Efficient Caging Planning Under Uncertainty Based on Configuration Spaces of Target Object and Finger Formation

(対象物体と指配置のコンフィグレーション空間を
用いた不確かさを扱える効率的なケージング計画)

万　偉偉

Abstract

This thesis discusses the planning algorithms of a novel type of grasping closure – namely caging. In the author's view, this is the first literature which systematically discusses caging planning algorithms and their applications in robotics. Although the idea of caging is not proposed by the author himself, he further develops the geometric definition of caging and makes it pragmatic.

The thesis contributes in three aspects. Firstly, in caging theory, the thesis initially explains the relationship between caging and traditional research in grasping. Namely, caging is the extension of immobilization. Secondly, in caging planning algorithms, the thesis initially employs caging to deal with uncertainty. It on the one hand proposes efficient algorithms to deal with caging test while on the other hand further proposes efficient algorithms to deal with caging optimization. Both caging test algorithms and caging optimization algorithms are explored in the configuration space of target object and the configuration space of finger formation. Thirdly, in the aspect of applications, the thesis applies the proposed caging planning algorithms to robotic hands and multi-robot cooperative transportation. It discusses how to select proper algorithms according to requirements of real-world applications. Results show that the algorithms are not only robust to various uncertainty but also helpful to reduce the number of fingers or mobile robots.

Main texts of the thesis are divided into four parts. The first three parts corresponds to the three contributions. The first part is the basic concepts of caging. It reviews time-of-the-art progresses in robotic manipulation and presents the contribution in theoretical aspect, namely relationship between caging and traditional research topics in grasping. The remaining two parts discuss caging by using two different spaces. One is the configuration space of target object and the other is the configuration space of finger formation. Details on how to solve the caging problems, namely the caging test problem and the caging optimization problem, in those two spaces are discussed respectively in these two parts. These two parts also involve applications of the caging algorithms like a distributed end-effector, a gripping manipulator and multi-robot cooperation.

The configuration space of target object and the configuration space of finger formation essentially equal with each other by a linear transformation. They actually provide different metrics to measure the robustness of caging. The fourth part of the thesis discusses the relationship of the algorithms in the two spaces and how to choose a proper algorithm according to mechanical structures and specific tasks. It also summarizes the thesis and proposes potential future directions.

The concepts and algorithms proposed in this thesis can efficiently solve 2D manipulation problems in the presence of uncertainty. The author believes that caging is a promising tool to deal with perception and control uncertainty. He would like to spread this tool and explore more about this tool in both theory and application aspects in the future.

Contents

List of Figures

Chapter 1

Introduction

1.1 Robotic Manipulation and Its Difficulties

Robotic manipulation involves two aspects. One is the working subject, namely a robot. The other one is the working motion, namely manipulation or rearranging the world by hands. Prof. Matthew T. Mason discussed a lot about how should a robot rearrange the world with robotic hands in his book "Mechanics of Robotic Manipulation"[Mason, 2001]. Mechanics is one essential problem in robotic manipulation. However, it is never the unique one. Generally speaking, a pragmatic robotic manipulation suffers from either difficulties from mechanism, sensing and mechanics. We can summarize them as following.

- **Mechanism:** Structures and organizations of a robotic hand
- **Sensing:** Perception and understanding of target objects
- **Mechanics:** Forces exerted by fingers and balances of those forces

Mechanism relates to structures of robotic hands. It concentrates on kinematics and actuation, especially how to select and organize mechanical components to obtain better control performance. Prof. Nancy Pollard in his course "Hands: Design and Control for Dexterous Manipulation"[Pollard, 2010] gave a good review of popular hands and related design issues. In most cases, robotic hands are designed according empirical requirements of specific tasks or designed by mimicking certain biological creatures. These hands are in front of difficulties like complicated control and expensive actuators. Low-cost, high-robustness and general-purpose robotic hands remains a popular research topic. One representative work in this topic is [Rodriguez and Mason, 2013](fundamentally based on [Rodriguez and Mason, 2012a]). It won the IEEE ICRA2013 best student paper award.

Sensing relates to perceiving and understanding target objects. Most manipulation systems install two kinds of sensors. One kind is global sensors which play the role of human eyes. The other kind is local sensors which play the role of tactile sensation. Take the WillowGarage PR2 robot[WillowGarage, 2012] for example. Before performing manipulation

tasks, the PR2 robot firstly perceive positions and geometric information of target objects with global sensors on its head. Then, when stretching out its hand for grasping, the PR2 robot exerts forces by using tactile sensors inside its gripping hand. The most important problems that relate to these sensors are their precisions and costs. Low-cost perception devices like KINECT[Microsoft, 2012][1] could have as much as 50mm errors while precise perception devices like a laser ranger could cost thousands of dollars or even overtake the cost of a robotic hand itself. Researchers are still devoting themselves to struggling with the difficulties caused by sensing devices. Two popular solutions, namely reducing the number of necessary sensors and improving the performance of manipulation under various uncertainties take up most of nowadays researches.

Mechanics relates to mathematical and physical analysis of manipulation. It includes but is not limited to form or force closure (see Chapter 7 of [Mason, 2001]). Given the geometric information of target objects, mechanics study calculates the formations that should be shaped to manipulate the target objects and calculates the forces that should be exerted to operate the target objects. The difficulty from mechanics is not independent. It is deeply coupled with mechanism and sensing. An under-actuated mechanism suffers from kinematics and thus changes analysis of mechanics. A noisy perception device offers uncertain surface normals and thus changes conditions of force or form closures. Lots of state-of-the-art works discuss how to optimize mechanics against mechanism and sensing.

None of the three difficulties are independent. They interplay each other and further improve the difficulties of robotic manipulation. Our group design a distributed end-effector to challenge these three difficulties. This distributed end-effector is a background work of my caging topic and I will briefly introduce this it in the next section.

1.2 The Distributed End-effector

Our group proposes a distributed end-effector to challenge the difficulties of mechanism, sensing and mechanics in robotic manipulation. During the design of this end-effector we try to avoid explicit contact between fingers and target object as well as try to reduce the number of motors as much as possible. Fig.1.1 illustrates our concept.

Novelty of this end-effector lies in the following four aspects. (1) Generality: As can be seen from Fig.1.1, our distributed end-effector is installed to a base robotic arm to transport target objects from place to place. The target objects could have any rigid shapes, from commodity packages to cups and plates. In this way, the distributed end-effector is more general than traditional robotic palletizers. (2) Conciseness: We install only one x-y-θ actuator to the end-effector to lower its cost. This actuator will attach, actuate and detach each of those four "hanging" fingers sequentially and each finger can be actuated "distributedly". (3) Fully distributed control: Each finger has some permanent magnets installed to its top and each finger is connected to the "palm" of the end-effector by the magnetic forces exerted

[1]News of KINECT2 has been released recently. KINECT2 is Time-of-Flight based depth sensor. It is more precise as well as low cost.

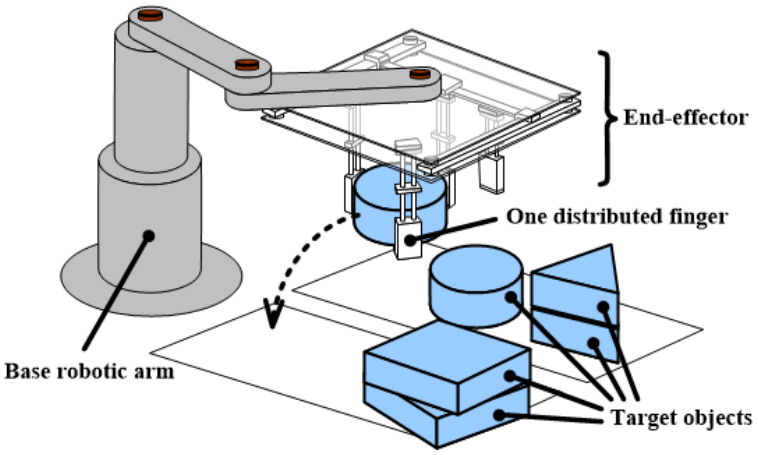


Figure 1.1: The concept of our distributed end-effector.

by those permanent magnets. Connection by magnets offer the freedom that when a finger is attached by the x-y-θ, it can be dragged and rotated freely in the "palm" plane. Each finger is controlled fully distributedly. (4) Prismatic finger body and nails: Each finger has a prismatic body and an inserting nail. The prismatic bodies help to enlength fingers without taking too much space while the inserting nails help to support objects during picking-up and transporting procedure. After perceiving the shape of a target object, the fingers stretch out their bodies to the bottom of the object and insert their nails underneath the object to avoid direct contact. In this way, we no longer need to consider about object materials or analyze contact frictions as long as the links of fingers constrain target objects and the nails have enough power to support object mass. Fig.1.2 shows in detail the mechanism of our x-y-θ actuator and a transporting procedure of the end-effector.

Along with these novelties, we encounter several problems like how to configure motors of the x-y-θ actuator, how to set permanent magnets and how to set the shape and strength of finger nails, etc. These problems relate heavily to mechanical design. However, besides these mechanical problems, we have an essential issue which relates to grasp synthesis. Say, given the shape of an object, how many fingers do we need and what kind of finger formation should we choose to constrain target objects? This problem looks like a traditional grasping problem, or more exactly looks like a form-closure problem since we do not consider frictions explicitly. Unfortunately, it is different. Form closure requires the equilibrium of wrenches. In contrast, the fingers in our case do not explicitly exert forces and we cannot build equations base on the convexity of form or force closure hulls. This problem is pure geometric. It is actually a different kind of closure which relates to caging.

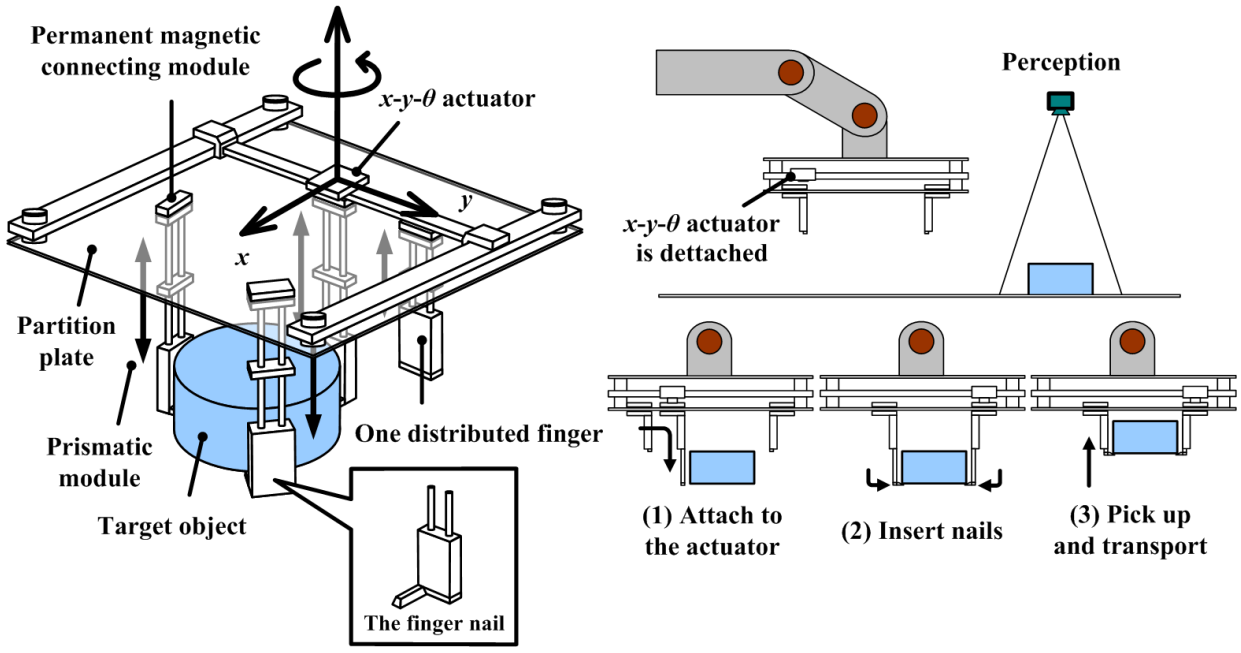


Figure 1.2: Mechanism of the x-y-θ actuator and a transporting procedure.

1.3 Controlling the End-effector by Caging

The first formal proposal of a caging problem is from Kuperberg's paper[Kuperberg, 1990] in 1990. In this paper, Kuperberg proposed the following question.

> Let P be a polygon in the plane, and let C be a set of n points in the complement of the interior of P. The points capture P if P cannot be moved arbitrarily far from its original position without at least one point of C penetrating the interior of P. Design an algorithm for finding a set of capturing points for P.

Kuperberg is a mathematician and he is mathematically strict in this proposal. Assume that any planar objects can be approximated by polygons, we can have a more general form. Inputs to the general form is the geometric shape or boundary clouds of a planar object. Outputs of it is a formation of capturing points that cages the geometric shape so that it can never go to infinity. The grasp closure issue we encountered in the end-effector is a variation of this general form. Actually, our issue is more difficult due to the limitation of hardware and uncertainty from engineering noises. Fig.1.3 illustrates the similarities and differences between our issue and Kuperberg's caging problem. The upper part of Fig.1.3 expresses Kuperberg's caging problem. It shows a caging formation which cages a circular object. In this case, the object is caged and cannot escape from the cage formed by the point finger formation. The lower part of Fig.1.3 expresses the caging issue of our end-effector. In this case, we need to consider extra problems like how many fingers should we employ to reduce

mechanical costs and how to deal with various uncertainty due to perception and control noises.

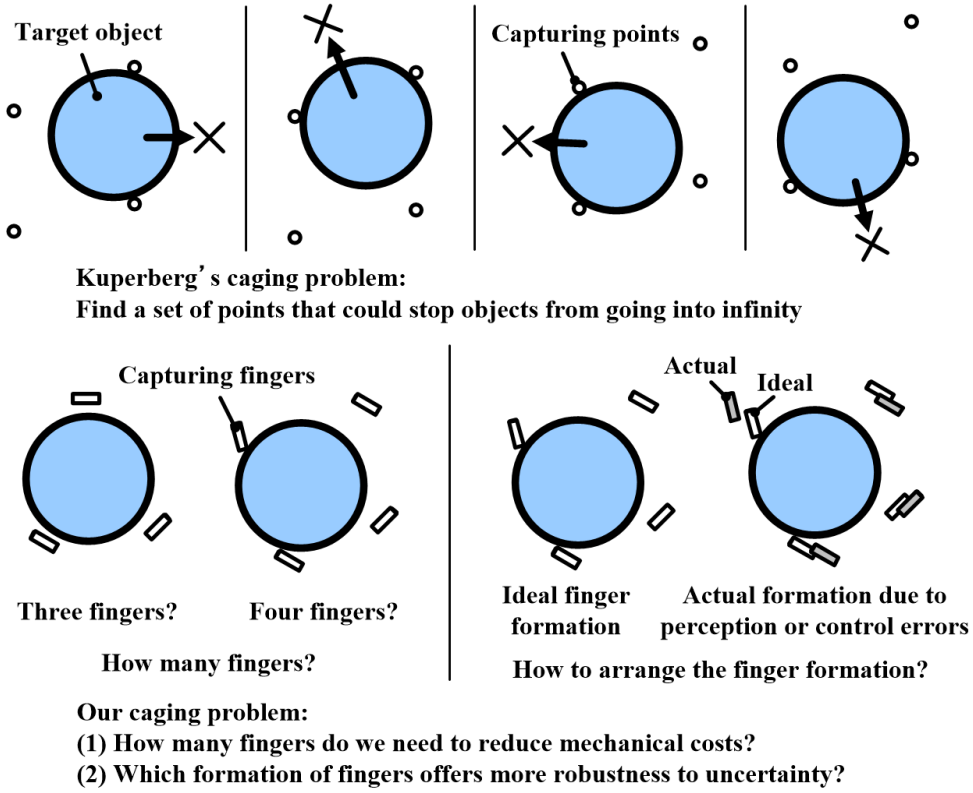


Figure 1.3: Similarities and differences between our issue and Kuperberg's caging problem.

Our issue has the same input as Kuperberg's proposal, namely the geometric shape of a planar object. It also has the same output, namely a formation of capturing finger positions that cages the target object[2]. Despite the similarities, our issue inherently suffers from engineering uncertainty. We not only need to consider whether the finger formation can capture or cage a target object, but also need to consider uncertainty caused by noises from the perception device and noises from control. The uncertainty makes our issue more difficult. Fig.1.4 shows in detail how Kuperberg's caging problem changes in the presence of perception and control noises.

When noises appear during perception procedure, the perceived object boundary could be dramatically different from its groundtruth shape. The difference may become even more dramatic after certain post-processing procedures. The upper part of Fig.1.4(a) illustrates

[2]Readers may notice that in our case finger have shapes. Surely finger shapes can simplify caging. For instance, our end-effector has rectangular finger shape and it can cage target objects more easily than point fingers. Nevertheless, let us temporarily take all fingers as point fingers. This is a sound assumption. Shaped fingers are super sets of point fingers. If an object can be caged by point fingers, it will sure to be caged by shaped ones.

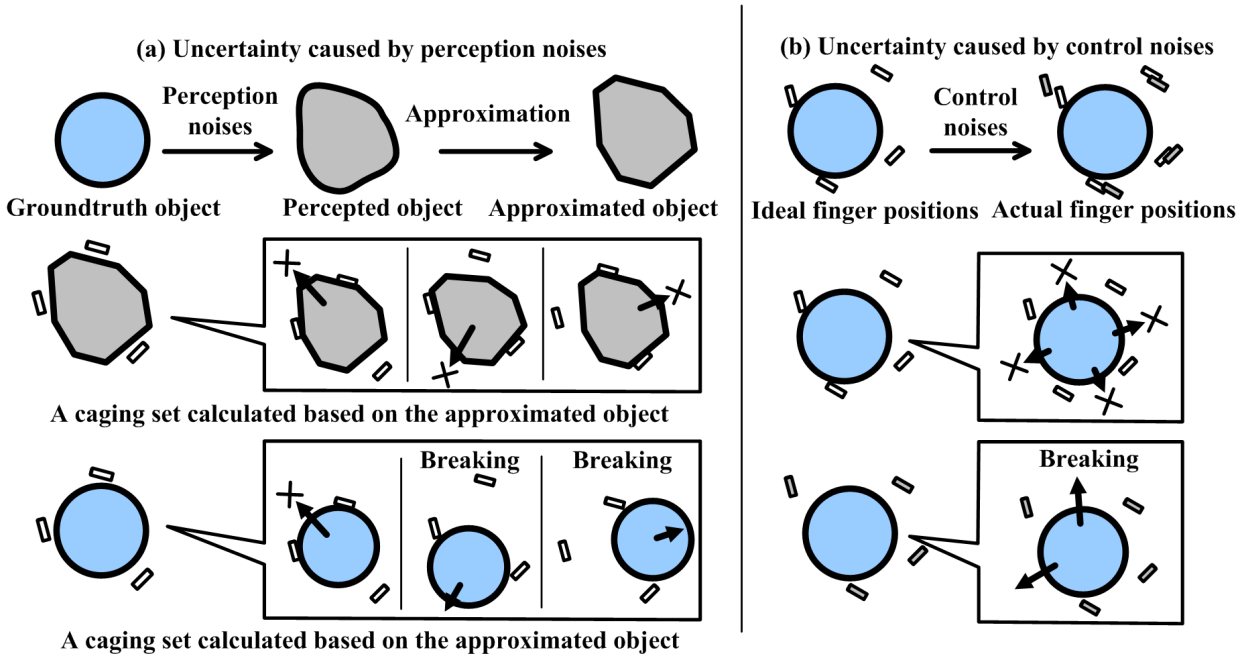


Figure 1.4: Our issue inherently suffers from uncertainty caused by engineering noises.

the perception noises and distortions in the shape. In this illustration, a groundtruth circular object becomes an irregular polygon after noisy perception and approximation. These noisy perception and approximation cause a fatal failure that the caging set calculated based the approximated polygon cannot cage the groundtruth circular object. The lower part of Fig.1.4(a) illustrates the fatal failures and caging breaking.

The same failure happens when control noises appears. Accumulation of noises from motor encoders, belt gears and permanent magnetic connecting modules could degenerate control and drive fingers to unexpected positions. Even though we can find a formation of finger positions to cage a target object, the caging may break due to noises during finger control and actuation. The upper part of Fig.1.4(b) illustrates the degenerated control and wrongly actuated fingers. These wrongly actuated fingers may either squash the target object or result into caging breakings. The lower part of Fig.1.4(b) illustrates a failure case (caging breaking).

Moreover, besides the noisy uncertainty we also need to take into account some other factors like length of the finger nails. In one word, our issue is a similar but complicated version of Kuperberg's caging problem. However, **there is no perfect solution to Kuperberg's caging problem, far from ours**. Consequently, I dive into the following research topic.

> How can we deal with the caging problem and apply it to our end-effector? Or more generally, how can we deal with the caging problem and apply it to robotic manipulation?

This thesis originates from this topic. The caging problem can indeed be divided into two sub-problems. The first one corresponds to Kuperberg's proposal, namely finding a set of caging formations or simply finding a caging set by solving the **caging test problem**. The second one corresponds to our newly encountered issue, namely finding a caging formation that is most robust to uncertainties or simply the **caging optimization** problem. I will generally call the **caging test problem** and the **caging optimization problem** by using **the caging problems** in the context.

In the problem of **caging test**, we need to develop a **caging test** algorithm that can test whether a specific finger formation cages the target object. We will need to find all the caging formations that can pass the caging test algorithm. In the problem of **caging optimization**, we need to develop an algorithm by using robust caging, namely find a measurement, evaluate the robustness of a caging formation with the measurement and picking out a caging formation that has satisfying robustness. Fig.1.5 illustrates the relationship of those caging problems.

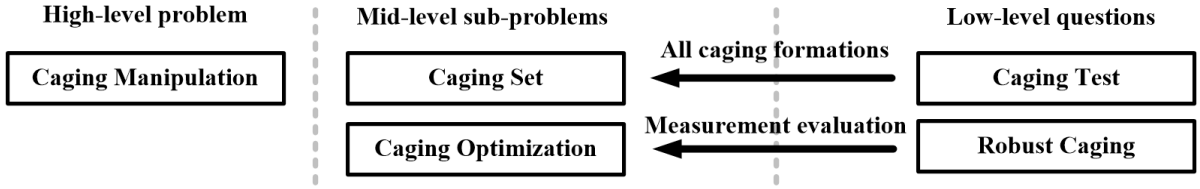


Figure 1.5: The caging problems of this thesis.

This thesis explores the caging problems. It summarizes my study in caging algorithms and presents some applications based on those study. The title of the thesis is named "caging planning" following "grasp planning". It borrows some ideas from path planning literature to solve the caging problems. I may not say the caging problems has been perfectly solved, but I believe my work can contribute to researches in related fields.

1.4 Organization of the Thesis

This thesis is composed of eight chapters. Besides the introduction chapter and the conclusion chapter, the remaining six chapters can be divided into four parts. They are,

Part I, Basic Concepts of Caging. This part introduces the basic concepts of caging and its recent development. It is not only a literature review but also includes some of my proposals that fill up the gap between caging and traditional robotic grasping research. My contribution in this part is the demonstration of both traditional grasping concepts and caging in the **configuration space of target object,** $\mathcal{C}^{\mathsf{obj}}$. I visualize the relationship between caging and traditional research in grasping with $\mathcal{C}^{\mathsf{obj}}$.

This Part I includes Chapter 2.

Part II, Caging in $\mathcal{C}^{\mathsf{obj}}$ and Its Applications. This part discusses in detail of how to deal with the caging problem in $\mathcal{C}^{\mathsf{obj}}$ and presents some applications based on my discussion. $\mathcal{C}^{\mathsf{obj}}$ can be seen as a tool to make caging analysis easier. By employing $\mathcal{C}^{\mathsf{obj}}$ we can review the caging problems from a different viewpoint and solve them intuitively.

Specifically, in this part, I firstly revisit, improve and implement a work by Prof. Jeff Erickson[Erickson et al., 2003][Erickson et al., 2007]. Erickson's idea is quite smart and my implementation improves his idea. However, both Erickson's idea and my implementation are limited to three-finger hands and known finger positions. How to push though those limitations and extend my implementation to general cases become a key issue. I propose an algorithm to solve this issue by fixing fingers alternatively and further reduce the computational complexity of that algorithm by decomposing caging into **translational caging** and **rotational constraints**. The performance of my algorithm are demonstrated and evaluated by the robustness of finger formations with WEBOT simulation software.

Applications in this part includes the distributed end-effector and a multi-robot cooperative transportation system. The applications work well with 2D convex objects. However, convexity is an inherent limitation and the algorithms in $\mathcal{C}^{\mathsf{obj}}$ can only work with 2D convex objects. Objects with concave boundaries invalidate the algorithms easily. This drawback motivates me to explore into another tool, say, **configuration space of finger formation, $\mathcal{C}^{\mathsf{frm}}$**.

This Part II includes Chapter 3 and Chapter 4.

Part III, Caging in $\mathcal{C}^{\mathsf{frm}}$ and Its Applications. This part analysizes the caging problems in $\mathcal{C}^{\mathsf{frm}}$. Like $\mathcal{C}^{\mathsf{obj}}$, $\mathcal{C}^{\mathsf{frm}}$ can also be seen as a tool to make easier caging analysis. The motivation that drive me to this tool is from two aspects. For one thing, I hope to make the caging algorithm work with any 2D shape, not only objects with either convex boundaries, but also concave boundaries, 1-order or high-order boundaries. For the other, I hope to make the caging algorithm complete as well as rapid. This is difficult in $\mathcal{C}^{\mathsf{obj}}$ since the complete algorithm in $\mathcal{C}^{\mathsf{obj}}$ may have a time cost as much as order nine. Therefore, approximation of the complete algorithm is employed in $\mathcal{C}^{\mathsf{obj}}$ to by a combination of **translational caging** and **rotational constraints**. However, the combination is not from strict mathematically analysis and lacks completeness. In $\mathcal{C}^{\mathsf{obj}}$, completeness and rapidness are reciprocal. High completeness implies low rapidness while high rapidness implies low completeness. Therefore, I choose to change to $\mathcal{C}^{\mathsf{frm}}$. Making both complete and rapid algorithms to the caging problems is the second motivation that drives me to $\mathcal{C}^{\mathsf{frm}}$.

In this part III, I will firstly discuss in detail why to change from $\mathcal{C}^{\mathsf{obj}}$ to $\mathcal{C}^{\mathsf{frm}}$. Employing $\mathcal{C}^{\mathsf{obj}}$ instead of $\mathcal{C}^{\mathsf{frm}}$ changes the center of my algorithm from target objects to fingers. We no longer need to bother with specific object shape features like concavity and convexity. This change exactly caters the expectation in the first aspect of my motivation. Then, I introduce the **space mapping** idea which caters the expectation in the second aspect of my motivation. *Raw space mapping* and especially its faster version, the *improved space mapping*, make it possible to update the whole space of $\mathcal{C}^{\mathsf{frm}}$. In other words, the algorithm is complete. At the same time, we can quickly find the **caging sets** in the updated $\mathcal{C}^{\mathsf{frm}}$ and locate an optimized caging formation. In other words, the algorithm is rapid.

My complete and rapid algorithm in this part is applied to the design and implementation of a gripping hand. The algorithm plays important roles in both design and implementation procedures. During design, the $\mathcal{C}^{\mathsf{frm}}$ algorithm is employed to simplify and evaluate deign models. During implementation, the $\mathcal{C}^{\mathsf{frm}}$ algorithm is employed to control the hand to cage and grasp objects. The design and implementation procedure could demonstrate advantages of caging.

This Part III includes Chapter 5 and Chapter 6.

Part IV, In-depth the Relationship Between $\mathcal{C}^{\mathsf{obj}}$ and $\mathcal{C}^{\mathsf{frm}}$. $\mathcal{C}^{\mathsf{frm}}$ is sometimes a more powerful tool comparing with $\mathcal{C}^{\mathsf{obj}}$. However, it is unwise to discuss which is better. Both algorithms in $\mathcal{C}^{\mathsf{obj}}$ and $\mathcal{C}^{\mathsf{frm}}$ have their advantages and disadvantages. The fourth part of the thesis proves that at different orientations $\mathcal{C}^{\mathsf{frm}}$ is the linear transformation of $\mathcal{C}^{\mathsf{obj}}$. Consequently, the metrics used in $\mathcal{C}^{\mathsf{frm}}$ and $\mathcal{C}^{\mathsf{obj}}$ are different. Both the two tools and their correspondent algorithms have reasons to exist. They therefore should be treated equally.

Actually, the two tools and their algorithms correspond to different solutions of **geometric modeling**. The algorithm in $\mathcal{C}^{\mathsf{obj}}$ uses **wireframe modeling** while the algorithm in $\mathcal{C}^{\mathsf{frm}}$ uses **solid modeling**. Both modeling technology plays important roles in **geometric modeling** and either algorithms in $\mathcal{C}^{\mathsf{obj}}$ and $\mathcal{C}^{\mathsf{frm}}$ should exist. I treat them equally and compile them into Part II and Part III of this thesis.

In real world, the algorithms in $\mathcal{C}^{\mathsf{obj}}$ and $\mathcal{C}^{\mathsf{frm}}$ should be chosen according to mechanical structure of robots and tasks. If all capture points are distributed and target objects are convex (like the distributed end-effector and multi-robot cooperative transportation), it is wise to do caging planning with the algorithms in $\mathcal{C}^{\mathsf{obj}}$. If capture points can be represented by certain formations or target objects have various shapes (like the gripping hand), it is wise to do caging planning with the algorithms in $\mathcal{C}^{\mathsf{frm}}$.

This Part IV discusses these in-depth relationships. It includes Chapter 7.

The last chapter, Chapter 8, concludes the thesis. It firstly summarizes the whole thesis, especially the algorithms in $\mathcal{C}^{\mathsf{obj}}$ and $\mathcal{C}^{\mathsf{frm}}$. Then, this chapter makes clear the contributions. In the third sub-section of this chapter, I discuss about some future directions in algorithms and applications aspects respectively. In the aspect of algorithms, future works could be the discussion of 2.5D/3D objects and the discussion of how to pre-define representative finger formations. In the aspect of applications, future works could be the deployment onto macro/nano manipulation and in-hand re-grasping systems.

Finally, Fig.1.6 shows all the works and their relationships. The sub-figures in Fig.1.6 are representatives of those works. We will see their details throughout the remaining contents.

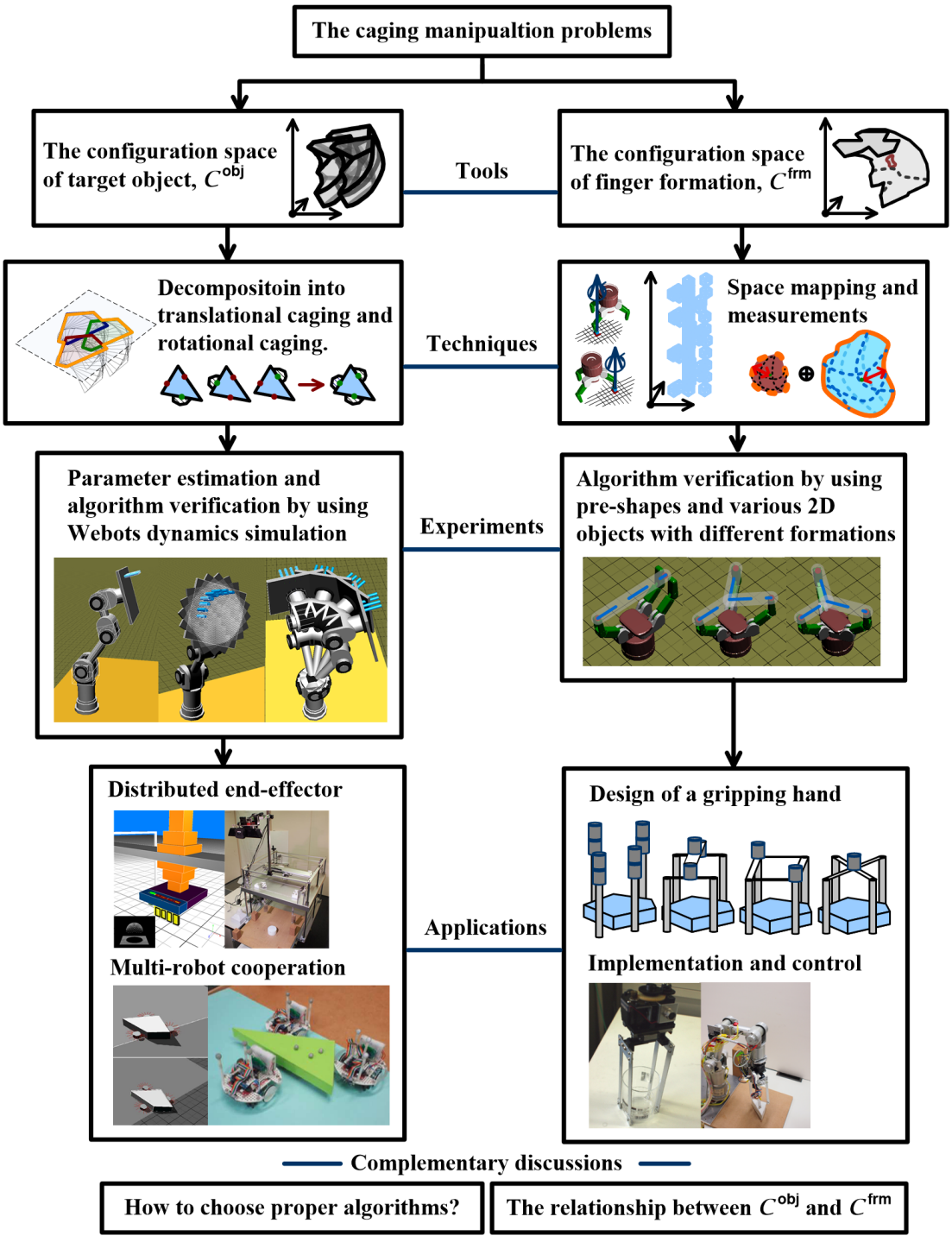


Figure 1.6: Organization of the works included in this thesis.

Part I

Basic Concepts of Caging

Chapter 2

Caging is the General Form of Grasping

2.1 Related Research Topics in Grasping

Before going deeper into caging, let us review some related researches and concepts in grasping. They are (1) force and form closure (2) immobilization and (3) grasping optimization.

2.1.1 Force closure and form closure

The force closure problem is one of the most fundamental problems in grasping. Basically, force closure describes a state in **wrench space**. Detailed reviews and discussions of the force closure problem can be found in Nguyen's publications[Nguyen, 1986a][Nguyen, 1986b]. I will not repeat those mathematically deductions here but would like to visualize the concept with figures. Like its name, **force closure means the wrench vectors exerted by fingers enclose the origin point of wrench space**. Or namely, the origin point of **wrench space** is enclosed by a convex hull which is spanned by wrenches exerted by fingers. Fig.2.1 visualizes this concept with two examples.

There are two points to explain about this figure. The first one is the forces exerted by those point fingers. Each point finger can exert not only normal forces along surface normals of target objects, but also friction forces along tangential directions. The synthesis of normal forces and friction forces is in a region. The areas in the middle of those orange, green and purple segments in the center part of Fig.2.1 illustrate this kind of regions. Since one finger can exert forces in a region, it is possible to ensure force closure with only two fingers. The first example of Fig.2.1 demonstrates this case.

The second point is **wrench space**. A wrench is a force plus a torque. Therefore, a wrench is a six dimensional vector shown in expression (2.1).

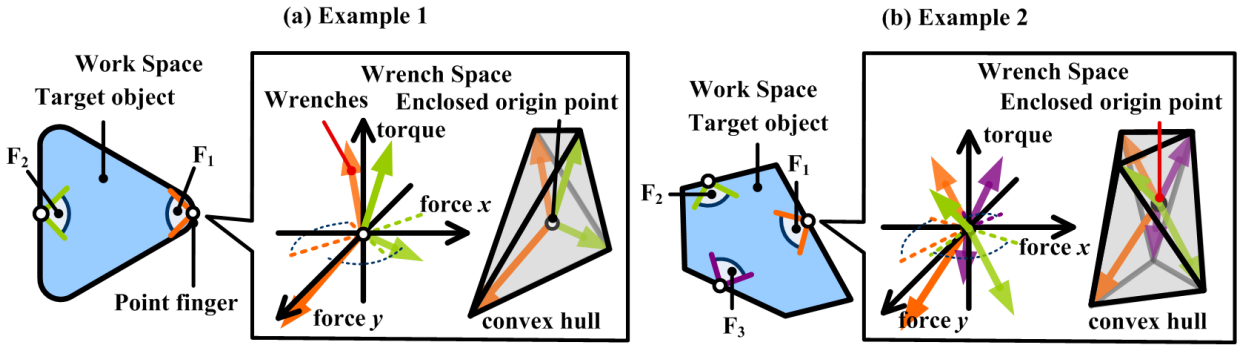


Figure 2.1: Two examples of force closure.

$$\boldsymbol{w_i} = \begin{pmatrix} \boldsymbol{F_i} \\ \boldsymbol{\tau_i} \end{pmatrix} = \begin{pmatrix} (F_{i_x}, F_{i_y}, F_{i_z})' \\ (\tau_{i_x}, \tau_{i_y}, \tau_{i_z})' \end{pmatrix} = \begin{pmatrix} \boldsymbol{F_i} \\ \boldsymbol{r_i} \times \boldsymbol{F_i} \end{pmatrix} = \begin{pmatrix} (F_{i_x}, F_{i_y}, F_{i_z})' \\ (r_{i_x}, r_{i_y}, r_{i_z}) \times (F_{i_x}, F_{i_y}, F_{i_z})' \end{pmatrix} \tag{2.1}$$

Here $\boldsymbol{F_i}$ is the force exerted by point fingers. According to foregoing explanation, this force is the synthesis of normal force and friction force. $\boldsymbol{r_i}$ is a position vector which indicates the relative position between the force and rotational center. All the six dimensional wrenches $\boldsymbol{w_i}$ constitute a six dimensional vector space named **wrench space**. In 2D case, F_{i_z} is always zero and $\boldsymbol{w_i}$ is always a three dimensional vector $\boldsymbol{w_i} = (F_{i_x}, F_{i_y}, \tau_{i_z})$. Therefore, the **wrench space** of a 2D object has three dimensions. The contents in the frames of Fig.2.1 illustrate the 3D **wrench spaces**.

