\mathcal{P}^+$? It is possible that such a finger placement may not be a valid finger placement for $\mathcal{P}$ because some fingers may lie in the interior of the uncertain shape $\mathcal{P}$. Two finger placements in the same solution set for the exact shape of $\mathcal{P}^-$ may not be path-connected if every path connecting the two finger placements consists of some finger trajectories that penetrate into $\mathcal{P}^+$.

To cage in the presence of the object uncertainty, we identify and make use of finger placements that cages the object of any shape inbetween $\mathcal{P}^-$ and $\mathcal{P}^+$.

Proposition 15. *A finger placement* $\mathbf{x}$ *is a caging placement for any shape* $\mathcal{P}$ *inbetween* $\mathcal{P}^-$ *and* $\mathcal{P}^+$*, if and only if,* $\mathbf{x}$ *is a caging placement for* $\mathcal{P}^-$ *and is a valid finger placement for* $\mathcal{P}^+$*.*

Proof. ($\rightarrow$): When a finger placement $\mathbf{x}$ is a caging placement for any shapes inbetween, it must be a caging placement for $\mathcal{P}^-$. It must also be a valid finger placement for $\mathcal{P}^+$; otherwise, it is not a caging placement for $\mathcal{P}^+$.

($\leftarrow$): By Proposition 14, $\mathbf{x}$ is a caging placement for any $\mathcal{P}$ inbetween. ■

For a demonstration, we apply the concept to the multi-finger squeezing caging. A solution set here is defined as a maximally connected set of finger placements that cages any object inbetween $\mathcal{P}^-$ and $\mathcal{P}^+$ as long as the constraints are imposed.

5.2 Algorithm Modifications

A straightforward approach to obtain all solution sets is to compute all cages for $\mathcal{P}^-$ then intersect them with the free configuration space induced by $\mathcal{P}^+$:

$\Omega^+ \equiv (\mathbb{R}^w \backslash \mathcal{P}^+)^n$. Yet, it is possible to avoid expensive boolean operations. Instead of decomposing the free workspace $\mathcal{F}^- \equiv \mathbb{R}^w \backslash \mathcal{P}^-$ derived from $\mathcal{P}^-$ as in the fundamental algorithm, it is decomposed in such a way that each decomposed convex subset of $\mathcal{F}^-$ is either contained in $\mathcal{P}^+$ or not overlap with the interior of $\mathcal{P}^+$, see Figure 5.1 (b). These subsets induce $\mathcal{K}$ and $\mathcal{I}$, ensuring that either of the following conditions:

(i) some fingers are in the interior of P^+, or

(ii) all fingers are in P^+;

is satisfied for all finger placements inside any member of $\mathcal{K}$ and $\mathcal{I}$. Any member with all finger placements satisfying (i) is not a subset of Ω^+, i.e., some fingers penetrate into $\mathcal{P}^+$. Its corresponding disjoint set entry will not be merged into a solution set.

The query algorithm in this setting operates almost in the same manner as the previous one. The first step is to locate a convex subset X containing a finger placement $\mathbf{x}$, the argument of a query. If X is contained in Ω^+, the subsequent steps follow those of the previous exactly. In case that the finger placement lies in Ω^- but not in Ω^+, the query algorithm should report that it does not form a cage for all shapes inbetween $\mathcal{P}^-$ and $\mathcal{P}^+$ (formed by the constraints). In practice, it is possible that some fingers actually enter the interior of $\mathcal{P}^+$. A reactive planner may still want to know whether the *current* finger placement forms a cage for $\mathcal{P}^-$ (i.e., caging the actual object). In this case, the query algorithm is exactly the fundamental one, treating $\mathcal{P}^-$ as an input object.

5.3 Applications and Results

The following are possible applications and guidelines to exploit this extension.

- *Shape uncertainty and/or deformable object*. The bounds $\mathcal{P}^-$ and $\mathcal{P}^+$ should con-

tain the uncertain object shape. In case that the object and/or the fingers are not perfectly rigid, their shape may slightly deform to some extent during grasping or caging. The bounds $\mathcal{P}^-$ and $\mathcal{P}^+$ should cover all possible shapes of the object that may vary during the operation, see an example in Figure 5.2.

- *Partial observation*. Some parts of the object may be occluded or not visible to the sensor. We assign $\mathcal{P}^-$ to observed surfaces, $\mathcal{P}^+$ to cover unobserved regions and the observed surfaces, see Figure 5.3. This method has a limitation that the observed surfaces must belong to a single rigid (or almost rigid) body. The output solution sets will only occupy the observed regions, i.e., using only the observed sections in caging.

- *Shape simplification*. For a detailed and noisy input object, e.g., obtained from range sensor, the exact algorithm may generate a lot of small solution sets at the cost of high computation time. A more efficient alternative is to compute simple bounding meshes that are a subset and a superset of input for $\mathcal{P}^-$ and $\mathcal{P}^+$, respectively. The simplification algorithm must be capable of preventing the result mesh from penetrating into the input mesh (Gumhold et al., 2003). See Figure 5.4, for an example.

- *Spherical fingers*. For spherical fingers, they can be treated as points by growing the object. The grown object can no longer represented by polyhedra. The bounds $\mathcal{P}^-$ and $\mathcal{P}^+$ are sampled in such a way that they are a subset and a superset of the grown object, see Figure 5.5.

Table 5.1 shows execution time of the results.

Table 5.1: Algorithm Execution Time (Object Uncertainty)

object	v^-	v^+	n	cvx op. (s)	total (s)
Figure 5.2	13	13	3	1.46	1.51
Figure 5.3	24	23	3	5.34	5.62
Figure 5.4	13	12	3	2.16	2.23
Figure 5.5	23	23	3	16.35	18.17

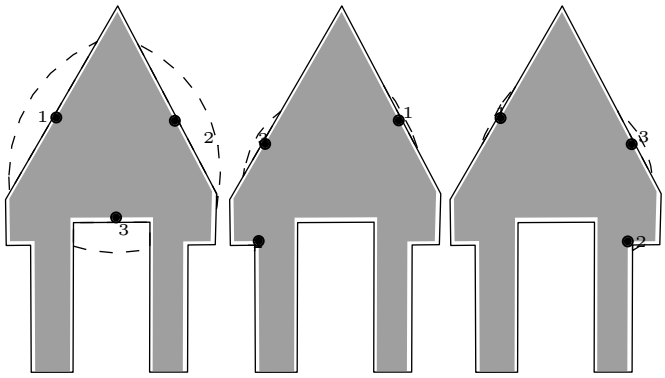

Figure 5.2: Solutions for caging under imperfect object measurement: uncertain/deformable object. A shaded region represents $\mathcal{P}^-$. A solid outline represents $\mathcal{P}^+$.

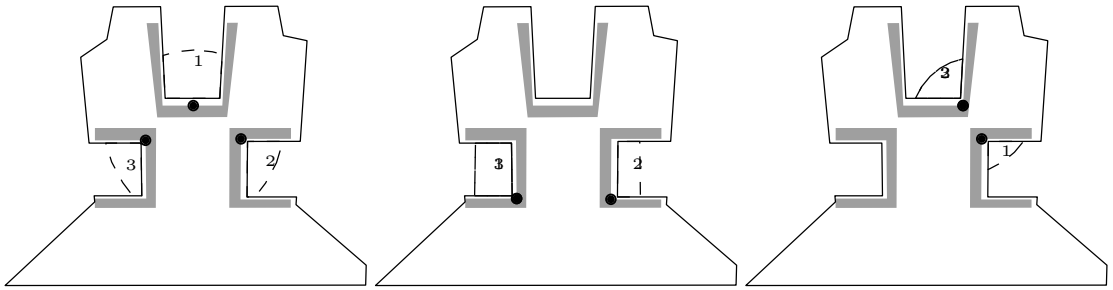

Figure 5.3: Solutions for caging under imperfect object measurement: partial observation. A shaded region represents $\mathcal{P}^-$, observed surface. A solid outline represents $\mathcal{P}^+$, unobserved region.

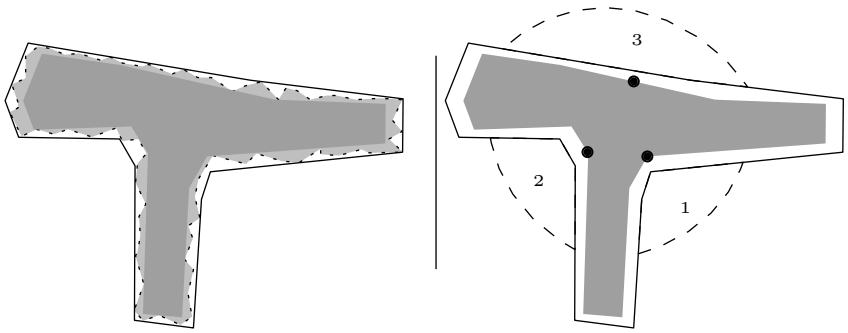

Figure 5.4: Solutions for caging under imperfect object measurement: simplification (dotted outline represents the object prior to the simplification).

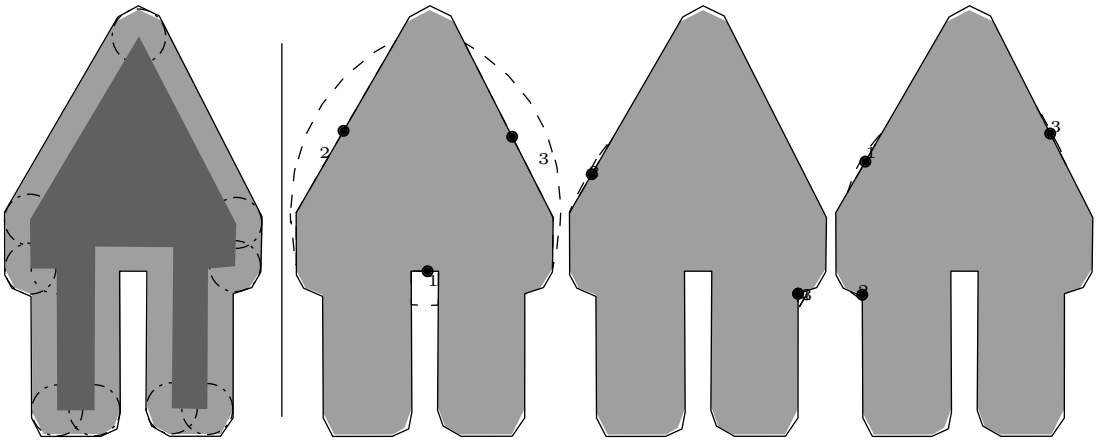

Figure 5.5: Solutions for caging under imperfect object measurement: spherical fingers (dark region represents the actual object).

CHAPTER VI

IMPERFECT FINGER CONTROL

6.1 Introduction

Occasionally, dispersion constraints alone are not sufficient to represent desirable traits of the manipulator critical to object caging. These traits are, for example, limited minimum distance or collision between pair of fingers, rotational/ball joint and joint limits possibly induced by the mechanics of the manipulator or the control policy. Proposition 10 shows that no combination of dispersion constraints can possibly simulate such behavior. This section focuses on approximating non-convex constraints to capture desirable traits of manipulator that dispersion constraints cannot. The concept applied to handle imperfect object shapes can be immediately upgraded to handle "imperfect c-obstacle", combining the effect of object measurement and finger control uncertainty. Instead of assuming exact c-obstacle, the c-obstacle is uncertain and/or time-varying but it must be inbetween an upperbound and a lowerbound.

6.2 Distance Constraint

Let us consider a simple constraint that prevents two arbitrary fingers x_1 and x_2 from getting too close to each other, given by: $\|\Delta_{1,2}\| \geq r_{\min}$ where $\Delta_{1,2} \equiv x_2 - x_1$ and $r_{\min}$ is a positive real constant. We construct a "lowerbound" $\mathcal{B}^-$ and an "upperbound" $\mathcal{B}^+$ for the constraint's infeasible region $\mathcal{B}$ ($\mathcal{B}^- \subseteq \mathcal{B} \subseteq \mathcal{B}^+$) in the coordinates of $\Delta_{1,2} \in \mathbb{R}^w$, see an example in Figure 6.1 (a). The exact version of the constraint is invariant to rigid transformations of the whole finger formation. Yet, the bounds are only "invariant to translation" but not "invariant to rotation" of the finger formation (consider a bound as a function that equals to 0 when in bound but ∞ otherwise). If we were to apply a bound as a constraint in an actual caging operation, the knowledge of relative rotation between the finger formation and the object is required. In practice, it is more convenient to impose the exact constraint while solution sets are derived from the approximated constraint. Similar to that

in the previous section, we avoid boolean operations by decomposing the compliment of $\mathcal{B}^-$ into convex polyhedra in such a way that each decomposed polyhedron that overlaps the interior of $\mathcal{B}^+$ must be contained in $\mathcal{B}^+$. Figure 6.1 (b) shows an example of such approximation for a bounding minimum constraint. The decomposed convex polyhedra are then transformed to the coordinates of the first and second fingers using the relationship $\Delta_{1,2} \equiv x_2 - x_1$, which is a linear transformation, preserving convexity (Boyd and Vandenberghe, 2004a). Let $B_{1,2}$ be the set of transformed convex polyhedra. The set of convex polytopes containing the feasible set of minimum distance constraint is given by: $\mathcal{B}_{1,2} \equiv \left\{ B \times (\mathbb{R}^w)^{n-2} \mid B \in B_{1,2} \right\}$.

The intersection among feasible sets induced by all constraints on fingers and the object forms the free configuration space. Let $\mathcal{K}$ and $\mathcal{K}'$ be sets of convex polytopes:

$$\left(\bigcup_{K \in \mathcal{K}} K \right) \cap \left(\bigcup_{K' \in \mathcal{K}'} K' \right) = \bigcup_{K \in \mathcal{K} * \mathcal{K}'} K$$

where: $K * K' \equiv \{ K \cap K' \mid K \in \mathcal{K}, K' \in \mathcal{K}' \}$. This $*$ operator will produce the intersection between unions of convex polytopes in the form of a union of convex polyhedra. Instead of directly decomposing the high-dimensional free configuration space into convex polyhedra, we decompose low-dimensional feasible sets first then apply the $*$ operator on them.

Under formation-translation invariant constraints, the whole finger formation can freely translate along an arbitrary direction without being obstructed by any constraints. This implies that once all the fingers and the object can be separated by a plane Π not intersecting the object, they can move arbitrarily far from the object by translating the whole finger formation. Let us denote the side of the plane Π containing the fixed object by Π^+, and the other side by Π^-.

Proposition 16. *For any escape path α starting from a feasible $\mathbf{x}$, there exists another path β from $\mathbf{x}$ to a point in $(\Pi^-)^n$ such that $\hat{\delta}(\beta) \leq \hat{\delta}(\alpha)$*

Proof. Let $\alpha = (a_1, a_2, ..., a_n)$ be an escape path from $\mathbf{x}$. It can be implied from

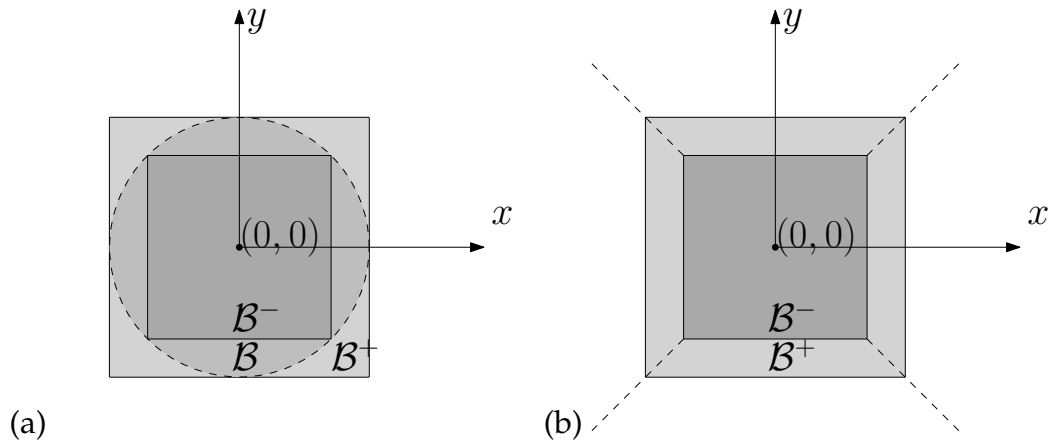

Figure 6.1: (a) a lowerbound $\mathcal{B}^-$ and an upperbound $\mathcal{B}^+$ for the infeasible region $\mathcal{B} \equiv \{(x, y) = \Delta_{1,2} \mid \|\Delta_{1,2}\| \geq r_{min}\}$. (b) decomposition into eight convex polyhedra (the center square is the void).

the definition of escape path that, for any $R \in \mathbb{R}$, there exists some t_0 that $\min(\|a_1\|, \|a_2\|, ..., \|a_n\|)(t) > R$ for any $t > t_0$. In other words, the fingers, after following the escape path for a period of time, must all be scattered outside the ball of radius R, which can be arbitrarily chosen. Since the object is bounded, at some sufficiently large radius R the finger formation can be translated without colliding the object arbitrarily far towards any direction. Hence, there exists some t_0 such that starting at $\alpha(t_0)$ it is possible to translate the finger formation to some placement in $(\Pi^-)^n$. ■

Vertices contained in $(\Pi^-)^n$ are exit vertices for the maximal dispersion propagation. At an exit vertex, the maximal dispersion value is equal to the dispersion value. To ensure that some vertices are in $(\Pi^-)^n$, the half plane Π^- should be in the subcover of $\mathcal{F}$ after the decomposition.

6.3 Plane Constraint

The fingers lying on a plane obey constraints in the form: $\langle n, x_i - p \rangle = 0$ where n and p is the normal and a point belongs to the plane. The constraint is linear when n does not depend on the positions of the fingers. However, this means that the relative rotation between the object and the finger formation other than that around the

normal axis is fixed, which is a rare case. Occasionally, the plane's normal depends on the positions of the fingers, e.g., the normal vector is derived from the cross product between finger position differences: $\mathrm{n} = (\mathrm{x}_2 - \mathrm{x}_0) \times (\mathrm{x}_1 - \mathrm{x}_0)$. In such case, the constraints in the form $\langle \mathrm{n}, \mathrm{x}_i - \mathrm{p} \rangle = 0$ are no longer linear and not convex. To simulate an articulated gripper, it is more convenience to use the following hinge constraints.