The wrenches exerted by those point fingers in Fig.2.1 are rendered with the same colors as their work space correspondence. **These wrenches span convex hulls which enclose their origin points. Enclosing the origin points indicates that the fingers in both examples of Fig.2.1 form force closures.**

Force closure ensures grasping. However, it is impractical since materials and frictions of target objects are difficult to be perceived and modelled. Therefore, researchers usually discusses force closure without considering frictions. That is the concept of form closure.

Form closure was first proposed by Reuleaux in the year 1875(see [Bicchi, 1995] for a detailed review of those historical work). When referring to form closure, researchers have a common assumption that the point fingers cannot exert friction force. This assumption means that the $\boldsymbol{w_i} = (F_{i_x}, F_{i_y}, \tau_{i_z})$ in the **wrench space** of a 2D object becomes $\boldsymbol{w_i} = (N_{i_x}, N_{i_y}, \tau_{i_z})$. Namely the synthesized force $\boldsymbol{F_i}$ is reduced into the simple normal force $\boldsymbol{N_i} = (N_{i_x}, N_{i_y}, 0)$. The assumption is quite practical since researchers no longer need to consider about frictions to ensure form-closure grasp.

However, form closure causes new problems. It requires a lot more fingers [Mishra et al., 1987] and it is not applicable to circular objects. Reader may compare Fig.2.1 and Fig.2.2 for example. The same point fingers and the same target object ensure force closures in Fig.2.1(a), however, they cannot ensure form closures in Fig.2.2(a). Without

friction, the wrenches exerted by the point fingers cannot span convex hulls to enclose the origin points. [Mishra et al., 1987] proved that (1) if the DoF (Degree of Freedom) of an object is n_{dof}, we need at least $n_{\text{dof}} + 1$ point fingers to ensure form closure and (2) if the object is a circle, we can never ensure form closure with point fingers. For a 2D object whose $n_{\text{dof}} = 3$, we need at least 4 point fingers to ensure form closure. Therefore, the point fingers in Fig.2.2 are insufficient and they cannot ensure form closures.

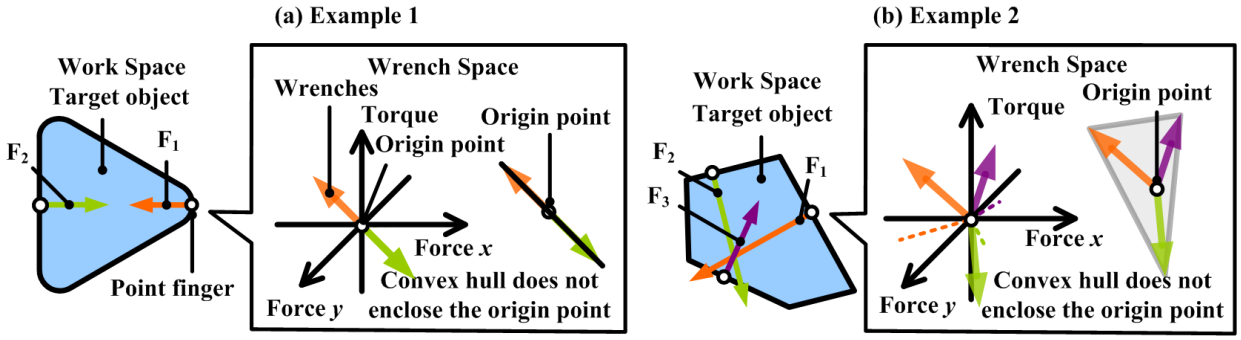


Figure 2.2: Two examples of non-form closure.

Mishra's theory seems to be contradictory to our common sense. Intuitively, people hold an feeling that the second example of Fig.2.2 can be successfully grasped by those point fingers. However, Mishra's theory shows that it would fail. Is there anything wrong with form closure? Elon Rimon tries to explain the contradiction by considering surface curvature of objects. His theory starts from [Ponce et al., 1995], [Rimon and Burdick, 1996] and [Rimon and Blake, 1996]. It becomes mature in [Rimon and Burdick, 1998a] and [Rimon and Burdick, 1998b]. The work is extended to 3D objects in [Rimon, 2001]. According to Rimon's theory, the contradiction between form closure and people's common sense was caused by surface curvatures of target objects. Take Fig.2.3 for example. In traditional definition of form closure, none of the cases in Fig.2.3 are form closure. Rimon improves the traditional definition by considering surface curvatures and introduces the concept of **2nd order form closure**. In Rimon's theory, the traditional form closure is named 1st order form closure since it only considers surface normals while his new closure is named 2nd order form closure since it further takes the derivative of surface normals, namely the curvatures of object surfaces into account. 2nd order form closure considers surface curvatures of target objects and therefore dissolves the contradiction. The surface curvatures at contact points in Fig.2.3(a) go outwards the contact tangent lines. They neither ensure 1st order nor ensure 2nd order form closure. The surface curvatures of the other cases in Fig.2.3(b) either run parallel to or go inwards contact tangent lines. Although they are not 1st order form closures. They fulfills Rimon's 2nd order definition and therefore can ensure successful grasp of target objects.

Besides **2nd order form closure**, another explanation of the contradiction is **infinitesimal motions**[Czyzowicz et al., 1999][van der Stappen, 2005][Cheong et al., 2006]. In force

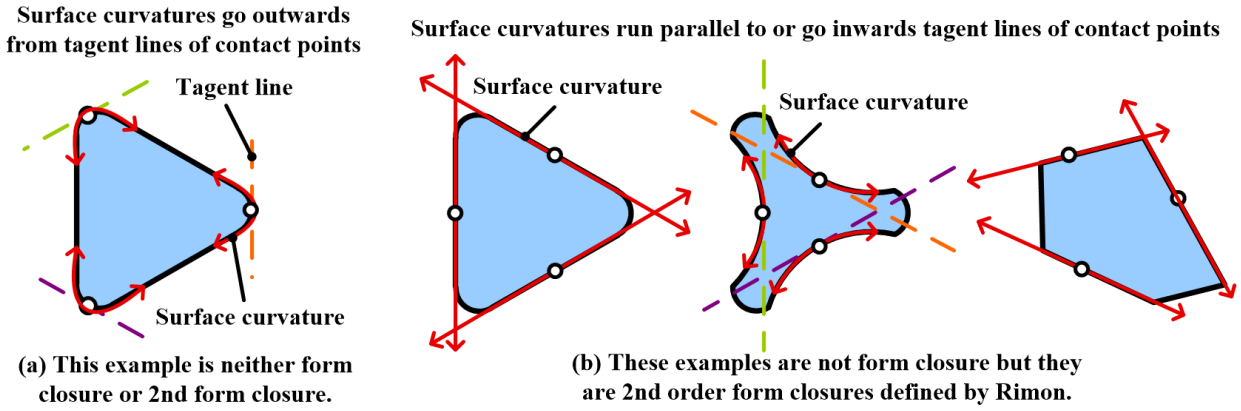


Figure 2.3: 2nd order form closure and surface curvatures.

closure, three fingers are enough to block both **finite motions** and **infinitesimal motions** since friction forces exist at every contacts. However, in form closure, three fingers can only block **finite motions** and target objects may oscillate infinitesimally due to the lack of friction. We may need at least 4 fingers to block both **finite motions** and **infinitesimal motions** and to ensure form closure. This conclusion corresponds to Mishra's $n_{\mathsf{dof}}+1$ theory.

Fig.2.4 illustrates the meanings of **finite motions** and **infinitesimal motions**. It is too strong for fingers to block both finite and **infinitesimal motions** in form closure and that's the reason why the contradiction between form closure and our common sense exists.

Rimon's 2nd order form closure can be recognized as an attempt to deprive **infinitesimal motions** from the definition of 1st order form closure. That is, even if there exists **infinitesimal motions**, a target object could be grasped. However, Rimon's analysis is quite complicated. A more concise way to deprive **infinitesimal motions** is to define closures from another viewpoint, namely immobilization.

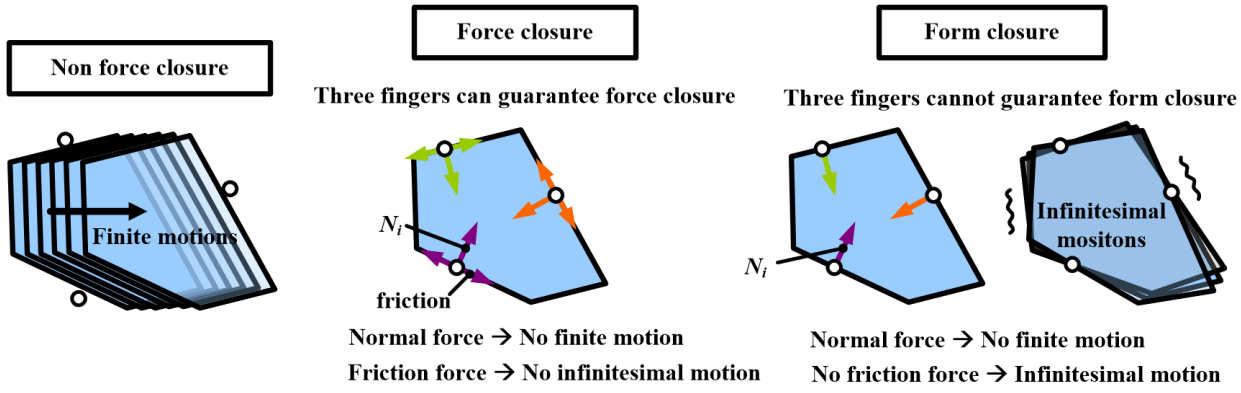


Figure 2.4: Form closure requires more fingers to constrain infinitesimal motions.

2.1.2 Immobilization

Either the definition of force or form closures relate heavily to forces and wrench space. In contrast, immobilization do not analyze forces. Jurek Czyzowicz[Czyzowicz et al., 1999] defines immobilization as following.

> The set of points I is said to immobilize a planar shape P if any rigid motion of P in the plane forces at least one point of I to penetrate the interior of P.

Comparing with force and form closure which require enclosing the origin point of wrench space, the definition of immobilization is pure geometric. It does not result into problems of surface curvatures or infinitesimal motions. **All we need to ensure is that the target object, in its configuration space, is at a fixed single configuration.** Instead of Jurek's definition, let us view the immobilization problem in **configuration space**. More exactly, we should call it **the configuration space of target object** and use symbol $\mathcal{C}^{\text{obj}}$ to indicate it. $\mathcal{C}^{\text{obj}}$ was originally a $\mathbb{R}^2 \times \mathcal{S}$ topology space. This $\mathbb{R}^2 \times \mathcal{S}$ topology space is homeomorphic to the three-dimension Euclidean space $\mathbb{R}^3$ (see Chapter 3 of reference [Choset et al., 2005] for more formal definition). We can therefore use $\mathbb{R}^3$ rather than $\mathbb{R}^2 \times \mathcal{S}$ to represent $\mathcal{C}^{\text{obj}}$ and to simplify the deductions. The first two dimension of $\mathcal{C}^{\text{obj}}$ denote the position of target objects and the third dimension of $\mathcal{C}^{\text{obj}}$ denotes the orientation of target objects. A point in $\mathcal{C}^{\text{obj}}$ is called a configuration and it corresponds the position and orientation of the target object in work space. Fig.2.5 shows the correspondences between a target object in work space and a configuration in $\mathcal{C}^{\text{obj}}$.

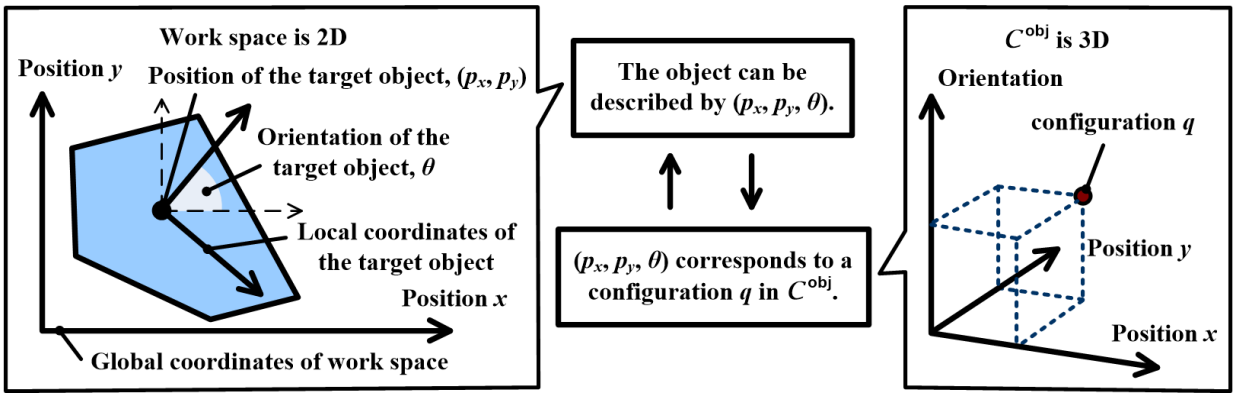


Figure 2.5: Correspondence between an object in work space and a configuration in $\mathcal{C}^{\text{obj}}$.

A point finger in work space corresponds to an **obstacle** in $\mathcal{C}^{\text{obj}}$. Take Fig.2.6 for instance. In the upper-left figure of Fig.2.6, the target object is at a configuration $\boldsymbol{q}$ and it does not collide with the point finger. In this case, the $\boldsymbol{q}$ is free. In the upper-right figure of Fig.2.6, as the target object moves along the red arrow, the target object and the point finger would collide with each other. In that case, the position and orientation of the target object, or namely the configuration $\boldsymbol{q}$ of the target object, becomes obstructed. A target object

could collide with a point finger at many different $\boldsymbol{q}$s. All the $\boldsymbol{q}$s constitute a compact set (sub-space) in $\mathcal{C}^{\text{obj}}$ and we name this set the configuration obstacle. This is shown in the middle row of Fig.2.6. One configuration obstacle is decided by the position of a point finger. Therefore, we say one configuration obstacle corresponds to one point finger. If there are three point fingers like the lower part of Fig.2.6, there will be three correspondent obstacles in $\mathcal{C}^{\text{obj}}$.

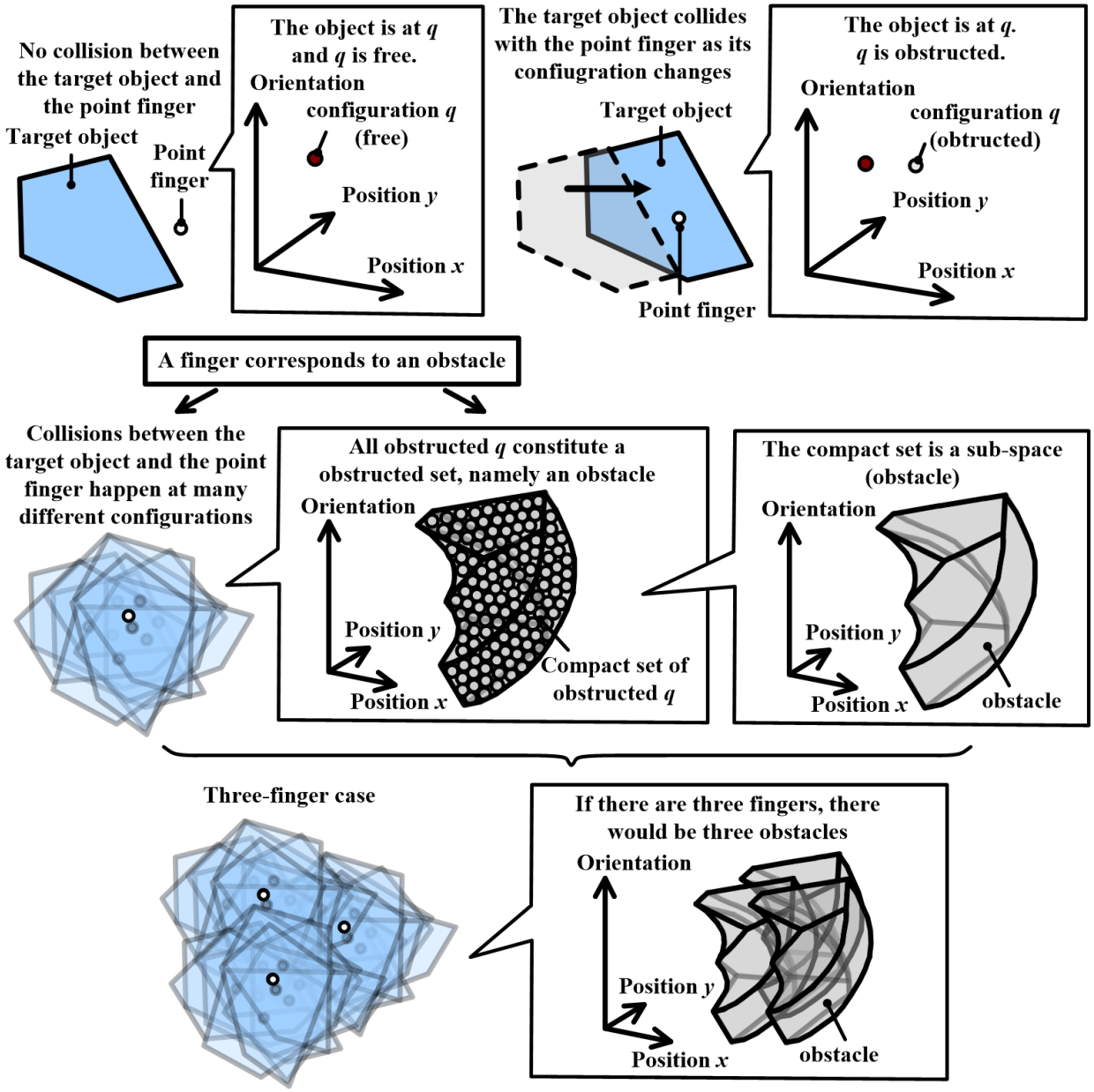


Figure 2.6: Correspondence between a finger in work space and an obstacle in $\mathcal{C}^{\text{obj}}$.

When a target object's configuration is fixed to a single point by the obstacles, we can say the object is immobilized. This is the same as Jurek's definition. When the configuration of a target object is fixed to a single point, any changes in object configuration, or namely any rigid motion of the target object, would cause the configuration to be obstructed by obstacles or namely would force at least one point finger to penetrate into the interior of the target object. An example of immobilization is the fingers and target object in the right part of Fig.2.1. Although those fingers cannot ensure form closure, they immobilize the target object. Fig.2.7 shows the $\mathcal{C}^{\text{obj}}$ of this immobilization example. In this figure, the obstacles are rendered with wire-frames to better illustrate the fixed single configuration. The three wire-framed obstacles compactly enclose an fixed single configuration and therefore the three fingers immobilize the target object. Readers may refer to the two images in the center part of this figure for better comprehension. In the center part, both a whole view which shows all $\mathcal{C}^{\text{obj}}$ and a sliced view which shows only the $\mathcal{C}^{\text{obj}}$ at an orientation θ are rendered. The target object cannot change its position or rotation as any changes in configuration would be obstructed by obstacles. The right part of Fig.2.7 demonstrates two obstructions.

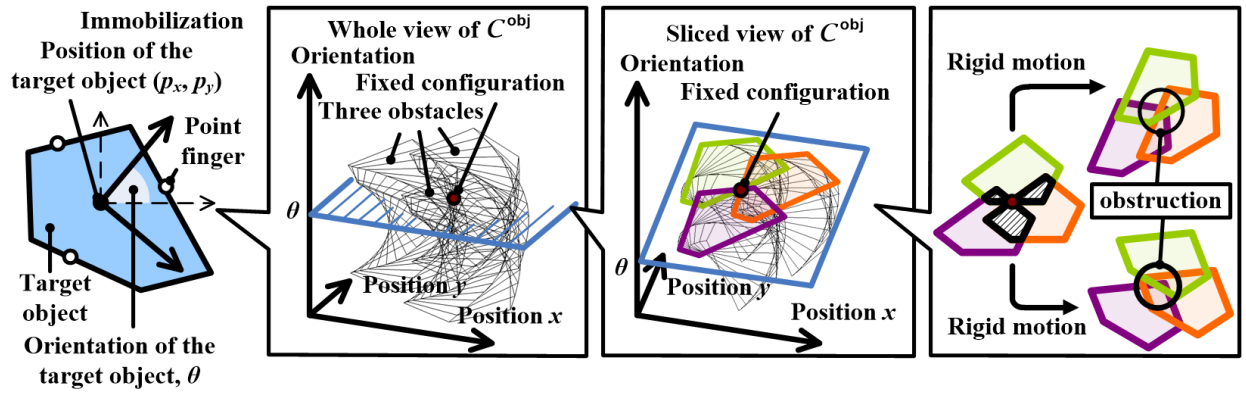


Figure 2.7: Immobilization means a fixed single object configuration.

Immobilization unifies 1st order and 2nd order form closure. 2nd order form closure belongs to immobilization while 1st order form closure is not immobilization. It makes the theory concise. However, we need to pay attention to that immobilization is different from fixture design. In fixture design, researchers expect to fix objects with force closure rather than immobilization. This is because immobilization inherits the problem of infinitesimal motions from form closure, it cannot "firmly" fix objects.

Jurek proved that four points are always sufficient to immobilize any shape. Comparing with force and form closures, immobilization is quite pragmatic. For one thing, it requires few fingers. For another, it involves no wrench analysis. However, that's not the ultimate theory. Caging, the major topic of this thesis, is more general comparing with immobilization. We will see their relationship later in this chapter. Before that, let us review another related topic, namely grasping optimization.

2.1.3 Grasping optimization

The third research related the topic of this thesis is grasping optimization. In accordance with the foregoing introduction, we divide grasping optimization into two aspects. The first one is optimization of force and form closures and the second one is optimization of immobilization.

2.1.3.1 Optimization of force closures and form closures

Basically, following the definition of force closure and form closure, traditional works tend to define a measurement and perform optimization of grasping in the **wrench space**. This **basic measurement** is usually **the radius of Largest Inscribed Sphere (LIS) of the convex hull spanned by wrenches**. Fig.2.8(a) illustrates this basic **basic measurement**.

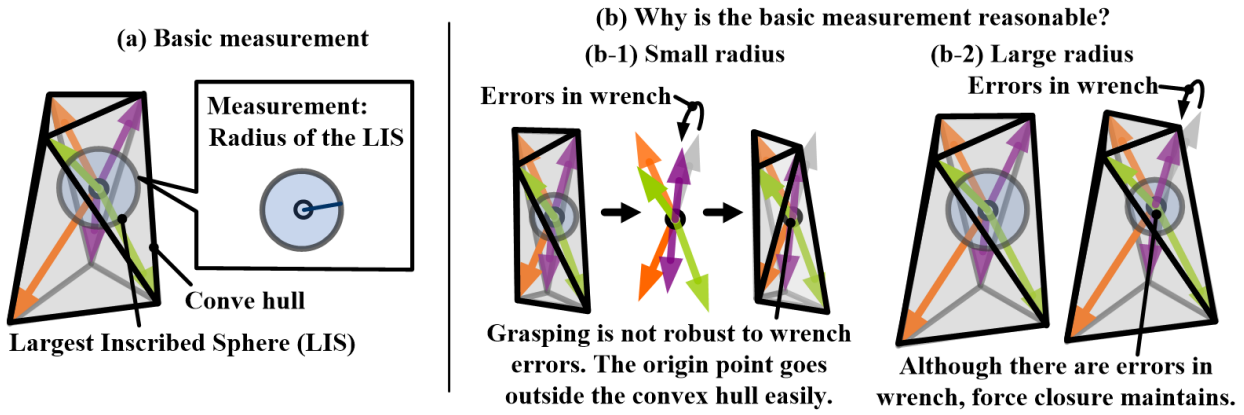


Figure 2.8: Grasping optimization is to find a convex hull whose LIS has largest radius.

The **basic measurement** is reasonable since the larger the radius of LIS is, the more robust a force closure or form closure would be. Fig.2.8(b) illustrates this reason. When the object is unknown, engineering errors, or namely noises from perception devices and control would easily cause errors in normal forces and friction forces. These errors in forces, in the **wrench space**, cause changes in wrench vectors. The middle part of Fig.2.8(b-1) demonstrates this kind of errors and changes. If the radius of LIS is too small, like the case in Fig.2.8(b-1), the origin point of **wrench space** may go outside the convex hull easily and force closure breaks. Therefore, Fig.2.8(b-1) is not robust to errors and changes. Optimization is needed. Comparing with Fig.2.8(b-1), Fig.2.8(b-2) has a larger radius of LIS and it is more robust. Interested readers may refer to the following three references for more details. The first one is reference [Mirtich and Canny, 1994] which gives some intuitive examples that demonstrate the efficacy of **basic measurement** in work space. The second and third one are references [Liu et al., 2004] and [Niparnan et al., 2009] which give formal expressions to calculate **basic measurement** and discuss how to compute force closure and form closures efficiently. Fig.2.8 is based on force closure. Nevertheless, it is not limited to

force closure. It can be applied to form closures in the same way. The **basic measurement** is a fundamental evaluation criteria in **wrench space** optimization.

Of course, there are lots of variations which make the **basic measurement** more pragmatic. For instance, sometimes we need to take into account the given external forces exerted on the target objects. Gravity force is one example of the given external forces. Some other times we need to consider some unavailable contact areas on the surface of target objects. The cutting edge of a knife is one example of those unavailable areas. Grasping the cutting edge does harm to fingers and hands. Fig.2.9 illustrates these pragmatic variations. When there are given external forces, as shown in Fig.2.9(b), the convex hull deforms with respect to the given forces. It results into different radius of LIS and different optimization results. Readers may see [Watanabe and Yoshikawa, 2007] for examples of optimization on given external forces. When there are certain unavailable areas, as shown in Fig.2.9(c), the convex hull deforms with respect to the changes of contacts. It also results into different radius of LIS and different optimization result. Readers may see [Li and Sastry, 1988] for examples of unavailable areas.

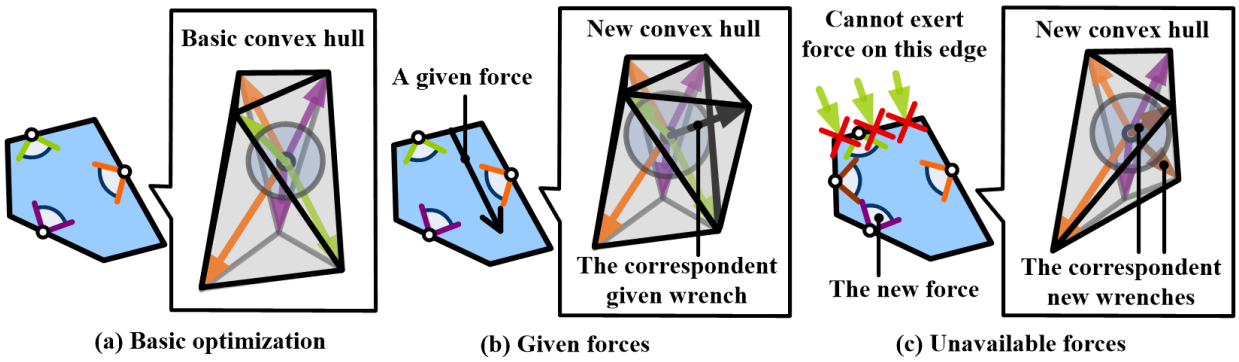


Figure 2.9: Some variations of grasping optimization.

Pure optimization of force closures and form closures is becoming less popular and there are few new publications on this topic in recent two or three years. What's more, it is not directly related to our topic in this thesis. Therefore we won't repeatedly review the historic works of this area. I refer readers to Watanabe's paper [Watanabe and Yoshikawa, 2007] for a good review and classification of literature. Interested readers may also refer to Eris Chinellato's work[Chinellato, 2002][Chinellato et al., 2003][Chinellato, 2008] to better understand different measurements and their performance. They may also refer to fixture design[Wallack and Canny, 1996][Wallack, 1996][Ponce, 1996] to see some practical applications. Readers may find that some figures in [Wallack and Canny, 1996][Wallack, 1996] [Ponce, 1996] are like translational and rotational decomposition in Chapter 3. The similarity, from another view, demonstrates the relationship between grasping closure and caging. We will see the details in Section 2.2.

2.1.3.2 Optimization of immobilization

Like optimization of force closures and form closures, we can also perform Immobilization optimization. Optimization of force closures and form closures are performed in **wrench space** and likewise optimization of immobilization could be performed in $\mathcal{C}^{\text{obj}}$. However, discussions on **immobilization optimization in** $\mathcal{C}^{\text{obj}}$ cannot be found in popular researches. I therefore develop a new measurement for immobilization optimization. I will briefly introduce the new measurement for immobilization in this sub-section. It is an essential connection between grasping and caging.

In previous sections we have seen that immobilization means **the target object, in** $\mathcal{C}^{\text{obj}}$**, is at a fixed single configuration**. In another word, when a target object is immobilized, its correspondent $\mathcal{C}^{\text{obj}}$ can be divided into three components. The first one is the fixed single configuration, which represents the position and orientation of the target object. When the target object is at this configuration, it is immobilized. The second one is the collection of obstacles, which compactly surrounds the single configuration. When the target object is at a configuration of this component, it collides with fingers. The third one is the other "free" space. When the target object is at a configuration of the third component, it can move freely. Considering the three components, we can **evaluate the quality of an immobilization grasp by measuring the minimum distance between the first component, namely the fixed single configuration, and the third component, namely the "free" space.** This measurement is quite intuitive and it is demonstrated in Fig.2.10.

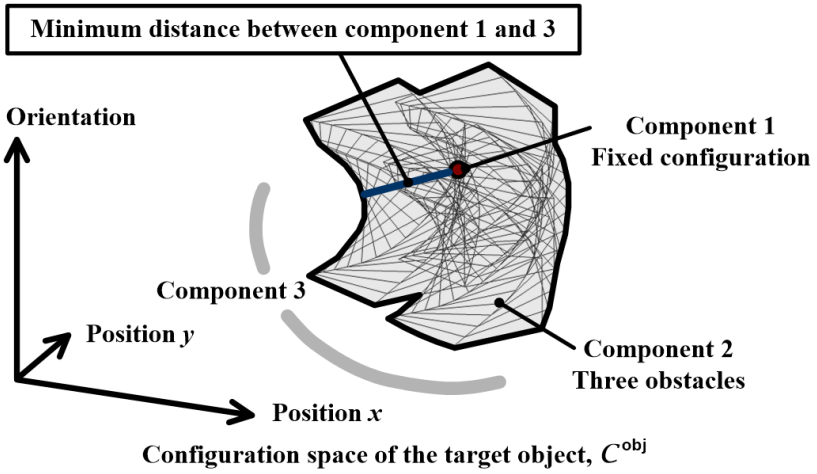


Figure 2.10: An intuitive measurement of immobilization optimization in $\mathcal{C}^{\text{obj}}$.

The same target object and the same fingers as Fig.2.7 is used in this figure. Since the target object is **convex** and the fingers are point fingers, component 2 is composed of three obstacles and it separates component 1 and component 2 compactly. There's no inner-holes in the obstacles. Note that some other objects, such as concave objects or objects with inner

holes, may break the compactness of these components and their measurement won't be as rigid as Fig.2.10.

The measurement in Fig.2.10 can indicate the quality of immobilization. However, it is a rough measurement since it did not take into account that objects are rigid. For rigid objects, it is unnecessary to measure a single obstacle and only the intersections between obstacles plays essential roles. The intersections represents the "distance" between fingers in work space. **The larger an intersection is, the smaller the distance between its correspondent fingers would be.** Fig.2.11 shows the correspondence between the intersections in $\mathcal{C}^{\mathsf{obj}}$ and the inter-finger "distance"s in work space.

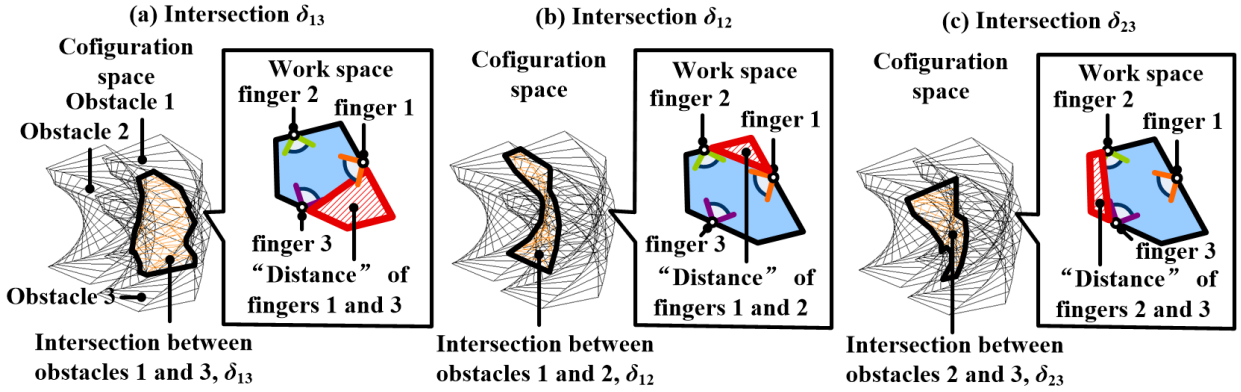


Figure 2.11: Correspondence between intersections of obstacles in $\mathcal{C}^{\mathsf{obj}}$ and inter-finger "distance"s in work space.

Consequently, we can have another measurement, namely **the minimum of intersections**. It is more advisable to use it for rigid objects. Nevertheless, this measurement is quite ambiguous. **The minimum of intersections** requires comparison between the "size"s of intersections. Unfortunately, it is difficult to define a measurement of the "size"s which is difficult to be defined. I rendered the "distance"s between fingers in Fig.2.11 with shadowed areas. Readers may refer to them to retrospect on this difficulty. How can we define a measurement to measure those shadow areas so that it can indicate the quality of immobilization? It remains an open question to immobilization optimization and relates intensively to **caging optimization**.

Caging optimization and immobilization optimization share lots of common backgrounds. They both need the measurement of distances in $\mathcal{C}^{\mathsf{frm}}$. However, I would like to emphasize the differences between them. In grasping optimization, fingers are always in contact with objects. This is not the case in caging. In caging, we would like to deal with engineering errors without considering contacts and forces. The robustness of caging is not to endure force errors, but to endure collisions between fingers and target objects or failures of constraining target objects caused by positioning errors. The difference makes caging optimization more complicated. I will revisit this problem and propose my solutions in relevant chapters later.

Comparing with grasping, caging is force-less and geometric. I am going to put them in the same context and show their relationship in the following texts.

2.2 The Relationship Between Grasping and Caging

Lots of related researches and concepts have been discussed in Section 2.1 and they can be summarized in the following words and figures.

Our discussion begins with force closure. Force closure requires considering both normal forces and friction forces. It is not practical since friction is difficult to model. Therefore, researchers come to the concept of form closure. Theoretical study showed that form closure requires as many as $n_{\text{dof}} + 1$ fingers which conflicts with our common sense. Researchers therefore propose two ways to conquer the contradiction. One is 2nd order form closure which takes surface curvatures of target objects into account. It is a bit complicated. The other is immobilization which divides object motions into finite motions and infinitesimal motions. Immobilization is concise and agrees with our common sense. Besides the basic concepts, researchers performed various attempts to optimize grasping. In force closure and form closure the quality measurement is mainly the radius of LIS. In immobilization, the quality measurement is mainly the minimum distance between the fixed single configuration and configurations in the "free" space.

Caging is the extension of immobilization. If we insert caging into the discussion, it should take an ensuing position after immobilization. Fig.2.12 summarizes our discussion in Section 2.1 with diagrams and shows the position of caging.

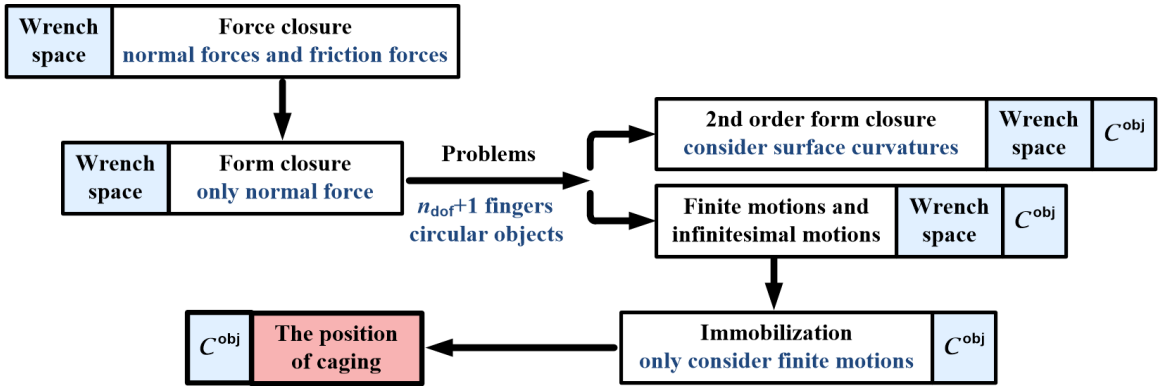


Figure 2.12: The position of caging and its relationship to grasping.