6.4 Hinge Constraint

For three dimensional workspace, a formulation of a hinge constraint imposed on a triplet of fingers' positions: $\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3$, and a "ghost" finger x_4. A hinge constraint consists of five distance constraints:

- $\|\Delta_{1,2}\| = r_a$,
- $\|\Delta_{2,3}\| = r_b$,
- $\|\Delta_{2,4}\| = h$,
- $\|\Delta_{1,4}\| = \sqrt{r_a^2 + h^2}$,
- $\|\Delta_{3,4}\| = \sqrt{r_b^2 + h^2}$,

see Figure 6.2 (a). The ghost finger's free workspace is the entire $\mathbb{R}^w$ as it does not actually collide with the object but, together with $\mathbf{x}_2$, they define the rotation axis of the hinge. The five distance constraints form two rigid triangles sharing a common edge $\overline{\mathrm{x}_2\mathrm{x}_4}$. Multiple hinge constraints can be applied by forming a mesh of distance constraints as in Figure 6.2 (b). The distance constraint $\|\Delta_{5,6}\| = r$ is added to fix the rotation between the two hinge axes.

Additional distance constraints can be added to the system to setup hinge joints' limit. For example, based on the example in Figure 6.2 (b), the following constraint

$$l_{min} \leq \|\mathrm{x}_1 - (c\mathrm{x}_3 - (1-c)\mathrm{x}_2)\| \leq l_{max} \tag{6.1}$$

with sufficiently large c and appropriate l_{min}, l_{max} can limit the angle $\widehat{x_1x_2x_3}$ within a subset of $[0, \pi]$.

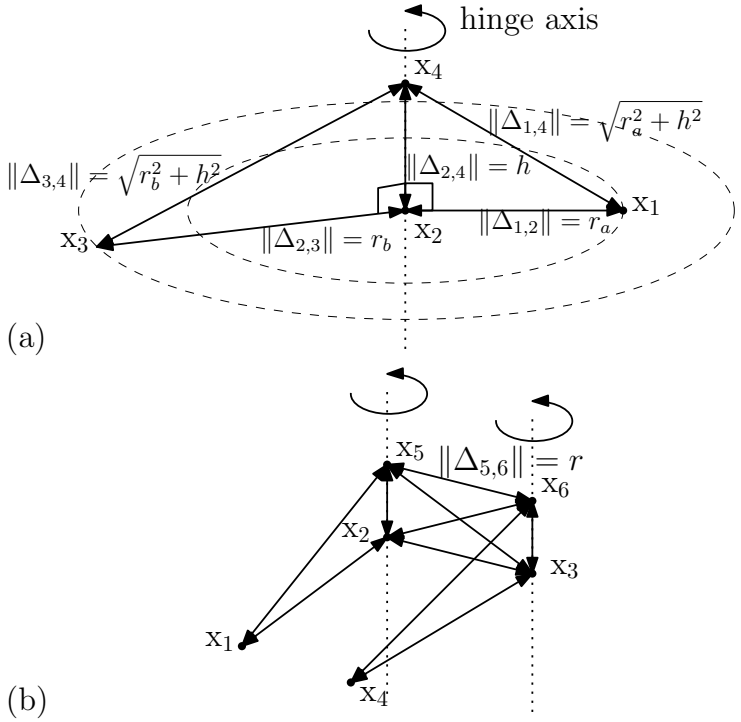

Figure 6.2: (a) a hinge constraint setup, (b) two hinge constraint setup. The solid lines with arrows on end points represent distance constraints and r_a, r_b, h, r are positive real constants.

6.5 Linear Constraints

A line segment or finger can be approximated with multiple fingers lying between two end points (fingers under a distance constraint). Let x_i and x_k be positions of two end points. The finger position x_j lying between x_i and x_k satisfies the linear constraint in the form: $x_j = (1 - c)x_i + cx_k$; for some real constant $c \in (0, 1)$. Such constraint is imposed along with the distance constraint between x_i and x_k is fixed. The more fingers inbetween, the more computation time; however, they are strictly constrained. We apply such fact to quickly reject some infeasible graph vertices from consideration. It should be noted that, for 2D workspace, an alternate approximation for line segment fingers can also be achieved by decomposing three dimensional (x, y, θ)-coordinates free configuration space of a single line segment

into (approximate) convex polyhedra; therefore, the cartesian among those convex polyhedra becomes members of $\mathcal{K}_0$ instead.

Apart from fixing the fingers along a line, applying distance constraints and appropriate linear constraints can fix the fingers in a rigid formation. For 2D workspace, consider a pair of fingers x_1 and x_2 that the distance between them is fixed, i.e., a rigid formation of two fingers. It is possible to attach another finger, say x_3, to the rigid formation by imposing constraints in the form: $x_3 = sR(x_2 - x_1) + x_1$ where s is a scalar and R is a rotation matrix, respectively.

6.6 Algorithm Modifications

We assume that all constraints are invariant to translation of the whole finger formation and at least a point in $(\Pi^-)^n$ is feasible. The system are under c constraints, a of which are non-convex constraints to be bounded. We assume that the set of appropriately decomposed convex subsets $\mathcal{B}_i$ for each i-th non-convex constraint is provided by user. The union of members in $\mathcal{B}_i$ is the lowerbound for the feasible set of the i-th constraint. Additionally, each member may not contain both the lowerbound and the upperbound's interior points.

The following are key steps for the extended algorithm when all c constraints are always active-constraints.

1. Same as step 1 of the fundamental algorithm, decompose the free workspace into convex polyhedra. The decomposition guarantees that Π^- is one of them.

2. Generate members of $\mathcal{K}_0$ and $\mathcal{I}_0$. Start from $X \equiv W_1 \times W_2 \times ... \times W_n = (\Pi^-)^n$, initially in $\mathcal{K}_0$. Enumerate neighboring convex polyhedra of X in the form $X' \equiv W_1' \times W_2' \times ... \times W_n'$ such that, for some i, $W_i \neq W_i'$ but $W_j = W_j'$ for any $j \neq i$. Ignore X' that is already in $\mathcal{K}_0$. Test if the intersection $I \equiv X \cap X'$ is feasible with respect to the constraints. If it is infeasible, reject I and continue the enumeration. Otherwise, put X' in $\mathcal{K}_0$, put I in $\mathcal{I}_0$, and have its neighboring convex polyhedra of X' enumerated recursively.

3. Construct the graph $(\mathcal{V}, \mathcal{E})$. Let $\mathcal{K}_1 \equiv \mathcal{K}_0 * \mathcal{B}_1 * \mathcal{B}_2 * ... * \mathcal{B}_a$ and $\mathcal{I}_1 \equiv \mathcal{I}_0 * \mathcal{B}_1 * \mathcal{B}_2 * ... * \mathcal{B}_a$. For each member X in $\mathcal{K}_1 \cup \mathcal{I}_1$, solve the convex optimization problem: find an associated vertex $\mathbf{v}$ in X, i.e., $\mathbf{v}$ is where δ attains its minimal value subject to the convex constraints. The vertex $\mathbf{v}$ is added to $\mathcal{V}$ if exists. For each member $X, Y \in \mathcal{K}_1$ such that $X \cap Y \in \mathcal{I}_1$ and the vertex associated to $X \cap Y$ exists, the edges $\{X, X \cap Y\}$ and $\{X \cap Y, Y\}$ are added to $\mathcal{E}$.

4. The maximal dispersion of the initial vertex is set to the dispersion of the initial vertex. Other vertices' maximal dispersions are initialized to $+\infty$.

5. Follow the step 5 of the fundamental algorithm in Chapter 4.6 to obtain all solution sets with respect to the constraints.

6. Trim the solution sets by discarding each disjoint set entry with the associated convex polyhedron contained in an upperbound.

The algorithm execution time now significantly depends on the constraints. Stricter constraints increase the chance of pruning infeasible convex polyhedra early, thus resulting in smaller $\mathcal{K}_0$, and $\mathcal{I}_0$. If all constraints are convex, we have $\mathcal{K}_1 = \mathcal{K}_0$ and $\mathcal{I}_1 = \mathcal{I}_0$; therefore, step 3 of the extended algorithm reduces to the step 3 of the fundamental one. Assuming that the number of convex polyhedra used in approximating those constraints be bounded by N. Cardinality of $\mathcal{K}_1$ and that of $\mathcal{I}_1$ are $O(|\mathcal{K}_0|N^a)$ and $O(|\mathcal{I}_0|N^a)$, respectively. The number of convex optimization subproblems to be solved are $O((|\mathcal{K}_0| + |\mathcal{I}_0|)N^a)$. Only feasible solutions to these subproblems will be used in generating nodes and edges of the graph. In the worst case situation that all nodes are feasible, the algorithm complexity becomes $O(aN^a n v^n \log(vN))$, which reduces to $O(v^n \log v)$ if n, N and a are treated as small constants.

One may query which solution set contains a given configuration by identifying a containing convex subset in $\mathcal{K}_1$. Like the fundamental query algorithm, we identify a convex subset of free workspace that contains each finger. In addition, for each i-th approximated constraint, we identify a convex subset in $\mathcal{B}_i$ containing the

given configuration. Though it is possible to apply an available point location algorithm to achieve a logarithmic query time in a higher dimensional space, the precomputation phase is complicated and time consuming (Snoeyink, 2004). Applying a naive approach to point location yields $O(\log v + aN)$ query. Occasionally, we can solve the point location problem in constant time, resulting in $O(\log v)$ query, because the convex subsets are regularly generated from known function.

6.7 Applications and Results

Consider a setting consists three fingers that $\delta \equiv \|x_1 - x_3\|^2 + \|x_2 - x_3\|^2$. Distance constraints are imposed between x_1 and x_2, the base fingers to restrict their distance in a small interval. The maximum distance constraint $\|x_1 - x_2\|^2 \leq r_{\max}$ is a dispersion constraint. We use unions of twelve polyhedra to define bounds for the minimum distance constraint $\|x_1 - x_2\|^2 \geq r_{\min}$. The experiment results on polygonal objects are shown in Figure 6.3. The key difference between our algorithm in this setting and the exact algorithm presented in (Vahedi and van der Stappen, 2008) is that: our algorithm takes $O(v^3 \log v)$ to approximate all cages parametrized by the maximal dispersion and the constraints. This is in contrast to the exact algorithm that takes $O(v^6 \log^2 v)$ to generate an exact solution represented by curves given fixed base fingers' positions, say p_1, p_2 as additional input.

In Figure 6.4, the dispersion function δ is δ^2_{adj}. We impose lowerbound distance constraints between pairs of fingers to prevent the fingers from collapsing to a point. This prevents degenerate solutions and make the fingers more scattered around to block the object.

Motivated by the fact that distance constraints depend only on positions of only two fingers and independent of the rest, we precompute whether a convex polyhedron $W_1 \times W_2$ satisfies the constraint for all possible W_1, W_2 being decomposed convex subsets of the free workspace. The precomputed results are queried in step 3 to quickly reject infeasible convex optimization subproblems. Table 6.1 shows the execution time of the extended algorithm using quick rejection. The nonconvex constraints significantly slow down the computation.

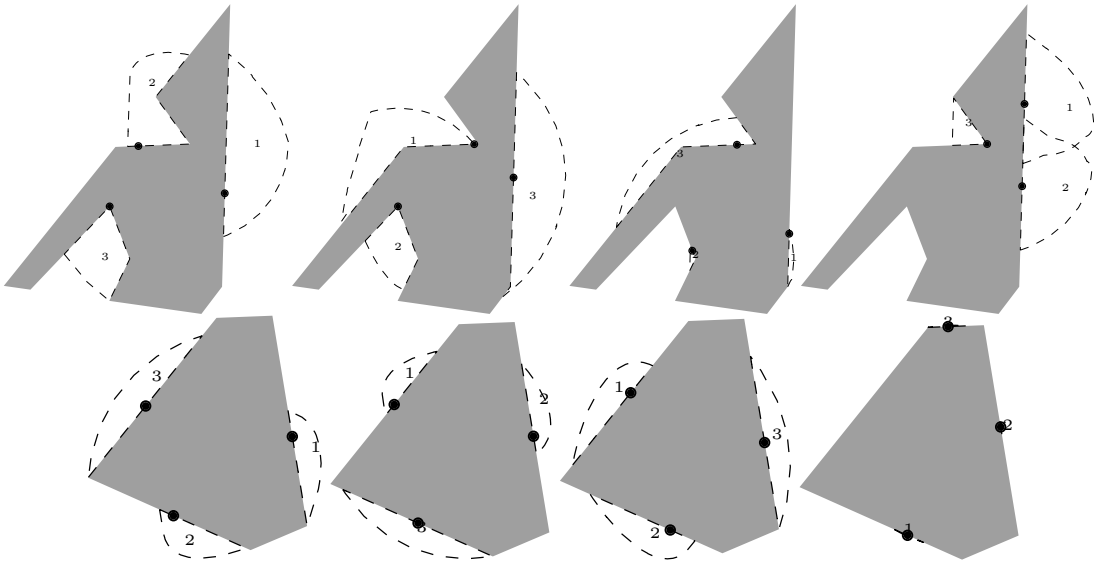

Figure 6.3: Top solutions for three finger caging via dispersion control, the distance between "base fingers" is constrained in a small interval. The dashed region shows projections of the solution sets.

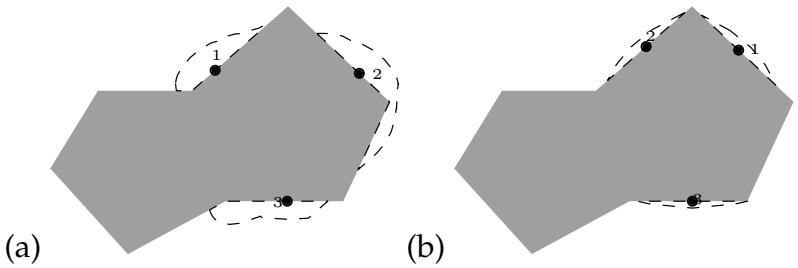

Figure 6.4: (a) a solution set for three fingers with three lowerbound distance constraints imposed on pairs of adjacent fingers, (b) similar solution set but the fingers are unconstrained.

Table 6.1: Algorithm Execution Time (Non-convex Constraint Approximation)

object	v	n	c	a	cvx op. (s)	total (s)
Figure 6.3 (top)	11	3	2	1	10.73	11.09
Figure 6.3 (bottom)	6	3	2	1	7.50	7.75
Figure 6.4 (a)	8	3	3	3	1720.27	9078.39
Figure 6.4 (b)	8	3	0	0	0.65	0.67

6.8 Summary

We present a $O(v^n \log v)$ algorithm for reporting all cages for a given object represented by a w-polyhedron, a set of n point fingers, and a set of dispersion constraints. The algorithm is extended to support object uncertainty and report approximate solutions for non-convex constraints. Information of a given finger placement, whether it is a in a cage and how much its maximal dispersion value is, can be queried with a $O(\log v)$ algorithm. Experiments show that our squeezing caging approach works well (reporting large solution sets) when the object has at least a concave section. The solution sets may be very small otherwise. Appropriate constraints can help improve solutions' quality. Yet, choosing a set of appropriate constraints to cage an object remains a challenging problem.

CHAPTER VII

COMBINED SQUEEZING AND STRETCHING CAGING

7.1 Introduction

Caging by fixing a finger formation does not allow the finger formation to deform. In constrast to caging by dispersion control, maintaining dispersion to be the same value or in an interval may permit the finger formation to change its shape. Consider Fig. 7.1, fixing the finger formation immobilizes the object. However, controlling the sum of square distance between adjacent fingers (a dispersion) below any value cannot cage the object since it allows the finger 2 and 4 to slide to where the finger 1 is and let the object escape without increasing dispersion. Conversely, if the object can be caged by maintaining dispersion after an initial finger formation setup, it can also be caged by fixing the initial finger formation. Though each solution set reported by this approach is a subset of a caging set, its description remains as simple as that of two-finger caging, i.e., a representative formation and a critical value. An important characteristic of caging via dispersion control is that the fingers can always shrink to a single point without increasing dispersion, as shown in Proposition 10. This implies that it is not possible to fix the finger formation by keeping dispersion below or above a value regardless of the choice of a dispersion function.

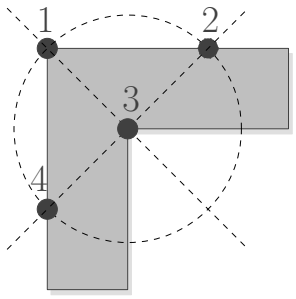

Figure 7.1: Four-finger caging by fixing the finger formation.

This chapter proposes an exact algorithm to identify approaches to cage an object. Instead of caging by keeping dispersion either below or above a value, the object is to be caged by controlling dispersion in an interval, using both the upperbound and the lowerbound. The algorithm is extended from that of squeezing and that of stretching caging to report solution sets for caging in an interval $[a, b)$, given a. The process of generating solutions for a polygonal or polyhedral object with v vertices and n fingers has $O(v^n \log n)$ time complexity. Each reported solution set remains in the form of a representative formation and an error tolerance. Unlike caging sets, the solution sets depend on the choice of a dispersion function which represents an "approach" to cage the object. The reported error tolerance is measured with respect to the chosen approach, the dispersion function. The higher the reported tolerance, the more the finger formation may deform without letting the object escape. Apart from identifying possible solutions to caging in the aforesaid manner, the algorithm may be applied in computing error tolerance for caging by fixing a finger formation on a plane. The finger formation possibly obtained from a grasp synthesis, current manipulator configuration or other caging method. We show that caging by fixing a given finger formation on a plane can be achieved by controlling an appropriate dispersion function in an interval. Therefore, the algorithm can be configured to report solutions to fixed formation caging – every distinct site to cage the object with a given fixed formation along with corresponding error tolerance.

This chapter is organized as follows. The next section is an overview for caging via dispersion control in an interval, aimed to setup a dispersion function capable of fixing a finger formation. Section 7.3 presents an approach to determine the upperbound value to control dispersion value below, given a lowerbound dispersion. The algorithms for computing all solution sets are in Section 7.4 followed with conclusion in Section 7.5.

7.2 Caging by Controling Dispersion in an Interval

Fixing a finger formation forbids the finger formation to deform, preventing the fingers from squeezing and stretching. Caging by fixing a finger formation is necessarily both squeezing and stretching caging. In two-finger setting, fixing a finger formation is equivalent to fixing the separation distance. For a setting with three or more fingers, an appropriate substitute for the separation distance, a dispersion function δ, has to be chosen to fix the finger formation a similar manner. The chosen function has to be invariant rigid transformation of the finger formation so that the fingers can perform caging without the knowledge of the object's configuration. Consider a three finger system with δ assigned to be $\delta^2_{\text{adj}} \equiv d^2_{1,2} + d^2_{2,3} + d^2_{3,1}$ where $d_{i,j}$ is the distance between the finger i and the finger j. Suppose that the fingers are in general positions ($d_{1,2} > 0, d_{2,3} > 0$ and $d_{3,1} > 0$), it can be observed that imposing the constraint $\delta^2_{\text{adj}} = d$ for some $d > 0$ does not fix the finger formation. To ensure that δ varies whenever the finger formation deforms from a given formation, δ has to depend on it.

7.2.1 Caging by Fixing a Finger Formation

Assuming a system of n fingers on a plane and $\mathbf{p} \equiv (\mathrm{p}_1, \mathrm{p}_2, ..., \mathrm{p}_n) \in \mathbb{R}^{2n}$ be the position of fingers representing a formation shape to fix. The measure on how much a finger formation deform or deviate from the given formation shape $\mathbf{p}$ is based on the difference:

$$\begin{aligned} \Delta_{\mathbf{p}} &= \mathbf{x} - \mathrm{A}_{\mathbf{p}}(\mathrm{r}, \mathrm{t}), \\ \mathrm{A}_{\mathbf{p}} &= \begin{pmatrix} \mathrm{p}_1 & \mathrm{p}_1^{\perp} & I_{2\times 2} \\ \mathrm{p}_2 & \mathrm{p}_2^{\perp} & I_{2\times 2} \\ \vdots & \vdots & \vdots \\ \mathrm{p}_n & \mathrm{p}_n^{\perp} & I_{2\times 2} \end{pmatrix} \end{aligned}$$

where:

- $\mathbf{x} = (\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_n) \in \mathbb{R}^{2n}$ is the current positions of the fingers,

- $A_{\mathbf{p}} \in \mathbb{R}^{2n\times 4}$ depends on the given formation $\mathbf{p}$ such that $A_{\mathbf{p}}(r,t)$ is the given formation similarity transformed with (r,t) as parameter,
- $r \in \mathbb{R}^2$ is an orientation-scale vector such that direction and magnitude of r correspond to rotation and scale of the transformation, respectively.
- $t \in \mathbb{R}^2$ is the translation part of the transformation,

Assuming that $n \geq 2$ and the formation to fix is not a single point, we assign $(r,t) = A_{\mathbf{p}}^{\dagger}\mathbf{x}$ so as to minimize $\Delta_{\mathbf{p}}^{\top}\Delta_{\mathbf{p}}$ which measures the formation shape variance between $\mathbf{x}$ and $\mathbf{p}$ "upto scale". It should be noted that the formation shape of $\mathbf{x}$ is the same as that of $\mathbf{p}$, if and only if, the scale of the transformation is one ($r^{\top}r = 1$), and the formation shape variance is zero ($\Delta_{\mathbf{p}}^{\top}\Delta_{\mathbf{p}} = 0$). To control both the scale and the variance, we exploit the following function:

$$\begin{aligned} \delta_{\mathbf{p}}^2 &\equiv f + g \\ f &\equiv \lambda(r^{\top}r - 1) \\ g &\equiv (1-\lambda)\Delta_{\mathbf{p}}^{\top}\Delta_{\mathbf{p}}, \end{aligned}$$

where λ is a constant in $(0,1)$. The function $\delta_{\mathbf{p}}^2$ is a dispersion function that maps a formation $\mathbf{x}$ to a real value because both f and g are. In addition, we claim that the finger formation cannot deform its shape from that of $\mathbf{p}$ without varying $\delta_{\mathbf{p}}^2$. Assume without loss of generality that $\mathbf{x} = \mathbf{p}$. Varying the formation from $\mathbf{p}$ while $\delta_{\mathbf{p}}^2 = 0$ either:

1. *Varies scale*: for any direction v such that the directional derivative of f at $\mathbf{p}$ along the direction v: $\nabla f(\mathbf{p}) \cdot v \neq 0$. The constraint $\delta_{\mathbf{p}}^2 = 0$ implies that $\nabla g(\mathbf{p}) \cdot v \neq 0$ as well. However, $\Delta_{\mathbf{p}}$ vanishes at $\mathbf{p}$ so $\nabla g(\mathbf{p}) \cdot v = 2\Delta_{\mathbf{p}}^{\top}\nabla\Delta_{\mathbf{p}}(\mathbf{p}) \cdot v = 0$, a contradiction. This case is not possible.
2. *Fixes scale*: $\nabla f(\mathbf{p}) \cdot v = 0$. The constraint $\delta_{\mathbf{p}}^2 = 0$ forces $\nabla g(\mathbf{p}) \cdot v = 0$. After the variation f remains the same so $\|r\| = 1$, i.e., no change in formation scale. Moreover, $\Delta_{\mathbf{p}}$ must remain vanished since $g = 0$ after the variation and g is a

norm of $\Delta_{\mathbf{p}}$. This implies that the formation shape must remain the same after the variation.

Hence, caging by fixing $\delta^2_{\mathbf{p}} = 0$ is exactly caging by fixing the formation shape of $\mathbf{p}$.

Controlling $\delta^2_{\mathbf{p}}$ in an interval $[0, e)$ for some $e > 0$ permits the finger formation to deform from the shape of $\mathbf{p}$. The parameter λ can be interpreted as a blending factor between two extreme caging behaviors: one is that of assigning f, and the other is assigning g, as a dispersion function. Setting λ close to zero will bias towards caging by controlling the "scale" with respect to the formation shape $\mathbf{p}$. On the other hand, setting λ close to one will bias towards caging by controlling the variance from $\mathbf{p}$ "upto scale". These two behaviors compliment each other, and are combined to control both the scale and the variance from $\mathbf{p}$.

7.2.2 Caging in a Bounded Workspace

To simplify the subsequent discussions in this chapter, we assume that:

(FB) any finger cannot move outside a sufficiently large box $\mathcal{B}$ bounding the object.

This assumption prevents the fingers from going arbitrarily far from the object, i.e., the usual caging condition is always satisfied so we use the condition that: the object is caged if all the fingers are prevented from going outside a sufficiently large box $\mathcal{B}'$, contained in $\mathcal{B}$, bounding the object. Recall that the condition is equivalent to the usual caging condition when (FB) is not assumed. When the fingers are all outside $\mathcal{B}'$, the finger formation can translate (the formation shape remains the same) arbitrarily far from the object. The size of $\mathcal{B}'$ must be large enough so that even if the fingers surround $\mathcal{B}'$, there exists a gap for the object to slip through. The size of $\mathcal{B}'$ depends on the number of fingers n and the size of the object, Conversely, if the fingers can move arbitrarily far from the object, they can move outside any box bounding the object. The assumptions allow us to work in a bounded domain, bounded free workspace and configuration space so as to avoid complications that

occur as a finger approaches infinity. In practice, it is also reasonable to assume (FB) as fingers that are assigned to cage the object should be controlled to be in vincinity.

7.3 Combined Squeezing and Stretching Caging

Like the presented two-finger squeezing or stretching caging algorithm, the one for combined squeezing and stretching caging is also based on convex decomposition of the configuration space and identifying "optimal path" among paths in each decomposed convex set. The free workspace of the system of interest is $\mathbb{R}^w$ subtracted by the polytope representing the object with the coordinates of reference set to that of the object. We assume that the free workspace is connected, ignoring inaccessible holes inside the object. Here, a path is a function that maps a value in $[0, 1]$ to a configuration in the free configuration space denoted by Ω. An optimal path in a set of paths $\mathcal{A}$ is defined as a path in $\mathcal{A}$ such that the dispersion of every point along the path is equal or greater than the lowerbound and the maximum value of dispersion among points along the path is the least among any other paths in $\mathcal{A}$. Without loss of generality, we assume the lowerbound dispersion to be the constant 0. In other words, α is an optimal path in $\mathcal{A}$, if and only if,

$$\hat{\delta}(\alpha) = \sup_{\mathbf{x}\in\text{img}(\alpha)} \delta(\mathbf{x}) = \inf_{\beta\in\mathcal{A}} \sup_{\mathbf{y}\in\text{img}(\beta)} \delta(\mathbf{y}),$$

subject to:

$$\inf_{\mathbf{x}\in\text{img}(\alpha)} \delta(\mathbf{x}) \;\geq\; 0. \tag{7.1}$$

Given that α is optimal in $\mathcal{A}$, the optimal dispersion of $\mathcal{A}$ is defined by $\hat{\delta}(\alpha)$. If an optimal path α in $\Gamma(\{\mathbf{x}\}, \{\mathbf{y}\}, \mathcal{A})$ does not exist because a path in $\Gamma(\{\mathbf{x}\}, \{\mathbf{y}\}, \mathcal{A})$ that satisfies the lowerbound constraint (7.1) does not exist, the fingers cannot reach $\mathbf{y}$ from $\mathbf{x}$ (and cannot reach $\mathbf{x}$ from $\mathbf{y}$) if the formation dispersion is controlled to be equal to or above 0. Similarly, an existence of an optimal path α in $\Gamma(\{\mathbf{x}\}, \{\mathbf{y}\}, \Omega)$ is a certificate that the fingers cannot reach $\mathbf{y}$ from $\mathbf{x}$ (and vice versa) if the formation dispersion is controlled in $[0, \sup_{\mathbf{x}\in\text{img}(\alpha)} \mathbf{x})$. To identify whether a finger place-

ment $\mathbf{x}$ forms a cage via dispersion control, we determine the optimal dispersion in $\Gamma(\{\mathbf{x}\}, \{\mathbf{e}\}, \Omega)$ where $\mathbf{e}$ is a remote configuration such that all fingers are outside $\mathcal{B}'$ and $\delta(\mathbf{e}) = 0$. For $\mathbf{e}$ to exist, $\mathcal{B}$ must be large enough to contain a smallest finger formation with zero dispersion outside $\mathcal{B}'$.

First, let us consider $\Gamma(\{\mathbf{x}\}, \{\mathbf{y}\}, K)$ where $K \subseteq \Omega$ is a bounded convex polytope, $\delta(\mathbf{x}) \geq 0$, and $\delta(\mathbf{y}) \geq 0$. If the straight line segment from $\mathbf{x}$ to $\mathbf{y}$ does not contain any formation with dispersion less than zero, the optimal path in $\Gamma(\{\mathbf{x}\}, \{\mathbf{y}\}, K)$ is the straight line segment with $\max(\delta(\mathbf{x}), \delta(\mathbf{y}))$ being the optimal dispersion of $\Gamma(\{\mathbf{x}\}, \{\mathbf{y}\}, K)$. This is because every path in $\Gamma(\{\mathbf{x}\}, \{\mathbf{y}\}, K)$ must visit $\mathbf{x}$ then $\mathbf{y}$, and a dispersion is a convex function. Every formation along inbetween the straight line segment has dispersion not greater than the end points, by Jensen's inequality (Boyd and Vandenberghe, 2004a). In case that the line segment contains some formation with dispersion less than zero, we claim that:

Proposition 17. *An optimal path α with optimal dispersion of $\Gamma(\{\mathbf{x}\}, \{\mathbf{y}\}, K)$ equal to $\hat{\delta}(\alpha) = \max(\delta(\mathbf{x}), \delta(\mathbf{y}))$ exists if there exists a path β from $\mathbf{x}$ to $\mathbf{y}$ that satisfies $\inf_{\mathbf{z} \in \text{img}(\beta)} \delta(\mathbf{z}) \geq 0$.*

The optimal path α can be constructed by first pulling each point on the path β towards a point $\mathbf{o} \in K$ that $\delta(\mathbf{o}) \leq 0$ until it hits the 0-level set of δ, see Figure 7.2. Since K is convex, such pull will not be blocked by the boundary of K. The pulled path is a path with maximum dispersion along the path lower or equal to β; therefore, is lower or equal to $\delta(\mathbf{x}, \mathbf{y})$. Connect the beginning and the end point of the pulled path using straight line segments to obtain the optimal path α with optimal dispersion equal to $\max(0, \delta(\mathbf{x}), \delta(\mathbf{y})) = \max(\delta(\mathbf{x}), \delta(\mathbf{y}))$. In case that such β does not exist, any path from $\mathbf{x}$ must pass to some point with dispersion lower than zero before reaching $\mathbf{y}$; therefore, the optimal path in $\Gamma(\{\mathbf{x}\}, \{\mathbf{y}\}, K)$ does not exist as well.

The existence of the path β in the previous paragraph can be verified using a strategy presented in stretching caging. Given that K is a bounded convex polytope and $\mathbf{x}$, $\mathbf{y}$ are at some vertices of K, an optimal path is simply a sequence of

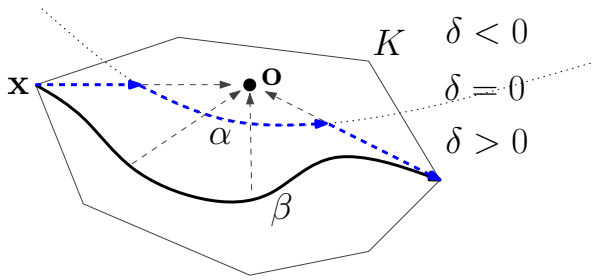

Figure 7.2: Path pulling example, the solid-line path β is pulled to the middle section of the dashed-line path. Connect the end points of the pulled path with straight lines to obtain the optimal path α, the entire dashed-line path.

edges in K (see Chapter 3.5). The underlying idea is that for any path α in K, it is possible to project each point $\mathbf{x}$ on the path along a straight ray enamating from some point $\mathbf{s}$ towards $\mathbf{x}$ until it hits the relative boundary of K, see Figure 7.3. Let $d = \inf_{\mathbf{z}\in\mathrm{img}(\alpha)} \delta(\mathbf{z})$. If the intersection between the interior of d-sublevel set of δ and the interior of K is not empty, the point $\mathbf{s}$ can be any point in the interior of K that $\delta(\mathbf{s}) < d$. Otherwise, any $\mathbf{s}$ in the interior of K not contained in the path is fine. Infimum over dispersion of points along the projected path is guaranteed to be equal to or higher to d. This projection operation is repeated on paths that lie on each facet of K, sending them to lower dimensional facets of K until the entired path lies on edges of K. When $\mathbf{x}$ and $\mathbf{y}$ are not at some vertices of K, they can be recursively projected to ones in the aforesaid manner, the trajectories during recrusive projection are used in connecting $\mathbf{x}$ to a vertex, and a vertex to $\mathbf{y}$. In fact, the relation that two vertices are connected by some path β with $\inf_{\mathbf{z}\in\mathrm{img}(\beta)} \delta(\mathbf{z}) \geq 0$ partitions vertices in K into connected components of vertices. Consequently, such β connecting $\mathbf{x}$ and $\mathbf{y}$ exists, if and only if, $\mathbf{x}$ and $\mathbf{y}$ are recursively projected to vertices belong to the same connected component of vertices in K. It should be noted that connected components of vertices in A are one-to-one mapped to connected components (of points) in A partitioned by the region with the dispersion lower than zero. An example is illustrated in Figure 7.4. In the example, the pairs of corresponding connected component of points : connected component of vertices are $A : \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\}$, $B : \{\mathbf{b}_1\}$, $C : \{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3\}$.

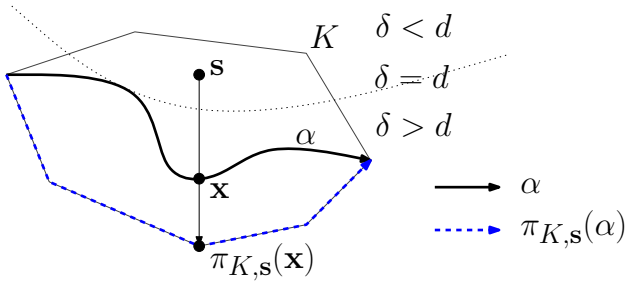

Figure 7.3: Projecting a path α (solid line) to a path on the relative boundary of K (dashed line).

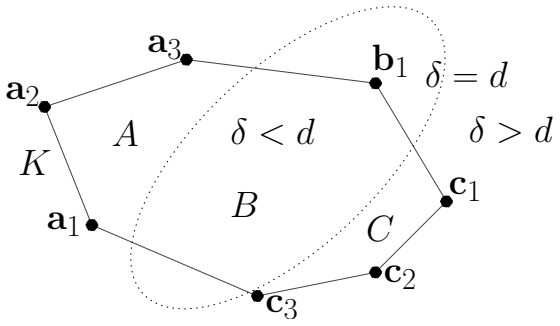

Figure 7.4: Points in a convex polyhedron are partitioned into multiply connected components of points. The vertices of the convex polyhedra are paritioned into connected components of vertices.

Recursively projecting a point with dipersion more than zero in A to a vertex of A maps a connected component of points containing the point to a connected component of vertices while each vertex of A can be mapped back to its containing component of points. For brevity, the partitioned connected component of points will be referred as "connected component".

From the discussions so far, it can be concluded that:

Proposition 18. *Let X be a connected component of a bounded convex polytope K. The optimal dispersion of $\Gamma(\{\mathbf{x} \in X\}, \{\mathbf{y} \in X\}, X)$ is $\max(\delta(\mathbf{x}), \delta(\mathbf{y}))$ and is equal to that of $\Gamma(\{\mathbf{x}\}, \{\mathbf{y}\}, K)$.*

Next, let us consider $\Gamma(\{\mathbf{x}\}, \{\mathbf{x}'\}, K \cup K')$ where:

- $K, K' \subseteq \Omega$ are bounded convex polytopes that share common vertices and $K \cap K'$ is exactly the convex hull of those vertices.
- X is a connected component of K and contains $\mathbf{x}$.
- X' is a connected component of K' and contains $\mathbf{x}'$.