Note that in Fig.2.12, each concept has an accompanying blue frame besides it. The blue frame denotes in which space is the concept developed. Force closure and form closure are developed in **wrench space**. In contrast, immobilization and caging are developed in $\mathcal{C}^{\text{obj}}$. Force closure and form closure are essentially the analysis of forces. In contrast, immobilization and caging are essentially the analysis of geometry. In Fig.2.7 we have seen

the $\mathcal{C}^{\text{obj}}$ of an immobilization grasp. Here I will go on to show the $\mathcal{C}^{\text{obj}}$ of caged objects in Fig.2.13.

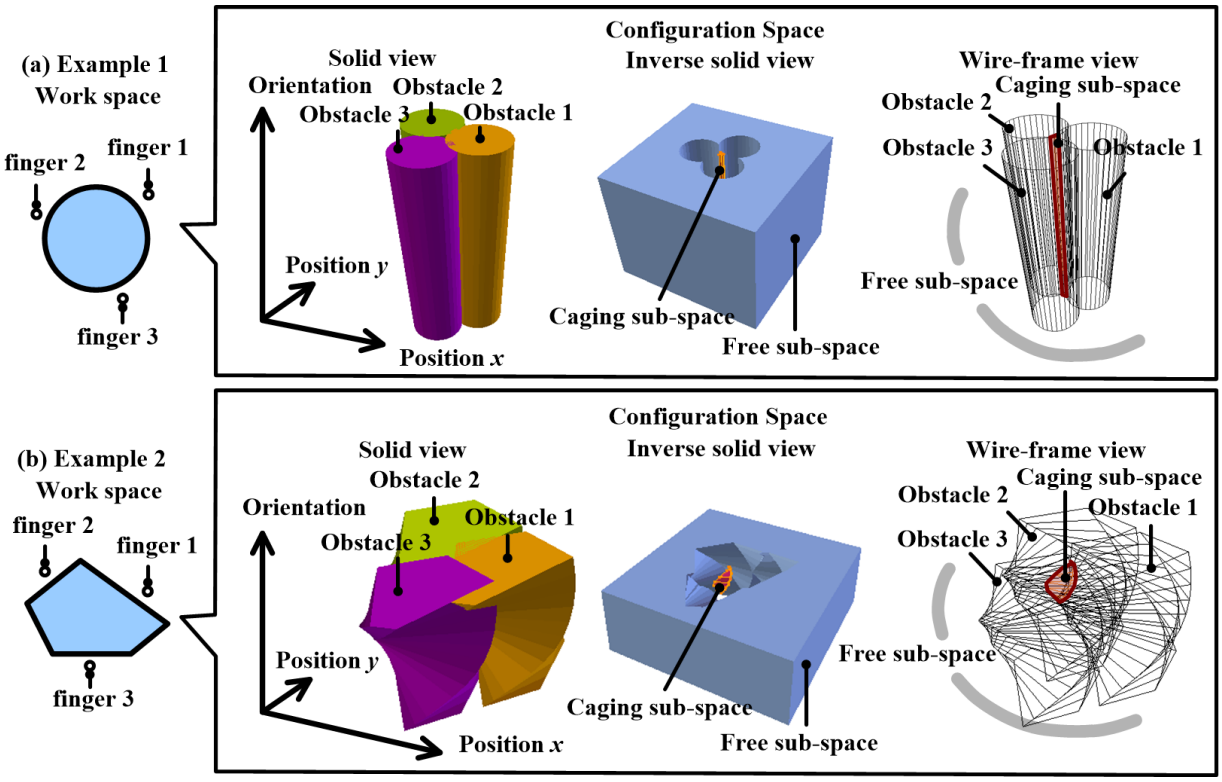


Figure 2.13: The $\mathcal{C}^{\text{obj}}$ of two caged objects.

Two examples of caging are given in Fig.2.13. The first one is a circular object which was used in Fig.1.3. The second one is a polygon which was used in Fig.2.7. According to Kuperberg's definition (see Section 1.3), caging, in work space, means the target object is constrained by fingers. The objects in Fig.2.13 are constrained by fingers in work space and they are caged. In $\mathcal{C}^{\text{obj}}$, caging means the configuration of a target object is constrained in a **caging sub-space**. In side this **caging sub-space**, the target object may move freely. However, it cannot go outside without being penetrated by fingers. The right part of Fig.2.13 illustrates the constraints in $\mathcal{C}^{\text{obj}}$. I made three different drawings to better present it. The first drawing is a solid view of obstacles. This solid view show each obstacle in with solid colors. Like the definition in immobilization (see Fig.2.6), each obstacle corresponds to a finger and there are totally three obstacles in each example. The second drawing is the inverse of the first drawing. In the second drawing, the complementary space of obstacles are rendered. When a target object is caged, the complementary space of obstacles can be divided into two sub-spaces. The first sub-space is a **free sub-space**, it is rendered with light blue in the second drawing of Fig.2.13. The configurations in this **free sub-space** can freely move into infinity without any penetration. The second sub-space is the caging sub-space.

If a target object is at a configuration inside this **caging sub-space**, the object is caged. The caging sub-space is emphasized with orange color in the second drawing. The third drawing is a wire-frame view of obstacles. The wire-frame view gives a more clear view that the caging sub-space is enclosed by obstacles and it is separated from **free sub-space**.

Now we can have the following conclusion that **caging in $\mathcal{C}^{obj}$ means the configuration of the target object is inside a caging sub-space. This sub-space is separated from the free sub-space. It could be either open or closed.** Recall our definition of immobilization. Immobilization means that **the target object, in its configuration space, is at a fixed single configuration.** The difference between caging and immobilization is caging implies a free sub-space while immobilization implies a single configuration. Caging is an extension of immobilization. Fig.2.14 demonstrates the relationship between caging and immobilization with the second example of Fig.2.13.

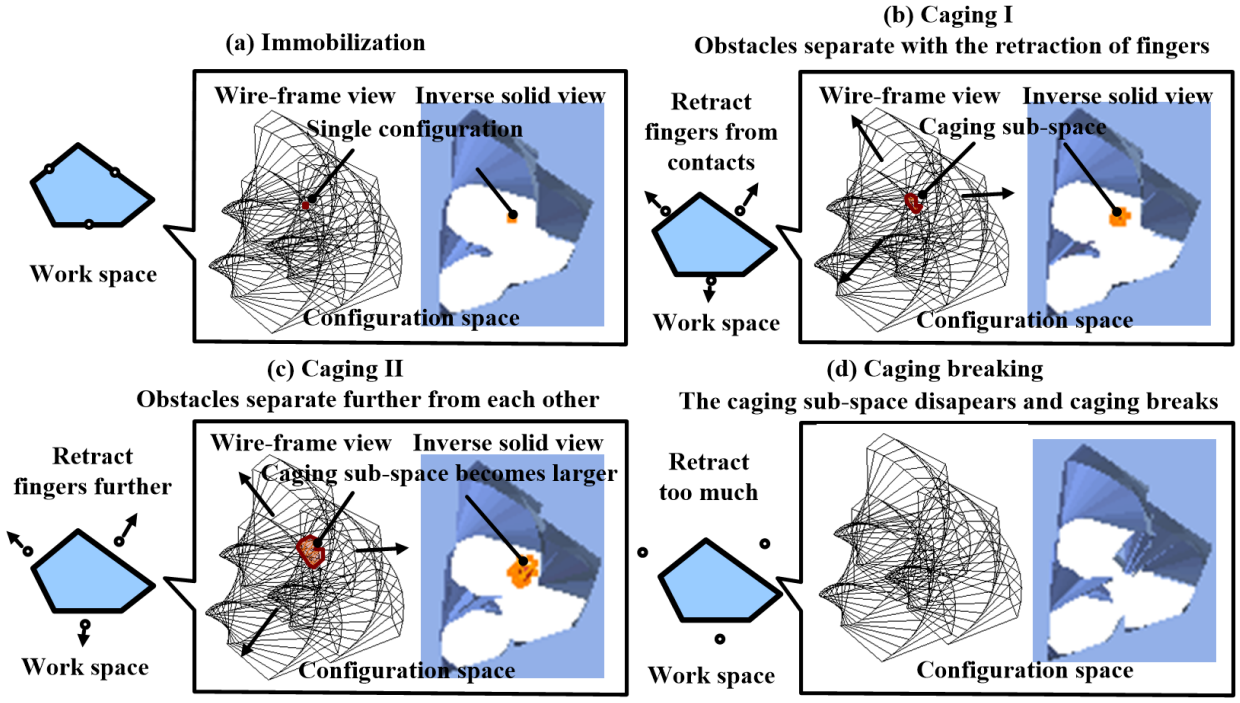


Figure 2.14: The relationship between immobilization, caging and caging breaking.

In figure Fig.2.14(a), the object is immobilized and its $\mathcal{C}^{obj}$ is the same as Fig.2.7. A fixed single configuration exists and it is separated from free spaces by obstacles. If we retract fingers back from the contact points, the single configuration will expand to a compact set, namely a caging sub-space. The immobilization grasping becomes caging. The further we retract fingers, the larger this caging sub-space would be. Fig.2.14(b) and Fig.2.14(c) shows two different cagings. The object in Fig.2.14(c) has higher freedom comparing with Fig.2.14(b) as its caging sub-space is larger. Fig.2.14 shows that if we retract fingers too far from the surface of objects, the caging sub-space may disappear. In that case, the

point fingers will never cage the target object and caging breaks. We can also see Fig.2.14 in inverse order. As we squeeze the fingers, the caging in Fig.2.14(c) and Fig.2.14(b) will finally degenerate into immobilization. The retracting and squeezing directions are important to ensure the continuity between caging and immobilization. We will discuss about that later.

Fig.2.14 demonstrates that caging is the extension of immobilization. However, the object in Fig.2.14 doesn't cover all cases. For example, when the target object is the concave object shown in Fig.2.15 and it is caged by a two-finger formation, no matter how we squeeze the two fingers, they will never immobilize the object. The final shape of the caging sub-space in this case will be an caging surface. That means there must be something else between caging and immobilization.

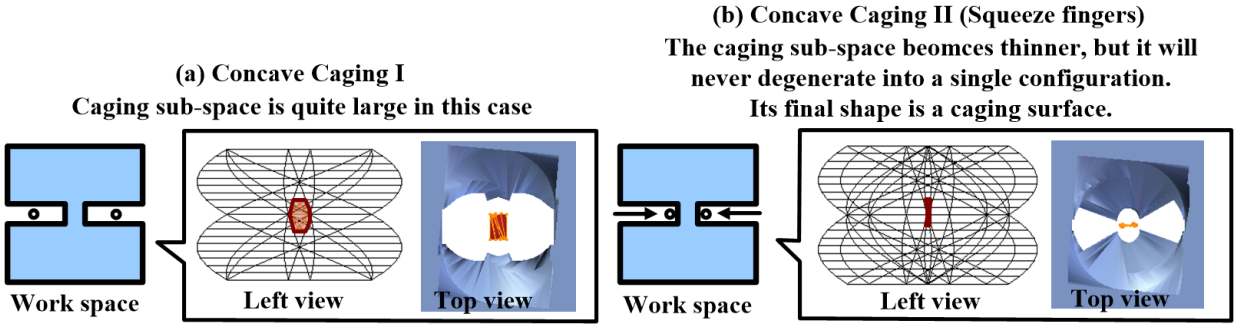


Figure 2.15: Gaps exist between traditional immobilization and caging.

I define this "something else" as **contacting caging**. Readers may compare the following lists for better comprehension.

immobilization	The target object, in its configuration sub-space, is at a fixed single configuration.
contacting caging	The target object, in its configuration sub-space, is on a caging curve and or a caging surface.
caging	The target object, in its configuration sub-space, is in a caging sub-space.

It is easy to get two conclusions from these concepts. (1) We can start from immobilization and move fingers to caging. But we cannot start from caging and move fingers to immobilization. Caging may end up at **contacting caging**. (2) When the target object is convex polygon, there is no **contact caging** and caging directly connects to immobilization. We can either move from immobilization to caging or move from caging to immobilization. That is the most ideal case.

Based on these conclusions, we can further solve a problem. That is, how many fingers are sufficient to cage any objects? In the first conclusion, I claimed that we can start from immobilization and move fingers to caging. That means, **the number of fingers that are**

sufficient to immobilize an object is the number of finger that are sufficient to cage an object. According to Jurek , it requires at least 3 to 4 fingers to immobilize an object in 2D space. When three edges of the object form a triangle, we need at least 3 fingers. Or else, we need at least 4 fingers. More generally, it requires $n_{\mathsf{dim}} + 1$ to $2n_{\mathsf{dim}}$ fingers to immobilize an object in n_{dim} space. (For example, in 3D space, we need at least 4 to 6 fingers to immobilize a 3D object.) Therefore, **the number of fingers that are sufficient to cage an object in n_{dim} space is the same as immobilization. It is $n_{\mathsf{dim}} + 1$ to $2n_{\mathsf{dim}}$.**

Comparing with form closure, the number of fingers that is sufficient to cage an object is much smaller. We need at least $n_{\mathsf{dof}} + 1$ fingers to grasp an object with form closure. A 2D object has 3 DoF and therefore 4 fingers are needed. A 3D object has 6 DoF and therefore 7 fingers are needed. In contrast, we need $n_{\mathsf{dim}} + 1 = 2 + 1 = 3$ to $2n_{\mathsf{dim}} = 2 \times 2 = 4$ fingers to sufficiently cage an object in 2D space and $n_{\mathsf{dim}} + 1 = 3 + 1 = 4$ to $2n_{\mathsf{dim}} = 2 \times 3 = 6$ fingers to sufficiently cage an object in 3D space. Note that this is the **sufficient** number. It is neither the least number nor the maximum number since there are lots of possibilities. When the target object is concave, 2 could be the least number of fingers that are required to cage an object. When the target object has inner holes, 1 could be the least number of fingers. When target objects are convex, 3 or 4 could be the least number of fingers. We will revisit this conclusion in later chapters.

2.3 State-of-the-art Works in Caging

We have connected caging to grasping and seen that caging is an extension of grasping. Connecting caging to traditional grasping research is one contribution of this thesis. To my best knowledge, no other researchers had ever discussed the connection of these concepts in $\mathcal{C}^{\mathsf{obj}}$. Actually, the development of caging is relatively slow due to its abstractness and high cost of computational resources. I will summarize state-of-the-art works in caging in this section. Note that I am not going to repeatedly trace back to those very old publications, but would like to discuss the recent development. Interested readers may refer to [Rodriguez, 2013] for a complete review of old publications.

Major contemporary researchers of caging in the robotics community include Elon Rimon, Andrew Blake, Attawith Sudsang, Vijay Kumar, Zhidong Wang, A. Frank van der Stappen, Yusuke Maeda and Alberto Rodriguez. They all have a series of publications on this topic. Some other researchers, like Jeff Erickson and David J. Cappelleri, also have some publications directly related to caging. Besides them, there are many other researchers who indirectly made great contributions. Let us review their works here.

The first paper that makes caging an independent research topic of robotics is written by Elon Rimon, [Rimon and Blake, 1996]. It is later extended into a journal paper [Rimon and Blake, 1999]. This work is limited to 2D objects and two-finger grippers. Some ensuing improvements of this work involve extension to three-finger grippers and application with visual sensors [Davidson and Blake, 1998a] [Davidson and Blake, 1998b]. The major idea used in these three works is to build a **contact space graph**. This idea is still in use

in the most recent publications of Rimon and his students [Allen et al., 2012]. The name of contact space comes from contacts. Each axis of the contact space corresponds to all the contacts between a finger and the surface of an object. The number of axes of a contact space is the number of fingers that contact with object surfaces. Therefore, the dimension of the contact space is the same as the number of fingers. This space suffers from the curse of dimensionality as finger number increases. That is one reason why the idea of **contact space graph** can only be applied to grippers with limited number of fingers.

The other researcher, Attawith Sudsang, begins his research of caging a little later than Rimon. His first debut is [Sudsang and Ponce, 1998]. In this work, he carries out a simulation on a 2D triangular object to demonstrate his contact-less grasping and manipulation idea. This work is based on a concept named Inescapable Configuration Space (ICS) proposed in [Sudsang et al., 1997] and [Sudsang et al., 2000]. Although this work is not explicitly named "caging", it is essentially the same idea. ICS can be recognized as another description of the **caging sub-space**. However, it is not as intuitive and complete as **caging sub-space**. This is because ICS is defined in work space, not configuration space. Its definition limited its further development. We can find some other similar concepts that are defined in work space and aim to describe **caging sub-space**. The attractive region proposed in [Qiao, 2001] and [Qiao, 2002] is such an concept. These concepts, from another view point, show the interests of robotics community in caging and the importance of choosing a proper mathematical tool. In [Sudsang et al., 1999], Sudsang implements his simulation in [Sudsang and Ponce, 1998] on real mobile robots. These two papers, [Sudsang and Ponce, 1998] and [Sudsang et al., 1999], start the research of performing caging transportation with multiple point robots. Sudsang claims that point fingers and point mobile robots are the same thing. Caging with point fingers shares the same principle as caging with point robots or multi-robot cooperative caging. Later in [Sudsang and Ponce, 2000], Sudsang proposes a basic solution which connects multi-robot cooperative caging and motion planning/obstacle avoidance in the presence of obstacles. Connecting caging to motion planning is quite practical and interesting. Unfortunately, I fail to find their ensuing implementation of this framework. In [Vongmasa and Sudsang, 2006], Sudsang and Vongmasa propose the concept of coverage diameters. This work gives a redundant solution to multi-robot cooperative caging. Although redundant, it is quite practical since caging can be ensured as long as the inter-robot distance is smaller than the coverage diameter. Their solution of coverage diameter is applicable to both convex and concave 2D objects. However, the coverage diameter results into redundant robots. Probably Sudsang realized the drawback of coverage diameter, he published another work [Suarod et al., 2007] which discusses about looser caging. The work [Suarod et al., 2007] is based on Zhidong Wang's proposals. We will review it later when discussing about Wang's works. These works of Sudsang concentrate on caging with point mobile robots. Besides multi-robot cooperative caging, Sudsang also published some works on caging with point fingers. In [Pipattanasomporn et al., 2008] and [Pipattanasomporn and Sudsang, 2011], Sudsang and Pipattanasomporn discusses two-finger squeezing caging. This work is extended to (1) a given formation of fingers and (2) both squeezing and stretching in [Pipattanasomporn et al., 2008].

This work decomposes objects into convex components. Its result is interesting. I am looking forward to their improvement in completeness and limitation of formations. Sudsang and his group are quite active in caging research and make lots of contributions. They also had the idea of using caging to deal with uncertainties [Pipattanasomporn and Sudsang, 2010]. The major difference between their work and mine is their concepts are built in work space. We can find that some of their publications implicitly imply the idea of **contact space graph** and **configuration space**. However, they fail to make explicit expression. Implicitly working in work space obscure their presentation and limits their development.

Zhidong Wang is the first researcher who explicitly expressed caging in $\mathcal{C}^{\mathsf{obj}}$. His work in caging concentrates on multi-robot cooperation. At the very beginning, Wang solves the multi-robot cooperative transportation by task allocation [Wang et al., 1999]. Task allocation requires to specify specific tasks to each robot during cooperative transportation. An impressive task allocation work can be found from [Cheng et al., 2008]. The requirements of task allocation limits its extensibility. I guess that's why Wang change to the idea of caging. Wang's early caging publications were [Wang and Kumar, 2002] and [Wang et al., 2003a]. In these two papers Wang together with Vijay Kumar proposes the concept of object closure and rendered it in $\mathcal{C}^{\mathsf{obj}}$. Object closure is exactly an alternative name of caging. Wang further implements his proposal with three mobile manipulators in [Wang et al., 2003b]. Each mobile manipulator in this work is simplified into a rectangular finger and consequently the implementation is the same as caging with three rectangular fingers. The implementation in [Wang et al., 2003b] controls a precomputed formation of the three mobile manipulators by maintaining certain offset margins from target objects. This formation control strategy is rough and encounters some problems. Therefore, Wang proposes a new control strategy in [Wang et al., 2004]. The control strategy in [Wang et al., 2004] mixes maximum margins and minimum margins of object closure and leader-follower formation control. The new control strategy is validated by using three circular robots and a concave object in [Wang et al., 2005]. These early works of Wang built up a solid basis of caging test. Given a formation of mobile robots and a target object, Wang's algorithm can tell whether the formation of robots could cage the target object[1]. He in [Wang et al., 2006] extends the caging test to a set of robot formations where he could not only test the caging of one formation but many formations in a set. The extended algorithm is named dynamic object closure. Dynamic object closure enables testing many formations of robots at different time intervals and enables robots to cage and transport target objects in real time. It can take the place of formation control. Wang proposes the real-time caging and transportation in [Wang et al., 2009]. The algorithm in this paper plans a path in configuration-time space to connect caging formations calculated by dynamic object closure at different time intervals. I am looking forward to Wang's implementation of the dynamic object closure on real robots. Wang's work is based on his discussion in $\mathcal{C}^{\mathsf{obj}}$. Some of my work borrow and improve his idea. I will refer to them when necessary.

In the same year as Wang and Kumar's publication [Wang and Kumar, 2002], some

[1]His test is not complete. But it is powerful to solve practical problems.

other researchers from Kumar's group published another work based on the idea of object closure [Pereira et al., 2002b][Pereira et al., 2002a]. This work is implemented with three car-like mobile robots and a triangle object with the same control strategy used in [Wang et al., 2003b]. It was later extended to a journal paper [Pereira et al., 2004]. The works of Pereira are not independent, they are more like a complementary branch of Wang. There are some other researches in Kumar's group that work caging. For example, [Fink et al., 2008]. Nevertheless, those works bias towards multi-robot control rather than geometric basis of caging. I am not going to discuss them here. Wang/Kumar's work explicitly discuss caging in $\mathcal{C}^{\mathsf{obj}}$. However, they concentrate too much on the application aspect of multi-robot cooperative transportation and fail to go further into the basic theory. The major difference between Wang/Kumar's work and mine can be concluded into the following three points. (1) They discuss the caging problems only in $\mathcal{C}^{\mathsf{obj}}$ while I explore different tools like $\mathcal{C}^{\mathsf{frm}}$. (2) They do not discuss optimization while I treat caging test and caging optimization as two parallel problems of caging. (3) They assume perfect object information and use redundant number of robots while I assume noisy perception and least number of capture points.

When Wang was trying to figure out a solution to caging test, he proposed the concept of CC-closure object. CC-closure object is **the Configuration obstacle of a Configuration obstacle**. This name is a little obscure but it do fully exploit the the configuration of configuration. It was interesting to find that Jeff Erickson independently developed the same concept in [Erickson et al., 2003] and [Erickson et al., 2007]. Erickon and Wang do not have any interaction with each other but they do proposed the same idea.

Erickson's idea was further developed by Vahedi and Stappen. Stappen's group have excellent background in theoretical grasping and computational geometry and consequently Vahedi's early work analyzes the caging problems in work space by geometric computation. He successfully described the relationship between immobilization and caging (This description is in work space. It is different from my contribution which is in $\mathcal{C}^{\mathsf{obj}}$). In [Vahedi and van der Stappen, 2006] and [Vahedi and van der Stappen, 2008c], Vahedi discusses the problem of caging with two fingers. His algorithm could both perform two-finger caging tests and report two-finger caging sets rapidly. We can find some relationship between these works and [Rimon and Blake, 1996], [Pipattanasomporn et al., 2008] and [Allen et al., 2012]. This is because the vertex-graspings are commonly recognized as the bounds where caging breaks. Vahedi's major contribution is the application of the caging breaking bounds to three-finger cases. In references [Vahedi and van der Stappen, 2007], [Vahedi and van der Stappen, 2008a], [Vahedi and van der Stappen, 2008b] and
[Vahedi and van der Stappen, 2009], Vahedi concentrates on three-finger caging and concentrates on the problem proposed by Erickson's, namely reporting the caging set of a third finger with two given ones or reporting the caging set of a third finger with the given distance between the other two fingers. Especially in [Vahedi and van der Stappen, 2009], Vahedi not only uses work space but also makes a program to show the critical patches in configuration space. Although Vahedi makes certain improvements in computational efficiency, I maintain that his work is essentially the same as Erickson. Vahedi's concept, for instance,

"vertex-grasping", "equilibrium grasping" and "critical patches" actually describe same thing as Erickson's "critical orientation". Nevertheless, Vahedi is the first researcher who purely works on caging theory and describes the relationship between caging and immobilization. He shows the difficulty of caging tests and finding caging sets and partially solves those problems. In his Ph.D thesis [Vahedi, 2009], Vahedi draws lots of important conclusions. For example, the Lemma 6.2.10 and Theorem 6.2.11 on page 68 and 69 show how to maintain caging when shrinking fingers. Many of my ideas in this thesis, like translational caging and accumulation, are borrowed from Erickson and Vahedi's work.

The early works of Maeda and Makita belong to the research field of non-prehensile manipulation [Maeda et al., 2004][Maeda and Makita, 2006]. Try comparing their works in grasping to those works in task allocation of multi-robot cooperative transportation, we can find that the non-prehensile manipulation works of Maeda and Makita are actually the same as multi-robot cooperation by using task allocation. The difference is they use multi-fingers instead of robots. It is very interesting that so many researchers who work with task allocation move to caging. The first caging work of Maeda and Makita is [Makita and Maeda, 2008]. In this paper, they measure the distances between fingers and evaluate with a target may go through those distances. Their solutions are intuitive and require redundant number of fingers. Later, a student from Maeda's group, Yokoi, published a multi-robot cooperation work by using not only mobile robots but also obstacles in the environment [Yokoi et al., 2009]. They take into account walls and transport objects along the walls. Maeda summarizes the work in both [Makita and Maeda, 2008] and [Yokoi et al., 2009] and implements with real robots in [Maeda et al., 2012]. He further discusses and installs some soft parts to rigid mobile robots and fingers in this paper. More recently, Maeda and Makita employ AR (Augmented Reality) markers to recognize shapes of 3D objects. AR markers are quite popular to retrieve shapes of target objects from pre-built databases. Employing AR markers make their caging work practical. However, the number of modeled shapes in the pre-built databases is limited. It is difficult to cover many objects. I in this thesis prefer real-time perception and modeling but I agree that with the help of modern database and machine learning, modeled shapes would bring bright future to robots. Maeda and Makita's work are much more practical comparing with previous works. They are, nevertheless, weak in the caging theories and all their implementations are based on [Makita and Maeda, 2008]. Details of Maeda and Makita's caging research are summarized in Makita's Ph.D thesis [Makita, 2010] (It is written in Japanese.).

Rodriguez is a graduate student of Prof. Matthew T. Mason. He proposes the idea of "from caging to grasping" [Rodriguez et al., 2011]. This idea is nearly the same as my "grasping by caging" proposal. It was interesting that we come to the same idea without any communication. This paper of Rodriguez is awarded the best student paper of Robotics, Science and Systems 2011 and is invited to be published in International Journal of Robotics Research [Rodriguez et al., 2012]. Rodriguez's work in caging is as theoretical as Vahedi. It contrasts significantly with Maeda and Makita's practical implementations. Rodriguez started his research in caging by studying the case of two fingers [Rodriguez and Mason, 2008], which is a well discussed topic of Rimon and Sudsang. The difference of Rodriguez's two-finger

research is he not only considers squeezing but also gives intensive discussions on stretching. He demonstrates that complete caging is a difficult problem as there are too many special cases and he maintains that "from caging to grasping" is essentially to find a F-caging function which could ensure continuous caging of target objects. Rodriguez's study in [Rodriguez and Mason, 2008] and [Rodriguez et al., 2011] is done in topology space of fingers. Using this space as the analyzing tool is quite abstract. He diverts the analyzing tool from topology space into $\mathcal{C}^{\mathsf{obj}}$ in [Rodriguez and Mason, 2012b]. I will discuss later in related chapters the difference between the topology space and $\mathcal{C}^{\mathsf{obj}}$. Rodriguez's most recent work is a review of caging research [Rodriguez, 2013]. His research on caging is theoretical and his review biases towards theoretical researches of caging too.

All the researches in discussed until now work on either fingers or mobile robots. The work of David J. Cappelleri brings something new to our vision. He employs caging in micro-manipulation [Cappelleri et al., 2011] to transport and assemble micro-parts. This paper is later extended to a journal version in [Cappelleri et al., 2012]. Although Cappelleri does not explicitly claim the "grasping by caging" concept, he actually makes use of it. In his publications, Cappelleri firstly cages a micro-object and then shrink the cage into force closure to transport or assemble it. Cappelleri's implementation is based a the redundant calculation like [Makita and Maeda, 2008]. From Makita and Cappelleri's work, we can find that there is a big gap between caging theories and pragmatic applications. Comparing with complicated caging theories, pragmatic applications prefer using simple and redundant algorithms. How to practically use the well developed theories and reduce the redundancy caused by simple algorithms is one challenge in front of us. Rodriguez tells us that complete caging theory involves too many unexpected cases and it is difficult to develop a complete algorithm that takes into account every aspect. Even if the target objects are 2D, the difficulty remains. Therefore, we should seek the balance between complete caging algorithms and pragmatic applications. I am going to discuss my solutions on caging in the next few chapters. In a certain degree, my solutions are complete and ready to be employed by practitioners.

In order to better illustrate the related works, I compile them chronically according to the relationship of the researchers and their ideas. Fig.2.16 shows the chronicle categorization. Each categorized box includes four items, namely the time period, the main researchers, the methodologies and the problems solved. Beside the related works, I attach my work in this thesis into this chronically categorization. Readers may compare my work and the other researchers to better understand the its position.

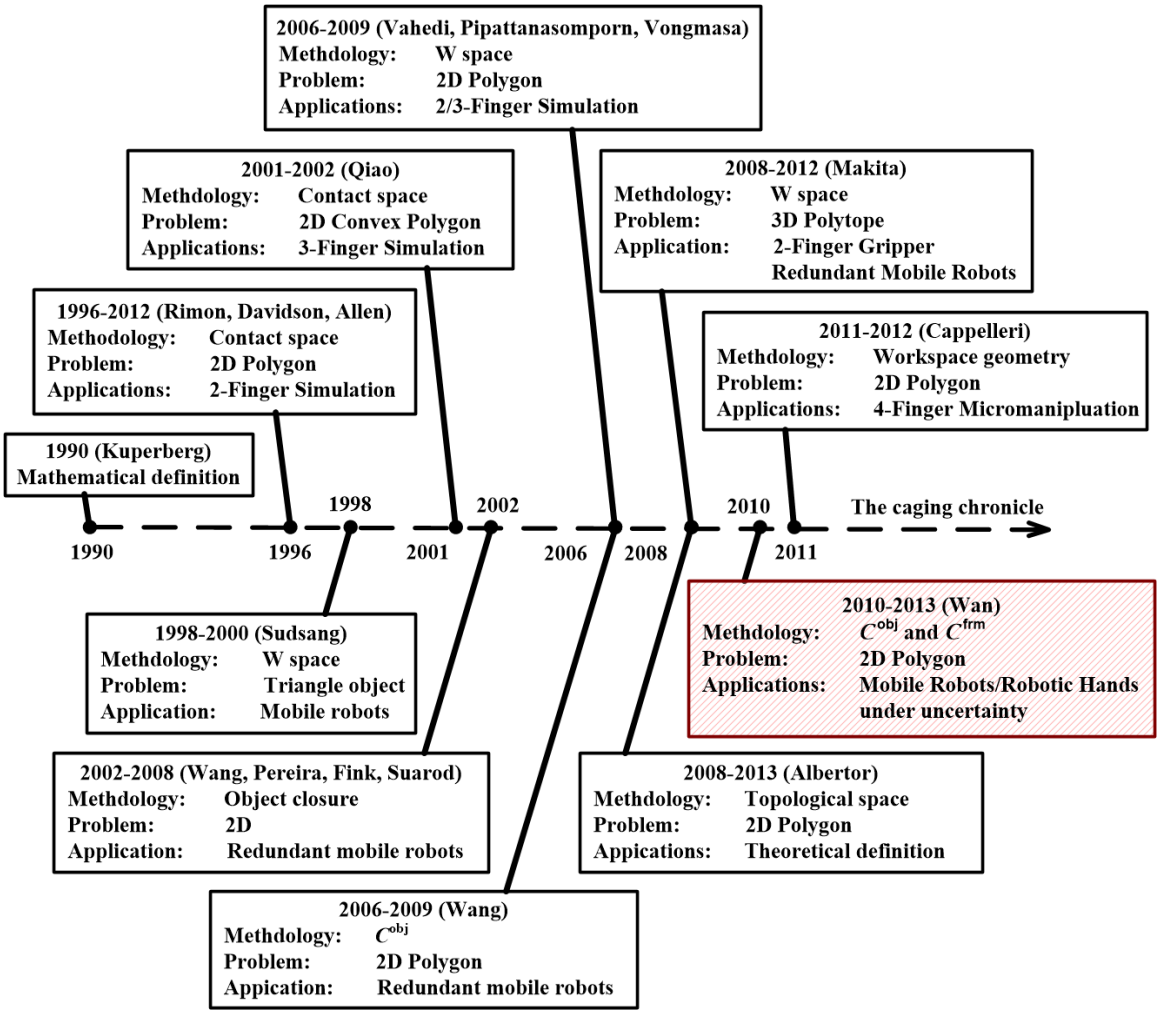


Figure 2.16: A summary of the related works and my contributions.

Part II

Caging in $\mathcal{C}^{\mathsf{obj}}$ and Its Applications

Chapter 3

Caging in The Configuration Space of Target Object

3.1 Caging Test in $\mathcal{C}^{\mathsf{obj}}$

Starting from this chapter, I am going to discuss in detail the tools, techniques, experiments and applications of caging.

Firstly, I list the symbols that will be used in the following part. Readers are recommended to revisit these symbols frequently during their reading process.

$\mathcal{W}$	Work space.
$\boldsymbol{p_i}$	A point in $\mathcal{W}$ space. Since we deal with 2D objects in 2D $\mathcal{W}$ space, the $\boldsymbol{p_i}$ has two coordinate elements $\{p_{i_\mathsf{x}}, p_{i_\mathsf{y}}\}$.
$\mathcal{C}^{\mathsf{obj}}$	The configuration space of target object.
O	The target object in $\mathcal{W}$ space.
$\partial\mathsf{O}$	Boundary of the target object in $\mathcal{W}$ space.
$\boldsymbol{f_i}$	A point finger in $\mathcal{W}$ space. Since we deal with 2D objects in 2D $\mathcal{W}$ space, the $\boldsymbol{f_i}$ has two coordinate elements $\{f_{i_\mathsf{x}}, f_{i_\mathsf{y}}\}$. Mathematically, $\boldsymbol{f_i} = \boldsymbol{p_i}$. I employ this extra symbol $\boldsymbol{f_i}$ rather than using an unified one with $\boldsymbol{p_i}$ to make clear the texts.
$\boldsymbol{q}^{\mathsf{obj}}/\boldsymbol{q_i}^{\mathsf{obj}}$	A configuration of $\mathcal{C}^{\mathsf{obj}}$. Since the configuration space of a 2D object is 3D, a configuration $\boldsymbol{q_i}^{\mathsf{obj}}$ has three coordinate elements $\{q_{i_\mathsf{x}}^{\mathsf{obj}}, q_{i_\mathsf{y}}^{\mathsf{obj}}, q_{i_\theta}^{\mathsf{obj}}\}$. The first two elements $q_{i_\mathsf{x}}^{\mathsf{obj}}$ and $q_{i_\mathsf{y}}^{\mathsf{obj}}$ are actually the same as a position in 2D $\mathcal{W}$ space. That is to say, $\{q_{i_\mathsf{x}}^{\mathsf{obj}}, q_{i_\mathsf{y}}^{\mathsf{obj}}\}$ and $\{f_{i_\mathsf{x}}, f_{i_\mathsf{y}}\}$ both indicate coordinates in 2D plane. The are only different in symbols.
$\mathsf{O}[\boldsymbol{q_i}^{\mathsf{obj}}]$	A target object O at configuration $\boldsymbol{q_i}^{\mathsf{obj}}$. Reader may assume that $\mathsf{O}[\boldsymbol{q_i}^{\mathsf{obj}}]$ represents a region in 2D workspace that is occupied by a target object. It

mathematically equals a set of 2D points. This expression can also be written in the following two forms, (1) $\mathsf{O}[\{q_{i_\mathsf{x}}^{\mathsf{obj}}, q_{i_\mathsf{y}}^{\mathsf{obj}}\}]$ and (2) $\mathsf{O}[q_{i_\theta}^{\mathsf{obj}}]$. The first form, $\mathsf{O}[\{q_{i_\mathsf{x}}^{\mathsf{obj}}, q_{i_\mathsf{y}}^{\mathsf{obj}}\}]$ indicates a region in 2D workspace that is occupied by a target object which could rotate arbitrarily through 0 to 2π. However, its position is fixed at the point $\mathsf{O}[\{q_{i_\mathsf{x}}^{\mathsf{obj}}, q_{i_\mathsf{y}}^{\mathsf{obj}}\}]$. The second form, $\mathsf{O}[q_{i_\theta}^{\mathsf{obj}}]$, indicates a region in 2D workspace that is occupied by a target object which could translate arbitrarily on the 2D plane. However, its orientation is fixed to $q_{i_\theta}^{\mathsf{obj}}$. Note that the superfix are sometimes omitted for conciseness.

$\partial\mathsf{O}[\boldsymbol{q_i}^{\mathsf{obj}}]$ The boundary of the 2D region occupied by a target object at configuration $\boldsymbol{q_i}^{\mathsf{obj}}$. Note that the symbols $\partial\mathsf{O}[\{q_{i_\mathsf{x}}^{\mathsf{obj}}, q_{i_\mathsf{y}}^{\mathsf{obj}}\}]$ and $\partial\mathsf{O}[q_{i_\theta}^{\mathsf{obj}}]$ do not make much sense as the first one is usually the boundary of a circle while the second one is not applicable ($\mathsf{O}[q_{i_\theta}^{\mathsf{obj}}]$ spans all 2D plane).