Any path from $\mathbf{x}$ to $\mathbf{x}'$ has to visit some point in $K \cap K'$ so the optimal path exists, if and only if, $X \cap X' \neq \varnothing$. We claim that:

Proposition 19. *$X \cap X' \neq \varnothing$, if and only if, X and X' shares a common vertex.*

Proof. $(\leftarrow)$ is obvious. For $(\rightarrow)$, for any point $\mathbf{z}$ on $X \cap X'$, it can be recursively projected to a vertex of $K \cap K'$ with greater or equal dispersion; therefore, X and X' must share a common vertex when their intersection is not empty. ■

Suppose that an optimal path in $\Gamma(\{\mathbf{x}\}, \{\mathbf{x}'\}, X \cup X')$ exists, it must consist of two parts: one from $\mathbf{x}$ to a point in $X \cap X'$ and the other from the point in $X \cap X'$ to $\mathbf{x}'$. Let $\mathbf{z}$ be a point that δ restricted on $X \cap X'$ attains its minimal value. The optimal path in $\Gamma(\{\mathbf{x}\}, \{\mathbf{x}'\}, X \cup X')$ is simply a concatenation between the optimal path in $\Gamma(\{\mathbf{x}\}, \{\mathbf{z}\}, X)$ and that in $\Gamma(\{\mathbf{z}\}, \{\mathbf{x}'\}, X')$. Both can be constructed by the discussed method. If $K \cap K'$ intersects the 0-level set of δ an empty set, then $X \cap X' = K \cap K'$. The optimal dispersion in this case is $\max(\max(\delta(\mathbf{x}), \delta(\mathbf{z})), \max(\delta(\mathbf{z}), \delta(\mathbf{x}'))) = \max(\delta(\mathbf{x}), \delta(\mathbf{x}'), \delta(\mathbf{z}))$. Otherwise, it is $\max(\delta(\mathbf{x}), \delta(\mathbf{x}'))$ because the minimal value of δ restricted on $X \cap X'$ is always zero.

Proposition 20. *Given that $X \cap X' \neq \varnothing$, the optimal dispersion of $\Gamma(\{\mathbf{x}\}, \{\mathbf{x}'\}, X \cap X')$ is $\max(\delta(\mathbf{x}), \delta(\mathbf{x}'), \delta(\mathbf{z}))$ and is equal to that of $\Gamma(\{\mathbf{x}\}, \{\mathbf{x}'\}, K \cap K')$.*

Suppose that the free configuration space Ω is a union of bounded convex polytopes $K_1, K_2, ..., K_c$ such that the intersection of every convex polytope pair sharing common vertices is exactly the convex hull of the common vertices. In this setting, connected components in convex polytopes serve as basic building blocks for the combinatorial search structure. A connected component X in a convex polytope

K is said to be adjacent to another connected component X' in convex polytope B if there exists a common connected component in $A \cap B$ that overlaps with both X and X'. The connected components in convex polytopes and their adjacency form a graph structure: each graph vertex is a connected component in a convex polytope K_i. The graph edge connecting the two graph vertices is present when the intersection between the graph vertices is nonempty. However, the graph edges defined this way are more than necessary. To improve the algorithm, the edge should present only when the intersection of the convex polytopes containing the graph vertices is their common facet with dimension less than that of the configuration space by one, ignoring lower dimensional gaps. Given a graph path, i.e., a sequence of adjacent graph vertices: $X_1, X_2, ..., X_m$; one could immediately construct an optimal path travelling through the vertices using the stated method and compute its optimal dispersion. The goal of computing the caging tolerance of an initial configuration $\mathbf{x}$, $\delta(\mathbf{x}) \geq 0$, or the optimal dispersion of $\Gamma(\{\mathbf{x}\}, \{\mathbf{e}\}, \Omega)$, can be achieved by finding an "optimal graph path" that starts at a graph vertex containing $\mathbf{x}$ and terminates at a graph vertex containing $\mathbf{e}$. Here, the optimal graph path is a graph path that induces the least optimal dispersion among all other graph paths linking the same end points, $\mathbf{x}$ and $\mathbf{e}$.

7.4 Algorithms

A single-source shortest path algorithm is modified to propagate optimal dispersion among graph vertices and a disjoint set data structure is used in keeping track of solution sets, connected sets of caging formations. The algorithm to report solution sets of combined squeezing and stretching caging is as follows:

1. *Decompose the free workspace* into convex polytopes $\mathcal{W} = \{W_1, W_2, ...\}$ such that intersection of every convex polytope pair sharing common vertices is exactly the convex hull of the common vertices. Each decomposed convex polytope subset of Ω is created by cartesian product among n members of $\mathcal{W}$. We denote $\mathcal{K}$ by the set of all such convex polytopes covering Ω. This step can be accomplished by polygon/polyhedra triangulation in $O(v^2)$ or better.

2. *Identify connected components*. Create a graph for stretching caging: each graph vertex, and each graph edge, is a vertex, and an edge, of a convex polytope in $\mathcal{K}$. Compute the minimal value of δ restricted on each edge as the edge's cost, using a convex optimizer. Identify connected components of the stretching caging graph by considering its edges whose cost is lower than 0 as removed. Note that each connected component may span across convex polytopes in $\mathcal{K}$, but we will be working only on the connected component containing $\mathbf{e}$, referred to as E. The other connected components cannot be reached while the constraint $\delta \geq 0$ is imposed. Solution to each optimization problem can be solved in constant time (treating n as a constant) independent of v because of triangulation. The time complexity of this step is $O(v^n)$ required for solving $O(v^n)$ convex optimization problems.

3. *Create the graph for combined squeezing and stretching caging*. Each graph vertex is the intersection of the connected component and a convex polytope in $\mathcal{K}$. Let $K, K' \in \mathcal{K}$ be convex polytopes that intersect E a nonempty set. The graph edge links the vertices $E \cap K$ and $E \cap K'$ if K and K' share a common facet. Formally, a graph edge connects vertices X, X' in the form:

$$\begin{aligned} X &= E \cap (W_{\iota(1)} \times W_{\iota(2)} \times ... \times W_{\iota(n)}), \\ X' &= E \cap (W_{\iota'(1)} \times W_{\iota'(2)} \times ... \times W_{\iota'(n)}), \end{aligned}$$

   such that $\iota(i) = \iota'(i)$ for any i except when i is equal to some index j that $W_{\iota(j)} \cap W_{\iota'(j)}$ is a common facet between $W_{\iota(j)}$ and $W_{\iota'(j)}$. At initialization, every graph vertex is associated with an element of disjoint set and has its "temporary optimal dispersion" (to be updated during propagation) set to ∞.

4. *Propagate optimal dispersion*. The propagation starts at a graph vertex containing $\mathbf{e}$ whose dispersion is trivially known to be equal to 0. Let X be such graph vertex, and δ^*_X be its optimal dispersion. For each adjacent vertex X' of X, performs 5) and 6).

5. Compute $\delta_{X \cap X'}$, the minimum value of δ restricted on $X \cap X'$. Such minimum value can be computed using a convex optimizer. If the minimum value is

below zero, it is clipped to zero, by Proposition 20.

6. Update the temporary optimal dispersion of X' with $\max(\delta_{X\cap X'}, \delta^*_X)$ if $\max(\delta_{X\cap X'}, \delta^*_X)$ is lower. Merge associated set elements of X and X' if $\delta^*_X \leq \delta_{X\cap X'}$.

7. Next, the algorithm picks another graph vertex X with the least temporary optimal dispersion value to promote to the optimal dispersion of $\Gamma(\{\mathbf{x}\}, \{\mathbf{e}\}, \Omega)$ where $\mathbf{x}$ is a point at which δ restricted on X attains its minimal value. If such X exists, performs 5) and 6) for each X' adjacent to X. Otherwise, the algorithm terminates, the solution sets can be enumerated from the disjoint set data structure. Some feassible finger formation may not be visited due to the hard constraint $\delta \geq 0$. If the fingers are considered as free fingers and the hard constraint is enforced by a control policy, the constraint can be ignored before the initial formation setup at such a formation. After the setup is complete, the object can be caged by controlling their dispersion greater or equal to 0. Those caging placements can be obtained by running the algorithm for reporting all stretching cages.

The time complexity of the algorithm is dominated by steps 4) through 7) which requires $O(v^n \log v)$ for $O(v^n)$ operations of a heap – updating and extracting least temporary optimal dispersion value.

Like squeezing and stretching caging solutions, it is possible query what solution set a finger formation $\mathbf{x}$ that $\delta(\mathbf{x}) \geq 0$ is in by applying the point location algorithm (Snoeyink, 2004) to identify containing W_i of each finger. Let K, which is a member of $\mathcal{K}$, be the cartesian product among such W_i. The formation $\mathbf{x}$ is then projected to a vertex of $\mathcal{K}$ to check if it is in the same connected component that contains $\mathbf{e}$. If it does not, apply the stretching caging's query algorithm for $\mathbf{x}$ instead. Otherwise, identify X, the graph vertex of combined caging, using the projected vertex and K. Of course, $\mathbf{x}$ is a caging formation and in a solution set overlapping X only if $\delta(\mathbf{x}) < \delta^*_X$.

The presented algorithm is for reporting all solution sets to caging by controlling a dispersion function in $[0, \delta^*)$, not an arbitrary interval. An approach to obtain solutions of caging by controlling dispersion in an arbitrary interval is to update topology of the graph and optimal dispersions of all graph vertices as the lowerbound dispersion (previously the constant 0) varies from ∞ to $-\infty$. Initially, the lowerbound dispersion is ∞, the constraint is too strict, inducing an empty graph of combined caging. The next lowerbound dispersion is chosen from the greatest edge cost belong to the graph of stretching. The graph of combined caging is then updated. As the constraint becomes more relaxed, connected components emerge. The less the lowerbound dispersion, the larger subset of free configuration space becomes accessible. As the lowerbound drop to $-\infty$, the combined caging becomes squeezing caging. Solution sets are to be reported as snapshot of the graph everytime it is updated. The time complexity of this naive approach is $O(v^{2n} \log v)$. For better performance, a dynamic connected component and a dynamic shortest path algorithms should be applied.

7.5 Summary

This chapter presents an algorithm that reports all possible approach to caging an object by controlling dispersion above a fixed lowerbound and a critical upperbound, determined by the algorithm. The solution reported by the algorithm depends on the given dispersion function which determines caging behavior and its error tolerance. Solutions of squeezing caging that forbid the finger formation to collapse to a point can be obtained from the algorithm by setting the dispersion function to δ_{p}^2 and an appropriate lower bound dispersion. Setting $\delta_{\mathbf{p}}^2$ as the dispersion function, the algorithm will report all distinct approaches to caging by fixing the finger formation on a plane to $\mathbf{p}$.

CHAPTER VIII

ROBUST CAGING

8.1 Introduction

This chapter proposes an approximate algorithm to report all solutions to caging by "attempting" to fix a given formation of n point fingers. Each solution is described by a representative finger placement along with a corresponding error tolerance based on a deformation measure. The deformation measure of a finger placement determines how much its formation deforms from the given one. A solution set represents a maximally connected set of finger placements that cages the object by maintaining the deformation less than the error tolerance. The algorithm produces solutions in $O(CM^2 \log(CM^2))$ where C and M determine the complexity of the configuration space and the quality of approximation, respectively. Maintaining the fingers inside a solution set, a cage, is equivalent to keeping the deformation measure below the error tolerance. The object is guaranteed to be caged as long as the fingers are inside a solution set. This approach to caging simplifies the complication in modeling and applying caging sets for three or more fingers. The alogrithm possibly applies:

- To locate where to cage an object with a given finger formation. The finger formation maybe a set of contact points from a manipulator at a grasping pose.
- To validate, improve or convert existing caging placements obtained by other algorithms.
- To determine a pregrasping cage, or a quality, for a grasp. The finger formation to fix is set to a finger placement that grasps the object.

Attempting to fix the finger formation is not possible by controlling any dispersion function below or above a value. This is a consequence from the fact that the fingers can always collapse to a single point without increasing dispersion, see

Proposition 10. The capability of caging by attempting to fix a finger formation is very close to the limit of traditional object caging. If a caging set of an object has an interior point, caging by attempting to fix a finger formation (to that interior point's formation) is possible.

One of the differences between this chapter and the previous one is that this chapter presents an approximate algorithm to support a wider range of constraints and deformation measures. It is designed to measure error tolerance of a caging placement under an intuitive measure, defined in the following section.

The chapter is organized as follows. The next two sections are an overview of the problem, formulation (Section 8.2), and simplification (Section 8.3). Basic notations, definitions, and their properties are also listed here. Section 8.4 concerns properties of functions related to caging. The properties leads to Section 8.5: construction of a roadmap that helps in computing the cages and their error tolerance. The algorithm for computing all solution sets, and the results are presented in Section 8.6, followed by the conclusion in Section 8.7.

8.2 Problem Formulation

We assume a system that consists of a bounded rigid object $\mathcal{P}$ without inaccessible holes and n point fingers on a plane. Each finger's position is a point in the coordinates of the object's frame of reference. Let the finger formation to fix be: $\mathbf{o} \equiv (o_1, o_2, ..., o_n) \in \mathbb{R}^{2n}$. We define the measure of how much a finger formation $\mathbf{x} \equiv (x_1, x_2, ..., x_n) \in \mathbb{R}^{2n}$ deforms from $\mathbf{o}$ based on the difference between $\mathbf{x}$ and a finger placement rigidly transformed form $\mathbf{o}$:

$$\begin{aligned} \Delta(\mathbf{x}, r, t) &\equiv (\Delta_1, \Delta_2, ..., \Delta_n)(\mathbf{x}, r, t) \\ \Delta_i(\mathbf{x}, r, t) &\equiv \begin{pmatrix} o_i & o_i^{\perp} \end{pmatrix} r + t - x_i \end{aligned} \tag{8.1}$$

where:

- $r \in \mathbb{R}^2, r^\top r = 1$, is a unit vector representing the rotation part of the rigid

transformation.

- $\mathrm{t} \in \mathbb{R}^2$ is a vector representing the translation part of the rigid transformation.
- $\mathrm{o}_i^{\perp}$ is obtained by counter-clockwisely rotating o_i by 90 degrees.

The deformation measure is $\delta_p : \mathbb{R}^{2n} \to \mathbb{R}$,

$$\delta_p(\mathbf{x}) \equiv \inf_{\mathrm{r}^\top \mathrm{r}=1, t \in \mathbb{R}^2} \|\mathrm{s}(\mathbf{x}, \mathrm{r}, \mathrm{t})\|_p,$$

where $\|\cdot\|_p$ is the L^p-norm, and $\mathrm{s} : \mathbb{R}^{2n} \times \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^n$,

$$\mathrm{s}(\mathbf{x}, \mathrm{r}, \mathrm{t}) \equiv (\|\Delta_1\|_2, \|\Delta_2\|_2, ..., \|\Delta_n\|_2)\,(\mathbf{x}, \mathrm{r}, \mathrm{t}),$$

is a vector of displacements between each finger and its position in the given formation shape. The choice of p depends on the desirable characteristics of the error tolerance. For example,

- δ_1 measures the sum of displacements.
- $(\delta_2)^2$ measures the sum of squared displacements.
- δ_∞ measures the maximum displacement of a finger.

By maintaining the fingers to satisfy $\delta_p < \epsilon$, for some small ϵ, we effectively limit their deformation from the given formation shape $\mathbf{o}$. A function δ_p is clearly invariant to rigid transformation of the finger formation, i.e., $\delta_p(\mathbf{x}) = \delta_p((\mathbf{R}\mathrm{x}_1 + \mathrm{t}, \mathbf{R}\mathrm{x}_2 + \mathrm{t}, ..., \mathbf{R}\mathrm{x}_n + \mathrm{t}))$ for some $\mathbf{R} \in \mathbb{SO}(1)$ and $\mathrm{t} \in \mathbb{R}^2$. However, δ_p is not necessarily convex. If δ_p were convex, it would be a dispersion function, and we could not attempt to fix the finger formation by maintaining $\delta_p < \epsilon$ since the finger can always collapse to a point without increasing the dispersion.

When the error tolerance ϵ gets sufficiently large, the fingers are free to move around the object and collapse to a single point. Imposing $\delta_p < \epsilon$ using too

large ϵ cannot cage the object. Let the object and the finger formation be contained in balls of diameter d_o and d_f, respectively. When the error tolerance is $\epsilon^* = \max(d_o, d_f)\|(1, 1, ..., 1)\|_p + e$ (for any $e > 0$), each finger are free to displace more than the maximum diameter $\max(d_o, d_f)$ so they cannot cage the object. The maximum diameter that the finger formation can expand under this error tolerance bound is below $d_f + 2\epsilon^*$. It is safe to bound the workspace by a square $\mathcal{B}$ that is at least $d_o + d_f + 2\epsilon^*$ wide and centered at the object. If any of the finger goes outside this bound, the finger formation either deforms beyond ϵ^* or is separated from the object by a plane. In either case, the fingers can no longer cage the object. Hence, the workspace and the configuration space are given by $\mathcal{F} \equiv \mathcal{B} \backslash \mathcal{P}$, and $\Omega \equiv \mathcal{F}^n \subseteq \mathcal{B}^n$, respectively.

A finger placement $\mathbf{x}$ is in a $\delta_p < \epsilon$ cage if the supremum value of δ_p restricted on (an image of) every *unbounded path* from $\mathbf{x}$ is not less than ϵ. If $\mathbf{x}$ is in such a cage, the finger formation must change in such a way that its deformation measure δ_p goes above or equal to ϵ at some point *regardless of a chosen escape route*. A path is an n-vector of syncronized finger trajectories that does not penetrate into $\mathcal{P}$, i.e., (an image of) a valid path is contained in Ω. An unbounded path is equivalently an escape path. This condition for a finger placement to lie in a cage can be rewritten in such a way that no unbounded paths involved. Let Π be a half plane that intersects $\mathcal{P}$ an empty set. The fingers at a finger placement in Π^n can translate while preserving their formation shape arbitrarily far from the object. Conversely, if the fingers can move arbitrarily far from the object, they can translate while preserving their formation shape to a finger placement in Π^n. This path-transformation process does not increase the supremum value of δ_p restricted on the path. Hence, we have the following equivalent caging condition. A finger placement $\mathbf{x}$ forms a $\delta_p < \epsilon$ cage if the supremum value of δ_p restricted on (an image of) every path that starts from $\mathbf{x}$ and terminates in $\mathcal{T} \equiv (\mathcal{B} \cap \Pi)^n$ is not less than ϵ.