$\mathcal{F}_i$ One configuration obstacle in $\mathcal{C}^{\mathsf{obj}}$. Please go back to Fig.2.11 for details. Mathematically, it is a set of compact 3D points and a sub-space of $\mathcal{C}^{\mathsf{obj}}$. The subscript indicates the correspondence between $\mathcal{W}$ space fingers and $\mathcal{C}^{\mathsf{obj}}$ space obstacles. For example, $\mathcal{F}_i$ is the $\mathcal{C}^{\mathsf{obj}}$ obstacle of finger $\boldsymbol{f_i}$. Note that the obstacles of different fingers indeed have the same shape in $\mathcal{C}^{\mathsf{obj}}$, they only differ in positions.

$\mathcal{F}_i[q_{i_\theta}^{\mathsf{obj}}]$ Generally, a configuration obstacle $\mathcal{F}_i$ spans the whole *orientation* axis of $\mathcal{C}^{\mathsf{obj}}$. I use $\mathcal{F}_i[q_{i_\theta}^{\mathsf{obj}}]$ to denote a sliced layer of the whole $\mathcal{F}_i$ at *orientation* $q_{i_\theta}^{\mathsf{obj}}$. This combination is not applicable to $q_{i_\mathsf{x}}^{\mathsf{obj}}$ and $q_{i_\mathsf{y}}^{\mathsf{obj}}$.

$\mathcal{C}_{\mathsf{otl}}^{\mathsf{obj}}$ All obstruction-free obstacles in $\mathcal{C}^{\mathsf{obj}}$. It is used to indicate both the caging sub-space and the free sub-space in Fig.2.13. Mathematically, $\mathcal{C}_{\mathsf{otl}}^{\mathsf{obj}}$ is the union of all $\mathcal{F}_i$. Assume there are n fingers, then $\mathcal{C}_{\mathsf{otl}}^{\mathsf{obj}} = \bigcup_{i=1}^{n} \mathcal{F}_i$.

$\mathcal{C}_{\mathsf{free}}^{\mathsf{obj}}$ All free sub-spaces in $\mathcal{C}^{\mathsf{obj}}$. Mathematically, it is complementary to $\mathcal{C}_{\mathsf{otl}}^{\mathsf{obj}}$. Assume there are n fingers, then $\mathcal{C}_{\mathsf{free}}^{\mathsf{obj}} = \mathcal{C}^{\mathsf{obj}} \setminus \mathcal{C}_{\mathsf{otl}}^{\mathsf{obj}} = \{\boldsymbol{q}^{\mathsf{obj}} | (\boldsymbol{q}^{\mathsf{obj}} \in \mathcal{C}^{\mathsf{obj}}) \wedge (\boldsymbol{q}^{\mathsf{obj}} \notin \bigcup_{i=1}^{n} \mathcal{F}_i)\}$

$R^{(\theta)}$ The rotation matrix with respect to an angle θ. For example, if we would like to rotate π, $R^{(\pi)} = \begin{bmatrix} cos(\pi) & sin(\pi) & 0 \\ -sin(\pi) & cos(\pi) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

A caging test problem offers the following conditions. (1) The target object and its initial configuration, say, $\mathsf{O}[\boldsymbol{q_0}^{\mathsf{obj}}] = \mathsf{O}[\{q_{0_\mathsf{x}}^{\mathsf{obj}}, q_{0_\mathsf{y}}^{\mathsf{obj}}, q_{0_\theta}^{\mathsf{obj}}\}]$. (2) The positions of fingers, say, $\boldsymbol{f_1} = \{f_{1_\mathsf{y}}, f_{1_\mathsf{y}}\}$, $\boldsymbol{f_2} = \{f_{2_\mathsf{y}}, f_{2_\mathsf{y}}\}$, ..., $\boldsymbol{f_n} = \{f_{n_\mathsf{y}}, f_{n_\mathsf{y}}\}$ when there are n fingers.

When caging is achieved, we have the following two necessary and sufficient conditions. (1) The $\mathcal{C}_{\mathsf{free}}^{\mathsf{obj}}$ is divided into several disconnected components. Most of the components are enclosed by obstacles, let us denote them with $\mathcal{C}_{\mathsf{fc}}^{\mathsf{obj}} = \bigcup_{i=1}^{u} \mathcal{C}_{\mathsf{fc_i}}^{\mathsf{obj}}$. Here $u = 1$ when the target

object is convex. A special component is the complementary of $\mathcal{C}_{\mathsf{fc}}^{\mathsf{obj}}$, let us denote it with $\mathcal{C}_{\mathsf{ff}}^{\mathsf{obj}}$. This condition follows the following expression, $\mathcal{C}_{\mathsf{free}}^{\mathsf{obj}} = \mathcal{C}_{\mathsf{fc}}^{\mathsf{obj}} \cup \mathcal{C}_{\mathsf{ff}}^{\mathsf{obj}} = (\bigcup_{i=1}^{u} \mathcal{C}_{\mathsf{fc_i}}^{\mathsf{obj}}) \cup \mathcal{C}_{\mathsf{ff}}^{\mathsf{obj}}$, $\mathcal{C}_{\mathsf{fc}}^{\mathsf{obj}} \cap \mathcal{C}_{\mathsf{ff}}^{\mathsf{obj}} = \emptyset$ (2) The configuration of the target object is inside one component of $\mathcal{C}_{\mathsf{fc}}^{\mathsf{obj}}$. Let us denote the configuration of the target object, when performing caging test, is $\boldsymbol{q_0}^{\mathsf{obj}}$. Then, caging requires $\boldsymbol{q_0}^{\mathsf{obj}} \in \mathcal{C}_{\mathsf{fc}}^{\mathsf{obj}}$ or more exactly, $\boldsymbol{q_0}^{\mathsf{obj}} \in \mathcal{C}_{\mathsf{fc_k}}^{\mathsf{obj}}$, $1 \leq k \leq u$. Note that the caging sub-space and the free sub-space of Fig.2.13 are examples of $\mathcal{C}_{\mathsf{fc}}^{\mathsf{obj}}$ and $\mathcal{C}_{\mathsf{ff}}^{\mathsf{obj}}$ respectively.

Recall our discussion of caging in Chapter 2.5: "The target object, in its configuration space, is in a caging sub-space." Following this discussion, caging test can be performed by checking the following expression.

$$(\mathcal{C}_{\mathsf{free}}^{\mathsf{obj}} = (\bigcup_{i=1}^{u} \mathcal{C}_{\mathsf{fc_i}}^{\mathsf{obj}}) \cup \mathcal{C}_{\mathsf{ff}}^{\mathsf{obj}}) \wedge (\boldsymbol{q_0}^{\mathsf{obj}} \in \mathcal{C}_{\mathsf{fc_k}}^{\mathsf{obj}}) \wedge (|\mathcal{C}_{\mathsf{fc_k}}^{\mathsf{obj}}| > 1), 1 \leq k \leq u \tag{3.1}$$

Here $|\mathcal{C}_{\mathsf{fc_k}}^{\mathsf{obj}}|$ means the cardinality, namely the number of elements, of $\mathcal{C}_{\mathsf{fc_k}}^{\mathsf{obj}}$. When $|\mathcal{C}_{\mathsf{fc_k}}^{\mathsf{obj}}| > 1$, the target object is either in the state of caging or in the state of contact caging. When $|\mathcal{C}_{\mathsf{fc_k}}^{\mathsf{obj}}| = 1$, the target object is in the state of immobilization.

Now the caging test problem becomes modeling and intersecting several 3D objects. We have discussed in Section 1.3 that there are two ways of modeling a 3D object. One is **wireframe modeling** while the other one is **solid modeling**. Here I choose the **wireframe modeling** technology to model $\mathcal{C}_{\mathsf{otl}}^{\mathsf{obj}}$. This is because in $\mathcal{C}^{\mathsf{obj}}$, we can easily know the vertices of $\mathcal{C}_{\mathsf{otl}}^{\mathsf{obj}}$. These vertices make **wireframe modeling** easier comparing with **solid modelling**. This is because **Solid modeling** models 3D objects with a set of voxels. If a user want to render an object modeled by **solid modeling**, he has to firstly convert the voxels into a wireframe model [Wikipedia, 2013a]. The conversion process makes **solid modeling** complicated.

Fig.3.1 shows the details of **wireframe modeling**. We have seen in Section 2.1.2 that the $\mathcal{C}^{\mathsf{obj}}$ has three axes, namely *position x*, *position y* and *orientation*. The symbol definition of a configuration $\boldsymbol{q_i}^{\mathsf{obj}} = \{q_{i_\mathsf{x}}^{\mathsf{obj}}, q_{i_\mathsf{y}}^{\mathsf{obj}}, q_{i_\theta}^{\mathsf{obj}}\}$ respectively denote coordinate values along *position x*, *position y* and *orientation*. In order to model the whole wireframe of a $\mathcal{F}_i$. We first discretize rotation, namely the *orientation* axis. With a granularity of $2m+1$, we can divide the rotation of a target object into $2m+1$ angles. In correspondence, the $[-\pi, \pi)$ domain of the *orientation* axis is divided into $2m+1$ coordinate values and the $\mathcal{C}^{\mathsf{obj}}$ between this domain is divided into $2m+1$ layers. Note that we do not need to consider the domains since 2π is the period of rotation and the $\mathcal{C}^{\mathsf{obj}}$ between the other domains are the same as the one between $[-\pi, \pi)$. The $2m+1$ layers have coordinate values along the *orientation* axis ranging from $q_{-m_\theta}^{\mathsf{obj}}$ to $q_{m_\theta}^{\mathsf{obj}}$. The whole model of $\mathcal{F}_i$ is accordingly discretized into $2m+1$ slices. Note that the whole model of $\mathcal{F}_i$ is periodical at every 2π rotation. We only discuss the part between $[-\pi, \pi)$. Fig.3.1(a) illustrates the granularity and divided layers.

When performing caging test, the configuration of the target object is known. It is $\boldsymbol{q_0}^{\mathsf{obj}} = \{q_{0_\mathsf{x}}^{\mathsf{obj}}, q_{0_\mathsf{y}}^{\mathsf{obj}}, q_{0_\theta}^{\mathsf{obj}}\}$. Here the position $\{q_{0_\mathsf{x}}^{\mathsf{obj}}, q_{0_\mathsf{y}}^{\mathsf{obj}}\}$ is equal to the position of a pivot

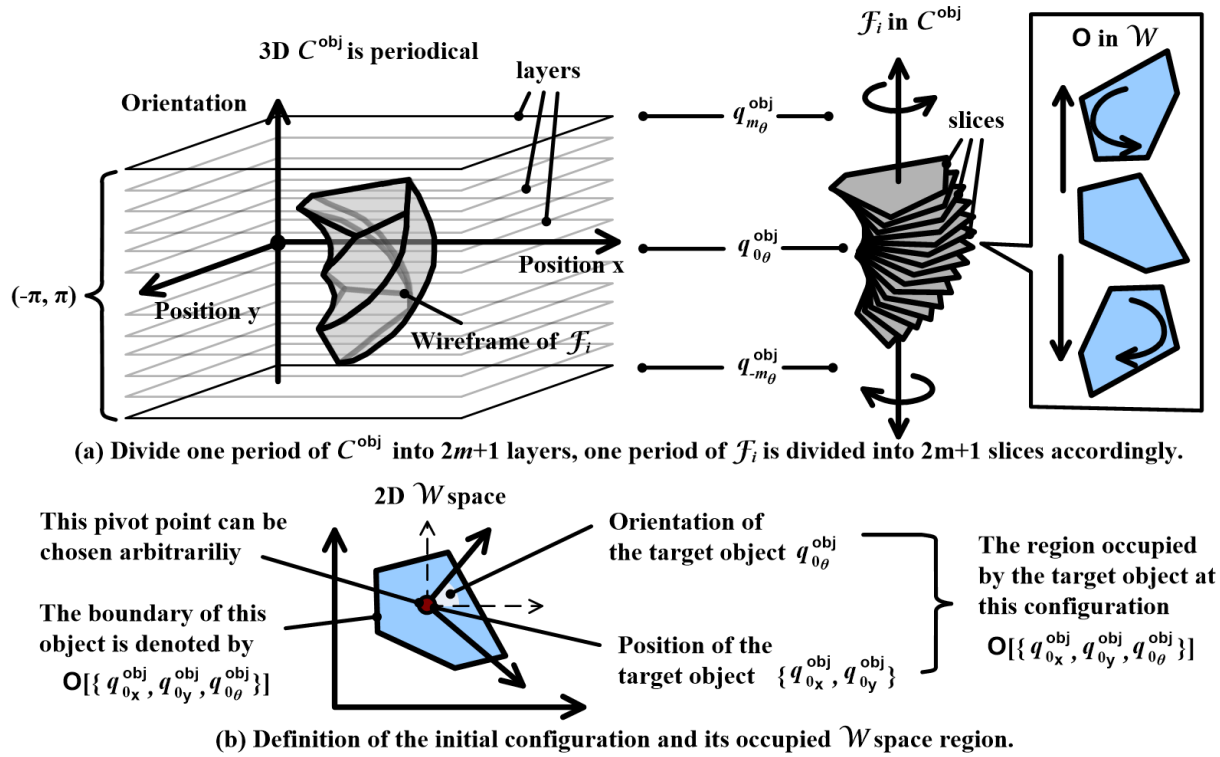


Figure 3.1: Wireframe modeling of a $\mathcal{F}_i$ obstacle and discretization.

point on the target object. This pivot point could be chosen arbitrarily. For instance, [Wang and Kumar, 2002] chooses the geometric center of the target object while [Pereira et al., 2002b] chooses a vertex of the target object. The pivot point is coordinate-invariant and there is no difference between different choices. The $\mathcal{W}$ space region occupied by the target object at its initial configuration therefore can be expressed as $\partial \mathsf{O}[\{q_{0_x}^{obj}, q_{0_y}^{obj}, q_{0_\theta}^{obj}\}]$. Fig.3.1(b) illustrates the initial configuration and the occupied workspace region.

Without any changes in orientation, the $\mathcal{F}_i$ is always a slice at layer $q_{0_\theta}^{obj}$, namely $\mathcal{F}_i[q_{0_\theta}^{obj}]$. If the target object rotates to $q_{j_\theta}^{obj}$, $-m \leq j \leq m$, then the the $\mathcal{F}_i$ becomes a slice at layer $q_{j_\theta}^{obj}$, namely $\mathcal{F}_i[q_{j_\theta}^{obj}]$. Since the target object may rotate to any orientation between $q_{-m_\theta}^{obj}$ and $q_{m_\theta}^{obj}$, $\mathcal{F}_i$ is composed of $2m+1$ layers naming from $\mathcal{F}_i[q_{-m_\theta}^{obj}]$ to $\mathcal{F}_i[q_{m_\theta}^{obj}]$.

Modeling the wireframe of a discretized $\mathcal{F}_i$ essentially equals to modeling $2m+1$ layers of polygon slices. Fig.3.2 shows the detail of how to model the $2m+1$ slices. Given a configuration of the target object, for example $\{q_{j_x}^{obj}, q_{j_y}^{obj}, q_{j_\theta}^{obj}\}$, $-m \leq j \leq m$ and a finger $\boldsymbol{f_1}$, we can generate $\mathcal{F}_1[q_{j_\theta}^{obj}]$ by $R^{(\pi)} \cdot \mathsf{O}[\{f_{1_x}, f_{1_y}, q_{j_\theta}^{obj}\}]$. The following part proves this conclusion.

According to the definition of $\mathcal{F}_0$,

$$\mathcal{F}_1[q_{j_\theta}^{obj}] = \Big\{ \boldsymbol{p_i} | \boldsymbol{f_1} \cap \mathsf{O}[\{p_{i_x}^{obj}, p_{i_y}^{obj}, q_{j_\theta}^{obj}\}] \neq \emptyset \Big\}. \tag{3.2}$$

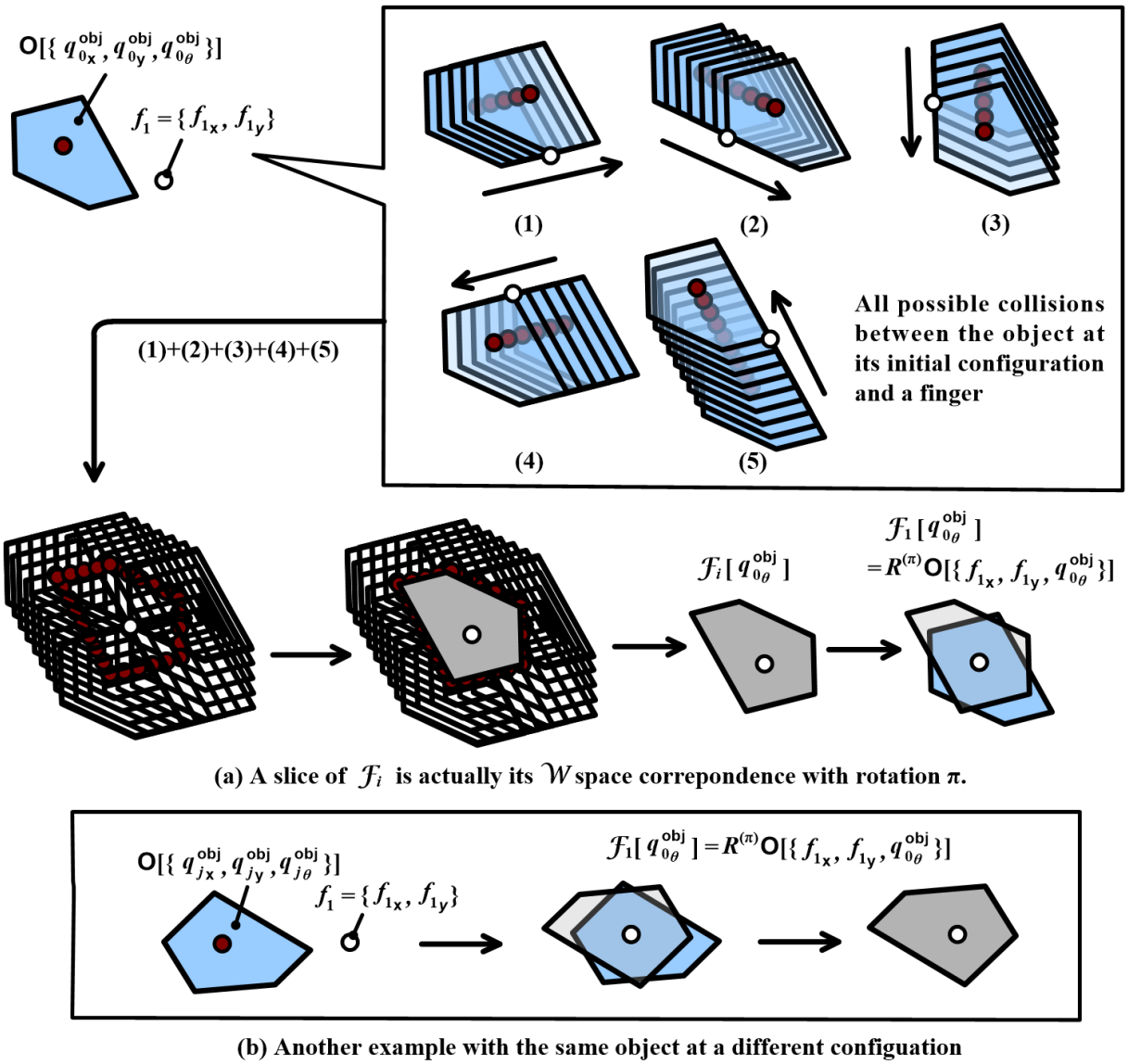


Figure 3.2: Modeling the discretized slices of $\mathcal{F}_i$.

This is equal to

$$\mathcal{F}_1[q_{j_\theta}^{\mathsf{obj}}] = \left\{ \boldsymbol{p_i} | \boldsymbol{p_i} \cap \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathsf{O}[\{f_{1_\mathsf{x}}, f_{1_\mathsf{y}}, q_{j_\theta}^{\mathsf{obj}}\}] \neq \emptyset \right\} \tag{3.3}$$

since for each point,

$$\boldsymbol{p_i} - \boldsymbol{f_1} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot (\boldsymbol{f_1} - \boldsymbol{p_i}). \tag{3.4}$$

In 2D plane,

$$R^{(\pi)} = \begin{bmatrix} cos(\pi) & sin(\pi) & 0 \\ -sin(\pi) & cos(\pi) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.5}$$

Therefore,

$$\mathcal{F}_1[q_{j_\theta}^{\mathsf{obj}}] = \left\{ R^{(\pi)} \cdot \partial\mathsf{O}[\{f_{1_\mathsf{x}}, f_{1_\mathsf{y}}, q_{j_\theta}^{\mathsf{obj}}\}] \right\} \tag{3.6}$$

Fig.3.2(a) graphically illustrates the geometric meaning of these expressions. All the key translations that result into collision between the target object and the finger are shown in Fig.3.2(a)(1-5). By summing up these key translations and connecting the pivot points, we can generate $\mathcal{F}_1[q_{0_\theta}^{\mathsf{obj}}]$. It is easy to find that this $\mathcal{F}_1[q_{0_\theta}^{\mathsf{obj}}]$ equals its $\mathcal{W}$ space correspondence $\mathsf{O}[\{f_{1_\mathsf{x}}, f_{1_\mathsf{y}}, q_{0_\theta}^{\mathsf{obj}}\}]$ with a π rotation. This is the same as our deduction. In Fig.3.2(b), I give another example with the same object as Fig.3.2(a). The difference is, in this case, the target object is at another configuration, say, $\boldsymbol{q_j}^{\mathsf{obj}}$.

Now we can model the whole $\mathcal{F}_i$ in the following way. Given a finger $\boldsymbol{f_i}$ and a target object at its initial configuration O, we model the $\mathcal{F}_i$ that corresponds to this object by using a set of polygon slices, namely,

$$\left\{ R^{(\pi)} \cdot \partial\mathsf{O}[\{f_{i_\mathsf{x}}, f_{i_\mathsf{y}}, q_{-m_\theta}^{\mathsf{obj}}\}], R^{(\pi)} \cdot \partial\mathsf{O}[\{f_{i_\mathsf{x}}, f_{i_\mathsf{y}}, q_{-m+1_\theta}^{\mathsf{obj}}\}], \ldots, R^{(\pi)} \cdot \partial\mathsf{O}[\{f_{i_\mathsf{x}}, f_{i_\mathsf{y}}, q_{m_\theta}^{\mathsf{obj}}\}] \right\}. \tag{3.7}$$

The caging test in expression (3.1) becomes performing the following procedures. (1) Check whether $\{q_{0_\mathsf{x}}^{\mathsf{obj}}, q_{0_\mathsf{y}}^{\mathsf{obj}}, q_{0_\theta}^{\mathsf{obj}}\}$ is enclosed by $\left\{ \mathcal{F}_1[q_{0_\theta}^{\mathsf{obj}}], \mathcal{F}_2[q_{0_\theta}^{\mathsf{obj}}], \ldots, \mathcal{F}_{n_f}[q_{0_\theta}^{\mathsf{obj}}] \right\}$ when there are n_f fingers. If $\boldsymbol{q_0}$ is enclosed, we can calculate the enclosed region that $\boldsymbol{q_0}$ exists. We denote this region by $\mathcal{C}_{\mathsf{fc_k}}^{\mathsf{obj}}[q_{0_\theta}^{\mathsf{obj}}]$. It is a set of 2D points. (2) For any $\boldsymbol{p_i} \in \mathcal{C}_{\mathsf{fc_k}}^{\mathsf{obj}}[q_{0_\theta}^{\mathsf{obj}}]$, check whether it is enclosed or obstructed by obstacles in neighbour layers. Since the layers adjacent to $q_{0_\theta}^{\mathsf{obj}}$ are $q_{1_\theta}^{\mathsf{obj}}$ and $q_{-1_\theta}^{\mathsf{obj}}$, We should check whether any $\boldsymbol{p_i}$ fulfills that $\{p_{i_\mathsf{x}}, p_{i_\mathsf{y}}, q_{1_\theta}^{\mathsf{obj}}\}$ is enclosed or obstructed by $\left\{ \mathcal{F}_1[q_{1_\theta}^{\mathsf{obj}}], \mathcal{F}_2[q_{1_\theta}^{\mathsf{obj}}], \ldots, \mathcal{F}_{n_f}[q_{1_\theta}^{\mathsf{obj}}] \right\}$ and enclosed or obstructed by $\left\{ \mathcal{F}_1[q_{-1_\theta}^{\mathsf{obj}}], \mathcal{F}_2[q_{-1_\theta}^{\mathsf{obj}}], \ldots, \mathcal{F}_{n_f}[q_{-1_\theta}^{\mathsf{obj}}] \right\}$. If all points in the 2D set are obstructed, then caging is true. If

any point is enclosed, we further calculate the new enclosed region $\mathcal{C}^{\mathsf{obj}}_{\mathsf{fc_k}}[q^{\mathsf{obj}}_{1_\theta}]$ or $\mathcal{C}^{\mathsf{obj}}_{\mathsf{fc_k}}[q^{\mathsf{obj}}_{-1_\theta}]$ that $\{p_{i_\mathsf{x}}, p_{i_\mathsf{y}}, q^{\mathsf{obj}}_{1_\theta}\}$ or $\{p_{i_\mathsf{x}}, p_{i_\mathsf{y}}, q^{\mathsf{obj}}_{-1_\theta}\}$ belongs to and replace the 2D point set with the points in the new enclosed region. For any point in the new enclosed regions, we repeat the procedure done in $q^{\mathsf{obj}}_{1_\theta}$ or $q^{\mathsf{obj}}_{-1_\theta}$ with $\left\{\mathcal{F}_1[q^{\mathsf{obj}}_{2_\theta}],\ \mathcal{F}_2[q^{\mathsf{obj}}_{2_\theta}],\ ...,\ \mathcal{F}_{n_f}[q^{\mathsf{obj}}_{2_\theta}]\right\}$, $\left\{\mathcal{F}_1[q^{\mathsf{obj}}_{3_\theta}],\ \mathcal{F}_2[q^{\mathsf{obj}}_{3_\theta}],\ ...,\ \mathcal{F}_{n_f}[q^{\mathsf{obj}}_{3_\theta}]\right\}$, ...until we reach $\left\{\mathcal{F}_1[q^{\mathsf{obj}}_{m_\theta}],\ \mathcal{F}_2[q^{\mathsf{obj}}_{m_\theta}],\ ...,\ \mathcal{F}_{n_f}[q^{\mathsf{obj}}_{m_\theta}]\right\}$ or with $\left\{\mathcal{F}_1[q^{\mathsf{obj}}_{-2_\theta}],\ \mathcal{F}_2[q^{\mathsf{obj}}_{-2_\theta}],\ ...,\ \mathcal{F}_{n_f}[q^{\mathsf{obj}}_{-2_\theta}]\right\}$, $\left\{\mathcal{F}_1[q^{\mathsf{obj}}_{-3_\theta}],\ \mathcal{F}_2[q^{\mathsf{obj}}_{-3_\theta}],\ ...,\ \mathcal{F}_{n_f}[q^{\mathsf{obj}}_{-3_\theta}]\right\}$, ...until we reach $\left\{\mathcal{F}_1[q^{\mathsf{obj}}_{-m_\theta}],\ \mathcal{F}_2[q^{\mathsf{obj}}_{-m_\theta}],\ ...,\ \mathcal{F}_{n_f}[q^{\mathsf{obj}}_{-m_\theta}]\right\}$. During this repetition, if a point is neither enclosed nor obstructed, caging is considered to be breaking. Or else, caging succeeds.

The narrative of this caging test algorithm seems complicated. Fortunately, it can be implemented with computer programs concisely. Fig.3.3 illustrates the basic ideas of this algorithms with the convex polygon we used in previous figures. The (ccw) part of Fig.3.3(a) shows the continuous caging test along counter-clockwise rotation, namely from $q^{\mathsf{obj}}_{1_\theta}$ to $q^{\mathsf{obj}}_{m_\theta}$. The (cw) part of Fig.3.3(b) shows the continuous caging test along clockwise rotation, namely from $q^{\mathsf{obj}}_{-1_\theta}$ to $q^{\mathsf{obj}}_{-m_\theta}$. At (ccw)-3 and (cw)-2, some points from the continuous refreshing 2D point set are neither enclosed nor obstructed, the caging breaks. The given formation of fingers is considered not able to cage the given target object. Fig.3.3(b) separately illustrates the correspondent failures at (ccw)-3 and (cw)-2 in $\mathcal{W}$ space.

Our figures in Fig.2.13, Fig.2.14 and Fig.2.15 are rendered by using this algorithm. They are programmed with Python and the figures are rendered with the Blender rendering engine [Blender, 2013].

With modern computers, this caging test algorithm can test whether a given formation of fingers could cage a given target object in a few seconds. Given a polygon of n_v boundary points, the computational complexity of this algorithm would be $O(n_v^{n_f} \cdot s \cdot m)$. Here, $O(n_v^{n_f})$ is the complexity of calculating an enclosed region while s is the average size of an enclosed 2D point set. The algorithm is complete with discretization and works with both convex and concave objects.

3.2 Robust Caging in $\mathcal{C}^{\mathsf{obj}}$

We have seen in last part how to discretize $\mathcal{C}^{\mathsf{obj}}$, how to model the wireframes of $\mathcal{F}_i$ at each layer and how to perform **caging test** with the discretization. Beyond **caging test**, we need to (1) find a set of finger formations that could cage the target object and (2) develop a **robust caging** algorithm to find an optimized formation of fingers that could be most robust to endure uncertainty. I refer readers to Fig.1.4 if they need a refresher about uncertainty. Let us firstly consider the item (1), namely how to find a set of finger formations that could cage the target object.

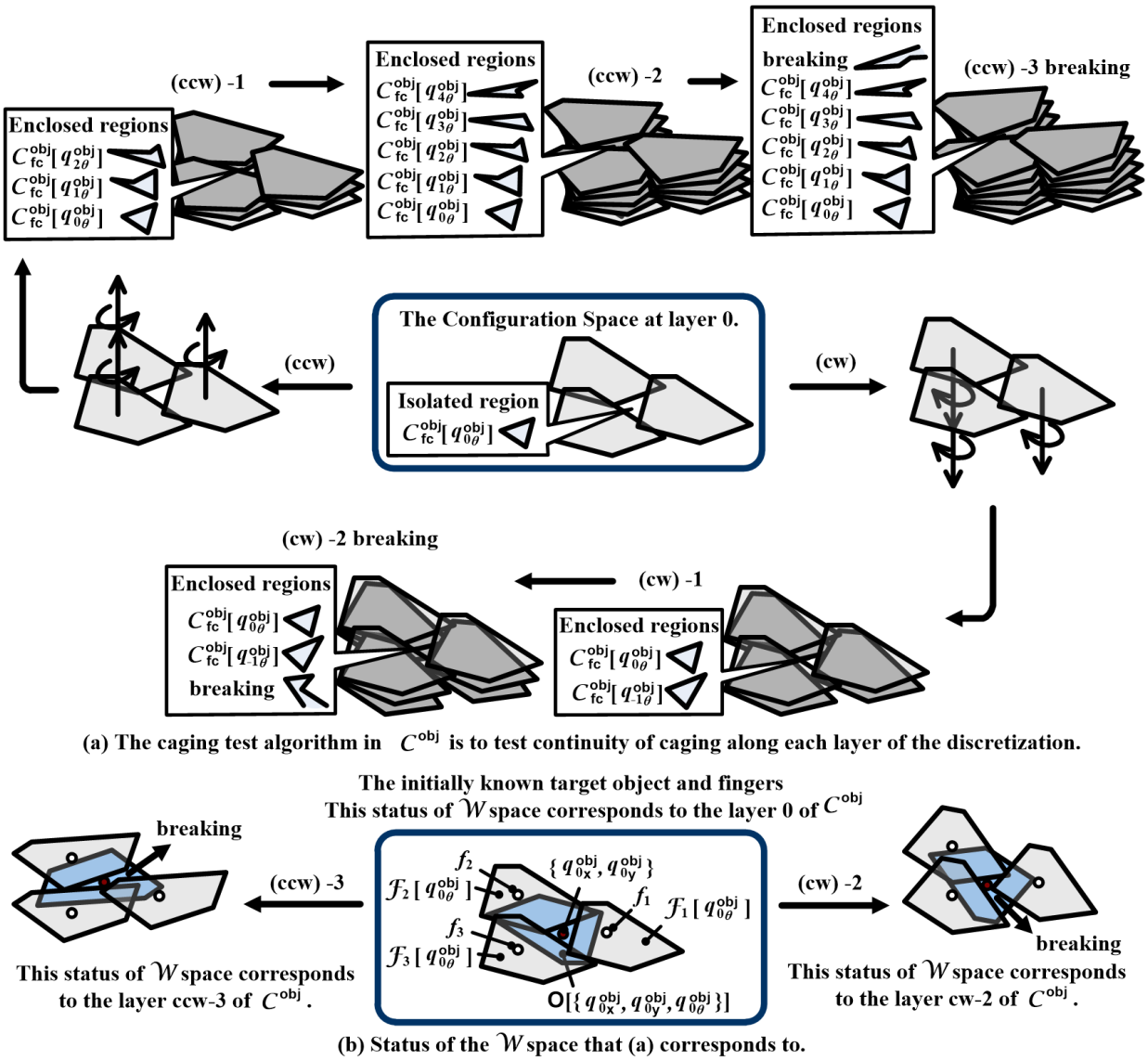


Figure 3.3: The caging test algorithm after discretization becomes testing the continuity of enclosure at each layer.

3.2.1 Finding all possible caging formations is costly

The most intuitive solution to find all possible caging formations is to perform caging test with all finger formations in a certain band that surrounds the target object. Fig.3.4 illustrates this intuitive solution. I proved in Section 2.2 that $n_{\text{dim}} + 1 = 2 + 1 = 3$ to $2n_{\text{dim}} = 2 \times 2 = 4$ fingers are sufficient to cage an object. It was emphasized there that **this is neither the least number nor the maximum number**. It is the sufficient number. That means it would be sufficient if we perform caging test with formations ranging from 2-finger formations to $2 \times 2 = 4$-finger formations. Of course more than 4 fingers would beyond sufficient and far enough for caging 2D objects. We can freely test 5-finger, 6-finger, ..., n_f-finger formations as we like. As examples, I am going to limit my analysis to the sufficient number, namely the 2-finger, 3-finger and 4-finger formations.

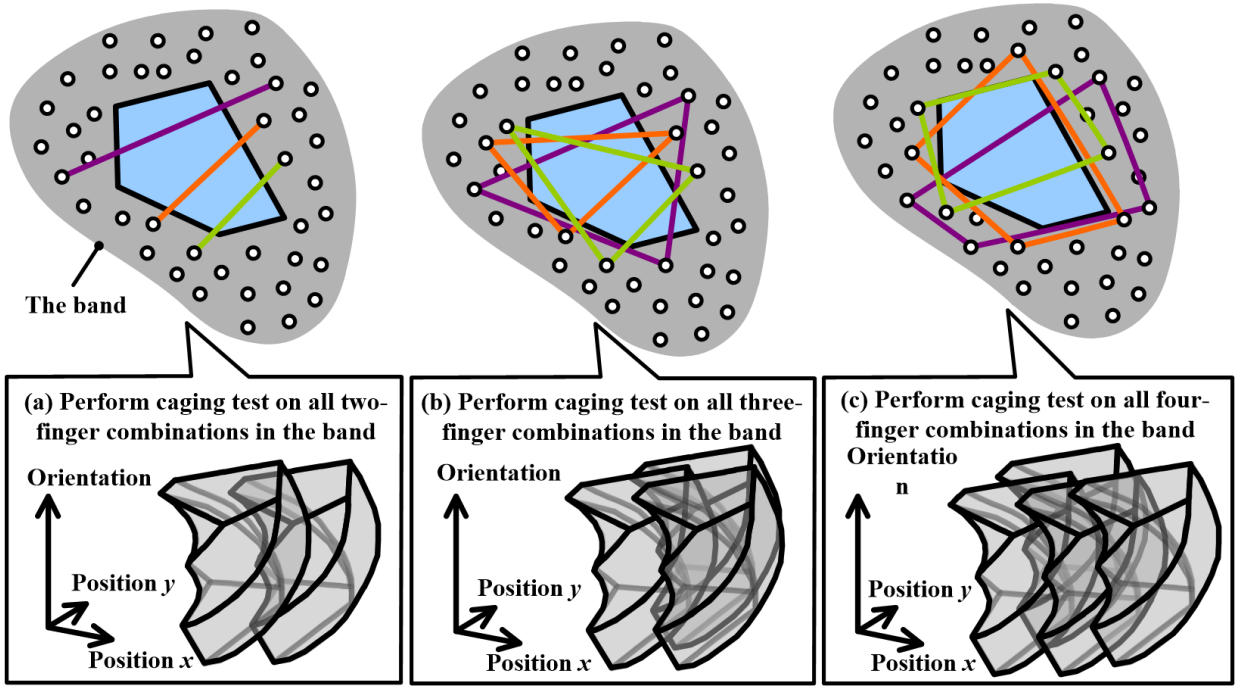


Figure 3.4: An intuitive way to find all caging formations.

Let us see again the intuitive solution in Fig.3.4. In the intuitive solution shown in Fig.3.4, finding all possible 2-finger, 3-finger and 4-finger caging formations means performing caging tests on all 2-finger, 3-finger and 4-finger combinations in the gray band. This is intuitive ans simple. However, it is computationally impossible. Performing one **caging test** with one given formation of fingers would cost $O(n_v^{n_f} \cdot s \cdot m)$. It is to the $(n_f + 2)$th order of variants. Assume that the band includes n_p points, then roughly there would be n_p^2, n_p^3 and n_p^4 finger formations for 2-finger, 3-finger and 4-finger cases. Totally, the cost of finding all caging formations may run as high as $O(n_p^2 \cdot n_v^2 \cdot s \cdot m + n_p^3 \cdot n_v^3 \cdot s \cdot m + n_p^4 \cdot n_v^4 \cdot s \cdot m) = O(n_p^4 \cdot n_v^4 \cdot s \cdot m)$. That's to the 10th order! Or more generally, it would be to the order of $2n_f + 2$ where n_f is

the maximum number of fingers that are employed for caging. Of course it is computational infeasible and we must choose another way to consider this "finding all caging formations" problem.

3.2.2 The caging region of a third finger – Concepts

One good starting point to consider the "finding all caging formations" problem is the idea introduced in [Erickson et al., 2007] and [Vahedi, 2009]. In these works, Erickson and Vahedi try to solve a problem like this: How to find the caging region of a third finger given the following two conditions. (1) A convex target object and its initial configuration. This is nearly the same as the "finding all possible caging formations" problem except that the object shape is limited to convex ones. (2) Positions of two fingers, say, $\boldsymbol{f_1} = \{f_{1_x}, f_{1_y}\}$ and $\boldsymbol{f_2} = \{f_{2_x}, f_{2_y}\}$. This condition is different from the "finding all possible caging formations" problem where all finger positions are unknown. This "finding the caging region of a third finger" problem is computationally feasible comparing with the "finding all possible caging formations" problem because of the two extra conditions.

Erickson proposes two concepts in his paper to solve this problem, namely the *canonical motion* and the *critical orientation*. *Canonical motion* is a motion that the target object moves as well as continues keeping in contact with the two given fingers. *Critical orientation* is an orientation at which the target object is potentially on the critical condition of detaching from either of the two given fingers. Fig.3.5 illustrates this problem and the two concepts[1]. As we can see, *canonical motion* and *critical orientation* are concepts specific to the two given fingers. If there are three given fingers, *canonical motion* and *critical orientation* may not exist. *Critical orientation* brings non-smoothness to *canonical motion*. The *canonical motion* changes drastically at *critical orientation*s.

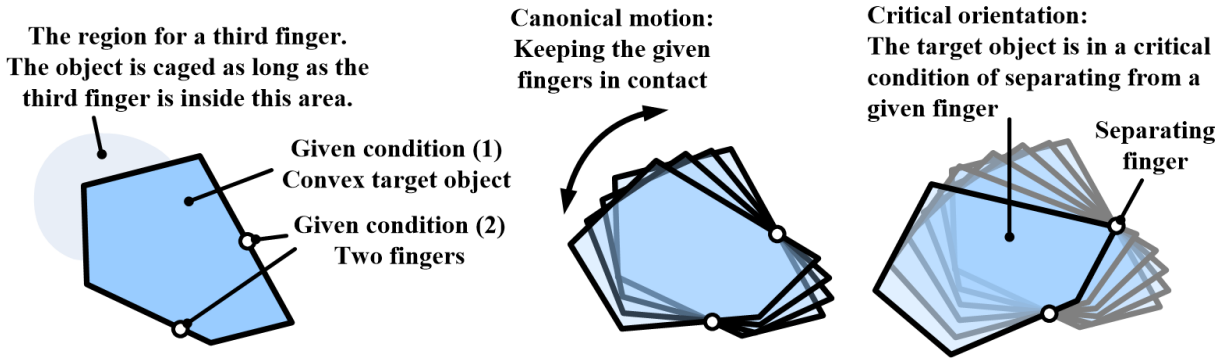


Figure 3.5: The "finding the caging region of a third finger" problem and two accompanying concepts.

[1] *Critical orientation* is defined with the configuration obstacle of a configuration obstacle shown in Fig.3.7. When segments of the two $\mathcal{F}_{\mathcal{F}_1}[q_{i_\theta}^{\mathsf{obj}}]$ and $\mathcal{F}_{\mathcal{F}_2}[q_{i_\theta}^{\mathsf{obj}}]$ are on the same line, the target object is in *critical orientation*.

These two concepts make it easier to find the region for a third fingers. We introduce a symbol $\mathcal{A}_\mathsf{c}$ to denote this region. We find this region by calculating its boundary. Any point on the boundary of $\mathcal{A}_\mathsf{c}$, say, $\boldsymbol{f_3}$, together with the given $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$ forms a 3-finger formation. This 3-finger formation is on the boundary of caging breaking. It is important to keep the target object in contact with the two given fingers so that we can calculate all those $\boldsymbol{f_3}$s on the boundary of $\mathcal{A}_\mathsf{c}$ and find $\mathcal{A}_\mathsf{c}$. Therefore, the concept of *canonical motion* plays an important role. When caging breaks, the object must be escaping through $\boldsymbol{f_1}$ and $\boldsymbol{f_3}$ or escaping through $\boldsymbol{f_2}$ and $\boldsymbol{f_3}$. The concept of *critical orientation* is used to monitor these escapings. The *critical orientation* plays an important role in recording the possible escaping orientations.

Besides the two concepts, Erickson also proposes an extended form of $\mathcal{F}_i$. It shares the same idea of CC-object proposed by [Wang et al., 2003b]. In Erickson's symbol definition, A_θ is used to denote the $\mathcal{F}_a[q_{j_\theta}^{\mathsf{obj}}]$ that corresponds to $\boldsymbol{f_a}$ while A_θ^* is used to denote the configuration of $\mathcal{F}_a[q_{j_\theta}^{\mathsf{obj}}]$ at $q_{j_\theta}^{\mathsf{obj}}$. Fig.3.6 and Fig.3.7 illustrate **the Configuration obstacle of a Configuration obstacle** and shows why employing this concept makes caging easier to be analyzed. If we would like to know at a certain layer, for example $q_{0_\theta}^{\mathsf{obj}}$, whether the three fingers can cage the target object, we need to make sure their correspondent configuration obstacles at this layer overlap with each other. Fig.3.6(a) demonstrates a caging state at layer $q_{0_\theta}^{\mathsf{obj}}$ and the overlapping of fingers. This is intuitive to human beings, but how can we check it with computer programs? Surely it is unwise to calculate all points between $\mathcal{F}_1[q_{0_\theta}^{\mathsf{obj}}]$ and $\mathcal{F}_2[q_{0_\theta}^{\mathsf{obj}}]$, between $\mathcal{F}_2[q_{0_\theta}^{\mathsf{obj}}]$ and $\mathcal{F}_3[q_{0_\theta}^{\mathsf{obj}}]$ and between $\mathcal{F}_3[q_{0_\theta}^{\mathsf{obj}}]$ and $\mathcal{F}_1[q_{0_\theta}^{\mathsf{obj}}]$ respectively. A smarter solution is to follow the idea of $\mathcal{C}^{\mathsf{obj}}$ and simplify this problem by using **the Configuration obstacle of a Configuration obstacle**. Fig.3.6(b) shows the overlapping after conversion. In this case, checking whether $\mathcal{F}_1[q_{0_\theta}^{\mathsf{obj}}]$ and $\mathcal{F}_3[q_{0_\theta}^{\mathsf{obj}}]$ overlaps becomes checking whether the finger $\boldsymbol{f_3}$ collides with or is inside $\mathcal{F}_{\mathcal{F}_1}[q_{0_\theta}^{\mathsf{obj}}]$. This smarter solution changes checking the overlapping of two polygons into a point in polygon problem. It is much easier to be processed by computer programs.

Fig.3.7 shows how **the Configuration obstacle of a Configuration obstacle**, or namely $\mathcal{F}_{\mathcal{F}_1}[q_{0_\theta}^{\mathsf{obj}}]$ in Fig.3.6(b) is modeled. This is similar to the procedure of modeling a $\mathcal{F}_i[q_{0_\theta}^{\mathsf{obj}}]$ in Fig.3.2 because $\mathcal{F}_i[q_{0_\theta}^{\mathsf{obj}}]$ aims to convert the relationship between a finger and a target object while $\mathcal{F}_{\mathcal{F}_1}[q_{0_\theta}^{\mathsf{obj}}]$ aims to convert the relationship between two $\mathcal{F}_i[q_{0_\theta}^{\mathsf{obj}}]$. Nevertheless, implementing the procedure illustrated in Fig.3.7 with computer programs is different from implementing the procedure in Fig.3.2. The implementation of Fig.3.2 is essentially based on expression (3.6). In contrast, the implementation of Fig.3.7 is more complicated and it requires to the following two steps. (1) Put boundary points of the convex, $\mathcal{F}_3[q_{0_\theta}^{\mathsf{obj}}]$, on to the boundary points of the convex, $\mathcal{F}_3[q_{0_\theta}^{\mathsf{obj}}]$ sequentially. (2) Calculate the convex hull of the sequential $\mathcal{F}_3[q_{0_\theta}^{\mathsf{obj}}]$s. The convex hull calculated in the (2) step is the result $\mathcal{F}_{\mathcal{F}_1}[q_{0_\theta}^{\mathsf{obj}}]$. Given a polygon of n_v boundary points, this algorithm would cost $O(n_v^2)$ where most of the computational resources is consumed by step (1).

Now let us recall our "finding the caging region of a third finger" problem, given two

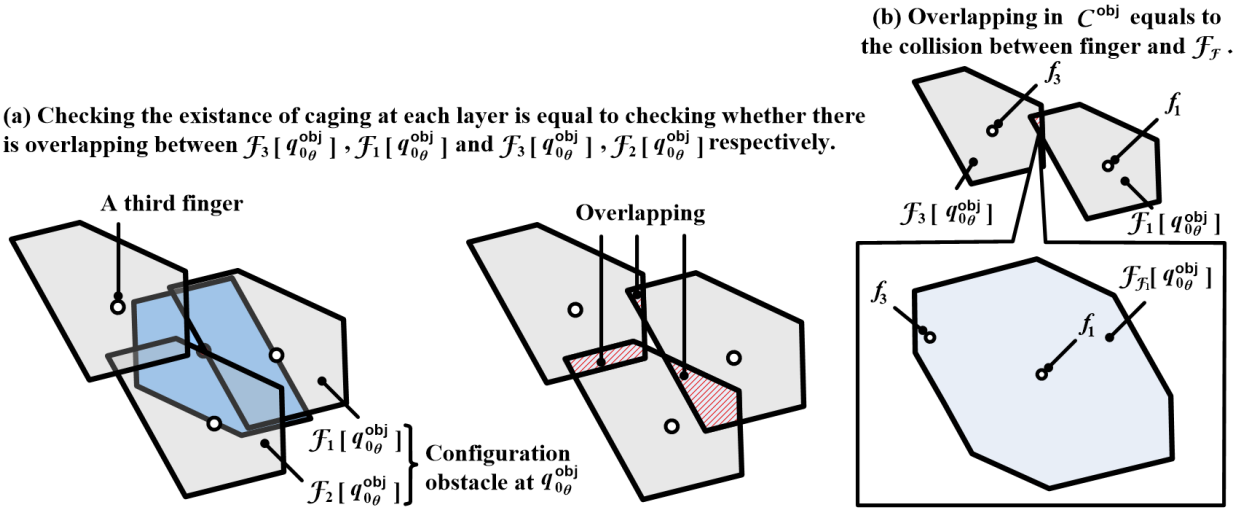


Figure 3.6: The "finding the caging region of a third finger" problem and two accompanying concepts.

fingers $\boldsymbol{f_1}$, $\boldsymbol{f_2}$ and the target object, we can model $\mathcal{F}_{\mathcal{F}_1}[q_{0_\theta}^{obj}]$ and $\mathcal{F}_{\mathcal{F}_2}[q_{0_\theta}^{obj}]$ according to the algorithm introduced in last paragraph. Then, the caging region of a third finger at a one layer of $\mathcal{C}^{obj}$, namely $\mathcal{A}_c[q_{0_\theta}^{obj}]$, can be expressed as the boolean sum of expression (3.8) and expression (3.9).

$$\mathcal{A}_c[q_{0_\theta}^{obj}] \in ((\mathcal{F}_{\mathcal{F}_1}[q_{0_\theta}^{obj}] \cap \mathcal{F}_{\mathcal{F}_2}[q_{0_\theta}^{obj}]) \setminus \mathsf{O}[\{q_{0_x}^{obj}, q_{0_y}^{obj}, q_{0_\theta}^{obj}\}]) \tag{3.8}$$

$$(\boldsymbol{f_1} \notin \mathcal{A}_c[q_{0_\theta}^{obj}]) \wedge (\boldsymbol{f_2} \notin \mathcal{A}_c[q_{0_\theta}^{obj}]) \tag{3.9}$$

The region calculated by expression (3.8) $\wedge$ expression (3.9) is illustrated in the left part of Fig.3.8. This region for the third finger is only the region at one layer. Vahedi in his Ph.D thesis [Vahedi, 2009] names the state when a third finger is in the region $\mathcal{A}_c[q_{0_\theta}^{obj}]$ the **translational caging**. This is a rational name since when a third finger is in $\mathcal{A}_c[q_{0_\theta}^{obj}]$, the target object can never escape from the fingers by translational motion. It has to rotate to escape. A complete caging not only involves the **translational caging** but also involves a **rotational constraint**. Calculating the complete $\mathcal{A}_c$ is like Fig.3.3 and it requires to accumulate the translational caging regions at different orientations $q_{-m_\theta}^{obj}$, $q_{-m+1_\theta}^{obj}$, ..., $q_{0_\theta}^{obj}$, ..., $q_{m_\theta}^{obj}$. The right part of Fig.3.8 shows the accumulation procedure. The translational caging regions in this right part are rendered with shadows decorated by gray and red textures. We should start from $\mathcal{A}_c[q_{0_\theta}^{obj}]$ and calculate $\mathcal{A}_c[q_{1_\theta}^{obj}]$, $\mathcal{A}_c[q_{2_\theta}^{obj}]$, ...and $\mathcal{A}_c[q_{-1_\theta}^{obj}]$, $\mathcal{A}_c[q_{-2_\theta}^{obj}]$, ...until $+/-m$ or until a $\mathcal{A}_c[q_{i_\theta}^{obj}]$ disappears. This procedure relates to the two concepts proposed by Erickson, namely the *canonical motion* and the *critical orientation*. However, this complete accumulation is more advanced comparing with Erickson's work since (1) it does not explicitly take *critical orientation*s into account and (2) it proposes an implicit way of generating

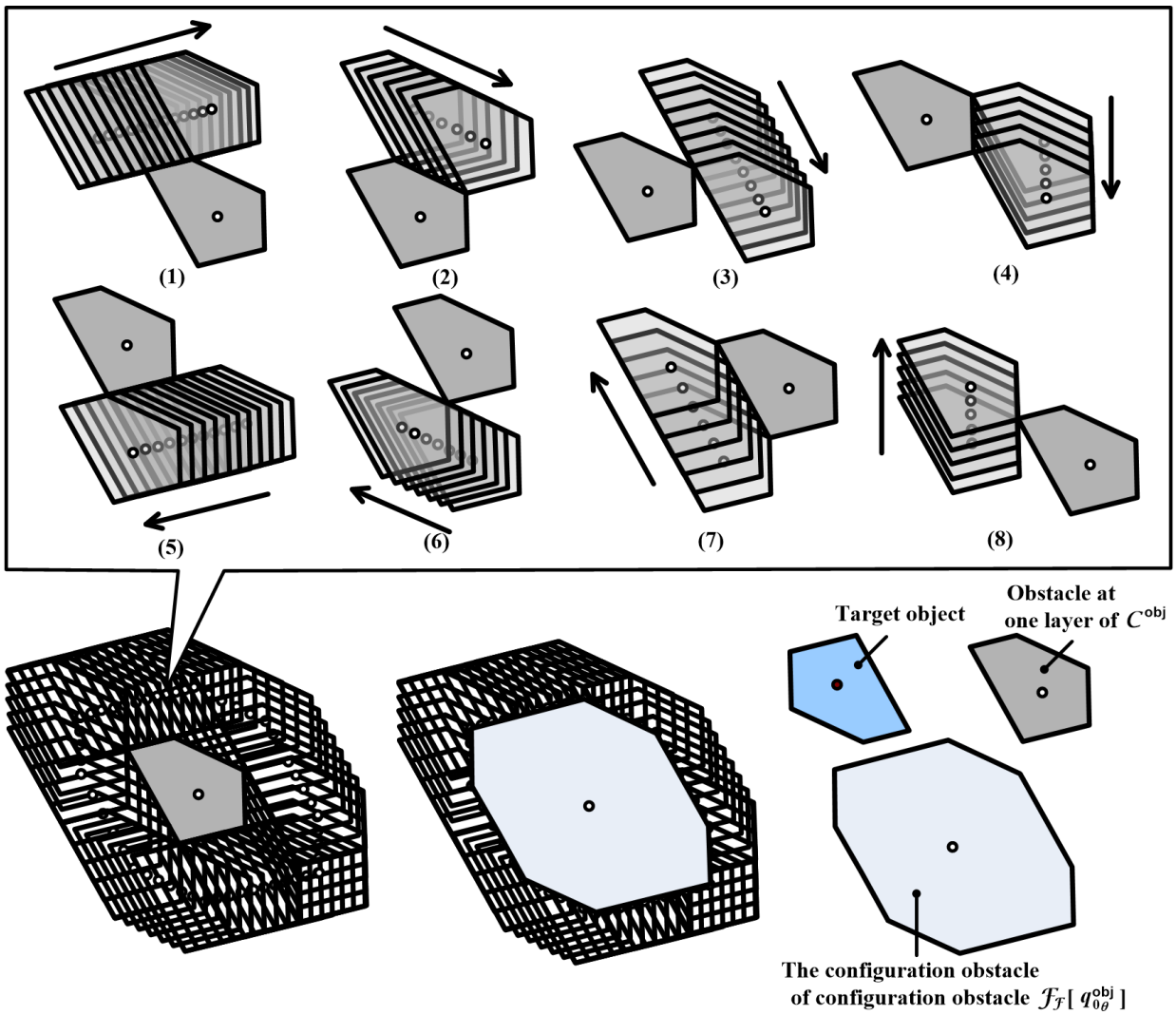


Figure 3.7: Modeling the configuration obstacle of a configuration obstacle.

the *canonical motion*s. We will discuss the details of my complete accumulation in the next sub-section. It is ready to co-operate with raw sensor data from real-world devices.

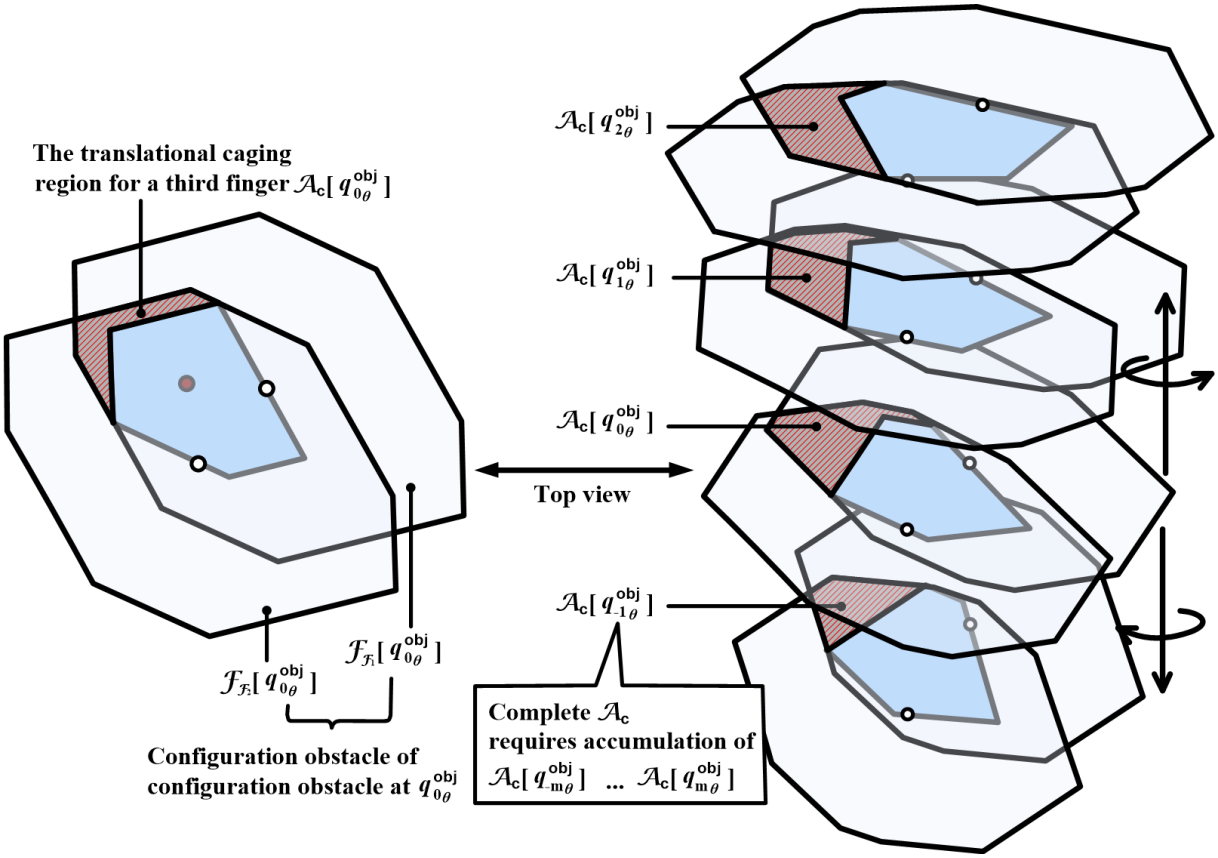


Figure 3.8: The translational caging region $\mathcal{A}_c[q_{0_\theta}^{obj}]$ for a third finger and region accumulation.

3.2.3 The caging region of a third finger – Algorithms

This sub-section is going to discuss details of how to completely accumulate $\mathcal{A}_c$. We have to solve two problems to smoothly accumulate the region. The two problems are (1) how to keep tracking the canonical motion of target objects and (2) what should be the termination condition of the accumulation.

3.2.3.1 Tracking the canonical motion

The first problem of accumulation is how to keep tracking the canonical motion of the target object. Canonical motion requires to continuously keep the target object in contact with the two given fingers. But how can we guarantee the continuity of contacts with programming languages? One solution is to keep in contact with one finger, say $\boldsymbol{f_1}$, rotate round $\boldsymbol{f_1}$ a

little and slip the target object to adjacent boundary points along $\boldsymbol{f_1}$ until it contacts with $\boldsymbol{f_2}$. This solution cannot mathematically ensure continuity and may produce a discretized contacts of "continuous" motion. Fig.3.9 illustrates its details. Note that I will use a simple triangular object to exemplify various algorithms in this sub-section since comparing with the polygons we used in foregoing texts, the triangular object would make the contents clearer.

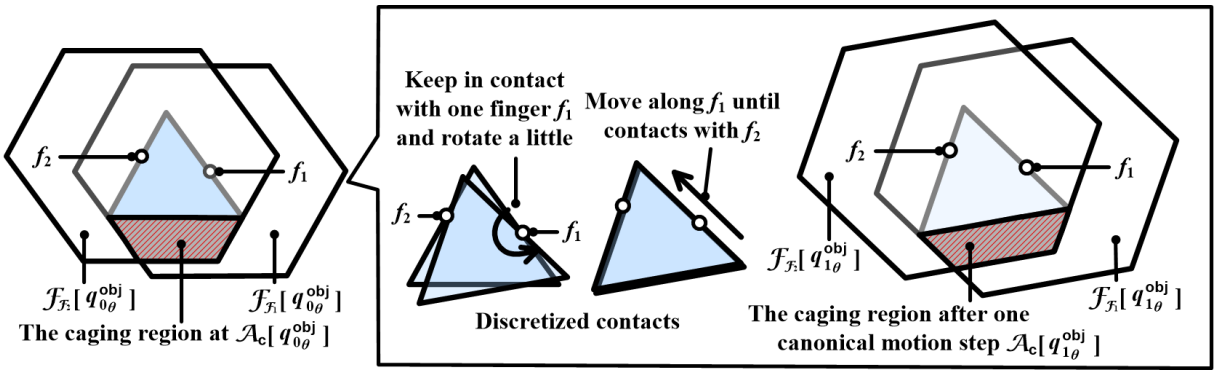


Figure 3.9: A discretized tracking of canonical motion.

This solution can be considered to be a solution of tracking canonical motion in $\mathcal{W}$ space. Although Vahedi does not explicitly describe how he tracks the canonical motion in his Ph.D thesis [Vahedi, 2009], he confirms using this solution during the e-mail discussions between him and me. This solution suffers from several problems. In the first place, it is quite difficult to be programmed with computer languages. For instance, it would be very difficult to "slip the target object to adjacent boundary points along $\boldsymbol{f_1}$ until it contacts with $\boldsymbol{f_2}$" with computer programs. In the second place, there's no ensurance on the continuity of the discretized motion. The solution depends too much on the continuous contacts with $\boldsymbol{f_1}$. It is doubtful whether this dependence can produce the ground-truth canonical motions.

In Erickson's papers, especially the lower part of Fig.3 in [Erickson et al., 2007]. He seems to be tracking the canonical motion in $\mathcal{C}^{\text{obj}}$. Tracking the canonical motion in $\mathcal{C}^{\text{obj}}$ is exactly the same solution as me. However, I am not sure whether Erickson programmed the tracking of canonical motion in this way. It is only a deduction from the lower part of Fig.3. Erickson does not discuss in detail about it.

I will show the details of my solution on how to track the canonical motion in $\mathcal{C}^{\text{obj}}$. Comparing with the discretized $\mathcal{W}$ space tracking adopted by Vahedi, tracking the canonical motion in $\mathcal{C}^{\text{obj}}$ offers more continuity.

Take Fig.3.10 for example. Keeping the target object in contact with both $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$ in $\mathcal{C}^{\text{obj}}$ is actually the same as keeping the configuration of target object in contact with both $\mathcal{F}_1$ and $\mathcal{F}_2$. Therefore, we only need to model $\mathcal{F}_1[q_{i_\theta}^{\text{obj}}]$ and $\mathcal{F}_2[q_{i_\theta}^{\text{obj}}]$ at each layer and calculate their intersections to decide where should the configuration of the target object be to keep in contact with both $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$. The upper part of Fig.3.10 illustrates the canonical motion of the target object in $\mathcal{W}$ space while the lower part of Fig.3.10 illustrates

the correspondent target object configurations in $\mathcal{C}^{\text{obj}}$. Besides this figure, I summarize how to track the target configuration in the next canonical motion step in Alg.1. Readers may combine the renderings in Fig.3.10 and the program flow in Alg.1 to better understand my idea of how to track the canonical motion.

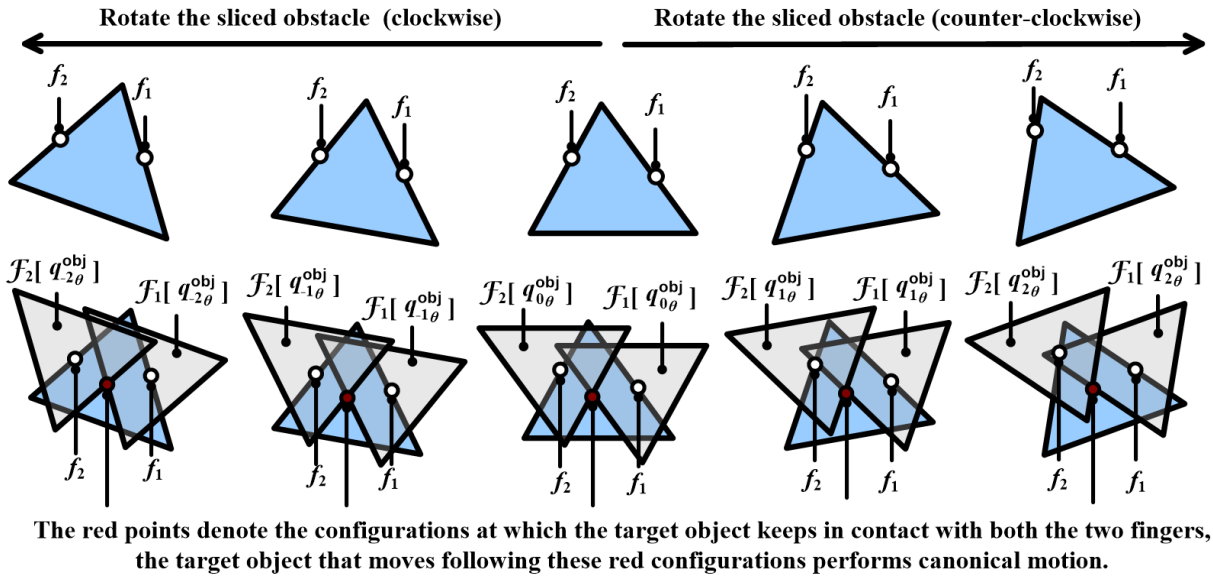


Figure 3.10: Tracking canonical motion in $\mathcal{C}^{\text{obj}}$.

Algorithm 1: Tracking the canonical motion

Data: $\boldsymbol{f_1}$, $\boldsymbol{f_2}$, $\mathsf{O}[\boldsymbol{q_0}]$, $q_{i_\theta}^{\text{obj}}$
Result: Target configuration in the next step $\boldsymbol{q_{i+1}}^{\text{obj}}$

begin
  $\mathcal{F}_1[q_{i_\theta}^{\text{obj}}] \leftarrow$`getConfigurationObstacle(`$\boldsymbol{f_1}$, $\mathsf{O}[\boldsymbol{q_0}]$, $q_{i_\theta}^{\text{obj}}$`)`
  $\mathcal{F}_2[q_{i_\theta}^{\text{obj}}] \leftarrow$`getConfigurationObstacle(`$\boldsymbol{f_2}$, $\mathsf{O}[\boldsymbol{q_0}]$, $q_{i_\theta}^{\text{obj}}$`)`
  /*$\mathcal{P}$ is a set of intersection points.*/
  $\mathcal{P} \leftarrow$`getIntersections(`$\mathcal{F}_1[q_{i_\theta}^{\text{obj}}]$, $\mathcal{F}_2[q_{i_\theta}^{\text{obj}}]$`)`
  if $\mathcal{P}==\emptyset$ **then**
    return null
  $\{q_{i+1_x}^{\text{obj}}, q_{i+1_y}^{\text{obj}}\} \leftarrow$`getNearestPoint(`$\{q_{i_x}^{\text{obj}}, q_{i_y}^{\text{obj}}\}$, $\mathcal{P}$`)`
  $\boldsymbol{q_{i+1}}^{\text{obj}} \leftarrow \{q_{i+1_x}^{\text{obj}}, q_{i+1_y}^{\text{obj}}, q_{i+1_\theta}^{\text{obj}}\}$
  return $\boldsymbol{q_{i+1}}^{\text{obj}}$
end

Specifically, tracking the canonical motion in $\mathcal{C}^{\text{obj}}$ involves two steps. In the first step, both

$\mathcal{F}_1[q^{\mathsf{obj}}_{0_\theta}]$ and $\mathcal{F}_2[q^{\mathsf{obj}}_{0_\theta}]$ are continuously rotated with small steps, say from $\mathcal{F}_1[q^{\mathsf{obj}}_{0_\theta}]$ to $\mathcal{F}_1[q^{\mathsf{obj}}_{1_\theta}]$, $\mathcal{F}_1[q^{\mathsf{obj}}_{2_\theta}]$, ..., from $\mathcal{F}_2[q^{\mathsf{obj}}_{0_\theta}]$ to $\mathcal{F}_2[q^{\mathsf{obj}}_{1_\theta}]$, $\mathcal{F}_2[q^{\mathsf{obj}}_{2_\theta}]$, ...and from $\mathcal{F}_1[q^{\mathsf{obj}}_{0_\theta}]$ to $\mathcal{F}_1[q^{\mathsf{obj}}_{-1_\theta}]$, $\mathcal{F}_1[q^{\mathsf{obj}}_{-2_\theta}]$, ..., from $\mathcal{F}_2[q^{\mathsf{obj}}_{0_\theta}]$ to $\mathcal{F}_2[q^{\mathsf{obj}}_{-1_\theta}]$, $\mathcal{F}_2[q^{\mathsf{obj}}_{-2_\theta}]$, ... At each $q^{\mathsf{obj}}_{i_\theta}$ there are two intersection points between $\mathcal{F}_1[q^{\mathsf{obj}}_{i_\theta}]$ and $\mathcal{F}_2[q^{\mathsf{obj}}_{i_\theta}]$. One of the intersection point locates in a further region while the other intersection point locates in a nearer region. The intersection points can be easily found in the lower part of Fig.3.10. The succeeding intersection in the nearer region to previous configuration is selected as an ensuing canonical motion step. For example, the intersection of $\mathcal{F}_1[q^{\mathsf{obj}}_{1_\theta}]$ and $\mathcal{F}_2[q^{\mathsf{obj}}_{1_\theta}]$ in the nearer region to $\{q^{\mathsf{obj}}_{0_\mathsf{x}}, q^{\mathsf{obj}}_{0_\mathsf{y}}, q^{\mathsf{obj}}_{0_\theta}\}$ is selected as $\{q_{1_\mathsf{x}}, q_{1_\mathsf{y}}, q^{\mathsf{obj}}_{1_\theta}\}$. The intersection of $\mathcal{F}_1[q^{\mathsf{obj}}_{2_\theta}]$ and $\mathcal{F}_2[q^{\mathsf{obj}}_{2_\theta}]$ in the nearer region to $\{q^{\mathsf{obj}}_{1_\mathsf{x}}, q^{\mathsf{obj}}_{1_\mathsf{y}}, q^{\mathsf{obj}}_{1_\theta}\}$ is selected as $\{q^{\mathsf{obj}}_{2_\mathsf{x}}, q^{\mathsf{obj}}_{2_\mathsf{y}}, q^{\mathsf{obj}}_{2_\theta}\}$, and so on. By calculating each configuration $\{q^{\mathsf{obj}}_{i_\mathsf{x}}, q^{\mathsf{obj}}_{i_\mathsf{y}}, q^{\mathsf{obj}}_{i_\theta}\}$ during the rotation, we can uniformly generate the whole canonical motion.

3.2.3.2 Termination conditions

Besides canonical motion tracking and intersection, the accumulation procedure should be stopped according to certain conditions. This is because complete accumulation along whole $[-\pi, \pi)$ introduces fatal excessive regions which break caging.

Specifically, we have two termination conditions for accumulation. The first one is the disappearance of intersection points and the other one is when $\mathcal{A}_\mathsf{c}[q^{\mathsf{obj}}_{i_\theta}]$ becomes empty. On the one hand, if the intersection points between $\mathcal{F}_1[q^{\mathsf{obj}}_{i_\theta}]$ and $\mathcal{F}_2[q^{\mathsf{obj}}_{i_\theta}]$ becomes null as the target object moves canonically, the target object could escape from the gap between $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$. We must make sure $\mathcal{F}_1[q^{\mathsf{obj}}_{i_\theta}]$ and $\mathcal{F}_2[q^{\mathsf{obj}}_{i_\theta}]$ stop the rotation before the intersections become null. Say, only the accumulated $\mathcal{A}_\mathsf{c}$ before the disappearance of intersections constitutes the complete caging region. On the other hand, when $\mathcal{A}_\mathsf{c}[q^{\mathsf{obj}}_{i_\theta}]$ becomes empty, we can never translationally cage the target object by posing a third finger $\boldsymbol{f_3}$ in a certain empty region and it is unnecessary to make further accumulation.

Now let us accumulate the $\mathcal{A}_\mathsf{c}$ of the triangle object and see the roles of the termination conditions. Here I introduce two extra concepts to accumulate the complete $\mathcal{A}_\mathsf{c}$. One concept is the potential caging region $\mathcal{A}_\mathsf{pc}$ and the other concept is the determinate caging region $\mathcal{A}_\mathsf{dc}$. Like their names, the potential caging region is a region which potentially includes a part of the final caging region. A finger in this region only potentially cage the target object. It is not a determinate caging region. Some parts of the potential caging region may be in the final $\mathcal{A}_\mathsf{c}$ while some other parts may be not. The determinate caging region is determinately one part of the final $\mathcal{A}_\mathsf{c}$. All of the determinate caging region belong to (are sub-regions of) the final $\mathcal{A}_\mathsf{c}$. Fig.3.11 demonstrates the procedure of how to calculate and accumulate $\mathcal{A}_\mathsf{pc}$ and $\mathcal{A}_\mathsf{dc}$ at two rotational orientations $q^{\mathsf{obj}}_{0_\theta}$ and $q^{\mathsf{obj}}_{1_\theta}$.