To determine the error tolerance of a given finger placement $\mathbf{x}$ is to find the largest possible ϵ that prevents the fingers from escaping from $\mathbf{x}$ if the constraint $\delta_p < \epsilon$ is imposed. That value of ϵ is referred to as the error tolerance, or the critical

value, at $\mathbf{x}$, given by:

$$\delta_p^*(\mathbf{x}) \equiv \inf_{\alpha \in \Gamma(\{\mathbf{x}\}, \mathcal{T}, \Omega)} \sup_{\mathbf{z} \in \mathrm{img}(\alpha)} \delta_p(\mathbf{z}).$$

If $\delta_p(\mathbf{x}) < \delta_p^*(\mathbf{x})$, then $\mathbf{x}$ is said to form a $\delta_p < \epsilon$ cage and the maximum possible ϵ is $\delta_p^*(\mathbf{x})$. On the other hand, $\mathbf{x}$ does not form a $\delta_p < \epsilon$ cage if $\delta_p(\mathbf{x}) \geq \delta_p^*(\mathbf{x})$.

A solution set is defined as a set of maximally connected finger placements that forms a $\delta_p < \epsilon$. In order to identify each solution set, we must find a representative finger placement inside the solution set and its error tolerance. In the implementation, an error tolerance at a representative finger placement $\mathbf{x}$: $\delta_p^*(\mathbf{x})$ will be obtained from a δ_p-*optimal escape path* from $\mathbf{x}$. A path α^* in $\Gamma(S, T, W)$ is said to be δ-*optimal* (among paths in $\Gamma(S, T, W)$) when:

$$\sup_{\mathbf{z} \in \mathrm{img}(\alpha^*)} \delta(\mathbf{z}) = \inf_{\alpha \in \Gamma(S,T,W)} \sup_{\mathbf{z} \in \mathrm{img}(\alpha)} \delta(\mathbf{z}).$$

The value $\sup_{\mathbf{z} \in \mathrm{img}(\alpha^*)} \delta(\mathbf{z})$ of a δ-optimal path α^* is called δ-*optimal value*. A δ_p-optimal escape path α^* from $\mathbf{x}$ is a path that is δ_p-optimal among paths in $\Gamma(\{\mathbf{x}\}, \mathcal{T}, \Omega)$ so $\delta_p^*(\mathbf{x}) = \sup_{\mathbf{z} \in \mathrm{img}(\alpha^*)} \delta_p(\mathbf{z})$. Any δ_p-optimal escape path from any $\mathbf{x}$ is contained in $\delta_p^{-1}(-\infty, \delta_p^*(\mathbf{x})]$, the $\delta_p^*(\mathbf{x})$-sublevel set of δ_p, but not contained in $\delta_p^{-1}(-\infty, \delta_p^*(\mathbf{x}))$.

8.3 Problem Simplification

To simplify the problem, we will deal with bounds of δ_p instead of δ_p itself. An upperbound (and a lowerbound) of δ_p must be greater (resp. less) or equal to δ_p pointwise. Consider $\bar{\delta}_p$, and $\underline{\delta}_p : \mathbb{R}^{2n} \to \mathbb{R}$ defined as follows.

$$\begin{aligned}
\bar{\delta}_p(\mathbf{x}) &\equiv \min\left\{\bar{\delta}_p^1(\mathbf{x}), \bar{\delta}_p^2(\mathbf{x}), ..., \bar{\delta}_p^M(\mathbf{x})\right\}, \\
\underline{\delta}_p(\mathbf{x}) &\equiv \min\left\{\underline{\delta}_p^1(\mathbf{x}), \underline{\delta}_p^2(\mathbf{x}), ..., \underline{\delta}_p^m(\mathbf{x})\right\}, \\
\bar{\delta}_p^j(\mathbf{x}) &\equiv \inf_{\mathrm{t} \in \mathbb{R}^2} \|\mathrm{s}(\mathbf{x}, \mathrm{r}_j, \mathrm{t})\|_p, \\
\underline{\delta}_p^j(\mathbf{x}) &\equiv \inf_{\mathrm{r} \in R_j, \mathrm{t} \in \mathbb{R}^2} \|\mathrm{s}(\mathbf{x}, \mathrm{r}, \mathrm{t})\|_p,
\end{aligned}$$

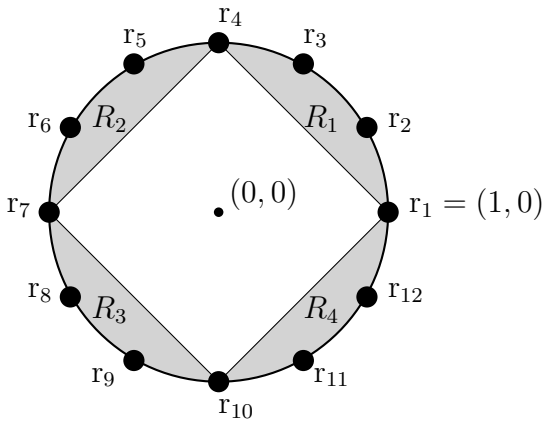

Figure 8.1: An example of r_j (big dots on the unit circle) and R_j (shaded convex regions) with $M = 12$ and $m = 4$.

where:

- r_j is a unit vector.
- R_j is a compact convex subset of $\mathbb{R}^2$ such that the union of every R_j, $j \in \{1, 2, ..., m\}$ contains $\left\{\mathrm{r} \in \mathbb{R}^2 \mid \mathrm{r}^\top \mathrm{r} = 1\right\}$.

Accuracy of error tolerances computed based on $\underline{\delta}_p$ depends on m, the number of convex subsets comprising the union. With more convex subsets, we obtain "better approximation" for a unit circle centered at the origin, i.e., the union of every R_j contain fewer points outside the unit circle. The condition that the union must contain the circle, as well as, the condition that each r_j must be on the circle are required so that $\underline{\delta}_p \leq \delta_p \leq \bar{\delta}_p$ pointwise. The error tolerance lowerbound defined based on $\underline{\delta}_p$ is:

$$\underline{\delta}_p^*(\mathbf{x}) \equiv \inf_{\alpha \in \Gamma(\mathbf{x}, \mathcal{T}; \Omega)} \sup_{\mathbf{z} \in \mathrm{img}(\alpha)} \underline{\delta}_p(\mathbf{z}).$$

Similarly, $\underline{\delta}_p^* \leq \delta_p^*$ pointwise. Figure 8.1 illustrates valid approximations of the unit circle for $\bar{\delta}_p$ and $\underline{\delta}_p$.

To cage an object, one must first move the fingers to a finger placement $\mathbf{x}$ that forms a $\delta_p(\mathbf{x}) < \epsilon$ cage, i.e., $\delta_p(\mathbf{x}) < \underline{\delta}_p^*(\mathbf{x})$. It is possible to simplify computation of δ_p by using the sufficient condition: $\bar{\delta}_p(\mathbf{x}) < \underline{\delta}_p^*(\mathbf{x})$; instead. Larger M is needed for better $\bar{\delta}_p$, closer to δ_p. When M is small, only finger placements in a large cage may satisfy the sufficient condition.

Our simplified goal is to obtain maximally connected sets of finger placements satisfying the sufficient condition.

Proposition 21. *A finger placement* $\mathbf{x}$ *and* $\mathbf{y}$ *are in the same solution set if*

1) $\bar{\delta}_p(\mathbf{x}) < \underline{\delta}_p^*(\mathbf{x})$, *and*

2) *there exists a path* α *that travels from* $\mathbf{y}$ *to* $\mathbf{x}$ *and is contained in* $\bar{\delta}_p^{-1}(-\infty, \underline{\delta}_p^*(\mathbf{x}))$.

Proof. By 1), $\delta_p(\mathbf{x}) \leq \bar{\delta}_p(\mathbf{x}) < \underline{\delta}_p^*(\mathbf{x}) \leq \delta_p^*(\mathbf{x})$; therefore, $\mathbf{x}$ is in a $\delta_p < \delta_p^*(\mathbf{x})$ cage.

Since $\underline{\delta}_p^* \leq \delta_p^*$ and $\delta_p \leq \bar{\delta}_p$, $\bar{\delta}^{-1}(-\infty, \underline{\delta}_p^*(\mathbf{x})) \subseteq \delta_p^{-1}(-\infty, \delta_p^*(\mathbf{x}))$. By 2), the path α is also contained in $\delta_p^{-1}(-\infty, \delta_p^*(\mathbf{x}))$. Let β be an escape path that starts from $\mathbf{x}$ and is contained in $\delta_p^*(\mathbf{x})$-sublevel set of δ_p. The concatenation of α and β is an escape path that starts from $\mathbf{y}$ and is contained in $\delta_p^*(\mathbf{x})$-sublevel set of δ_p. This implies that $\delta_p^*(\mathbf{y}) \leq \delta_p^*(\mathbf{x})$. Suppose for a contradiction that $\delta_p^*(\mathbf{y}) < \delta_p^*(\mathbf{x})$, i.e., there exists an escape path γ that starts from $\mathbf{y}$ and lies in $\delta_p^*(\mathbf{y})$-sublevel set of δ_p. The concatenation of reversed α and γ is an escape path that starts from $\mathbf{x}$ and lies in $\delta_p^*(\mathbf{y})$-sublevel set of δ_p. This results in a contradiciton that: $\delta_p^*(\mathbf{x}) \leq \delta_p^*(\mathbf{y}) < \delta_p^*(\mathbf{x})$. ■

Functions such as $\bar{\delta}_p$ and $\underline{\delta}_p$, $p \in \{1, 2, \infty\}$, are minimization among convex functions. They possess several excellent properties that help in identifying all possible cages to be shown in the following section.

Proposition 22. $\bar{\delta}_1^j, \bar{\delta}_2^j, \bar{\delta}_\infty^j, \underline{\delta}_1^j, \underline{\delta}_2^j$ *and* $\underline{\delta}_\infty^j$ *are convex.*

Proof. The readers are referred to (Boyd and Vandenberghe, 2004a) for theorems on convexity applied in this proof. The function in the form $\|\Delta_j\|_2$ is convex over the variables $\mathbf{x} \in \mathbb{R}^{2n}$, $\mathrm{r} \in \mathbb{R}^2$, and $\mathrm{t} \in \mathbb{R}^2$. The same holds for the following functions:

- $\|s\|_1 = \|\Delta_1\|_2 + \|\Delta_2\|_2 + ... + \|\Delta_n\|_2$,
- $\|s\|_2 = (\|\Delta_1\|_2^2 + \|\Delta_2\|_2^2 + ... + \|\Delta_n\|_2^2)^{1/2} = \|\Delta\|_2$,
- $\|s\|_\infty = \max(\|\Delta_1\|_2, \|\Delta_2\|_2, ..., \|\Delta_n\|_2)$.

The reasons are: $\|s\|_1$ is a sum of convex functions, $\|s\|_2$ is a 2-norm of linear functions, and $\|s\|_\infty$ is a maximization among convex functions. The functions $\bar{\delta}_1^j$, $\bar{\delta}_2^j$, $\bar{\delta}_\infty^j$ are convex because the infimum of $\underline{\delta}_p^j$ over $t \in \mathbb{R}^2$ preserves convexity. Since R_j is convex, the infimum of $\underline{\delta}_p^j$ over $\mathrm{r} \in R_j, \mathrm{t} \in \mathbb{R}^2$ also preserves convexity. Hence, $\underline{\delta}_1^j$, $\underline{\delta}_2^j$ and $\underline{\delta}_\infty^j$ are convex. ■

8.4 Minimization Among Convex Functions

In this section, we present properties of a function in the form of $\delta \equiv \min(f_1, f_2, ..., f_m)$ where f_i is a real-valued convex function for any $i \in \{1, 2, ..., m\}$. The goal is to construct an f-optimal path from any two points in the domain of f. The domain of f and each f_i must be a compact convex set denoted by D. Each f_i must have a strict global minimal point. Yet, it still works even if each f_i is a convex function with a pit of minimal points. By slightly "pushing" the pit, we obtain a close approximate that satisfies the assumption. The less the pit is pushed, the more accurate the approximation will be.

We define a subcover of D: $\mathcal{D} \equiv \{D_1, D_2, ..., D_m\}$ where each

$$D_i \equiv \{\mathbf{x} \in D \mid f_i(\mathbf{x}) = f(\mathbf{x})\},$$

and $\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_m$ as the unique minimal points of $f_1, f_2, ..., f_m$, respectively.

Lemma 23. *A straight line path from any point $\mathbf{x} \in D_i$ to $\mathbf{m}_i$ is f-optimal among paths in $\Gamma(\{\mathbf{x}\}, \{\mathbf{m}_i\}, D)$. Its f-optimal value is $f(\mathbf{x})$.*

Proof. By Jensen's inequality (Boyd and Vandenberghe, 2004a) and $f_i(\mathbf{m}_i) \leq f_i(\mathbf{x})$, f_i is nonincreasing along the straight line path from $\mathbf{x}$ to $\mathbf{m}_i$. Additionally, $f \leq f_i$ and $f(\mathbf{x}) = f_i(\mathbf{x})$ due to $\mathbf{x} \in D_i$; therefore, the straight line path is contained in $f(\mathbf{x})$-sublevel set of f. It is also f-optimal in $\Gamma(\{\mathbf{x}\}, \{\mathbf{m}_i\}, D)$ because f-optimal value of any path starts at $\mathbf{x}$ is at least $f(\mathbf{x})$. ■

To find an f-optimal path connecting any two points now reduces to find an f-optimal path connecting any two minimal points $\mathbf{m}_i$ and $\mathbf{m}_j$.

Consider the following optimization problem denoted by $(\mathrm{KP})_{i,j}^{A}$:

$$\begin{array}{ll} \text{minimize} & l \\ \text{subject to} & \mathbf{z} \in A \\ & f_i(\mathbf{z}) \leq l \\ & f_j(\mathbf{z}) \leq l \end{array} .$$

A must be convex for $(\mathrm{KP})_{i,j}^{A}$ to be a convex optimization problem. Let $l_{i,j}$ be the minimal value and $\mathbf{z}_{i,j}$ be a point satisfying the constraints of $(\mathrm{KP})_{i,j}^{D}$ when it is minimized. Solving $(\mathrm{KP})_{i,j}^{D}$ is equivalent to finding the lowest level $l = l_{i,j}$ for l-sublevel set of f_i and l-sublevel set of f_j to coincide, see Figure 8.2 for an illustration. When $l < l_{i,j}$, the l-sublevel set of f_i and that of f_j are disjoint. It should be noted that:

- any nonempty sublevel set of f_i contains $\mathbf{m}_i$, and
- the l-sublevel set of f is exactly the union of all l-sublevel set of f_i, $i \in \{1, 2, ..., m\}$.

For any $l < l_{i,j}$, if the l-sublevel set of f contains both $\mathbf{m}_i$ and $\mathbf{m}_j$, they will not be in the same connected component until $l = l_{i,j}$.

Lemma 24. *If $f_i(\mathbf{z}_{i,j}) > f_j(\mathbf{z}_{i,j})$, the following holds:*

1) $\mathbf{z}_{i,j}$ *minimizes* f_i,

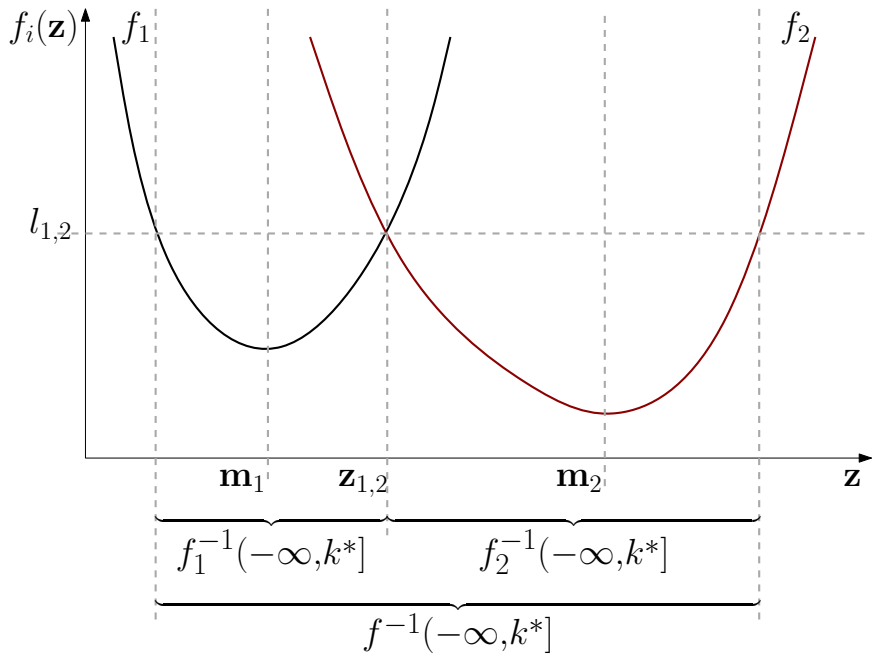

Figure 8.2: A solution for an instance of $(\text{KP})^D_{1,2}$.

2) $f_i(\mathbf{m}_i) > f_j(\mathbf{m}_j)$*, and*

3) there exists an f*-optimal path from any* $\mathbf{x} \in D_i$ *to* $\mathbf{m}_j$ *with an* f*-optimal value of* $f(\mathbf{x})$*.*

Proof. 1) Suppose for a contradiction that $f_i(\mathbf{z}_{i,j}) > f_j(\mathbf{z}_{i,j})$ but $\mathbf{z}_{i,j}$ does not minimize f_i. The function f_j is continuous in its relative interior because of its convexity (Boyd and Vandenberghe, 2004a). By continuity of f_j, there exists a finite radius ball at $\mathbf{z}_{i,j}$ such that $f_j(\mathbf{z}) < f_i(\mathbf{z}_{i,j})$ for any $\mathbf{z}$ inside the ball. Let $\mathbf{w}$ be the global minimizer of f_i. By Jensen's inequality and $f_i(\mathbf{w}) < f_i(\mathbf{z}_{i,j})$, any point $\mathbf{z}$ inbetween $\mathbf{z}_{i,j}$ and $\mathbf{w}$ must satisfy $f_i(\mathbf{z}) < f_i(\mathbf{z}_{i,j})$. Any point $\mathbf{z}$ inside the ball and lies inbetween $\mathbf{z}_{i,j}$ and $\mathbf{w}$ must satisfy: $f_j(\mathbf{z}) < f_i(\mathbf{z}_{i,j}) = l_{i,j}$, and $f_i(\mathbf{z}) < f_i(\mathbf{z}_{i,j}) = l_{i,j}$. Hence, $l_{i,j}$ is not the minimal value for $(\text{KP})^D_{i,j}$, a contradiction.