At the initial layer $q^{\mathsf{obj}}_{0_\theta}$ the potential caging region $\mathcal{A}_\mathsf{pc}$, or $\mathcal{A}_\mathsf{pc}[q^{\mathsf{obj}}_{0_\theta}]$ is exactly the same as $\mathcal{A}_\mathsf{c}[q^{\mathsf{obj}}_{0_\theta}]$. As we track the canonical motion, or as we rotate the sliced obstacles, the $\mathcal{A}_\mathsf{pc}[q^{\mathsf{obj}}_{i_\theta}]$ no longer equals $\mathcal{A}_\mathsf{c}[q^{\mathsf{obj}}_{i_\theta}]$. For example, at layer $q^{\mathsf{obj}}_{1_\theta}$, the $\mathcal{A}_\mathsf{pc}[q^{\mathsf{obj}}_{1_\theta}]$ is the intersection between $\mathsf{O}[\boldsymbol{q_1}] \cup \mathcal{A}_\mathsf{c}[q^{\mathsf{obj}}_{1_\theta}]$ and $\mathcal{A}_\mathsf{pc}[q^{\mathsf{obj}}_{0_\theta}]$. The orange polygon in the dialogue box of Fig.3.11 shows

this accumulation between $\mathcal{A}_{\mathsf{pc}}$ and $\mathcal{A}_{\mathsf{c}}$. More generally this accumulation can be written as expression (3.10) and expression (3.11).

$$\mathcal{A}_{\mathsf{pc}}[q_{i_\theta}^{\mathsf{obj}}] = (\mathsf{O}[\boldsymbol{q_i}] \cup \mathcal{A}_{\mathsf{c}}[q_{i_\theta}^{\mathsf{obj}}]) \cap \mathcal{A}_{\mathsf{pc}}[q_{i-1_\theta}^{\mathsf{obj}}],\ i = 1, 2, \ldots, m \tag{3.10}$$

$$\mathcal{A}_{\mathsf{pc}}[q_{i_\theta}^{\mathsf{obj}}] = (\mathsf{O}[\boldsymbol{q_i}] \cup \mathcal{A}_{\mathsf{c}}[q_{i_\theta}^{\mathsf{obj}}]) \cap \mathcal{A}_{\mathsf{pc}}[q_{i+1_\theta}^{\mathsf{obj}}],\ i = -1, -2, \ldots, -m \tag{3.11}$$

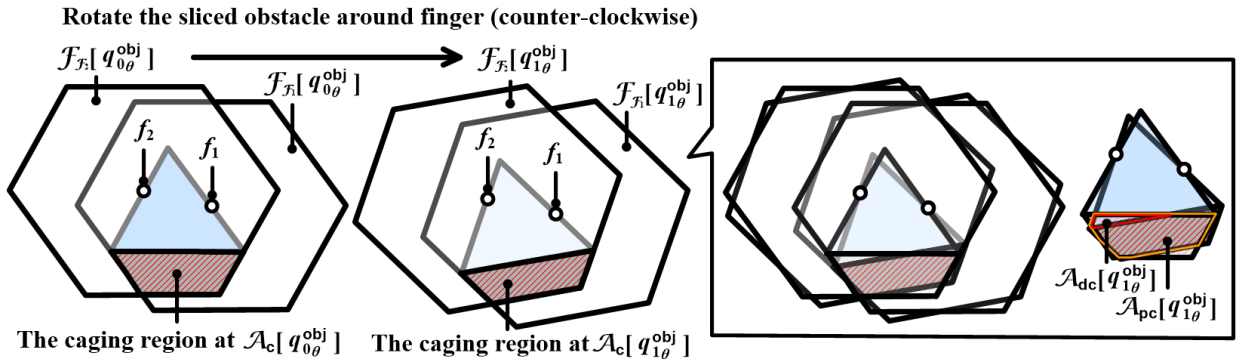


Figure 3.11: Accumulation of the potential caging region and the determinate caging region.

The determinate caging region of a layer, or namely $\mathcal{A}_{\mathsf{dc}}[q_{i_\theta}^{\mathsf{obj}}]$, is a sub-region of the potential caging region at layer $q_{i-1_\theta}^{\mathsf{obj}}, i > 1$ or $q_{i+1_\theta}^{\mathsf{obj}}, i < 1$. The red polygon in the dialogue box of Fig.3.11 demonstrates this determinate caging region of a layer. It can be formally expressed as following.

$$\mathcal{A}_{\mathsf{dc}}[q_{i_\theta}^{\mathsf{obj}}] = \mathsf{O}[\boldsymbol{q_i}] \cap \mathcal{A}_{\mathsf{pc}}[q_{i-1_\theta}^{\mathsf{obj}}],\ i = 1, 2, \ldots, m \tag{3.12}$$

$$\mathcal{A}_{\mathsf{dc}}[q_{i_\theta}^{\mathsf{obj}}] = \mathsf{O}[\boldsymbol{q_i}] \cap \mathcal{A}_{\mathsf{pc}}[q_{i+1_\theta}^{\mathsf{obj}}],\ i = -1, -2, \ldots, -m \tag{3.13}$$

The whole $\mathcal{A}_{\mathsf{dc}}$ along all layers is the union of the determinate caging regions at each $q_{i_\theta}^{\mathsf{obj}}$. When we are calculating the whole $\mathcal{A}_{\mathsf{dc}}$, we need to take care of the termination conditions. If the intersection points between $\mathcal{F}_1[q_{1_\theta}^{\mathsf{obj}}]$ and $\mathcal{F}_2[q_{1_\theta}^{\mathsf{obj}}]$ becomes null, then caging will break between $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$. The whole $\mathcal{A}_{\mathsf{dc}}$ does not exist. If the intersection points always exists but the $\mathcal{A}_{\mathsf{c}}[q_{i_\theta}^{\mathsf{obj}}]$ becomes empty at certain layers, we need to make union of all available $\mathcal{A}_{\mathsf{dc}}[q_{i_\theta}^{\mathsf{obj}}]$ before empty. More concretely, if the $\mathcal{A}_{\mathsf{c}}[q_{i_\theta}^{\mathsf{obj}}]$ exists between $-j$ and k where $-j$ denotes a number in $(-m, 0)$ and k denotes a number in $(0, m)$. The whole $\mathcal{A}_{\mathsf{c}}$ would be

$$\mathcal{A}_{\mathsf{dc}} = \mathcal{A}_{\mathsf{dc}}^{+} \cap \mathcal{A}_{\mathsf{dc}}^{-} \tag{3.14}$$

where $\mathcal{A}_{\mathsf{dc}}^{+}$ denotes the union of $\mathcal{A}_{\mathsf{c}}[q_{i_\theta}^{\mathsf{obj}}]$ along positive axis or along counter-clockwise rotation while where $\mathcal{A}_{\mathsf{dc}}^{-}$ denotes the union of $\mathcal{A}_{\mathsf{c}}[q_{i_\theta}^{\mathsf{obj}}]$ along negative axis or along clockwise rotation. They can be formally written as

$$\mathcal{A}^{+}_{\mathsf{dc}} = \bigcup_{i=1}^{k} (\mathsf{O}[\boldsymbol{q_i}] \cap \mathcal{A}_{\mathsf{pc}}[q^{\mathsf{obj}}_{i-1_\theta}]) \tag{3.15}$$

$$\mathcal{A}^{-}_{\mathsf{dc}} = \bigcup_{i=-1}^{-j} (\mathsf{O}[\boldsymbol{q_i}] \cap \mathcal{A}_{\mathsf{pc}}[q^{\mathsf{obj}}_{i+1_\theta}]) \tag{3.16}$$

Alg.2 summaries the algorithm flow of how to make union and terminate along the positive axis. Note that the second termination condition in this algorithm is not exactly the same as our analysis which requires $\mathcal{A}_{\mathsf{c}}[q^{\mathsf{obj}}_{i_\theta}] == \mathsf{null}$. It might be clearer if we employ $\mathcal{A}_{\mathsf{pc}}[q^{\mathsf{obj}}_{i_\theta}] == \mathsf{null}$. Nevertheless, I did not employ $\mathcal{A}_{\mathsf{pc}}[q^{\mathsf{obj}}_{i_\theta}] == \mathsf{null}$ is because when $\mathcal{A}_{\mathsf{c}}[q^{\mathsf{obj}}_{i_\theta}] == \mathsf{null}$, the following $\mathcal{A}_{\mathsf{dc}}[q^{\mathsf{obj}}_{i+1_\theta}] == \mathsf{null}$ will always be a subset of $\mathcal{A}_{\mathsf{dc}}[q^{\mathsf{obj}}_{i_\theta}] == \mathsf{null}$. It is unnecessary to repeatedly make union of them. We are actually making union of the caging regions that will be obstructed by the canonical motion of O. Surely we need to consider a special case where all caging regions are outside O. That is a very special case and we will discuss about it soon. Alg.2 makes union of the caging regions along positive axis. Like the previous discussion of expression (3.14), the whole $\mathcal{A}_{\mathsf{dc}}$ not only involves the positive union but also involves the negative union. It should be a intersection of these two region unions.

Note that the whole $\mathcal{A}_{\mathsf{dc}}$ is not the complete caging region. There are some special cases where we need to further take into account the caging regions that are outside the target object O. The example in the upper part of Fig.2.13 illustrates one of this case. A circular object, no matter how it changes the orientation, its $\mathcal{A}_{\mathsf{pc}}[q^{\mathsf{obj}}_{i_\theta}]$ maintains the same through out all $q^{\mathsf{obj}}_{i_\theta}$ and its $\mathsf{O}[\boldsymbol{q_{i+1}}]$ never intersects with $\mathcal{A}_{\mathsf{pc}}[q^{\mathsf{obj}}_{i_\theta}]$. Fig.3.12 demonstrates this idea[2]. The empty intersections between $\mathsf{O}[\boldsymbol{q_{i+1}}]$ and $\mathcal{A}_{\mathsf{pc}}[q^{\mathsf{obj}}_{i_\theta}]$ mean that the result of Alg.2 is always $\emptyset$. This is contradictory with our previous solution in expression (3.14). Actually, we should take special treatment to deal with this special case. Comparing with Alg.2, the special treatment is simple, namely $\bigcup_{i=-m}^{m} (\mathcal{A}_{\mathsf{c}}[q^{\mathsf{obj}}_{i_\theta}])$. The complete caging region $\mathcal{A}_{\mathsf{c}}$, should be a union of $\mathcal{A}_{\mathsf{dc}}$ and this special treatment (see expression (3.17)).

$$\mathcal{A}_{\mathsf{c}} = \mathcal{A}_{\mathsf{dc}} \cup (\bigcup_{i=-m}^{m} (\mathcal{A}_{\mathsf{c}}[q^{\mathsf{obj}}_{i_\theta}])) = (\mathcal{A}^{+}_{\mathsf{dc}} \cap \mathcal{A}^{-}_{\mathsf{dc}}) \cup (\bigcap_{i=-m}^{m} (\mathcal{A}_{\mathsf{c}}[q^{\mathsf{obj}}_{i_\theta}])) \tag{3.17}$$

3.2.4 The caging region of a third finger – Demonstrations

Now let us see how my proposal works with the triangle object. The triangle object is simple and it can be analyzed step by step to observe the performance of my algorithms. Fig.3.13 shows the details. The upper row of Fig.3.13 illustrates the changes of $\mathcal{A}_{\mathsf{c}}[q^{\mathsf{obj}}_{i_\theta}]$ as i increases

[2]According to [van der Stappen, 2005], circular objects are the only special case. But I did not find any explicit proof about that. Therefore my algorithm follows exactly expression (3.17)

Algorithm 2: Computing the $\mathcal{A}^{+}_{\mathsf{dc}}$

Data: $\boldsymbol{f_1}$, $\boldsymbol{f_2}$, $\mathsf{O}[\boldsymbol{q_0}]$
Result: $\mathcal{A}^{+}_{\mathsf{dc}}$

begin
  i$\leftarrow 0$
  $\mathcal{A}^{+}_{\mathsf{dc}} \leftarrow$null
  $\mathcal{F}_1[q^{\mathsf{obj}}_{0_\theta}] \leftarrow$`getCObstacle(`$\boldsymbol{f_1}$, $\mathsf{O}[\boldsymbol{q_0}]$, $q^{\mathsf{obj}}_{0_\theta}$`)`
  $\mathcal{F}_2[q^{\mathsf{obj}}_{0_\theta}] \leftarrow$`getCObstacle(`$\boldsymbol{f_2}$, $\mathsf{O}[\boldsymbol{q_0}]$, $q^{\mathsf{obj}}_{0_\theta}$`)`
  $\mathcal{F}_{\mathcal{F}_1}[q^{\mathsf{obj}}_{0_\theta}] \leftarrow$`getCCObstacle(` $\mathcal{F}_1[q^{\mathsf{obj}}_{0_\theta}]$`)`
  $\mathcal{F}_{\mathcal{F}_2}[q^{\mathsf{obj}}_{0_\theta}] \leftarrow$`getCCObstacle(` $\mathcal{F}_2[q^{\mathsf{obj}}_{0_\theta}]$`)`
  $\mathcal{A}_{\mathsf{c}}[q^{\mathsf{obj}}_{0_\theta}] \leftarrow$`getTranslationalCagingRegion(` $\mathcal{F}_{\mathcal{F}_1}[q^{\mathsf{obj}}_{0_\theta}]$, $\mathcal{F}_{\mathcal{F}_2}[q^{\mathsf{obj}}_{0_\theta}]$, $\mathsf{O}[\boldsymbol{q_0}]$`)`
  $\mathcal{A}_{\mathsf{pc}}[q^{\mathsf{obj}}_{0_\theta}] == \mathcal{A}_{\mathsf{c}}[q^{\mathsf{obj}}_{0_\theta}]$
  while $i \leq m$ **do**
    $\mathcal{F}_1[q^{\mathsf{obj}}_{i_\theta}] \leftarrow$`getCObstacle(`$\boldsymbol{f_1}$, $\mathsf{O}[\boldsymbol{q_0}]$, $q^{\mathsf{obj}}_{i_\theta}$`)`
    $\mathcal{F}_2[q^{\mathsf{obj}}_{i_\theta}] \leftarrow$`getCObstacle(`$\boldsymbol{f_2}$, $\mathsf{O}[\boldsymbol{q_0}]$, $q^{\mathsf{obj}}_{i_\theta}$`)`
    $\boldsymbol{q_{i+1}} \leftarrow$`Alg.1(`$\mathcal{F}_1[q^{\mathsf{obj}}_{i_\theta}]$, $\mathcal{F}_2[q^{\mathsf{obj}}_{i_\theta}]$, $\mathsf{O}[\boldsymbol{q_0}]$, $q^{\mathsf{obj}}_{i_\theta}$`)`
    /*Termination condition 1*/
    if $\boldsymbol{q_{i+1}}$ ==null **then**
      break
    $\mathcal{F}_{\mathcal{F}_1}[q^{\mathsf{obj}}_{i+1_\theta}] \leftarrow$`getCCObstacle(` $\mathcal{F}_1[q^{\mathsf{obj}}_{i+1_\theta}]$`)`
    $\mathcal{F}_{\mathcal{F}_2}[q^{\mathsf{obj}}_{i+1_\theta}] \leftarrow$`getCCObstacle(` $\mathcal{F}_2[q^{\mathsf{obj}}_{i+1_\theta}]$`)`
    $\mathcal{A}_{\mathsf{c}}[q^{\mathsf{obj}}_{i+1_\theta}] \leftarrow$`getTranslationalCagingRegion(` $\mathcal{F}_{\mathcal{F}_1}[q^{\mathsf{obj}}_{i+1_\theta}]$, $\mathcal{F}_{\mathcal{F}_2}[q^{\mathsf{obj}}_{i+1_\theta}]$, $\mathsf{O}[\boldsymbol{q_{i+1}}]$`)`
    $\mathcal{A}^{+}_{\mathsf{dc}} \leftarrow \mathcal{A}^{+}_{\mathsf{dc}} \cup (\mathsf{O}[\boldsymbol{q_{i+1}}] \cap \mathcal{A}_{\mathsf{pc}}[q^{\mathsf{obj}}_{i_\theta}])$
    $\mathcal{A}_{\mathsf{pc}}[q^{\mathsf{obj}}_{i+1_\theta}] \leftarrow \mathcal{A}_{\mathsf{pc}}[q^{\mathsf{obj}}_{i_\theta}] \cap (\mathcal{A}_{\mathsf{c}}[q^{\mathsf{obj}}_{i+1_\theta}] \cup \mathsf{O}[\boldsymbol{q_{i+1}}])$
    /*Termination condition 2*/
    if $\mathcal{A}_{\mathsf{c}}[q^{\mathsf{obj}}_{i_\theta}]$ ==null **then**
      break
    $i \leftarrow i + 1$
  end
  return $\mathcal{A}^{+}_{\mathsf{dc}}$
end

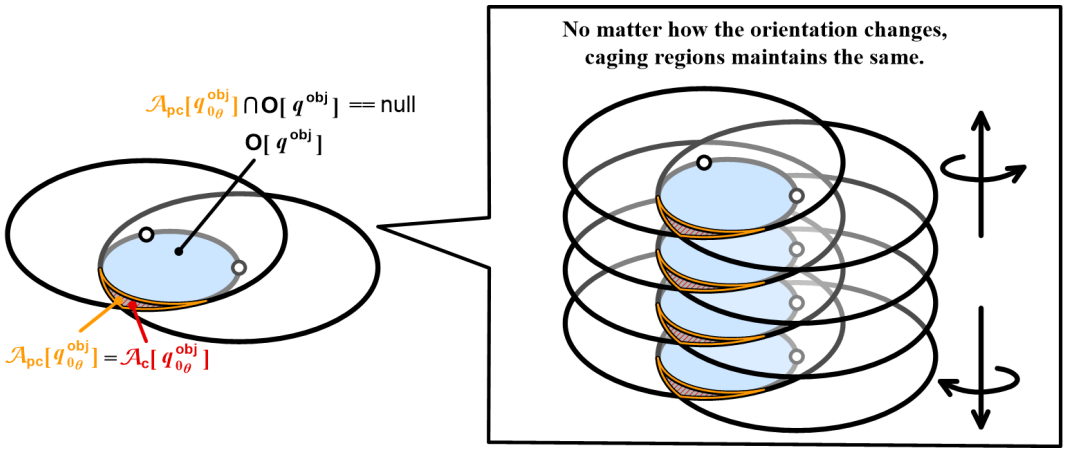


Figure 3.12: A special case where all caging regions are outside the target object.

from 0. The granularity of between $q^{obj}_{i_\theta}$ and $q^{obj}_{i+1_\theta}$ in this case is 10°. The lower row of Fig.3.13 illustrates the changes of $\mathcal{A}_c[q^{obj}_{i_\theta}]$ as i decreases from 0. It uses the same granularity as the upper row. The accumulation stops when at certain $-j$ and k the $\mathcal{A}_c[q^{obj}_{-j_\theta}]$ and $\mathcal{A}_c[q^{obj}_{k_\theta}]$ disappears. In Fig.3.13, $-j$ and k are -7 and 4 respectively. By using these $\mathcal{A}_c[i^{obj}_{k_\theta}]$ and by following the algorithm proposed in Alg.2 and expression (3.17), we can accumulate the complete caging region for a third finger. Fig.3.14 shows the accumulation procedure.

In correspondence with Fig.3.13, Fig.3.14 is divided into an upper row and a lower row too. The upper row of Fig.3.14 illustrates the changes of $\mathcal{A}_{pc}[q^{obj}_{i_\theta}]$ and $\mathcal{A}_{dc}[q^{obj}_{i_\theta}]$ as i increases from 0 to 4. The lower row of Fig.3.14 illustrates the changes of $\mathcal{A}_{pc}[q^{obj}_{i_\theta}]$ and $\mathcal{A}_{dc}[q^{obj}_{i_\theta}]$ as i decreases from 0 to -7. Readers may compare expressions (3.10), (3.11), (3.12) and (3.13) to better understand how are the orange polygons and the red polygons calculated. $\mathcal{A}^+_{dc}$ and $\mathcal{A}^-_{dc}$, as has been shown in expression (3.15) and expression (3.16) respectively, are the union of the caging regions along positive axis and the union of the caging regions along negative axis. The overlapping polygons in the right part of Fig.3.14 illustrate the unions. Those overlapping polygons rendered with red color are the union along positive axis, namely $\mathcal{A}^+_{dc}$ while the overlapping polygons rendered with green color are the union along negative axis, namely $\mathcal{A}^-_{dc}$.

The complete $\mathcal{A}_c$, following expression (3.17), is the intersection of $\mathcal{A}^+_{dc}$ and $\mathcal{A}^-_{dc}$ plus the $\cup(\bigcap_{i=-m}^{m}(\mathcal{A}_c[q^{obj}_{i_\theta}]))$. In the case of this triangle object, $\cup(\bigcap_{i=-m}^{m}(\mathcal{A}_c[q^{obj}_{i_\theta}]))$ is null since when $i == -4$ and $i == -7$, $\mathcal{A}_c[q^{obj}_{i_\theta}]$ is null. Therefore, the complete $\mathcal{A}_c$ of this triangle object and the fingers $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$ is $\mathcal{A}^+_{dc} \cap \mathcal{A}^-_{dc}$. This region is rendered with white color in Fig.3.15[3].

[3]Since I the difference between an orientation $q^{obj}_{i_\theta}$ and its adjacent orientation $q^{obj}_{i+1_\theta}$ is set to $10 degree$, the final result in this figure involves some zigzags. The zigzags can be eliminated by reducing the difference between orientation steps.

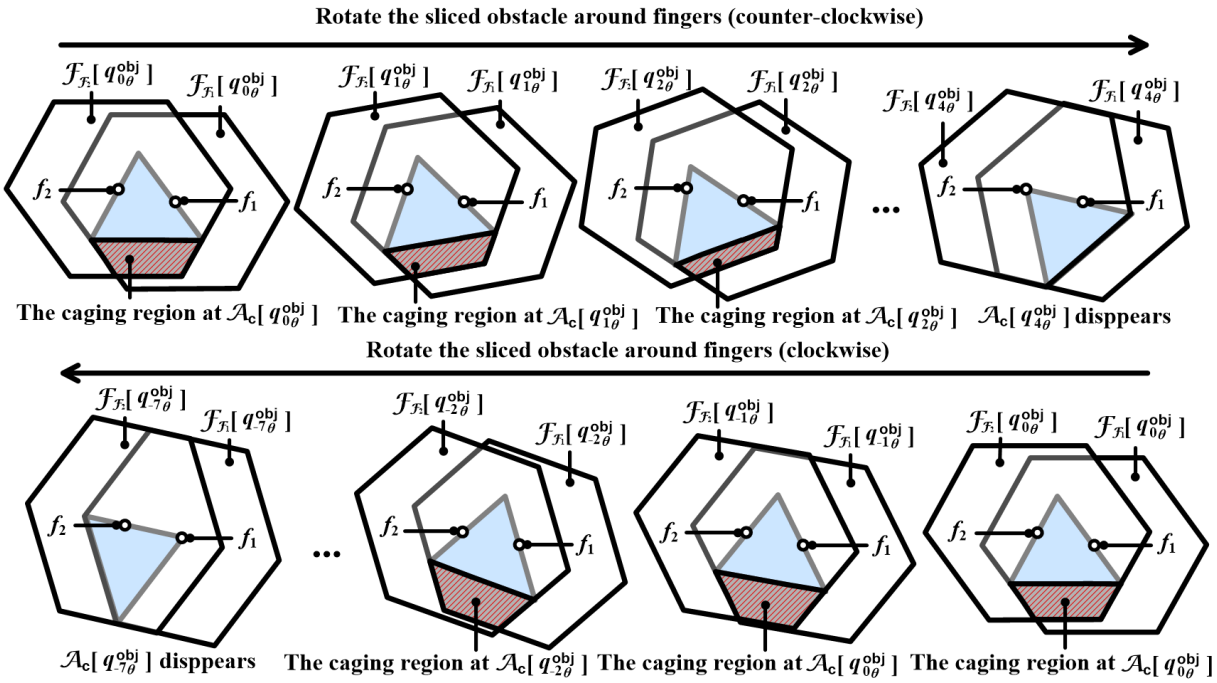


Figure 3.13: A step-by-step analysis of the $\mathcal{A}_c[q^{obj}_{i\theta}]$.

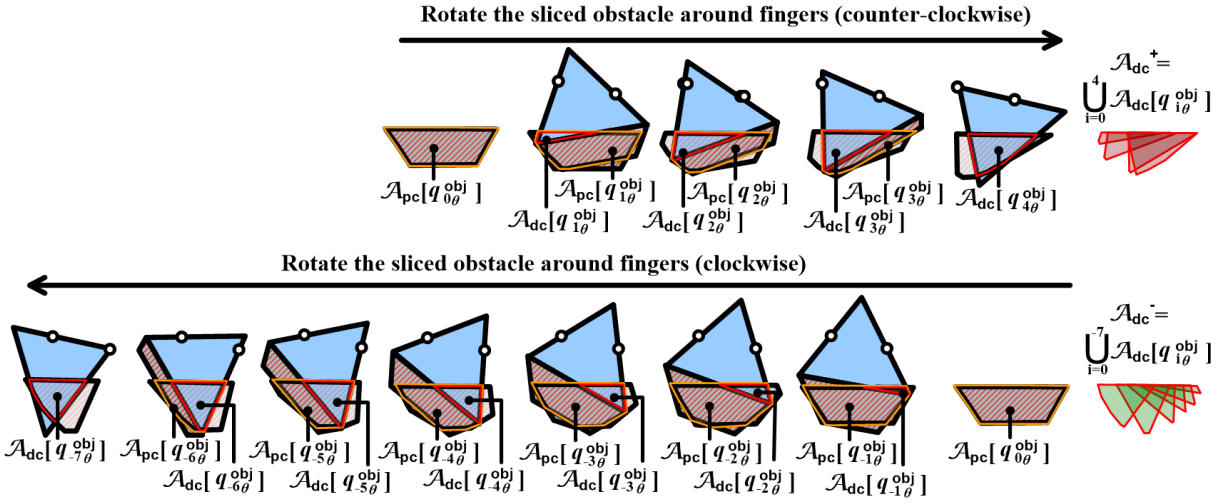


Figure 3.14: A step by step analysis of the $\mathcal{A}_{pc}[q^{obj}_{i\theta}]$ and $\mathcal{A}_{dc}[q^{obj}_{i\theta}]$.

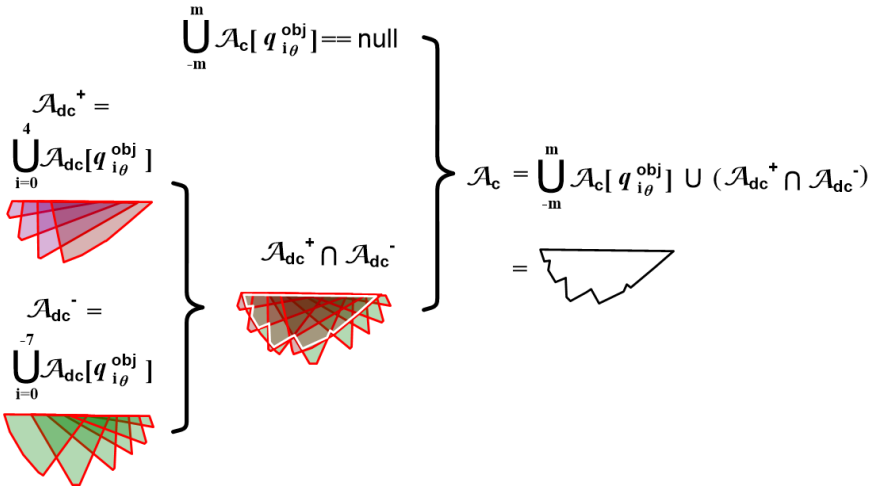


Figure 3.15: The complete caging region for a third finger of the triangle example.

Given a polygon of n_v boundary points, the computational complexity of this algorithm would be $O(n_v^2 \cdot m)$. This exponential complexity is acceptable to deal with one pair of given $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$.

Actually, for practical usage, it is unnecessary to union all the $\mathcal{A}_c[q_{i\theta}^{obj}])$ before it becomes null. We can stop at user-defined orientations, or namely we can perform the algorithm interactively. Take Alg.2 for example. If the second termination condition of this algorithm is not $\mathcal{A}_c[q_{i\theta}^{obj}] ==$ null but another one, say break the loop when $i == l$ where l is a terminating layer defined by users, then, the result caging region for a third finger would be a subset of the groundtruth value. Sometimes this subset is enough for practical usage and we can use it to reduce the cost of computational resources.

Fig.3.16 demonstrates with the same triangle object and shows the result of accumulation when $l = +/-m$, $l = +/-4$, $l = +/-3$, $l = +/-2$ and $l = +/-1$. Note that when $l = +/-m$, the algorithm is actually the one showed in Alg.2 and the accumulated $\mathcal{A}_c$ is the groundtruth region.

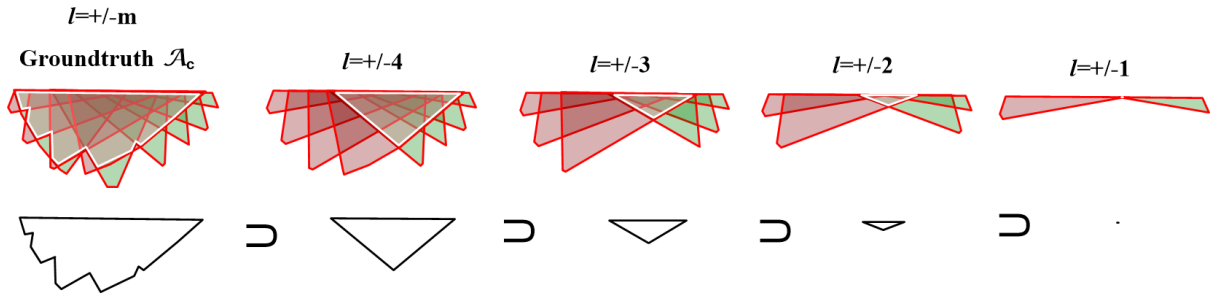


Figure 3.16: Terminating layers can be set manually to save computational resources.

As we can see in Fig.3.16 that the reduced results at $l = +/-4$, $l = +/-3$, $l = +/-2$ and $l = +/-1$ are subsets of the groundtruth region. I name these reduced results the sub-caging regions. In rough environments, these sub-caging regions are enough for practical usage. The idea of setting terminating layers share the same basis with [Pereira et al., 2004] which sets some stopping angles for rotation. [Pereira et al., 2004]'s algorithm with stopping angles is successfully applied to multi-robot cooperative transportation. Their results support my claim here. If we calculate the sub-caging regions instead of the groundtruth caging region, the computational complexity would decrease from $O(n_v^2 \cdot m)$ to $O(n_v^2)$ since m becomes a constant.

One significant problem of setting the terminating layers is we have no apriori-knowledge of how to set the positive terminating layer and the negative terminating layer with different values. In Fig.3.16, both positive terminating layer and negative terminating layer are set to the same absolute value. However, it causes certain problems. When $l = +/-m$, the actual positive terminating layer and the actual negative terminating layer are $+4$ and -7 respectively. We have analyzed these two values in foregoing contexts. Since the their absolute value differs from each other, there is a bias in the goundtruth caging region. I recommend readers refer to the left shape in the second row of Fig.3.16 for better comprehension. This shape is not symmetric. Its left part is a fatter than its right part. That is because the different $+4$ and -7 values bring into bias. When $l = +/-4$, $l = +/-3$, $l = +/-2$ or $l = +/-1$, the actual positive terminating layer and actual negative terminating layer share the same absolute value. The same value eliminates the inherent bias of groundtruth caging region. We can see from the other shapes in the lower row of Fig.3.16 that they are nearly symmetric. The left parts are no longer fatter than the right parts. Therefore, **we should not use the sub-caging regions for caging optimization**. These sub-caging regions no longer include metrics of caging. They can only denote whether a third finger inside them could cage the target objects.

3.2.5 The caging region of a third finger – Implementations

The algorithms of "finding the caging region of a third finger" are implemented and tested with Python programming language and Webots robot simulation software. Specific information about Webots can be found in its official website [Webots, 2013] while a brief introduction can be found in this journal paper [Michel, 2004]. I use Webots to simulate a range scanner. With Webots, we can change the resolution of the range scanner very easily. In my implementation, the resolutions are set to 1∼4 *pixels per centimeter*. Fig.3.17 shows the scanning procedure and the 3D cloud points collected by scanning a sphere object. The red line in Fig.3.17(a) illustrates the virtual laser beam of the range scanner in $\mathcal{W}$ space while Fig.3.17(b) shows the 3D cloud points collected by the scanning procedure. The 3D cloud points collected are filtered into boundary cloud points and employed for calculation of $\mathcal{A}_c$. The red frame in the right part of Fig.3.17 shows the details of filtering. The algorithms in the filtering procedure, like edge detection and boundary following, are quite common in image processing literature. I am not going to repeatedly cite them here. As

for simplification, the strategy employed here is re-sampling of the original boundary points. This simplification strategy has no special purpose. It only aims at improving calculation speed. The scene of this simulation is modeled according to our distributed end-effector introduced in section 1.2. I will introduce the other details of this scene in the next chapter when discusses about real-world applications. Let us concentrate on the ranger scanner and the implementation of my algorithms in this section.

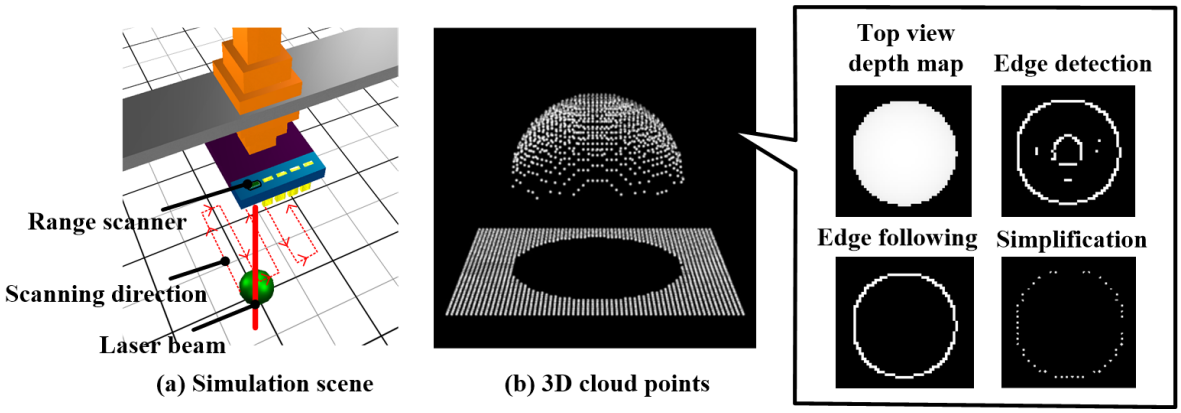


Figure 3.17: The scanning procedure of a range scanner and processing of 3D point clouds.

I build three different rigid polytopes into the simulation environment to validate the algorithms. The scanned and filtered boundary cloud points of these three rigid polytopes, together with their positions and orientations, will be employed as target objects O and initial configuration $\boldsymbol{q_0}$. These three objects and their boundary point clouds are shown in Fig.3.18. They are named Target Object A, Target Object B and Target Object C, respectively.

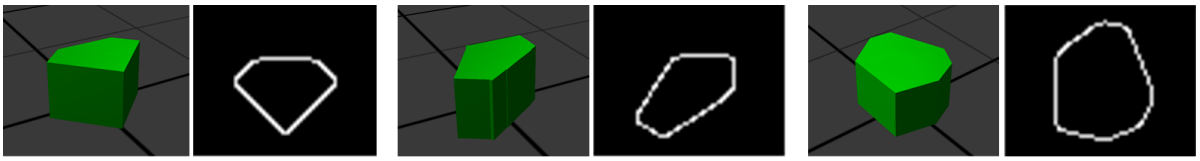


Figure 3.18: Three different objects are employed to validate the implementation.

Fig.3.19 shows the results of Target Object B. When performing the algorithm on this object, I give the positions of $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$ respectively in advance. They are rendered with blue points in Fig.3.19. The blue region in Fig.3.19(a) shows the region taken up by $\mathsf{O}[\boldsymbol{q_0}]$ while the green region in Fig.3.19(a) shows the $\mathcal{F}_{\mathcal{F}_1}[q_{0_\theta}^{\mathsf{obj}}]$. Likewise, the blue region in Fig.3.19(b) shows the region taken up by $\mathsf{O}[\boldsymbol{q_0}]$ while the red region in Fig.3.19(a) shows the $\mathcal{F}_{\mathcal{F}_2}[q_{0_\theta}^{\mathsf{obj}}]$. The intersections of $\mathcal{F}_{\mathcal{F}_1}[q_{0_\theta}^{\mathsf{obj}}]$ and $\mathcal{F}_{\mathcal{F}_2}[q_{0_\theta}^{\mathsf{obj}}]$ are rendered with white color in Fig.3.19(c). Note that the $\mathcal{A}_{\mathsf{c}}[q_{i_\theta}^{\mathsf{obj}}]$ at this layer should not only fulfill expression (3.8) but also expression

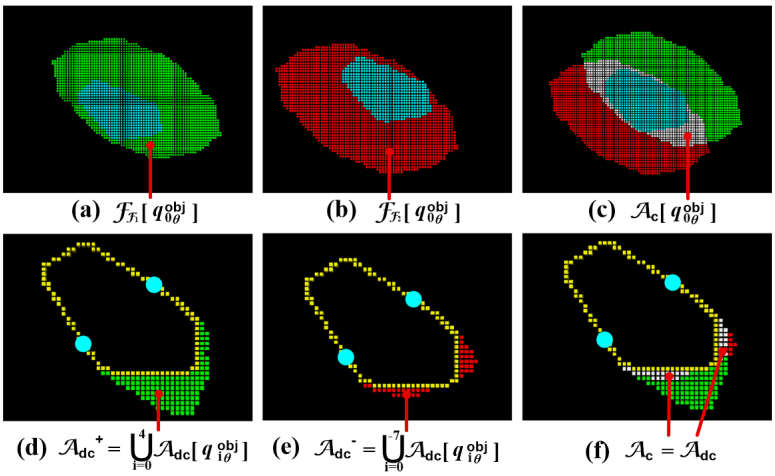


Figure 3.19: The detailed results of my algorithms on Target Object B.

(3.9). Therefore, only the lower part of the white regions in Fig.3.19(c) is kept as $\mathcal{A}_{\mathsf{c}}[q_{i_\theta}^{\mathsf{obj}}]$. Fig.3.19(d), Fig.3.19(e) and Fig.3.19(f) show $\mathcal{A}_{\mathsf{dc}}^{+}$, $\mathcal{A}_{\mathsf{dc}}^{-}$ and $\mathcal{A}_{\mathsf{dc}}$ respectively. They are rendered with red color in Fig.3.19(d), green color in Fig.3.19(e) and white color in Fig.3.19(f). Note that since in the case of Target Object B, where $\bigcup_{i=-m}^{m} (\mathcal{A}_{\mathsf{c}}[q_{i_\theta}^{\mathsf{obj}}]) == \mathsf{null}$, the complete $\mathcal{A}_{\mathsf{c}} = \mathcal{A}_{\mathsf{dc}}$. It is the white region in Fig.3.19(f).

Fig.3.20 shows some more results of the three target objects with different given positions of fingers. Like Fig.3.19, the complete $\mathcal{A}_{\mathsf{c}}$ is rendered with white color. $\mathcal{A}_{\mathsf{dc}}^{+}$ and $\mathcal{A}_{\mathsf{dc}}^{-}$ are rendered with red color and green color respectively. Note that the given fingers in these experiments are chosen randomly and may result into empty $\mathcal{A}_{\mathsf{c}}$. Fig.3.20(b-2) is an example of such an empty case. The $\mathcal{A}_{\mathsf{c}}$ in this figure is empty so that no region is rendered with white color.

Fig.3.21 shows results of the algorithm under a user-defined terminating layer. In this figure, the terminating layer l is set to $+/-2$ to have better visual effect. Like my analysis in previous sub-sections, These results are subsets of their corresponding complete $\mathcal{A}_{\mathsf{c}}$ in Fig.3.20(a-1), Fig.3.20(b-1) and Fig.3.20(c-1).

Moreover I carry out some experiments with the same objects presented by Erickson [Erickson et al., 2007]. Fig.3.22 demonstrates these objects and their correspondent results. The boundaries and finger positions are made with best similarity to the examples of [Erickson et al., 2007]. My results in Fig.3.22 are equal or a little smaller compared with Erickson' s results. This is because Erickson uses *critical orientation*s. The *critical orientation*s depend highly on the vertices of the target shape. It can be recognized as a rougher form of my discretization. In my discretization, the orientation axis is divided into $-m$, $-m+1$, ..., 0, ..., m layers. I calculate and accumulate the $\mathcal{A}_{\mathsf{c}}[q_{i_\theta}^{\mathsf{obj}}]$ along the discretized layers. In contrast, Erickson calculates and accumulate the $\mathcal{A}_{\mathsf{c}}[q_{i_\theta}^{\mathsf{obj}}]$ when the orientation

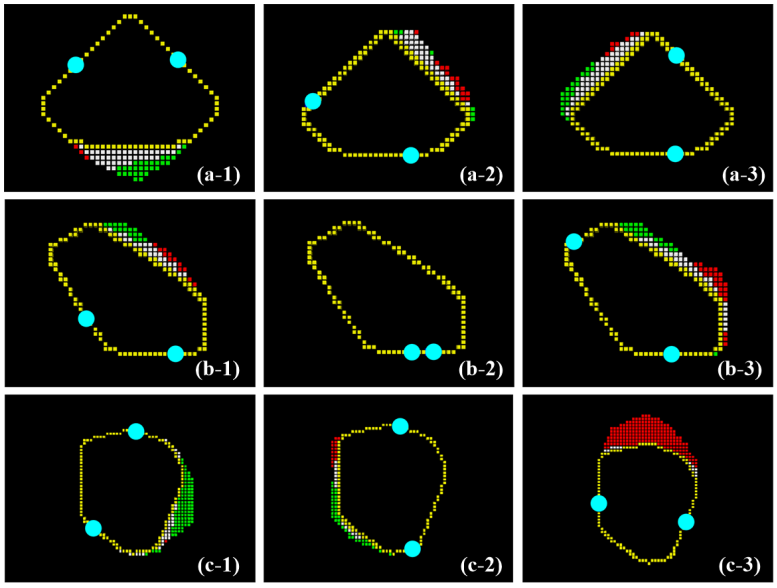


Figure 3.20: Results with different finger positions.

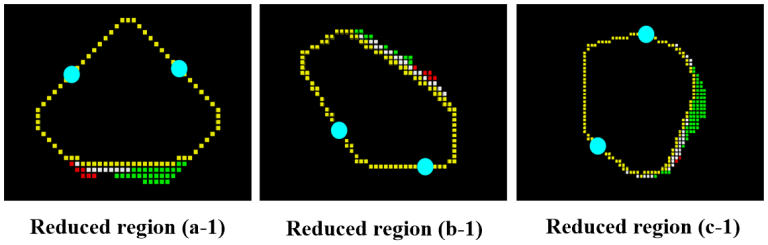


Figure 3.21: The results of Fig.3.20(a-1), (b-2) and (c-1) with user-defined terminating layer $l = +/-2$.

$q_{i_\theta}^{\mathsf{obj}}$ is at a *critical orientation*. This is even rougher so that the complete $\mathcal{A}_{\mathsf{c}}$ not only has zigzag form like Fig.3.15 but also changes from a region with smooth boundary to a polygonal region. The smooth result should be roughly the same or smaller comparing with the polygon region. That's why my results in Fig.3.22 are equal or a little smaller comparing with Erickson' s results in [Erickson et al., 2007]. The right three columns in Fig.3.22 are generated with respect to different resolutions of the range scanner. They correspond to 40×30, 80×60 and 160×120, respectively. It is easy to draw a conclusion that a higher resolution endows more continuity to the *critical orientation*s and consequently implies higher precision. However, higher resolution dramatically lowers computational efficiency. Recall that the computational complexity of my algorithm is $O(n_v^2 \cdot m)$. It highly depends on n_v, namely the number of boundary points or the resolution of a boundary.

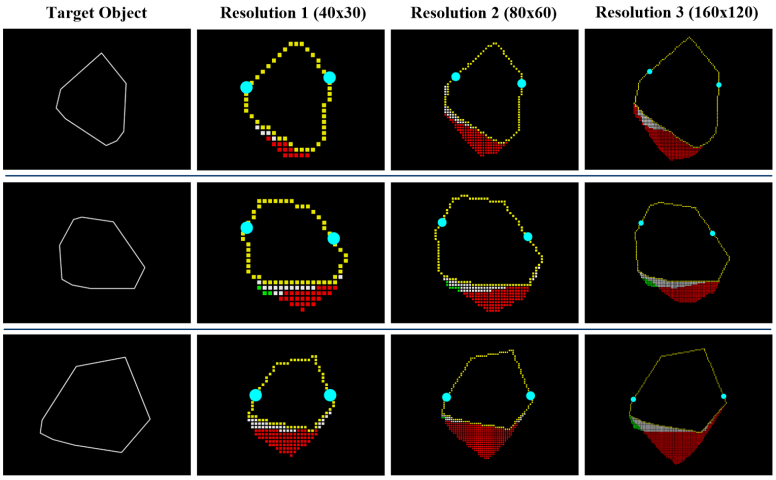


Figure 3.22: The results of my algorithm on Erickson's target objects with different scanner resolutions.

3.2.6 Robust three-finger caging and its complexity

In the previous four subsections, I presented how to find the caging region of a third finger given the shape of target object and two finger positions. Since the target object is limited to convex polygons. This algorithm can be further extended to replace the three-finger caging optimization by considering the immobilization optimization introduced in section 2.1.3.

3.2.6.1 The measurement of a three-finger immobilization

Now let us recall the open problem that was left in section 2.1.3. That is, how can we define a measurement to measure those shadow areas in Fig.2.11 so that it can indicate the quality of immobilization? If we present this problem in another way, it could be as following. Given a three-finger immobilization formation $\{\boldsymbol{f_1}, \boldsymbol{f_2}, \boldsymbol{f_3}\}$ and a target object, how to measure the robustness of each finger with respect to caging breaking? Let us take measuring the robustness of $\boldsymbol{f_3}$ for example. Delightfully, when measuring the robustness of $\boldsymbol{f_3}$, the problem becomes a familiar form discussed in the previous four sections. Here we have two given fingers $\{\boldsymbol{f_1}, \boldsymbol{f_2}\}$ and a target object. With these given information, we can calculate the caging region $\mathcal{A}_c$ of $\boldsymbol{f_3}$ by using Alg.2 and expression (3.17). Then, the robustness of $\boldsymbol{f_3}$ with respect to caging breaking can be measured by calculating the distance between $\boldsymbol{f_3}$ and the boundary of its $\mathcal{A}_c$. More specifically, it is the minimum Euclidean distance between $\boldsymbol{f_3}$ and $\partial\mathcal{A}_c$.

$$d_{\boldsymbol{f_3}} = \mathsf{min}(|\boldsymbol{f_1} - \partial\mathcal{A}_c|) \tag{3.18}$$

The measurement of the whole three-finger immobilization formation can be done by alternatively repeating the procedure of expression (3.18). That is, (1) alternatively fix

two fingers and take them as the given ones, (2) calculate the $\mathcal{A}_c$ of the third finger and (3) measure the minimum distance between the third finger and the boundary of $\mathcal{A}_c$. With these three steps, we can get three distances, say d_{f_1}, d_{f_2}, d_{f_3}. The measurement of the whole three-finger immobilization formation depends on the smallest value of the three distances and it could be defined as

$$Q_{\{f_1,f_2,f_3\}} = \min(d_{f_1}, d_{f_2}, d_{f_3}) \tag{3.19}$$

Fig.3.23 demonstrates the measurement of expression (3.19). Readers may better understand how I fix two fingers as given fingers alternatively and how I employed Alg.2 by referring to the frame of this figure. The algorithm based on expression (3.19) and demonstrated in Fig.3.23 is named the "immobilization optimization algorithm".

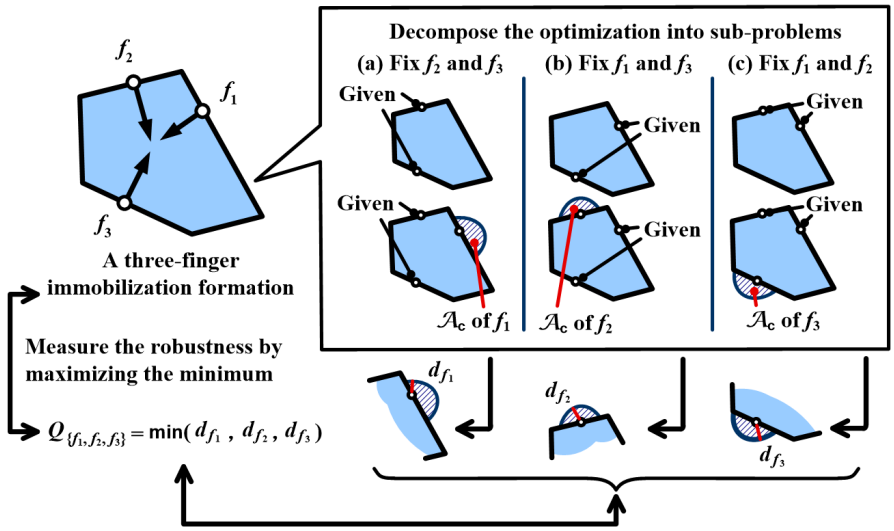


Figure 3.23: Measuring the robustness of a three-finger formation with a convex object.

3.2.6.2 Retracting to a robust three-finger caging

Expression (3.19) enables us to measure the robustness of a three-finger immobilization formation. However, it is still quite far from the robustness of any caging formation introduced in Fig.3.4. Calculating the robustness of all caging formations is computationally infeasible and therefore I propose another solution by replacing the three-finger caging optimization with the immobilization optimization. Recall that we have shown in section 2.1.3 and section 2.2 the relationship of **caging**, **contact caging** and **immobilization** of a convex object. **Caging** is the extension of **contact caging** or **immobilization**. In the case of three fingers and convex polygons, **contact caging** does not exist and **caging** is the extension of **immobilization**. Or we can say **immobilization** is the minimum form of **caging** in the case of three fingers and convex target objects. Accordingly, I do not calculate the robustness

of all caging formations and find an optimized three-finger caging explicitly but implicitly perform optimization by the following two steps. (1) Finding an optimized three-finger immobilization with the immobilization optimization algorithm and (2) retracting fingers to obtain certain robustness to avoid collisions. Fig.3.24 illustrates these two steps.

Note that this two-step algorithm is not based on rigid mathematical deduction. It is a patchwork method and it requires delicate design to get a good caging. However, comparing with Fig.3.4, the two steps in Fig.3.24 does not explicitly solve the problem of "finding all possible caging formations" but pays more attention to finding a formation of fingers that could be quite robust to endure uncertainties. On the one hand, an optimized three-finger immobilization has large robustness to caging breaking. On the other hand, the retraction of fingers keeps finger far from collision with target objects.

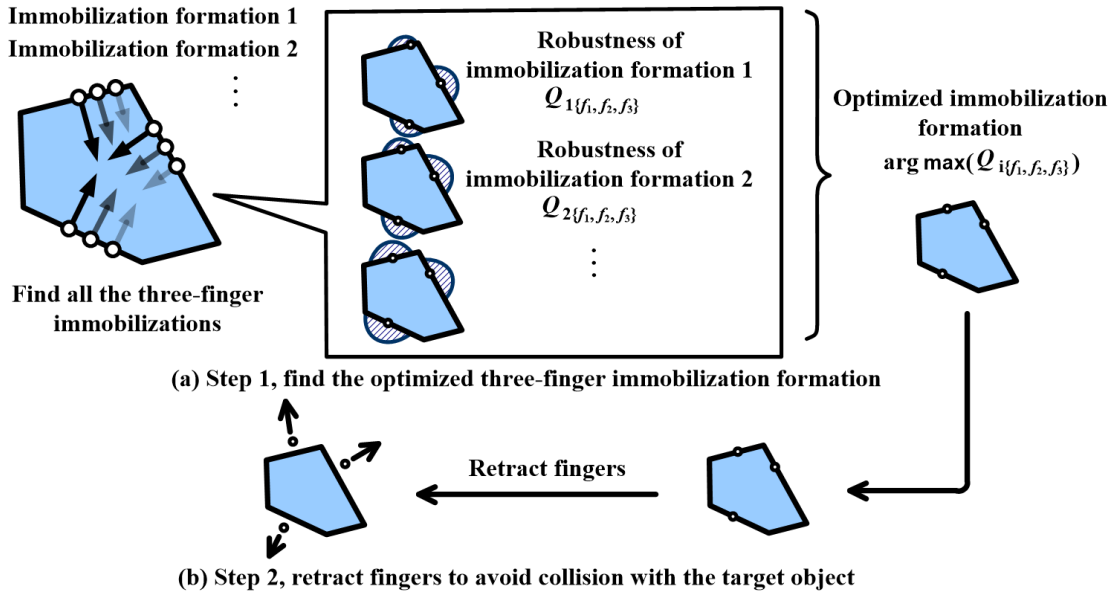


Figure 3.24: Two steps to get a robust caging formation.

As I told in last paragraph, this two-step solution requires delicate design, or else it may easily fail into caging breaking. There are generally two problems that relate to detailed design of these two steps. They are (1) how to find an optimized three-finger immobilization with the robustness measurement in expression (3.19) and (2) how to retract fingers. The first problem can be solved by finding all immobilization finger formations and evaluate their robustness one by one. The immobilization finger formation with largest measurement $Q_{\{f_1,f_2,f_3\}}$ is an optimized result. Fig.3.4(a) illustrates this procedure. The second problem is difficult to be strictly solved. We can further divide it into two sub-problems, namely (2.1) along which direction should a finger be retracted and (2.2) how much should a finger be retracted. Although the second problem cannot be solved precisely, we can find a certain retracting direction and give a threshold of retracting distance. These retracting direction and threshold of retracting distance will be discussed later with simulations.

The reason why I would like to delay discussing the second problem is the two-step algorithm to find a robust three-finger caging formation still costs too much computational resources and I would like to simplify it further. What's more I will introduce how to extend this two-step algorithm to multiple fingers and practical grasping problems. Let us firstly briefly analyze and discuss its complexity in the following texts. The cost of calculating one $\mathcal{A}_{\mathsf{c}}$, as was discussed in previous section, is $O(n_v^2 \cdot m)$. Measuring the distances $d_{\boldsymbol{f_1}}$, $d_{\boldsymbol{f_2}}$ and $d_{\boldsymbol{f_3}}$ can be roughly considered $O(n_v)$. Therefore, the total cost of measuring one formation is $O(n_v^3 \cdot m)$. We should multiply this cost with the cost of finding all immobilization formations, $O(n_v^3)$[4]. In total, the cost of the two-step algorithm to find a robust three-finger caging formation would be as high as $O(n_v^3 \cdot n_v^3 \cdot m) = O(n_v^6 \cdot m)$. It is much better than the intuitive solution introduced in section 3.2.1 which costs $O(n_p^4 \cdot n_v^4 \cdot s \cdot m)$. Note that n_p is much larger than n_v since n_p is the number of points in a band surrounding the boundary. However, it still costs too much. We need further simplification. I will introduce an simplification to this two-step algorithm by using **translational constraints** and **rotational constraints** in the next section. The simplified two-step algorithm will be named faster robust caging.

3.3 Faster Robust Caging

3.3.1 Translational constraints and rotational constraints

The high cost of the two-step algorithm mainly depends on the immobilization optimization algorithm which is half caused by finding all immobilization formations and half caused by accumulation of $\mathcal{A}_{\mathsf{c}}$. The second reason relates closely to Alg.2 and expression (3.17). Further, the second reason can be divided into two aspects, namely (1) the canonical motion and (2) the continuous accumulation. The canonical motion limits finger number to three, it results into the m component of $O(n_v^2 \cdot m)$. The continuous accumulation results into the n_v^2 component.

Vahedi in his Ph.D thesis [Vahedi, 2009] proves that the translational caging region at layer $q_{0_\theta}^{\mathsf{obj}}$, namely $\mathcal{A}_{\mathsf{c}}[q_{0_\theta}^{\mathsf{obj}}]$ is a superset of the complete caging region $\mathcal{A}_{\mathsf{c}}$. The same conclusion can be drawn from Fig.3.13 and expression (3.10) and (3.11). $\mathcal{A}_{\mathsf{pc}}[q_{j_\theta}^{\mathsf{obj}}]$ is always a subset of $\mathcal{A}_{\mathsf{pc}}[q_{i_\theta}^{\mathsf{obj}}]$ when the absolute value of j is larger than the absolute value of i. Therefore, the accumulated $\mathcal{A}_{\mathsf{c}}^{+}$ and $\mathcal{A}_{\mathsf{c}}^{-}$ are always subsets of $\mathcal{A}_{\mathsf{c}}[q_{0_\theta}^{\mathsf{obj}}]$. Can we improve the efficiency of this procedure by adding certain constraints to $\mathcal{A}_{\mathsf{c}}[q_{0_\theta}^{\mathsf{obj}}]$? Comparing with the complete $\mathcal{A}_{\mathsf{c}}$, computing $\mathcal{A}_{\mathsf{c}}[q_{0_\theta}^{\mathsf{obj}}]$ requires no canonical motions and no continuous accumulation. It is advisable to improve efficiency with $\mathcal{A}_{\mathsf{c}}[q_{0_\theta}^{\mathsf{obj}}]$ instead of $\mathcal{A}_{\mathsf{c}}$ by adding certain constraints.

[4]Some reader may maintain the idea that the multiplication here should be an addition. I agree with this argument. The computational complexity can be $O(n_v^3 + n_i \cdot n_v^3 \cdot m)$=$O(n_i \cdot n_v^3 \cdot m)$ where n_i is the number of immobilization formations. However, I decide not to employ n_i to keep its conciseness. Although n_i is always smaller than n_v^3, changing to it does not essentially change the comparison with other algorithms in this thesis.

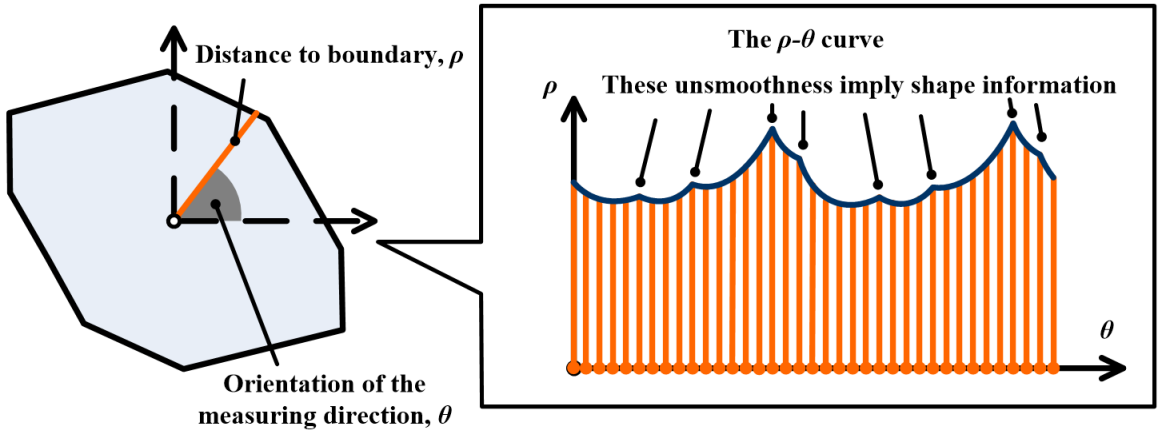
I propose employing the constraints along rotational axis to $\mathcal{A}_{\mathsf{c}}[q_{0_\theta}^{\mathsf{obj}}]$ to substitute for the time-consuming calculation of $\mathcal{A}_{\mathsf{c}}$. The constraints along rotational axis is named "rotational constraints" in this thesis. $\mathcal{A}_{\mathsf{c}}[q_{0_\theta}^{\mathsf{obj}}]$ is the translational caging region. Translational caging region $\mathcal{A}_{\mathsf{c}}[q_{0_\theta}^{\mathsf{obj}}]$ together with rotational constraints can highly improve time efficiency and break through the limitation of three fingers. Therefore, I do not accumulate the $\mathcal{A}_{\mathsf{c}}$ but divide the immobilization optimization algorithm into the collaboration of translational constraints, namely the procedure of calculating translational caging region, and rotational constraints.

The idea of introducing some rotational constraints is based on [Wang et al., 2005]'s idea which employs bounded rotational angle from a $\rho - \theta$ curve for caging test. $\rho - \theta$ curve represents the relationship between the distance from rotation anchor to the boundary of $\mathcal{F}_{\mathcal{F}_i}[q_{0_\theta}^{\mathsf{obj}}]$ and the angle of the measuring direction. Fig.3.25(a) demonstrates this curve.

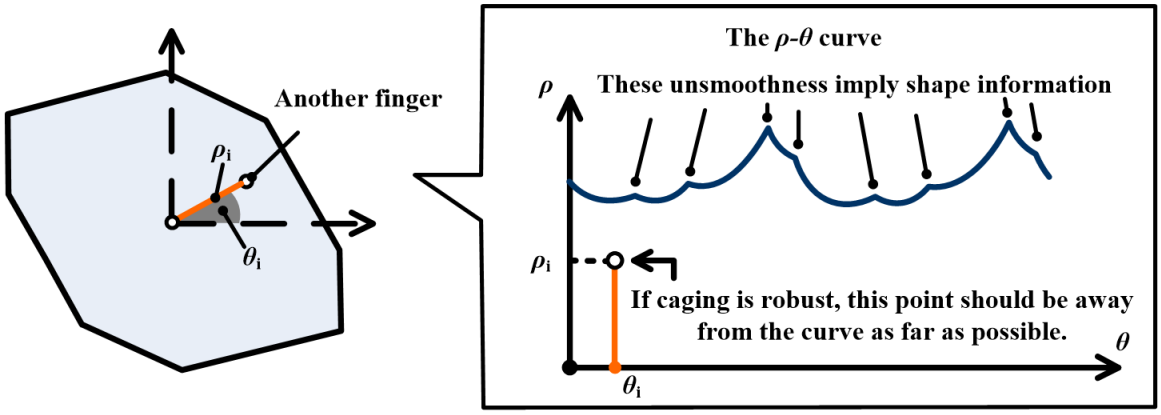
In Fig.3.25, I use the same target object as Fig.3.7. Since this object is a polygon and its boundary is composed of segments, the correspondent $\rho - \theta$ curve has several unsmooth connecting points. These connecting points, together with curvature of the curve, imply shape information of the original target object. Readers may compare the two peaks of the $\rho - \theta$ curve in Fig.3.25(a) and the shape of the original target object to better understand this explanation.

In order to make a formation as robust as possible, we need to make the distance between a finger and the boundary of $\mathcal{F}_{\mathcal{F}_i}[q_{0_\theta}^{\mathsf{obj}}]$ as far as possible. As is explained in Fig.3.6, the boundary of $\mathcal{F}_{\mathcal{F}_i}[q_{0_\theta}^{\mathsf{obj}}]$ is the boundary where two $\mathcal{F}_i$ separate. This "distance" depends on the shape of the $\rho - \theta$ curve and the position of the finger. It is the minimum euclidean distance between the finger and any point on the $\rho - \theta$ curve. Fig.3.25(b) shows a finger and the $\rho - \theta$ curve. If there's no apriori information about the shape of the target object, we cannot calculate the minimum euclidean distance between the finger and any point on the $\rho - \theta$ curve. We can only know the distance between this finger and the pivot of $\mathcal{F}_{\mathcal{F}_i}[q_{0_\theta}^{\mathsf{obj}}]$. That is to say, without any apriori target shape information, we can only measure the robustness by measuring the "distance" between a finger and the pivot of $\mathcal{F}_{\mathcal{F}_i}[q_{0_\theta}^{\mathsf{obj}}]$. The description seems complicated. However, it is intuitive to our common knowledge. Given an arbitrary shape, what is the best way to grasp or immobilize it with three fingers? Note that the condition here is "given an arbitrary shape". There is no extra information about the "arbitrary shape". We know no details about it. The answer of most people would be make the three fingers an equilateral triangle and make the edge of this triangle as small as possible. Because without any detailed information of a given shape, equalize the "distance" between fingers is the best choice. Equalizing the "distance" between fingers and making the "distance" as small as possible is like making the "distance" between a finger and the pivot of $\mathcal{F}_{\mathcal{F}_i}[q_{0_\theta}^{\mathsf{obj}}]$ as small as possible. Without any apriori target shape information, an equilateral formation with smallest inter-finger distances would be the best choice. Fig.3.25(c) shows the case when shape information is unavailable.

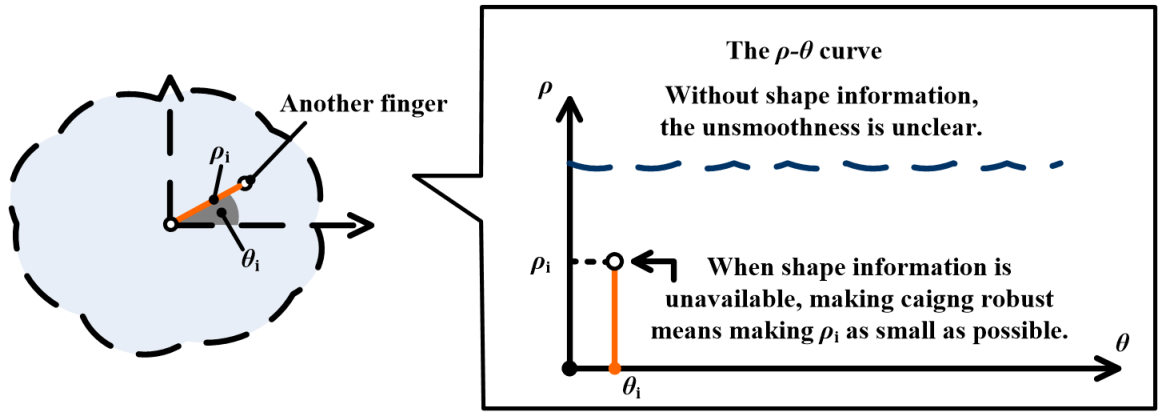
This conclusion is drawn from a $\rho - \theta$ curve which relates to rotation of the target object. That is why I name it the rotational constraints. **Rotational constraints re-**



(a) Procedure of building the ρ-θ curve



(b) Robustness of caging means distance to the curve



(c) When shape information is unavailable, robustness means the length of ρ_i

Figure 3.25: $\rho - \theta$ curve and the rotational constraints.

quire to making the distances between fingers as small as possible. An optimized three-finger immobilization under the rotational constraints would therefore be defined as an immobilization formation that has smallest inter-finger distances. Its measurement is as following.

$$d_{\{\boldsymbol{f_i},\boldsymbol{f_j}\}} = |\boldsymbol{f_i} - \boldsymbol{f_j}| \tag{3.20}$$

$$Q^{\mathsf{R}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}} = \frac{1}{\mathsf{max}(d_{\{\boldsymbol{f_1},\boldsymbol{f_2}\}}, d_{\{\boldsymbol{f_1},\boldsymbol{f_3}\}}, d_{\{\boldsymbol{f_2},\boldsymbol{f_3}\}})} \tag{3.21}$$

Only taking the rotational constraints into account means not considering the shape of target objects but only considering its rotation. Only taking into account rotational constraints loses lots of information. Nevertheless, we can remedy the lost by collaborating rotational constraints together with the translational constraints. Translational constraints relate to translational regions and it pays all its attention to layer $q^{\mathsf{obj}}_{0_\theta}$. The measurement of translational constraints is the revised version of expression (3.18) and (3.19). They are as following.

$$d^{\mathsf{T}}_{\boldsymbol{f_i}} = \mathsf{min}(|\boldsymbol{f_i} - \partial\mathcal{A}_{\mathsf{c}}[q^{\mathsf{obj}}_{0_\theta}]|) \tag{3.22}$$

$$Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}} = \mathsf{min}(d^{\mathsf{T}}_{\boldsymbol{f_1}}, d^{\mathsf{T}}_{\boldsymbol{f_2}}, d^{\mathsf{T}}_{\boldsymbol{f_3}}) \tag{3.23}$$

Note that only taking into account translational constraints results into a superset of the complete $\mathcal{A}_{\mathsf{c}}$. Only the collaboration of these two constraints works. We can have a new $Q_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ by collaborating $Q^{\mathsf{R}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ and $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$. $Q^{\mathsf{R}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ biases on rotation while $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ biases on translation, they collaborate together to offer a faster robust caging algorithm.

The analysis and collaboration is qualitative. We cannot strictly demonstrate and evaluate the performance of this collaboration. In section 3.2.6, we can measure the robustness of an immobilization by expression (3.18). That is because $\partial\mathcal{A}_{\mathsf{c}}$ is the complete caging region. In expression (3.22) and expression (3.23), complete caging region is no longer explicitly calculated so that we can no longer measure the robustness. The procedure is like finding a robust caging formation blindly. However, we can estimate its performance with some simple examples and simulation softwares. I will discuss about that in the following contexts and the implementation section.

3.3.2 Collaboration of $Q^{\mathsf{R}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ and $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$

Let us take a simple triangle object to demonstrate the collaboration of $Q^{\mathsf{R}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ and $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$. Like section 3.2.4, triangle object can help to demonstrate the collaboration clearly. Fig.3.26 shows the whole procedure of the faster robust caging algorithm by collaborating $Q^{\mathsf{R}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ and $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$.

In the first step, we need to find all the immobilization finger formations on the boundary of the target object and calculate all their translational constraints $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$. This step is

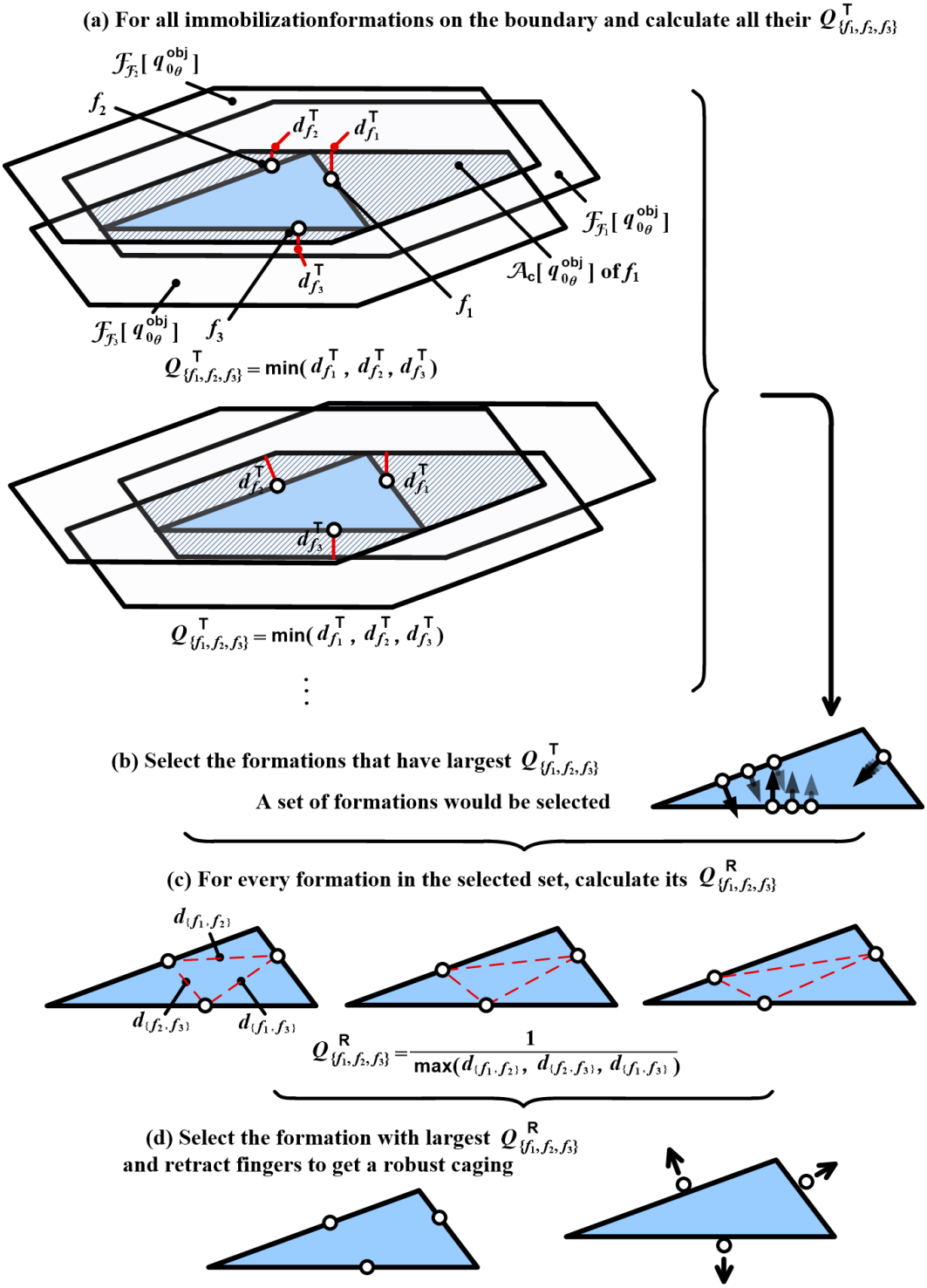


Figure 3.26: Procedure of the faster robust caging algorithm with a triangular example.

shown in Fig.3.26(a). The step consumes lots of resources. It would cost $O(n_v^6)$. Luckily, that's all the costly component. The other steps are addition components and they cost much smaller than the first one. The total cost of this faster algorithm is $O(n_v^6)$. Comparing with the original immobilization optimization algorithm, it reduces the coefficient m. This is an exciting reduction. m involves both accumulation and calculation of canonical motions. Reducing m could improve the algorithm remarkably. At the same time, this improvement by collaborating $Q^{\mathsf{R}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ and $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ is different from the terminating layers introduced in Fig.3.21. Although the terminating layer can also reduce m, it suffers from a significant problem. We have discussed about the significant problem in the last part of section 3.2.4. The collaboration reduces m without suffering from this problem.

In the second step, the finger formations with largest $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ will be filtered into a candidate set. Fig.3.26(b) illustrates this second step. The formations in the candidate set would be further processed in step three where the $Q^{\mathsf{R}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ of each formation is calculated. Fig.3.26(c) illustrates the third step. Finally, the finger formation with largest $Q^{\mathsf{R}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ will be selected as the result of immobilization optimization. We will retract it to get a robust caging. Fig.3.26(d) illustrates the last step.

Note that in the illustration of Fig.3.26, the procedure in Fig.3.26(b) can filter out a set of formations. That's not always the case, especially when the object information involves some noises. I propose to employ an extra parameter τ to endure the outliers. The extra parameter τ works as following. In the second step, I do not select the finger formations with largest $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ but select the finger formations whose $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ is larger than certain threshold τ. This τ should be chosen according to the largest $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ to obtain better performance. We will see how to choose it in the implementation part. Fig.3.27 shows the flow chart of the faster robust caging algorithm, including its two steps and the role of the extra parameter τ as a summary of this demonstration section.

3.3.3 Some extensions

Another topic before discussing the details of retraction is how to extend this faster robust caging algorithm to multiple fingers and practical grasping problems. They will be discussed in the following two sub-sections.

3.3.3.1 Multiple fingers

The collaboration of $Q^{\mathsf{R}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ and $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ not only improves the efficiency of original algorithms from $O(n_v^6 \cdot m)$ to $O(n_v^6)$ but also extend the number of fingers.

The original immobilization optimization algorithm depends on $\mathcal{A}_{\mathsf{c}}$ which accumulates with canonical motion and requires three fingers. The first step in faster robust caging uses collaboration of $Q^{\mathsf{R}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ and $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$. It does not need canonical motions and accumulation. Only the translational caging region at layer $q^{\mathsf{obj}}_{0_\theta}$ needs to be calculated.

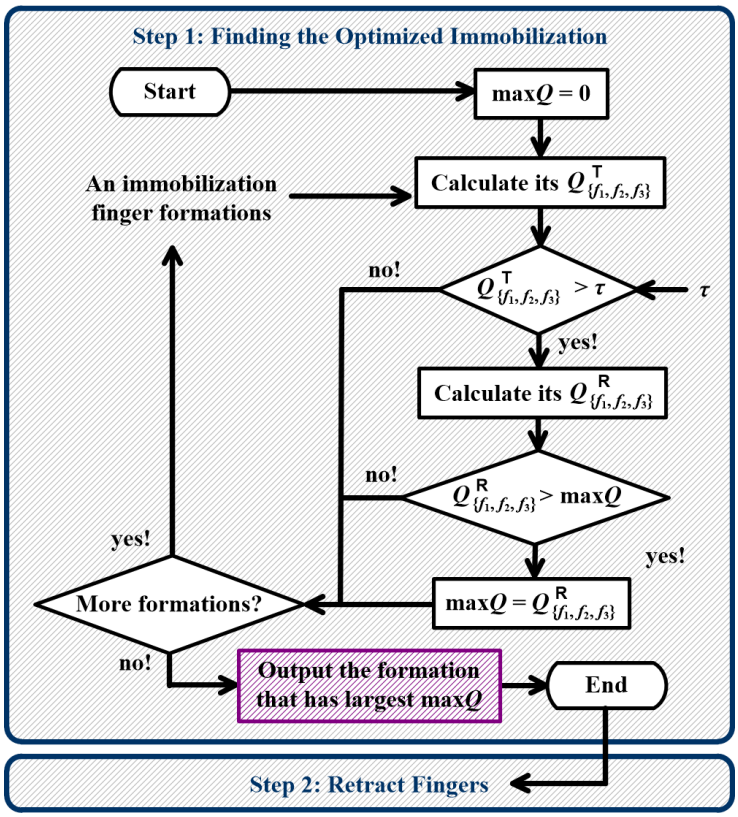


Figure 3.27: Flow chart of faster robust caging.