It follows from 1) that:

2) $f_i(\mathbf{z}_{i,j}) = f_i(\mathbf{m}_i) > f_j(\mathbf{z}_{i,j}) \geq f_j(\mathbf{m}_j)$.

3) $\mathbf{z}_{i,j} = \mathbf{m}_i$ when $f_i(\mathbf{z}_{i,j}) > f_j(\mathbf{z}_{i,j})$. By Lemma 23, a straight line from any

$\mathbf{x} \in D_i$ to such $\mathbf{m}_i$ is f-optimal and contained in $f(\mathbf{x})$-sublevel set of f. By Jensen's inequality and $f(\mathbf{m}_i) \leq f(\mathbf{x})$, a straight line path from $\mathbf{m}_i$ to $\mathbf{m}_j$ is contained in $f(\mathbf{x})$-sublevel set of f. As a result, the concatenation between two described straight line paths forms the f-optimal path. ■

The minimal points $\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_m$ form nodes of a graph. A directed edge from $\mathbf{m}_i$ to $\mathbf{m}_j$ is present if $f_i(\mathbf{z}_{i,j}) > f_j(\mathbf{z}_{i,j})$. Since it is necessary that $f_i(\mathbf{m}_i)$ must be greater than $f_j(\mathbf{m}_j)$ for $f_i(\mathbf{z}_{i,j})$ to be greater than $f_j(\mathbf{z}_{i,j})$, the graph must be a directed acyclic graph. A node without an outgoing edge is called a *sink*. For $\mathbf{x} \in D_i$ such that $\mathbf{m}_i$ is not a sink, there exists an f-optimal path to its sink with f-optimal value of $f(\mathbf{x})$.

Without loss of generality, we assume a path α contained in D to satisfy the following properties:

- α passes through a sequence of overlapping members in $\mathcal{D}$: $D_{\iota(1)}, D_{\iota(2)}, ..., D_{\iota(k)}$. That is, α is a concatenation of k subpaths. Its i-th subpath is contained in $D_{\iota(i)}$.
- α passes through a sequence of points $\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_k$ such that $\mathbf{p}_0 \in D_{\iota(1)}$, $\mathbf{p}_k \in D_{\iota(k)}$ and $\mathbf{p}_i \in D_{\iota(i)} \cap D_{\iota(i+1)}$ for $i \in \{1, 2, ..., k-1\}$.

Based on α, we define $\mathbf{q}_0 = \mathbf{p}_0$, $\mathbf{q}_k = \mathbf{p}_k$, and $\mathbf{q}_i = \mathbf{z}_{\iota(i),\iota(i+1)}$ for $i \in \{1, 2, ..., k-1\}$.

Theorem 1 (f-optimal path in a convex set)**.** *If the path α is f-optimal among paths in $\Gamma(\{\mathbf{p}_0\}, \{\mathbf{p}_k\}, D)$, its f-optimal value is:*

$$\max \Big\{ f_{\iota(1)}(\mathbf{q}_0), \min(f_{\iota(1)}, f_{\iota(2)})(\mathbf{q}_1), \min(f_{\iota(2)}, f_{\iota(3)})(\mathbf{q}_2), ..., \min(f_{\iota(k-1)}, f_{\iota(k)})(\mathbf{q}_{k-1}), f_{\iota(k)}(\mathbf{q}_k) \Big\} \quad (8.2)$$

Proof. We claim that the path α can be transformed to a path α' passing through the points $\mathbf{q}_0, \mathbf{q}_1, ..., \mathbf{q}_k$. The transformation does not increases the supremum value of

f restricted on the path, i.e., preserves optimality. To begin with, let us transform each $\mathbf{p}_i$, $i \in \{1, 2, ..., k-1\}$, to $\mathbf{q}_i$. Note that $f(\mathbf{p}_i) = f_{\iota(i)}(\mathbf{p}_i) = f_{\iota(i+1)}(\mathbf{p}_i)$ by the assumptions on $\mathbf{p}_i$. For $i \in \{0, k\}$, $\mathbf{p}_0$ and $\mathbf{p}_k$ are initially at $\mathbf{q}_0$ and $\mathbf{q}_k$, respectively. For $i \in \{1, 2, ..., k-1\}$, we have that:

- $f_{\iota(i)}(\mathbf{q}_i) \leq f_{\iota(i)}(\mathbf{p}_i)$, (inclusive) or
- $f_{\iota(i+1)}(\mathbf{q}_i) \leq f_{\iota(i+1)}(\mathbf{p}_i)$.

Otherwise:

- $f_{\iota(i)}(\mathbf{p}_i) < f_{\iota(i)}(\mathbf{q}_i) = f_{\iota(i)}(\mathbf{z}_{\iota(i),\iota(i+1)}) \leq l_{\iota(i),\iota(i+1)}$, and
- $f_{\iota(i+1)}(\mathbf{p}_i) < f_{\iota(i+1)}(\mathbf{q}_i) = f_{\iota(i+1)}(\mathbf{z}_{\iota(i),\iota(i+1)}) \leq l_{\iota(i),\iota(i+1)}$.

This is a contradiction since $l_{\iota(i),\iota(i+1)}$ does not minimize $(\mathrm{KP})^D_{\iota(i),\iota(i+1)}$ ($\mathbf{p}_i$ satisfies the feasibility conditions with lower l). Let j be either $\iota(i)$ or $\iota(i+1)$ such that $f_j(\mathbf{q}_i) \leq f_j(\mathbf{p}_i)$. By Jensen's inequality, f_j is nonincreasing along the straight line from $\mathbf{p}_i$ to $\mathbf{q}_i$. This implies that: $f(\mathbf{q}_i) \leq f_j(\mathbf{q}_i) \leq f_j(\mathbf{p}_i) = f(\mathbf{p}_i)$.

The construction of α' is as follows, see Figure 8.3.

1. Connect $\mathbf{q}_0 = \mathbf{p}_0$ to $\mathbf{m}_{\iota(1)}$ by a straight line path. According to Lemma 23, the path is contained in $f(\mathbf{q}_0) = f_{\iota(1)}(\mathbf{q}_0) = f(\mathbf{p}_0)$-sublevel set of f.
2. For $i \in \{1, 2, ..., k-1\}$, connect $\mathbf{m}_{\iota(i)}$ to $\mathbf{q}_i$. by a straight line path. We claim that the path is contained in $\min(f_{\iota(i)}, f_{\iota(i+1)})(\mathbf{q}_i)$-sublevel set of f. Note that $\min(f_{\iota(i)}, f_{\iota(i+1)})(\mathbf{q}_i) \leq f(\mathbf{p}_i)$ by construction of $\mathbf{q}_i$.
   (a) Case $f_{\iota(i)}(\mathbf{q}_i) > f_{\iota(i+1)}(\mathbf{q}_i)$: by Lemma 24, $\mathbf{q}_i = \mathbf{m}_{\iota(i)}$. The claimed sublevel set contains the path which is a single point.
   (b) Case $f_{\iota(i)}(\mathbf{q}_i) \leq f_{\iota(i+1)}(\mathbf{q}_i)$: by Jensen's inequality and $f_{\iota(i)}(\mathbf{m}_{\iota(i)}) \leq f_{\iota(i)}(\mathbf{q}_i) = \min(f_{\iota(i)}(\mathbf{q}_i), f_{\iota(i+1)})(\mathbf{q}_i)$, The claimed sublevel set contains the path.

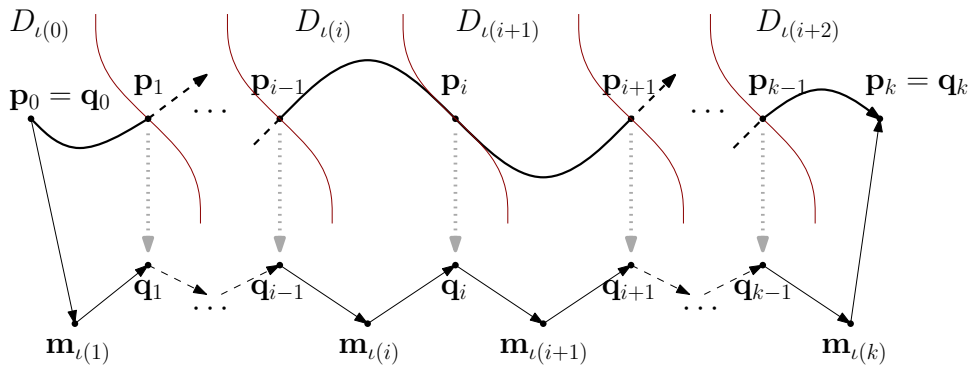

Figure 8.3: Abstracted illustration for the optimality-preserving path transformation.

3. Similar to the previous step, connect $\mathbf{q}_i$ to $\mathbf{m}_{\iota(i+1)}$ by a straight line path. The path is contained in $\min(f_{\iota(i)}, f_{\iota(i+1)})(\mathbf{q}_i)$-sublevel set of f.

4. Finally, connect $\mathbf{m}_{\iota(k)}$ to $\mathbf{q}_k = \mathbf{p}_k$ by a straight line path. The path is contained in $f(\mathbf{q}_k) = f_{\iota(k)}(\mathbf{q}_k) = f(\mathbf{p}_k)$-sublevel set of f

Given that α is f-optimal, α' is f-optimal as well by construction. The f-optimal value of α' is as stated in the theorem because $f \leq f_j$, for any j. ■

Theorem 1 leads to an approach to compute the f-optimal value for an f-optimal path connecting any two terminal points in D. The computation reduces to finding a sequence of indices: ι that minimizes (8.2) such that $D_{\iota(1)}$ and $D_{\iota(k)}$ contains the terminal points. The sequence ι does not need to satisfy the condition that $D_{\iota(i)} \cap D_{\iota(i+1)}$ is nonempty. The theorem just gurantees that a sequence producing the f-optimal value is among those that satisfy the condition. For any sequence that does not satisfy the condition, the value computed from (8.2) will be greater than or equal to the f-optimal one.

8.5 Roadmap Construction

Based on the derived properties, we construct a roadmap (Choset et al., 2005a) for optimal paths contained in a nonconvex domain. The definition of f here remains the same except that its domain D is not necessarily convex. For a compact

convex set K contained in D, we define its "local roadmap" $RM(K)$ by a complete graph of m nodes. The i-th node is the global minimal point of $f_i|_K$, $i \in \{1, 2, ..., m\}$. The "cost" between an edge connecting node i and j is given by $\min(f_i, f_j)(\mathbf{z}_{i,j}^K)$ where $\mathbf{z}_{i,j}^K$ is a point satisfying the constraints of $(\text{KP})_{i,j}^K$ when it is minimized. It is possible to optimize local roadmaps to some extent by collapsing nodes (minimal points) that are not sink within its convex domain to its sink. Then, remove duplicated edges that appear after collapsing, keeping only one with the minimal cost.

For the global roadmap of f, let $\mathcal{K} = \{K_1, K_2, ..., K_c\}$ be a finite subcover of the configuration space such that every member of $\mathcal{K}$ is compact. The global roadmap of f over the compact convex subsets consists of:

1. the local roadmaps: $RM(K_1)$, $RM(K_2)$, ..., $RM(K_c)$; and
2. "transitions" among the local roadmaps.

According to Lemma 23 and 24, the global roadmap can be accessed from any point $\mathbf{x}$ in D:

1. find the decomposed convex subset K containing $\mathbf{x}$.
2. find k that $f_k(\mathbf{x}) = f(\mathbf{x})$.
3. find the sink of the minimizer of $f_k|_K$.

The access cost is: $f_k(\mathbf{x})$. For K_i and K_j that has a nonempty intersection, their local roadmaps are linked together by a transition part. The transition part consists all of the nodes in the local roadmap of $K_i \cap K_j$ and additional edges the local roadmaps of K_i and K_j. The additional edges are generated by connecting nodes of $RM(K_i \cap K_j)$ to those of $RM(K_i)$ and those of $RM(K_j)$. This is similar to accessing the global roadmap except that K is given. For each $K \in \{K_i, K_j\}$ and a node $\mathbf{m}$ in $RM(K_i \cap K_j)$, an edge links $\mathbf{m}$ to a node $\mathbf{n}$ of $RM(K)$ if $\mathbf{n}$ is a sink (node) of a minimizer of $f_k|_K$ such that $f_k(\mathbf{m}) = f(\mathbf{m})$. The edge's cost is $f_k(\mathbf{m})$. Conforming the definition of

f-optimal, the distance of a path contained in the global roadmap of f is defined as the maximum cost of the edges comprising the path. A single-source shortest path algorithm is applied to compute shortest distances from every node to a given escape node $\mathbf{e}$ which globally minimizes f. The shortest distance from a node $\mathbf{n}$ to $\mathbf{e}$ is the f-optimal value of some f-optimal path among paths in $\Gamma(\{\mathbf{n}\}, \{\mathbf{e}\}, D)$.

Will this global roadmap works with any two finger placements in the configuration space? If the finger placements are in the same convex subset in $\mathcal{K}$ an f-optimal path is already contained in a local roadmap by construction. For a path α that traverses across convex polyhedra, we can assume without loss of generality that

- α passes through a sequence of overlapping convex polyhedra in the finite subcover $\mathcal{K}$: $K_{\iota(1)}, K_{\iota(2)}, ..., K_{\iota(k)}$. That is, α is a concatenation of k subpaths. Its i-th subpath is contained in $K_{\iota(i)}$.
- α passes through a sequence of points $\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_k$ such that $\mathbf{p}_0 \in K_{\iota(1)} \in K_{\iota(1)}$, $\mathbf{p}_k \in K_{\iota(k)} \in K_{\iota(k)}$ and $\mathbf{p}_i \in K_{\iota(i)} \cap K_{\iota(i+1)}$.

It is possible to transform α to a path that visits another sequence of points $\mathbf{q}_0, \mathbf{q}_1, ..., \mathbf{q}_k$ where $\mathbf{q}_0 \equiv \mathbf{p}_0, \mathbf{q}_k \equiv \mathbf{p}_k$, and $\mathbf{q}_i$ is a node of $RM(K_{\iota(i)} \cap K_{\iota(i+1)})$ for $i \in \{1, 2, ..., k-1\}$. The transformation simply inserts a loop (path) that starts at $\mathbf{p}_i$ and visits $\mathbf{q}_i$, for any $i \in \{1, 2, ..., k-1\}$. Each loop is constructed by concatenating a descent path from $\mathbf{p}_i$ to a sink obtained by Lemma 23 and 24 with its reverse. The transformed path is still f-optimal when α is f-optimal. This shows that there exists an f-optimal path visiting a sequence of nodes in transition parts, i.e., the global roadmap works.

8.6 Cage Identification Algorithm

Let us return to the problem of identifying all solution sets. The following are the steps for precomputing $\underline{\delta}_p^*$:

1. Decompose the compact workspace space $\bar{\mathcal{F}} \equiv \mathcal{B} \backslash \mathcal{P}$ into convex polyhedra $W_1, W_2, ..., W_w$. The decomposition can be an optimal convex decomposition, or just a triangulation. For simplicity, we assume that the decomposition does not increase the number of vertices. Construct the decomposed convex polyhedra of the compact configuration space $K_1, K_2, ..., K_c$ as a cartesian product among n decomposed polyhedra, i.e.,

$$\begin{aligned} K_1 &= W_1 \times W_1 \times \cdots \times W_1 \times W_1, \\ K_2 &= W_1 \times W_1 \times \cdots \times W_1 \times W_2, \\ &\cdots \\ K_c &= \underbrace{W_w \times W_w \times \cdots \times W_w \times W_w}_{n \text{ terms}}. \end{aligned}$$

2. Construct the global roadmap of $\underline{\delta}_p$ over the decomposed subsets $K_1, K_2, ..., K_c$. Transition parts are created only for nonempty intersection K_i and K_j.

3. Run the SSSP algorithm with $\mathbf{e}$ assigned to the global minimal point of $\underline{\delta}_p$ restricted to $\mathcal{T}$.

At this point, it is possible to query for approximated error tolerance at any finger placement and enumerate all nodes in the global roadmap of $\underline{\delta}_p$ that lies inside a solution set. To check if a node $\mathbf{n}$ that $\underline{\delta}_p(\mathbf{n}) < \underline{\delta}^*_p(\mathbf{n})$ is in a solution set, we compute $\bar{\delta}_p(\mathbf{n})$ and test if it is less than $\underline{\delta}^*_p(\mathbf{n})$. Recall the sufficient condition that a node $\mathbf{n}$ is in a solution set if it satisfies $\bar{\delta}_p(\mathbf{n}) < \underline{\delta}^*_p(\mathbf{n})$.

The remaining steps (optional, depending on applications) are to group nodes that are in the same cage. By Proposition 21, nodes $\mathbf{m}$ and $\mathbf{n}$ are in the same cage if there exists a path that connects them and is contained in $\underline{\delta}^*_p(\mathbf{m}) = \underline{\delta}^*_p(\mathbf{n})$-sublevel set of $\bar{\delta}_p$.

4) Assume without loss of generality that $K_1, K_2, ..., K_b$, $b \leq c$, are convex polyhedra containing some node $\mathbf{n}$ with $\bar{\delta}_p(\mathbf{n}) < \underline{\delta}^*_p(\mathbf{n})$. Construct the global roadmap of $\bar{\delta}_p$ over the decomposed subsets $K_1, K_2, ..., K_b$. Remove every node

n (and edges associated to **n**) that does not satisfy: $\bar{\delta}_p^*(\mathbf{n}) < \underline{\delta}_p^*(\mathbf{n})$ during the roadmap creation.

5) Group nodes contained in the same solution set. Nodes **m** and **n** connected by an edge are in the same solution set if the edge's cost is below $\underline{\delta}_p^*(\mathbf{m}) = \underline{\delta}_p^*(\mathbf{n})$.

After completing all the steps, we obtain the global roadmap of $\bar{\delta}_p$. Unlike the previous one, it is for identifying which solution set a finger placement is in and enumerating all possible solution sets.

The time complexity of step 3) is $O(cm^2 \log(cm^2))$ dominating the others. Step 2) involves solving $O(cm^2)$ convex optimization problems of size $O(n)$ which maybe a bottle neck when n is small. The time complexity of Step 8.6 is proportional to n times the number of caging nodes, which is much lower than the number of nodes in practice. The exact time complexity is $O((c + t_c)m^2 \log((c + t_c)m^2)) + (b + t_b)M^2 \log((b + t_b)M^2))$ where t_c and t_b are the number of transition parts (overlapping convex sets) for the first and the second global roadmap, respectively. The sum between c and t_c represents the complexity of the configuration space. Yet, it depends on the convex decomposition. An optimal convex decomposition should produce the least number.

We rely on point location algorithm (Snoeyink, 2004) to query for a solution set containing a finger placement $\mathbf{x} = (\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_n)$. A solution set query requires n point location queries for identifying a convex subset containing each x_i and $O(M)$ tests for a sink of **x**. The time complexity of a solution set query is $\bar{\delta}_p$ is $O(\log v + M)$.

8.6.1 Approximation for Input Object

In practice, the input polygon is usually complex, inaccurate and incomplete, e.g., scanned from range sensors. Processing complex input can be time consuming. On the other hand, traditional polygon simplification may lead to false solutions. We apply the proposed approach in Chapter V It resembles to the strategy applied when we approximate δ_p with $\underline{\delta}_p$ and $\bar{\delta}_p$. In addition to the deformation measure bounds, the unknown actual input polygon $\mathcal{P}$ must lie inbetween a lower-

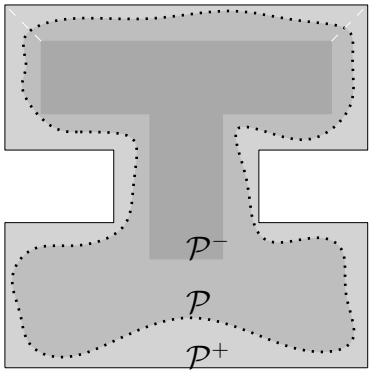

Figure 8.4: The set $\mathcal{P}$ represents the unknown actual object inbetween the lowerbound $\mathcal{P}^-$ and the upperbound $\mathcal{P}^+$.

bound and an upperbound: $\mathcal{P}^-$ and $\mathcal{P}^+$, see Figure 8.4. The lowerbound and the upperbound ensures the presence and absence of the object mass inside $\mathcal{P}^-$ and outside $\mathcal{P}^+$, respectively. Now, there are three versions of the configuration space: Ω, $\Omega^- \equiv (\mathcal{B}\backslash(\mathcal{P}^-))^n$, $\Omega^+ \equiv (\mathcal{B}\backslash(\mathcal{P}^+))^n$. Because of the object's uncertainty, we must prepare for the worst case. The more spacious Ω^- is used when computing the (approximated) error tolerance. Since $\Omega^- \subseteq \Omega$, the set of all escape paths between two specific sets of terminal points $\Gamma(A, B, \Omega^-) \subseteq \Gamma(A, B, \Omega)$; as a result, error tolerance for $\mathcal{P}^-$ is lower than that for $\mathcal{P}$ pointwise. On the other hand, the less spacious Ω^+ applies when identifying valid finger placements.

To handle imperfect input, the algorithm requires a few modifications. In the first step, $K_1, K_2, ..., K_c$ must be obtained from the convex decomposition of $\mathcal{B}\backslash(\mathcal{P}^-)$ instead. In the forth step, $K_1, K_2, ..., K_b$ must be replaced by decomposed convex subsets of $\mathcal{B}\backslash(\mathcal{P}^+)$ that overlap with $K_1, K_2, ..., K_b$.

8.6.2 Implementation and Results

We implement the algorithm in C++ and solve convex optimization problems with MOSEK (mos, 2011). Figure 8.5 (a), (b) and (c) show projections of top-*quality* solution sets on the workspace for an object generated after step 3) with $p = 2$, (a) $m = 8$, (b) $m = 16$, and (c) $m = 32$. Each solution set contains finger placements

that form $\underline{\delta} < \underline{\delta}^*(\mathbf{x})$ cages where $\mathbf{x}$ is a finger placement inside. The object is shown as a shaded region. The black dots represents the finger placement where the deformation measure $\underline{\delta}_p$ restrict in a corresponding solution set attains its minimal value. Each displayed loop surrounds all the possible positions of a finger inside a solution set. The formation to fix is the same as the leftmost set of points which is also an immobilizing stretching grasp. Quality of a solution set is determined by the gap between its approximated error tolerance and the minimal value of the deformation measure restricted inside the solution set.

The second solution set from the left in Figure 8.5 (a) breaks into the second and the third solution sets as m increases. In addition, the quality value and its order of similar solution sets are not preserved as m varies.

It is possible that the finger placement where the deformation measure restricted on a solution set attains its minimal value is not zero. The fingers can never be at the finger formation to fix as long as they remain inside such a solution set. This approach to caging, caging by attempting to fix a finger formation, may include cages similar to squeezing and stretching ones. In the figures, each set of cross marks is a finger placement with zero deformation that minimizes L_∞ norm of the displacement vector. Perfectly overlapping dots and crosses indicates that their corresponding solution set contains a member with zero deformation; otherwise, it does not.

Figure 8.6 shows projections generated after step 5) with $p = 2$, $m = 16$, (a) $M = 16$, (b) $M = 32$ and (c) $M = 64$. Each projected region will be a subset of a corresponding region in Figure 8.5. Small M leads to smaller solution sets and may cause them to break apart even if they are contained in the same $\delta_p < \epsilon$ cage.

Figure 8.7 shows solution sets with $p = \infty$, each circle surrounding a cross mark represents a radius that each finger may displace independently from its position in the formation without breaking the cage.

We apply the algorithm to identify error tolerance of immobilizing grasps (Fig-

ure 8.8) and convert dispersion-control cages (Figure 8.9).

The results generated by the modified algorithm are shown in Figure 8.10. Error tolerance decreases in exchange for ambiguity of the shape.

The running time taken to generate the results are shown in Table 8.1 where:

- $\underline{t}_{\text{opt}}$: time spent in performing convex optimization for the first global roadmap,
- $\underline{t}_{\text{prop}}$: time spent in propagating the error tolerance for the first global roadmap,
- $\bar{t}_{\text{opt}}$: time spent in performing convex optimization for the second global roadmap, and
- $\bar{t}_{\text{prop}}$: time spent in propagating the error tolerance for the second global roadmap.

Our implementation performs all the tasks sequentially on an intel i5 CPU 650 at 3.2 GHz. The algorithm running time can be greatly improved by parallel computing since each convex optimization subproblems can be solved separately. Applying available parallel SSSP algorithm also helps when the roadmap becomes more complex.

8.7 Summary

This paper presents an algorithm to identify cages by limiting deformation of an n-point formation on a plane. The deformation measure is based on a given formation to maintain. It is defined as the minimized norm of distances between each finger and its position in a rigid-transformed formation. An object is caged by preventing the deformation greater than an error tolerance after appropriately placing the fingers. The algorithm approximates the cages by providing approximated error tolerance. The approximated error tolerance is guaranteed to be less than or equal

Table 8.1: Algorithm Execution Time (Robust Caging)

Figure		p	$c+t_c$	$b+t_b$	m	M	$\underline{t}_{\text{opt}}$	$\underline{t}_{\text{prop}}$	$\bar{t}_{\text{opt}}$	$\bar{t}_{\text{pro}}$	total
8.5	(a)	2	1960	-	8	-	111.60	1.31	-	-	112.91
	(b)	2	1960	-	16	-	365.05	7.06	-	-	372.11
	(c)	2	1960	-	32	-	1398.74	109.53	-	-	1508.27
8.6	(a)	2	1960	60	16	16	365.05	7.06	7.50	0.00	379.61
	(b)	2	1960	60	16	32	365.05	7.06	43.74	0.02	415.87
	(c)	2	1960	60	16	64	365.05	7.06	163.20	0.17	535.48
8.7		∞	1960	60	16	64	499.01	9.17	184.58	0.33	693.09
8.8	I	∞	864	70	16	64	166.79	3.60	192.16	6.17	368.72
	J	∞	972	9	16	64	191.15	4.60	67.58	0.12	263.45
	R	∞	864	51	16	64	183.92	3.23	187.19	2.32	376.66
	R	∞	864	49	16	64	183.96	2.96	185.73	0.68	373.33
	O	∞	1280	4	16	64	274.53	3.48	11.47	0.00	289.48
	C	∞	864	33	16	64	169.38	3.76	107.58	0.84	281.56
	A	∞	1666	12	16	64	315.29	8.36	56.50	0.02	380.17
	G	∞	1372	166	16	64	254.47	14.41	400.24	11.98	681.10
	E	∞	1666	54	16	64	292.33	12.97	126.58	1.72	433.6
8.9	(a)	∞	1188	16	16	64	284.56	6.01	44.98	0.16	335.71
	(b)	∞	1188	17	16	64	324.69	2.35	55.05	0.05	382.14
	(c)	∞	3200	73	16	64	757.32	77.46	291.04	2.18	1128.00

to the real value. Higher quality approximation makes it approaches the real value. Under lower quality setting, the algorithm will only report sufficiently large cages, ignoring cages with low actual error tolerance.

The algorithm is designed for caging an object on a plane. Straightforwardly applying the algorithm to three-dimensional workspace, feeding three-dimensional convex subsets as input, yields solutions under the assumption that the finger formation and the object can only rotate around a fixed axis.

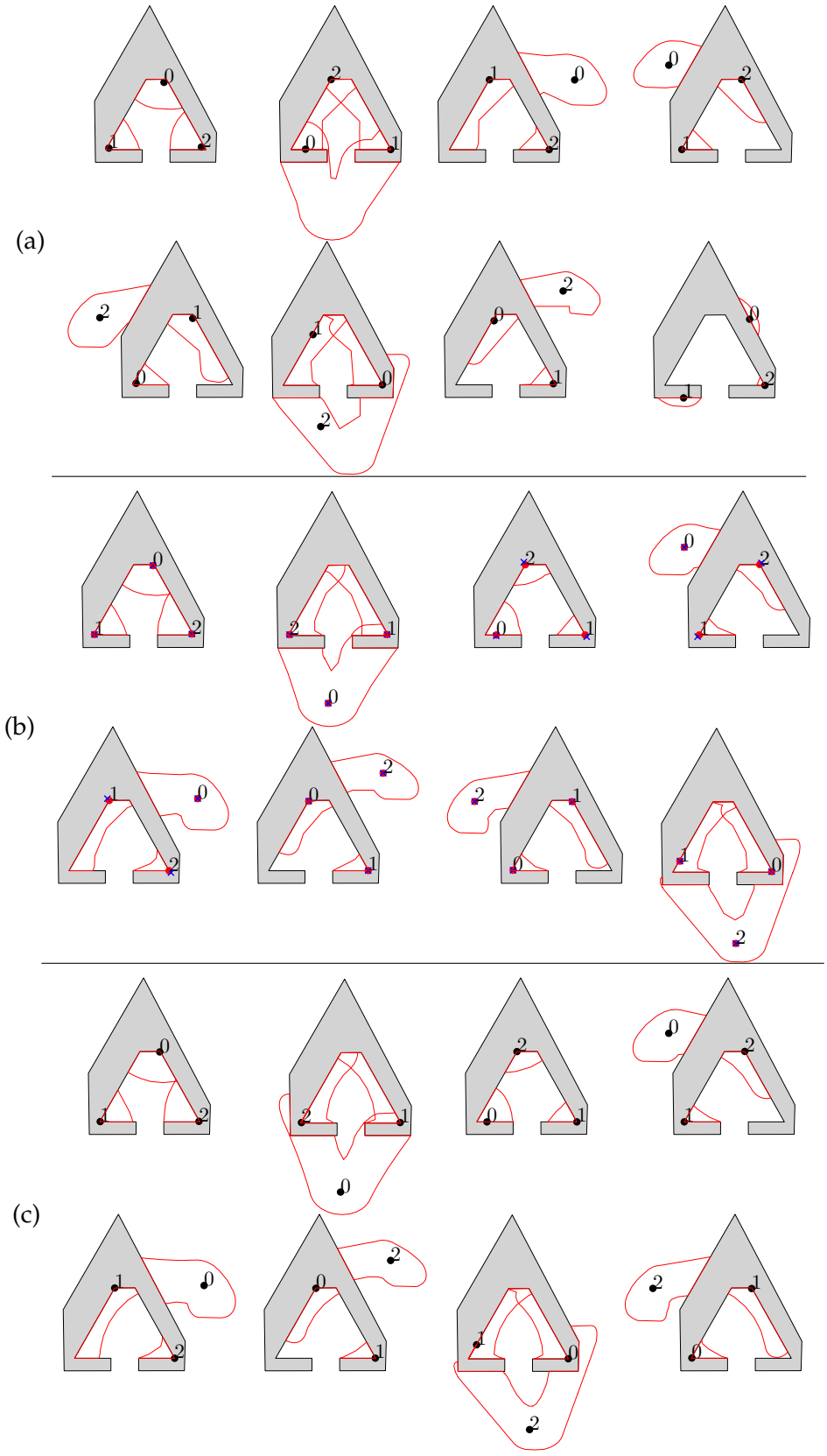

Figure 8.5: Projections onto the workspace of top-quality solutions generated after step 3) with $p = 2$ a) $m = 8$, b) $m = 16$, and c) $m = 32$.

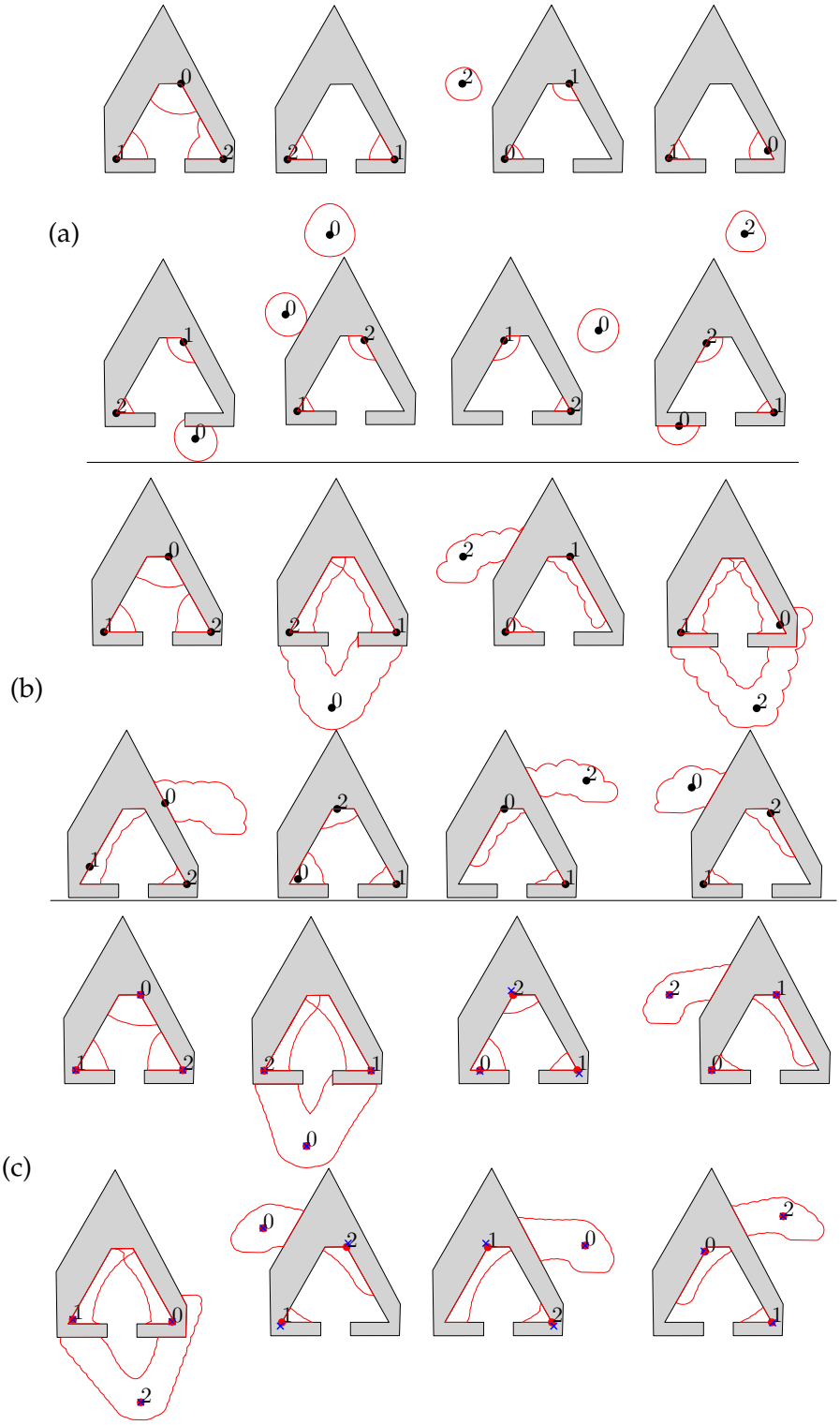

Figure 8.6: Projections onto the workspace of top-quality solutions generated after step 5) with $p = 2$, $m = 16$, a) $M = 16$, b) $M = 32$, and c) $M = 64$.

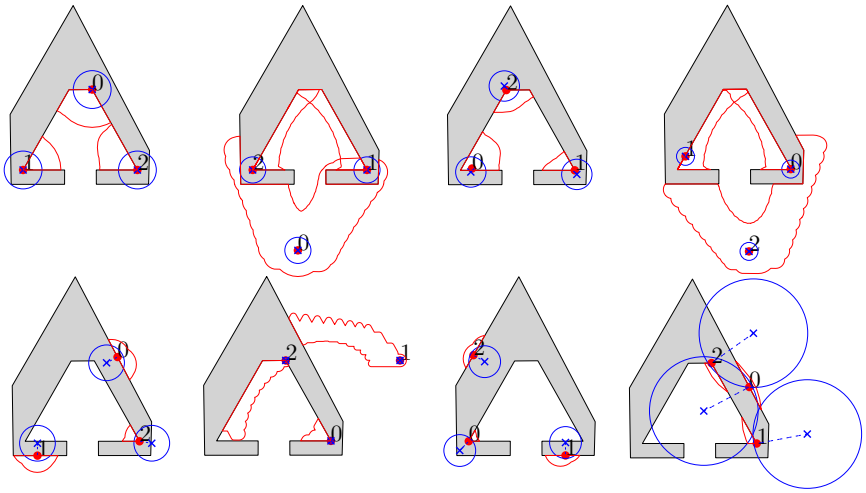

Figure 8.7: Solution sets when $p = \infty$, $m = 16$ and $M = 64$. *Top:* top-quality solution sets. *Bottom:* lower-quality solution sets that does not contain any finger placement with zero deformation measure inside.

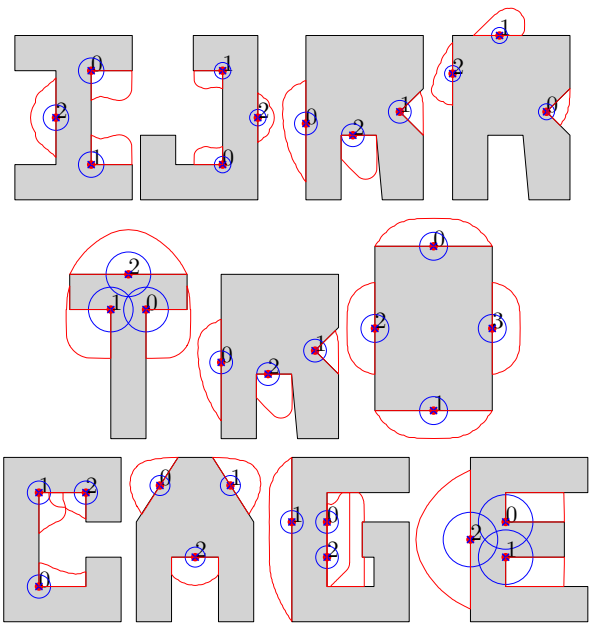

Figure 8.8: Cages from grasps. The paramters are $p = \infty$, $m = 16$, $M = 64$.

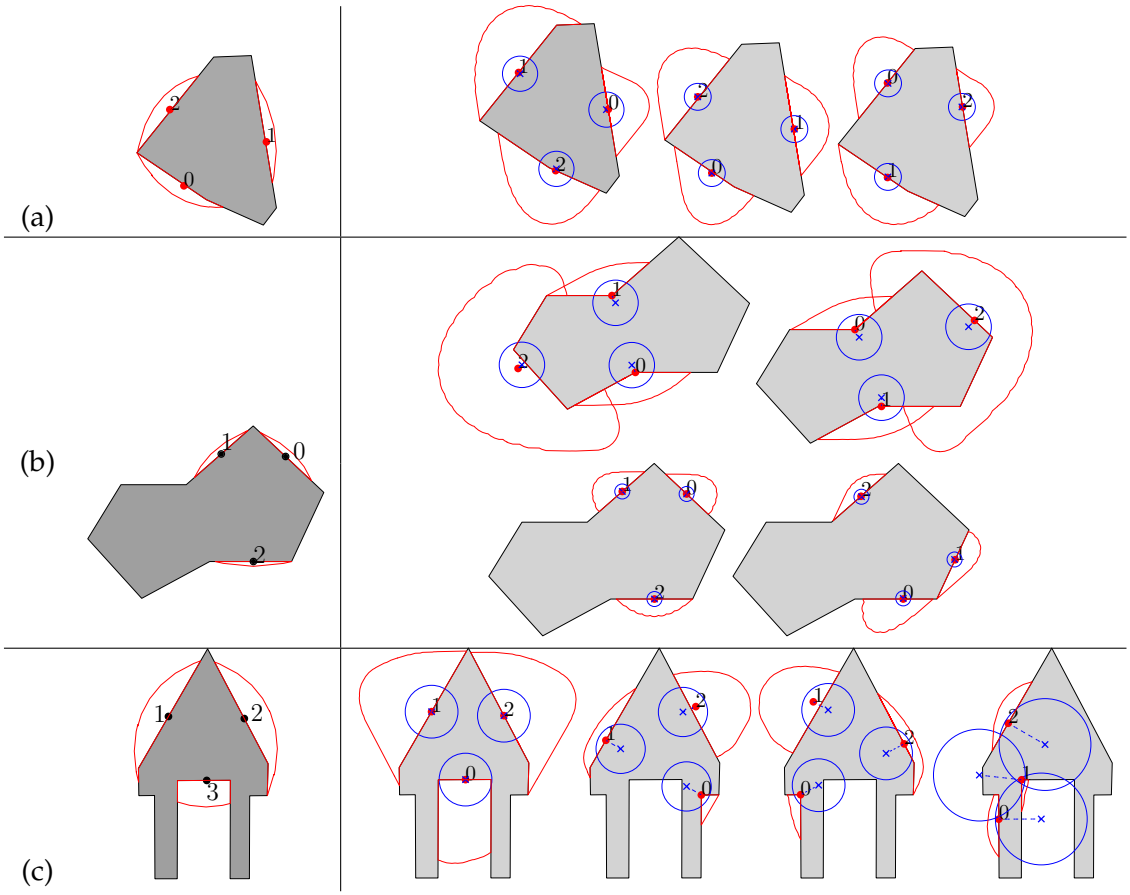

Figure 8.9: *Left:* the formations to fix and the object. Each red region bounds possible movement of a finger under $\delta^2_{\text{adj}} < \epsilon$ cage. *Right:* top-quality approximated $\delta_\infty < \epsilon$ cages with $m = 16$ and $M = 64$. Note: formation in (c) is symmetric, symmetric cages are not shown.

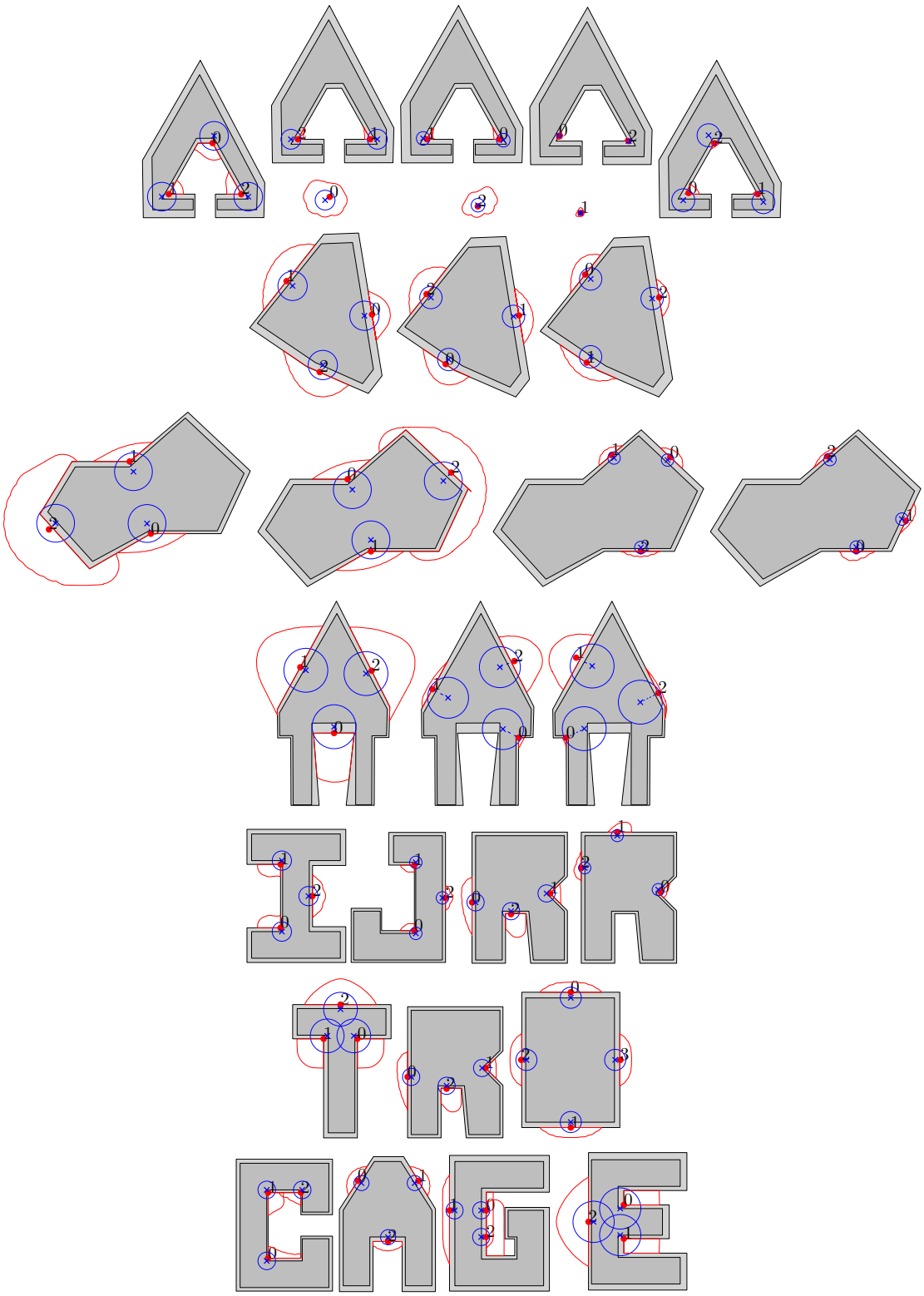

Figure 8.10: Imperfect shape cages capable of caging any object inbetween the bounds. Symmetric cages are not shown.

CHAPTER IX

CONCLUSION

In this work, we consider the problem of caging an object. The range of the problem concerns two and three dimensional setting, two or multiple fingers, free or constrained fingers, rigid or slightly deformable object and uncertainty. We present various approaches to caging and analyse appropriate conditions to apply each of them. When a large number of free moving fingers is available it is best to employ the diameter-based method, surrounding the object by limiting the distance between the fingers. This ensures the object is caged for subsequent manipulation task. When only two point fingers are available, the object must have at least a concave section in order to cage. More fingers leads to multi-finger squeezing and stretching cages. However, without concave sections, mutli-finger cages can be very small and provide low-error tolerance because a single dispersion constraint is not strict enough to control the increasing freedom of the finger formations. We present a framework that capable of handling multiple constraints. The constraints can be from the manipulator's physical limits or additional control policies. The constraints have to be convex otherwise approximation is required. The approximation technique is presented which turns out to be a general method for dealing with uncertainty of measurement and control. Nevertheless, the approximation is not required for some nonconvex constraints as presented in the approach of combining squeezing and stretching caging. A variation of the approximation technique also leads to the solution presented in the previous chapter designed to determine a very intuitive error tolerance measure.

More fingers, more constraints, better quality of approximation leads to more resource consuming computation. Yet, as stated, our algorithm are suitable for a system with several fingers. The caging problem can just be solved by surrounding the object strategy when more fingers are available. Fortunately, the computation speed is not a problem in the near future. The reason is because that the computation tasks of our algorithms can be processed in parallel. Especially, the most time

consuming task: solving small convex optimization problems.

References

The MOSEK Optimization Software. 2011.

Bicchi, A., and Kumar, V. Robotic grasping and contact: A review. In Preecedings of IEEE International Conference on Robotics and Automation, volume 1, pp. 348–353. San Francisco, CA, USA. April 2000.

Blind, S.J., McCullough, C.C., Akella, S., and Ponce, J. Manipulating parts with an array of pins: A method and a machine. The International Journal of Robotics Research 20(10) (2001): 808–818.

Boyd, S., and Vandenberghe, L. Convex Optimization, chapter Convex functions. Cambridge University Press. 2004a: pp. 67–126.

Boyd, S., and Vandenberghe, L. Convex Optimization, chapter Convex functions. Cambridge University Press. 2004b: pp. 67–113.

Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., and Thrun, S. Principles of Robot Motion, chapter Roadmaps. The MIT Press. 2005a: pp. 107–128.

Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., and Thrun, S. Principles of Robot Motion, Theory, Algorithms and Implementations, chapter 6. Cell Decompositions. MIT Press. 2005b: pp. 170–178.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C. Introduction to Algorithms, chapter Graph Algorithms. The MIT Press. 2001: pp. 595–599.

Davidson, C., and Blake, A. Caging planar objects with a three-finger one-parameter gripper. In Proceedings of IEEE International Conference on Robotics and Automation, volume 3, pp. 2722–2727. Leuven. May 1998a.

Davidson, C., and Blake, A. Error-tolerant visual planning of planar grasp. In Proceedings of IEEE International Conference on Conference on Computer Vision, pp. 911–916. Bombay. January 1998b.

Erickson, J., Thite, S., Rothganger, F., and Ponce, J. Capturing a convex object with three discs. In Proceedings of IEEE International Conference on Robotics and Automation, pp. 2242–2247. Taipei, Taiwan. September 2003.

Gopalakrishnan, K.G., and Goldberg, K. Gripping parts at concave vertices. In Proceedings of IEEE International Conference on Robotics and Automation, volume 2, pp. 1590–1596. Washington, DC, USA. May 2002.

Gumhold, S., Borodin, P., and Klein, R. Intersection free simplification. In In The 4th Israel-Korea Bi-National Conference on Geometric Modeling and Computer Graphics, pp. 11–16. Tel-Aviv, Israel. 2003.

Joe, B. Geompack – a software package for the generation of meshes using geometric algorithms. Advances in Engineering Software and Workstations 13(5-6) (1991): 325 – 331.

Joe, B. Geompack++. 2010.

Kriegman, D.J. Let them fall where they may: capture regions of curved objects and polyhedra. International Journal of Robotics Research 16 (1997): 448–472.

Kuperberg, W. Problems on polytopes and convex sets. DIMACS Workshop on Polytopes (January 1990): 584–589.

Pipattanasomporn, P., and Sudsang, A. Two-finger caging of concave polygon. In Proceeding of IEEE International Conference on Robotics and Automation, pp. 2137–2142. Orlando, FL. May 2006.

Pipattanasomporn, P., Vongmasa, P., and Sudsang, A. Caging rigid polytopes via finger dispersion control. In Proceeding of IEEE International Conference on Robotics and Automation, pp. 1181–1186. Pasadena, CA. 2008.

Rimon, E., and Blake, A. Caging 2d bodies by 1-parameter two-fingered gripping systems. In Proceedings of IEEE International Conference on Robotics and Automation, volume 2, pp. 1459–1464. Minneapolis, MN, USA. April 1996.

Rimon, E., Mason, R., Burdick, J.W., and Or, Y. A general stance stability test based on stratified morse theory with application to quasi-static locomotion planning. IEEE Transactions on Robotics 24 (June 2008): 626–641.

Rodriguez, A., and Mason, M. Two finger caging: squeezing and stretching. In Workshop on the Algorithmic Foundations of Robotics (WAFR) 2008. Guanajuato, México. December 2008.

Rodriguez, A., Mason, M., and Ferry, S. From caging to grasping. In Proceedings of Robotics: Science and Systems. Los Angeles, CA, USA. June 2011.

Snoeyink, J. Handbook of Discrete and Computational Geometry, chapter 34. Point Location. Chapman & Hall/CRC, New York, 2 edition. 2004: pp. 767–785.

Sudsang, A. A sufficient condition for capturing an object in the plane with disc-shaped robots. In Proceedings of IEEE International Conference on Robotics and Automation, volume 1, pp. 682–687. Washington, DC, USA. 2002.

Sudsang, A., Ponce, J., Hyman, M., and Kriegman, D.J. On manipulating polygonal objects with three 2-dof robots in theplane. In Proceedings of IEEE International Conference on Robotics and Automation, volume 3, pp. 2227–2234. 1999.

Sudsang, A., Ponce, J., and Srinivasa, N. Grasping and in-hand manipulation: Experiments with a reconfigurable gripper. Advanced Robotics 12(5) (1997): 509–533.

Sudsang, A., Rothganger, F., and Ponce, J. Motion planning for disc-shaped robots and pushing a polygonal object in the plane. IEEE Transactions on Robotics and Automation 18.

Vahedi, M., and van der Stappen, A.F. Caging polygons with two and three fingers. In Workshop on the Algorithmic Foundations of Robotics (WAFR) 2006. New York. July 2006.

Vahedi, M., and van der Stappen, A.F. Caging polygons with two and three fingers. The International Journal of Robotics Research 27(11-12) (2008): 1308–1324.

Van den Bergen, G. A fast and robust gjk implementation for collision detection of convex objects. J. Graph. Tools 4(2) (March 1999): 7–25.

Vongmasa, P., and Sudsang, A. Coverage diameters of polygons. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4036–4041. Beijing. October 2006.

Biography

Peam Pipattanasomporn was born in Bangkok, Thailand, on December, 1982. He received B.Eng. and M.Eng., both in computer engineering, from Chulalongkorn University, Thailand, in 2004 and 2006, respectively. In his master degree and the first few years in his doctorate, he also serves as a teaching assistant at the Department of Computer Engineering, Chulalongkorn University. His bachelor degree and his master master degree have been supervised by Dr. Attawith Sudsang. His doctorate has been also under the supervision of Dr. Attawith Sudsang. From 2007 to 2009, he has received a grant from Chulalongkorn University graduate scholarship to commemorate the 72^{nd} anniversary of his majesty King Bhumibol Adulyadej and the Thailand Research Fund. From 2009, he has received a grant from the Thailand Research Fund through the Royal Golden Jubilee Ph.D. Program under Grant No. Ph.D. 1.O.CU/51/D.1. Additionally, he received a grant from the 90^{th} Anniversary of Chulalongkorn University Fund through the Ratchadapiseksomphot Fund. His field of interest includes various topics in Robotics with emphasis on object caging and grasp planning.