Therefore, we can take this advantage and extend the faster robust caging algorithm to arbitrary fingers.

Firstly, let us discuss about how to calculate $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ in multi-finger cases. Calculating $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ in multi-finger cases shares the same principle as three-finger cases. They both consider adjacent fingers. The following lemma presents how to calculate the translational caging region with multiple fingers. When caging a 2D convex object, the translational caging region $\mathcal{A}_{\mathsf{c}}[q^{\mathsf{obj}}_{0_\theta}]$ of a finger $\boldsymbol{f_i}$ is determined by its two adjacent neighbours $\boldsymbol{f_{i-1}}$ and $\boldsymbol{f_{i+1}}$. Here we name $\mathcal{A}_{\mathsf{c}}[q^{\mathsf{obj}}_{0_\theta}]$ of a finger $\boldsymbol{f_i}$ with $\mathcal{A}^{\boldsymbol{f_i}}_{\mathsf{c}}[q^{\mathsf{obj}}_{0_\theta}]$. $\mathcal{A}^{\boldsymbol{f_i}}_{\mathsf{c}}[q^{\mathsf{obj}}_{0_\theta}]$ can be calculated by the following expressions.

$$\mathcal{A}^{\boldsymbol{f_i}}_{\mathsf{c}}[q^{\mathsf{obj}}_{0_\theta}] \in ((\mathcal{F}_{\mathcal{F}_{i-1}}[q^{\mathsf{obj}}_{0_\theta}] \cap \mathcal{F}_{\mathcal{F}_{i+1}}[q^{\mathsf{obj}}_{0_\theta}]) \setminus \mathsf{O}[\{q^{\mathsf{obj}}_{0_\mathsf{x}}, q^{\mathsf{obj}}_{0_\mathsf{y}}, q^{\mathsf{obj}}_{0_\theta}\}]) \tag{3.24}$$

$$(\boldsymbol{f_1} \notin \mathcal{A}^{\boldsymbol{f_i}}_{\mathsf{c}}[q^{\mathsf{obj}}_{0_\theta}]) \wedge (\boldsymbol{f_2} \notin \mathcal{A}^{\boldsymbol{f_i}}_{\mathsf{c}}[q^{\mathsf{obj}}_{0_\theta}]) \tag{3.25}$$

The two expressions here are nearly the same as expression (3.24) and (3.25) except that they are extended to an arbitrary finger $\boldsymbol{f_i}$. Note that the description of i, $i+1$ and $i-1$ is not strict. When $i > 1$, it is the same as the description. However, when $i = 1$, $i-1$ should be index of the last finger, say $i - 1 = n_f$ when n_f is the total number of fingers.

When $i = n_f$, $i+1$ should be index of the first finger, say $i+1 = 1$. $i-1$ and $i+1$ are modulated by n_f. Since calculating one $\mathcal{A}_{\mathsf{c}}[q_{0_\theta}^{\mathsf{obj}}]$ requires two adjacent fingers, the minimum required finger number is three. This is coherent with our discussion in section 2.2. 3 fingers are required to calculate the translational caging region and 3 or 4 is the least number of fingers that are required to cage a 2D convex target object.

When a planar object is translationally caged, all $\mathcal{F}_i[q_{0_\theta}^{\mathsf{obj}}]$ form a chain or the position of any finger f_i falls inside both $\mathcal{F}_{i-1}[q_{0_\theta}^{\mathsf{obj}}]$ and $\mathcal{F}_{i+1}[q_{0_\theta}^{\mathsf{obj}}]$. Despite the extension to multiple fingers, the chain shares the same concept as our illustration in Fig.3.6. More strictly speaking, we only consider the fingers that form a chain. There are some exceptions where the position of one finger doesn't simultaneously fall inside both $\mathcal{F}_{i-1}[q_{0_\theta}^{\mathsf{obj}}]$ and $\mathcal{F}_{i+1}[q_{0_\theta}^{\mathsf{obj}}]$. These exceptions are not taken into account. This is because our target objects are convex objects and Our faster robust caging algorithm start from immobilization formations[5] of a convex object and retract fingers to a robust caging formation. Fingers in immobilization formations are always in contact with target objects and therefore all finger in the immobilization formations are sure to form a chain. See Fig.3.28 for example. Both the fingers in Fig.3.28(a-1) and Fig.3.28(b-1) immobilize the target object. Especially, the fingers in Fig.3.28(b-1) are multi-finger caging. Some three fingers in Fig.3.28(b-1) already immobilize the target object. For example the fingers that are at the same position as Fig.3.28(a-1) are one of those "some three fingers". The other fingers besides those three fingers can be viewed as extra fingers. The positions of the extra fingers are always simultaneously in both $\mathcal{F}_{i-1}[q_{0_\theta}^{\mathsf{obj}}]$ and $\mathcal{F}_{i+1}[q_{0_\theta}^{\mathsf{obj}}]$ since boundary of the target object always belongs to any $\mathcal{F}_i[q_{0_\theta}^{\mathsf{obj}}]$. It is unnecessary to consider the exceptions that does not form chains.

Fig.3.28(b-2) and Fig.3.28(b-3) illustrates how $\mathcal{A}_{\mathsf{c}}^{\boldsymbol{f_i}}[q_{0_\theta}^{\mathsf{obj}}]$ is calculated. Readers may better understand these two figures by referring to expression (3.24) and (3.25).

Since the calculation of $\mathcal{A}_{\mathsf{c}}^{\boldsymbol{f_i}}[q_{0_\theta}^{\mathsf{obj}}]$ is nearly the same as three-finger cases, there is not too much difference between the expression to calculate translational constraints $Q_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}^{\mathsf{T}}$ and expression (3.23). It can be expressed as following. Note that the symbol n_f is the total number of fingers.

$$Q_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}^{\mathsf{T}} = \mathsf{min}(d_{\boldsymbol{f_1}}^{\mathsf{T}}, d_{\boldsymbol{f_2}}^{\mathsf{T}}, \ldots, d_{\boldsymbol{f_{n_v}}}^{\mathsf{T}}) \tag{3.26}$$

Then, let us discuss about how to calculate $Q_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}^{\mathsf{R}}$ in multi-finger cases. Like translational constraints, the rotational constraints of multi-finger case shares the same principle as the rotational constraints of three-finger case. Its expression is like expression (3.21).

$$Q_{\{\boldsymbol{f_1},\boldsymbol{f_2},\ldots\boldsymbol{f_{n_f}}\}}^{\mathsf{R}} = \frac{1}{\mathsf{max}(d_{\{\boldsymbol{f_1},\boldsymbol{f_2}\}}, d_{\{\boldsymbol{f_1},\boldsymbol{f_{n_f}}\}}, d_{\{\boldsymbol{f_3},\boldsymbol{f_4}\}}, d_{\{\boldsymbol{f_3},\boldsymbol{f_2}\}}, \ldots, d_{\{\boldsymbol{f_{n_f-1}},\boldsymbol{f_{n_f}}\}}, d_{\{\boldsymbol{f_{n_f-1}},\boldsymbol{f_{n_f-2}}\}})} \tag{3.27}$$

[5]In the multi-finger case, a finger formation is supposed to be immobilizing a target object as long as any three-finger or four-finger combination from the formation can immobilize it. Although not all fingers play the role of immobilizers, they should be in contact with the target object. Or else, the discussion becomes nonsense.

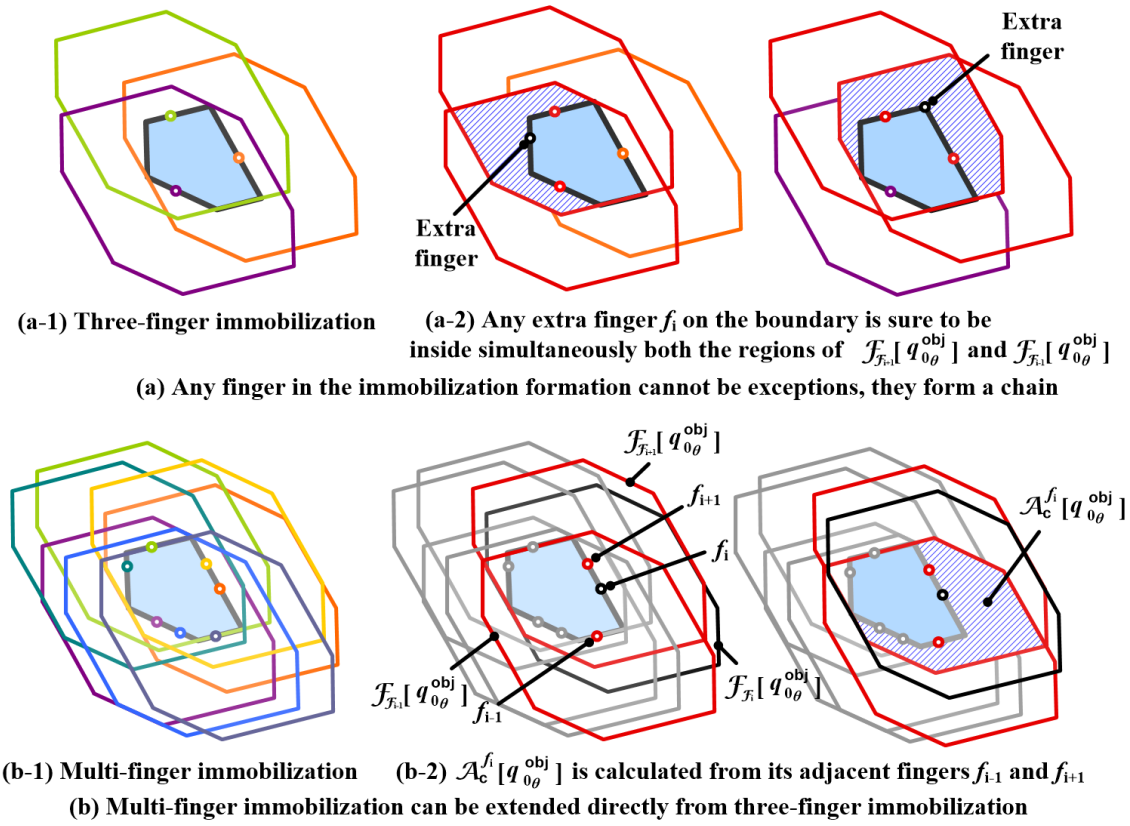


Figure 3.28: $\mathcal{A}_c^{f_i}[q_{0_\theta}^{obj}]$ in multi-finger case.

An essential point of the rotational constraints of multi-fingers in expression (3.27) is the meaning of "inter-finger distances". I have been using this name since section 3.3.1. However, we were dealing with three fingers before this sub-section and the "inter-finger distances" meant the distances between any two fingers. When the finger number is extended to more than three, the "distances between any two fingers" can no longer represent "inter-finger distances" correctly. It should be the distances between any two adjacent fingers. The distances between any two adjacent fingers are shown in the denominator component of expression (3.27), namely the $d_{\{f_1,f_2\}}, d_{\{f_1,f_{n_f}\}}, d_{\{f_3,f_4\}}, d_{\{f_3,f_2\}}, \ldots, d_{\{f_{n_f-1},f_{n_f}\}}, d_{\{f_{n_f-1},f_{n_f-2}\}}$ part.

Like the three-finger case, we can have a new $Q_{\{f_1,f_2,\ldots,f_{n_f}\}}$ by collaborating $Q^{\mathsf{R}}_{\{f_1,f_2,\ldots,f_{n_f}\}}$ and $Q^{\mathsf{T}}_{\{f_1,f_2,\ldots,f_{n_f}\}}$. It follows the same flowchart in Fig.3.27 to calculate a robust caging for multi-fingers.

3.3.3.2 Grasping by caging

Grasping by caging is another extension of the my faster robust caging algorithm. If one would like to perform a manipulation task with strict grasping as well as robustness to uncertainty. "grasping by caging" is a good choice.

By reviewing section 1.3, Fig.1.3 and Fig.1.4, we can find that the major aim of caging is to avoid collision with the target object as well as to keep constraining of the target object. The faster robust caging algorithm proposed in foregoing texts aims at finding a good solution to fulfill this requirements. Result of the faster robust caging algorithm, on the one hand, can cage and constrain the target object. This is because we started from immobilization which is the minimum form of caging. The caging maintains as long as we do not retract fingers too much from the positions of an optimized immobilization formation. On the other hand, it can avoid collision with the target objects since fingers are retracted from target object boundaries. The faster robust caging algorithm can solve the caging problems to a certain degree.

However, caging is not as strict as grasping or immobilization. Grasping and immobilization control objects strictly while caging only controls object loosely. Objects can move freely in a free configuration region $\mathcal{C}_{\mathsf{fc}}^{\mathsf{obj}}$. Recall the discussion in Chapter 2 and we may find that caging makes us confront a situation where

- Caging is robust to uncertainties but controls loosely.
- Grasping controls strictly but suffers from uncertainties.

I propose the "grasping by caging" concept to make up the drawback of caging. In certain occasions, people may employ caging as a pre-grasping step to practical grasping and take the advantages of both caging and grasping. Before this proposal, lots of impressive works have devoted themselves to deal with uncertainties in grasping. Those works do not involve caging but they interest me a lot and directly drive me to the "grasping by caging" idea. These researches involve but not limited to the following works. (1) Approaching regions with hand primitives, examples include [Ekvall and Kragic, 2007] and [Berenson et al., 2011], (2) database matching, examples include [Goldfeder et al., 2009] and [Glover et al., 2009], (3) heuristic shape recovery, examples include [Rao et al., 2010], [Bohg et al., 2011] and [Harada et al., 2013], (4) local reactive sensing, examples include [Leeper et al., 2010] and [Hsiao et al., 2010] and (5) machine learning, examples include [Saxena et al., 2006] and [Jiang et al., 2011], etc. Here I will discuss some of them in details. The most interesting work to me that relates to making grasping robust to uncertainty is Berenson' s research. He started his work by considering distance to obstacles, namely the distmap, in the scene [Berenson et al., 2007][Berenson and Srinivasa, 2008]. The distmap can be viewed as a criterion to evaluate the contact space of target objects. Probably it is criterion that drove him to the later concept – Task Space Region (TSR) [Berenson et al., 2009b]. The name is changed into Workspace Goal Regions (WGR) in

[Berenson et al., 2009a] and finally back to TSR in [Berenson et al., 2011] and his Ph.D thesis [Berenson, 2011]. Distmap calculates a measurement to obstacles in the scene while TSR calculates a measurement to fingers. Their final goals are the same, namely find a pre-grasping that could be robust to uncertainties in control or perception. Berenson's TSR is interesting. However it is defined manually so that the solution seems a little tricky. [Saxena et al., 2006] and [Jiang et al., 2011] defines implicitly and explicitly a window to detect gripping points. This window is actually the same idea as manually defining a TSR. Prof. Kragic's group published many works on grasping in the presence of uncertainties. In their publications, [Ekvall and Kragic, 2007] and [Bohg et al., 2011] interested me most. These two works also try to calculate a pre-grasping, either pre-approaching vector or pre-grasping shape according to recovered 3D models. They do not manually define a region or window. Nevertheless, [Bohg et al., 2011] recovers 3D models by using heuristics of known shapes. It is a heuristic way to build a bounding box and hence a heuristic way to define a region. The heuristic approach is an interesting strategy to calculate pre-graspings. [Leeper et al., 2010] and [Hsiao et al., 2010]'s research concentrate on the PR2 gripper of WillowGarage. Their work is different from pre-grasping and it plans re-grasping in real time according to dynamic information collected from sensors mounted on end-effectors. Database approaches can use pre-computed results. However, the range of target objects is limited. It depends on the range of pre-computed database.

We can have a conclusion by reviewing these publications. That is **in order to deal with uncertainties in grasping, we had better calculate a good pre-grasping**. In the "grasping by caging" proposal, calculating robust caging plays the role of calculating a good pre-grasping. Specifically, "grasping by caging" firstly finds a robust caging formation and initiate fingers to positions of this robust caging formation. This step initializes a good pre-grasping and it is essential the faster robust caging algorithm introduced in previous sections. Then the "grasping by caging" algorithm shrinks fingers into immobilization or contact caging to perform a practical grasping. Note that shrinking from a caging formation into a practical grasping is not limited to convex target objects. This is because of two reasons.(1) The minimum state of caging is contact caging or immobilization. (2) Contact caging and immobilization are in a state of equilibrium. When fingers in equilibrium state are endowed with friction, they become force closure. Fig.3.29 illustrates the two reasons. Although force closure suffers from perception problems, force closure itself is practical to grasping. The caging-based pre-grasping procedure filters out the perception problems of force closures. Even if the friction force are not large enough to support force closure, the result of "grasping by caging" is still practical strict control comparing with single caging.

There are two notes about my "grasping by caging" algorithm. The first one is since the faster robust caging algorithm requires convex target objects, the "grasping by caging" algorithm here only works with convex target objects. I will present another robust caging algorithm which is not limited to convex target objects in the third part of this thesis. In that case, the "grasping by caging" algorithm will work with both concave and convex target objects. The second one is we are discussing about "shrinking caging" but "stretching caging". Grasping by stretching caging must be dealt in a different way. Ro-

driguez [Rodriguez et al., 2012] shows that considering grasping by both shrinking caging and stretching caging at the same time is quite difficult since there are lots of possibilities. Readers may refer to Fig.4 of [Rodriguez et al., 2012] to better understand the difficulties.

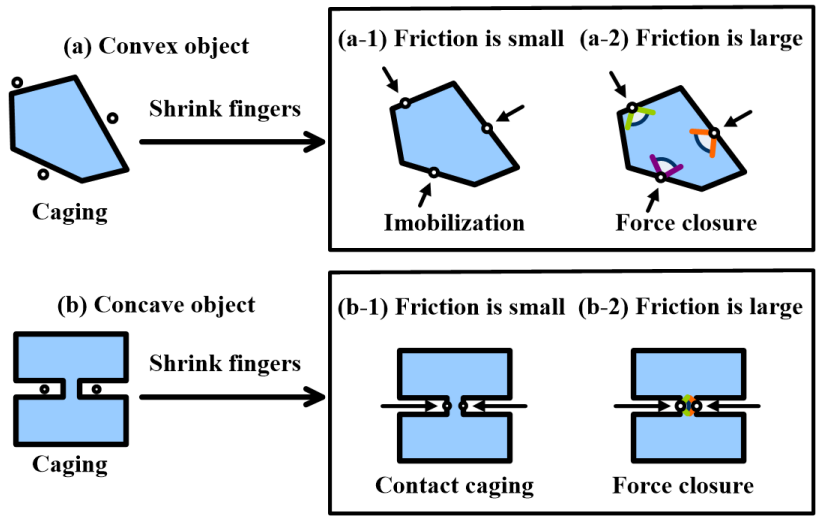


Figure 3.29: "Grasping by caging" can be applied to both convex and concave objects.

Fig.3.30 summarizes a whole methodology of practical grasping by using the extensions proposed in this section, it includes the faster robust caging algorithm, the multi-finger extension and the "grasping by caging" algorithm. During a procedure of grasping, there could be two kinds of uncertainties. One comes into being during the perception procedure shown in Fig.3.30(a) while the other one comes into being during the actuation procedure shown in the right part of Fig.3.30(b). The methodology summarized here could endure these uncertainties. In the beginning, the faster robust caging algorithm is applied to the approximated target object shape of Fig.3.30(a). Fig.3.30(b-1) shows the details of applying the faster robust caging algorithm. It finds an optimized immobilization formation by considering $Q^{\mathsf{R}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\ldots,\boldsymbol{f_{n_f}}\}}$ and $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\ldots,\boldsymbol{f_{n_f}}\}}$. Note that the subscript $\{\boldsymbol{f_1},\boldsymbol{f_2},\ldots,\boldsymbol{f_{n_f}}\}$ is omitted for conciseness. The faster robust caging algorithm starts calculation with three fingers and increase finger number one by one until certain condition is satisfied. We will discuss about this "certain condition" in the implementation section. In Fig.3.30(b-1), a three-finger immobilization formation is considered to be optimized since three fingers are enough and it is unnecessary to use four. The formation will be retracted from the boundary of target objects with some distance. The retraction distance will also be discussed in the implementation section. After retraction, we can get a robust caging formation and this formation will be applied to groundtruth target object. Fig.3.30(b-2) shows this procedure. Note that during this procedure, the finger may suffer from uncertainties caused by actuation noises. Thanks to our calculation in Fig.3.30(b-1). The uncertainties won't cause too much trouble since the caging formation is a robust one and it could endure actuation uncertainties to a certain degree. When the formation is applied to groundtruth target objects, the pre-grasping

formation is inititiated. We can then follow the 'grasping by caging" algorithm and shrink fingers to obtain more strict manipulation. Fig.3.30(c) illustrates this procedure.

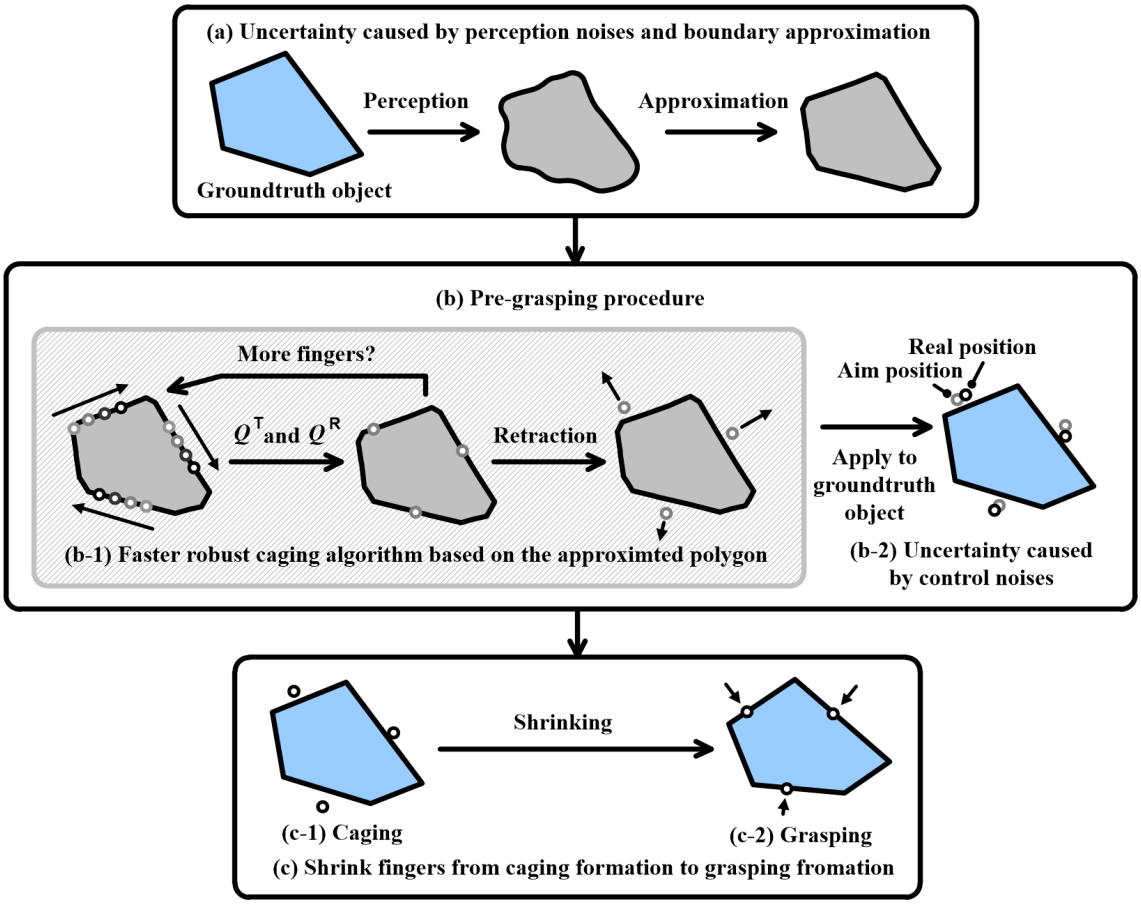


Figure 3.30: The whole approach to realize strict manipulation.

There is one extra problem in Fig.3.30(c), namely how can we maintain caging during the shrinking procedure. This is an important issue as losing caging during the shrinking procedure makes the pre-grasping procedure nonsense. Vahedi in his Ph.D thesis [Vahedi, 2009] proposed a lemma in page 78.

> Let c be a placement of the third finger that cages $P[q]$. Then any point c_0 vertically above $P[q]$ and below c also cages $P[q]$.

This lemma can be demonstrated with Fig.3.31. It means that when the object in Fig.3.31 is caged by the three point fingers $\boldsymbol{f_1}$, $\boldsymbol{f_2}$ and $\boldsymbol{f_3}$, the caging maintains if we shrink one finger vertically towards the connecting line of the other two fingers. For example, when we shrink $\boldsymbol{f_3}$ vertically towards the connecting line of $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$ in Fig.3.31, caging will continuously maintained until grasping. The inherent reason of this lemma is when shrinking

$\boldsymbol{f_3}$ vertically towards the connecting line of $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$, the distances $d_{\{\boldsymbol{f_1},\boldsymbol{f_3}\}}$ and $d_{\{\boldsymbol{f_2},\boldsymbol{f_3}\}}$ would continuously become smaller while the distance $d_{\{\boldsymbol{f_1},\boldsymbol{f_2}\}}$ remains the same. Accordingly, caging maintains continuously.

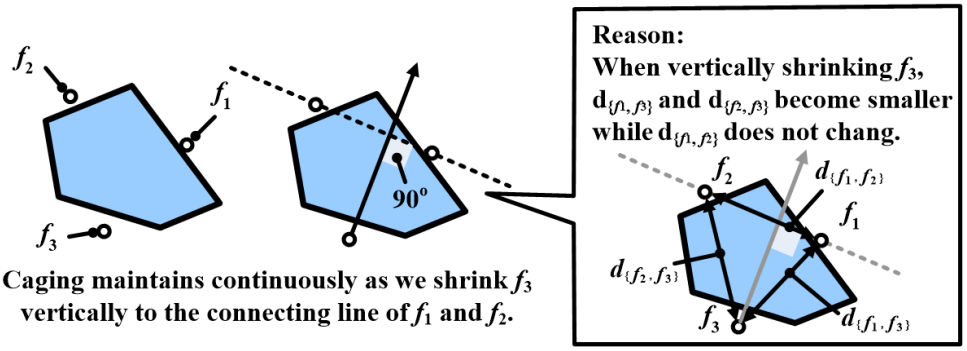


Figure 3.31: Caging maintains when vertically shrinking $\boldsymbol{f_3}$ in three-finger caging.

This lemma can be extended to multi-finger case intuitively. It should be as following. If we shrink a finger $\boldsymbol{f_i}$ vertically towards the connecting line of its adjacent fingers $\boldsymbol{f_{i-1}}$ and $\boldsymbol{f_{i+1}}$, caging will maintain continuously. Note that both Vahedi's lemma and my extension to multi-finger case are subject to shrinking caging. I define the shrinking caging as following. **Shrinking caging means when shrinking one finger vertically towards the connecting line of its adjacent neighbours, it should not go over the connecting line.** Surely this is a "stricter" shrinking caging. But it would be satisfying to make clear the coverage of my "grasping by caging" algorithm.

Taking into account the discussion on shrinking, Fig.3.30(c) becomes shrink randomly a finger towards the connecting line of its adjacent neighbours until strict grasping.

3.3.4 Implementation with Webots

In the past few sections we have successfully collaborate $Q^{\mathsf{R}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ and $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ and propose the basic flow of "grasping by caging". But at the same time, we have unfortunately collected a batch of unsolved problems. They are (1) how to retract fingers, (2) how is the performance of the collaboration， (3) how to choose the selecting parameter τ and (4) how to decide the number of fingers. Strictly deducing them with mathematical formulae is infeasible and consequently I estimate them by simulations.

The simulation uses the same Webots simulation software as section 3.2.5. Nevertheless, a different scene is built for this estimation work. This scene is based on a revised Katana Arm [Katana, 2013]. Fig.3.32 demonstrates this simulation scene. The original Katana Arm has five DoFs. It can be installed with extra grippers or other end-effectors. I install a $20.0cm \times 20.0cm$ board to the fifth DoF. The board is modeled with a box and it acts as the palm of the end-effector. On this board there are four fingers and each of them can be actuated inside the board along x and y axes. The fingers are modeled with cylinders. The

height of a finger, namely the height the cylinder is $6.0cm$ while its radius is $0.5cm$. These fingers are assumed to play the role of those point fingers in our algorithms. Since each finger can be actuated independently, it is like our end-effector introduced in section 1.2.

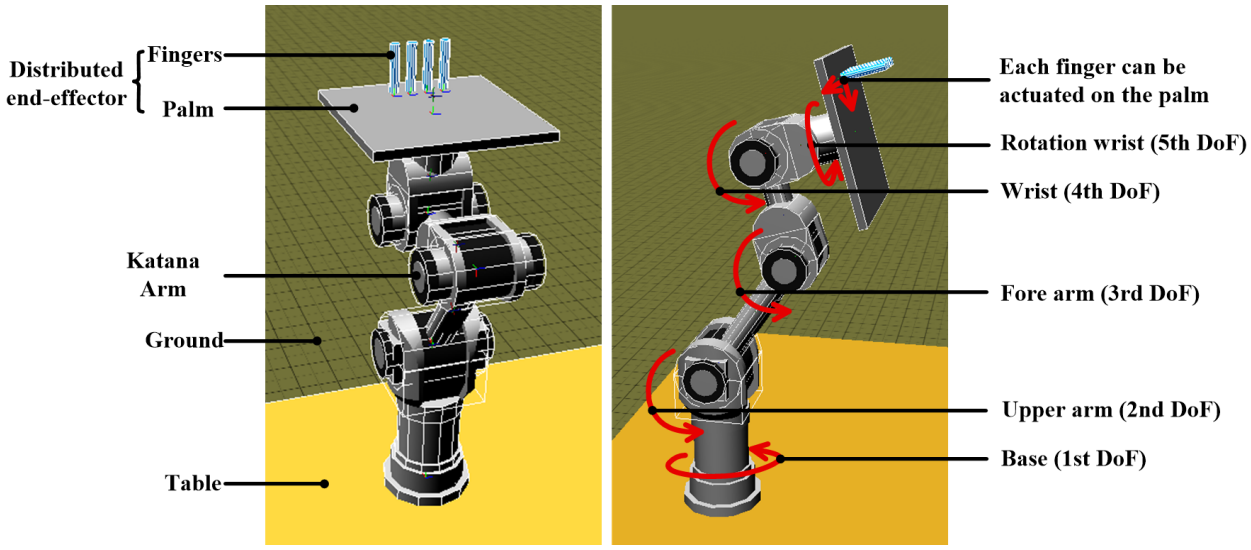


Figure 3.32: A Katana Arm with a distributed end-effector.

Five different target objects are modeled for the simulation. They are all convex 2D target objects. Fig.3.33 shows their geometrical settings.

I carry out two groups of implementations with the simulation to find answers to the three problems. The first implementation aims at problems (1) and (3), namely how to retract fingers and how to choose the selecting parameter τ. The second implementation aims at problem (2), namely how is the performance of the collaboration.

A common point of both groups is how to test whether a finger formation cages the target objects. Note that this is how to perform caging test with simulation. It is different from the caging test algorithm discussed in the beginning of this chapter. I propose to perform twelve tests to check whether a finger formation cages the target objects in simulation. The major idea of these tests are to check caging by rotating the end-effector. Their settings are as following. (1) The end-effector is inclined to induce random motions of the target object during rotation. I set the upperarm joint (3rd DoF in Fig.3.32) to $1.3rad$ to make inclination. This joint parameter is chosen because the total inclination of the end-effector palm becomes $0.14rad$ after forward kinematic calculation[6]. The $0.14rad$ inclination of the palm could induce random motions of the target object inside the palm as well as stop the target object from falling out of the palm. (2) The rotation wrist (the 5th DoF in Fig.3.32) is rotated with two different speeds to make the random motions more dramatic. During

[6]Each joint of the Katana Arm has an initial rotation and a mechanical stopper. Readers may refer to its document for details.

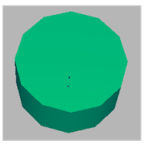
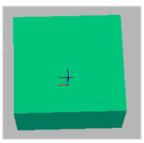
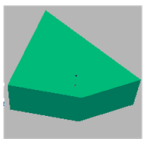
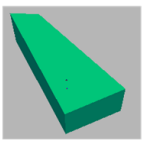
Names	Geometry (*cm*)	Figure
Cylinder	***radius*=5.0 *height*=5.0 *subdivision*=12**	
Box	***radius*=10.0 *length*=10.0 *height*=5.0**	
Polygon A	***vertexlist*={{6.0, -2.5}, {0.0, -5.0}, {-5.0, -2.5}, {2.5, 5.0}, {6.0, -2.5}} *height*=5.0**	
Polygon B	***vertexlist*={{5.0, 5.0}, {6.0, -2.5}, {0.0, -5.0}, {-5.0, -2.5}, {2.5, 5.0}, {5.0, 5.0}} *height*=5.0**	
Polygon C	***vertexlist*={{5.0, 5.0}, {0.0, -5.0}, {-5.0, -2.5}, {2.5, 5.0}, {5.0, 5.0}} *height*=5.0**	

Figure 3.33: Geometric settings of the convex 2D target objects.

the tests, the rotation wrist is actuated in a slower speed $1rad/s$ and a faster speed $10rad/s$. The slower and faster speed together make more random motions and therefore make caging test more convincing. If only one rotating speed is employed, random motions of the target object during rotation might be de-randomized and the testing may return a false result. (3) Along with different rotating speeds of the rotation wrist, the base of the Katana Arm is actuated to different orientations. This is also a strategy to make random motions more dramatic. The more dramatic the random motions are, the more convincing our caging tests. Orientations of the base are actuated to every $\frac{\pi}{6}$ in $[0, \pi)$ to perform caging tests. Making together the three settings, the total test number would be twelve with $0.14rad$ inclination of the palm. A clear list of each test is shown as following. If a formation could pass the twelve tests during simulation, it is supposed to be able to cage the target object. The twelve tests are boring procedure but they are a convincing way to ensure caging[7].

[7]I agree that the tests can be alternatively performed by using the caging test algorithms introduced in the beginning of this chapter. However, I prefer this simulation due to two reasons. The first one is, as finger number increases the caging test algorithms increases dramatically since n_f is an exponential number in $O(n_v^{n_f} \cdot s \cdot m)$. The second one is, we can see the performance of our algorithms more intuitively with the Webots simulation.

Test 1 Base Orientation: 0, UpperArm: $1.3rad$, Rotation wirst speed: $1rad/s$.

Test 2 Base Orientation: 0, UpperArm: $1.3rad$, Rotation wirst speed: $10rad/s$.

Test 3 Base Orientation: $\frac{\pi}{6}$, UpperArm: $1.3rad$, Rotation wirst speed: $1rad/s$.

Test 4 Base Orientation: $\frac{\pi}{6}$, UpperArm: $1.3rad$, Rotation wirst speed: $10rad/s$.

Test 5 Base Orientation: $\frac{\pi}{3}$, UpperArm: $1.3rad$, Rotation wirst speed: $1rad/s$.

Test 6 Base Orientation: $\frac{\pi}{3}$, UpperArm: $1.3rad$, Rotation wirst speed: $10rad/s$.

...

Note that Webots uses Open Dynamics Engine (ODE) [Smith, 2013] for physical computation. I set the gravity and columb friction coefficient of the ODE engine in Webots $(0, -9.81, 0)$ and 1×10^{-5}. ODE does not allow 0 friction coefficient and therefore I set it to a small value 1×10^{-5}. Besides the common point, each group involves different details. These details will be discussed separately in their own contexts.

3.3.4.1 Group I – Choosing parameters for the faster robust caging

The faster robust caging algorithm proposed in Fig.3.27 is only a framework. We need to further consider the problems (1), (3), (4) and choose two parameters to make it working. They are

- τ The parameter to collaborate $Q^{\mathsf{R}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ and $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$. This symbol has been defined in previous contexts.
- d_{ret} This parameter denotes the maximum distance that the fingers could be retracted from the boundary of a target object. This symbol is new and it depends on the direction of retraction.

Choosing the parameters requires testing different τ values and testing whether caging maintains continuously during a retracting procedure. Say, we can set τ with different values and check the d_{ret} of each τ. The τ that induces largest d_{ret} would be the most satisfying parameters. Fig.3.34 shows the procedure.

In the beginning, I initialize τ with 0 and increase it step by step. Let us denote one step with τ_{step}. Then, at a step i, the value of τ would be $\tau = 0 + i \cdot \tau_{\mathsf{step}}$. This would be the threshold to the immobilization optimization algorithm. The immobilization optimization algorithm has been shown in Fig.3.27. Readers can replace the purple box in Fig.3.34 with the upper frame of Fig.3.27 to better understand its role. Result of the immobilization optimization algorithm is an optimized immobilization formation. This optimized immobilization depends on the settings of parameter τ. We can see different τ may correspond to different optimized immobilization formations.

The optimized immobilization formation would be an input to the blue frame of Fig.3.34. In this blue frame, the offset between fingers and boundary of target objects are increased

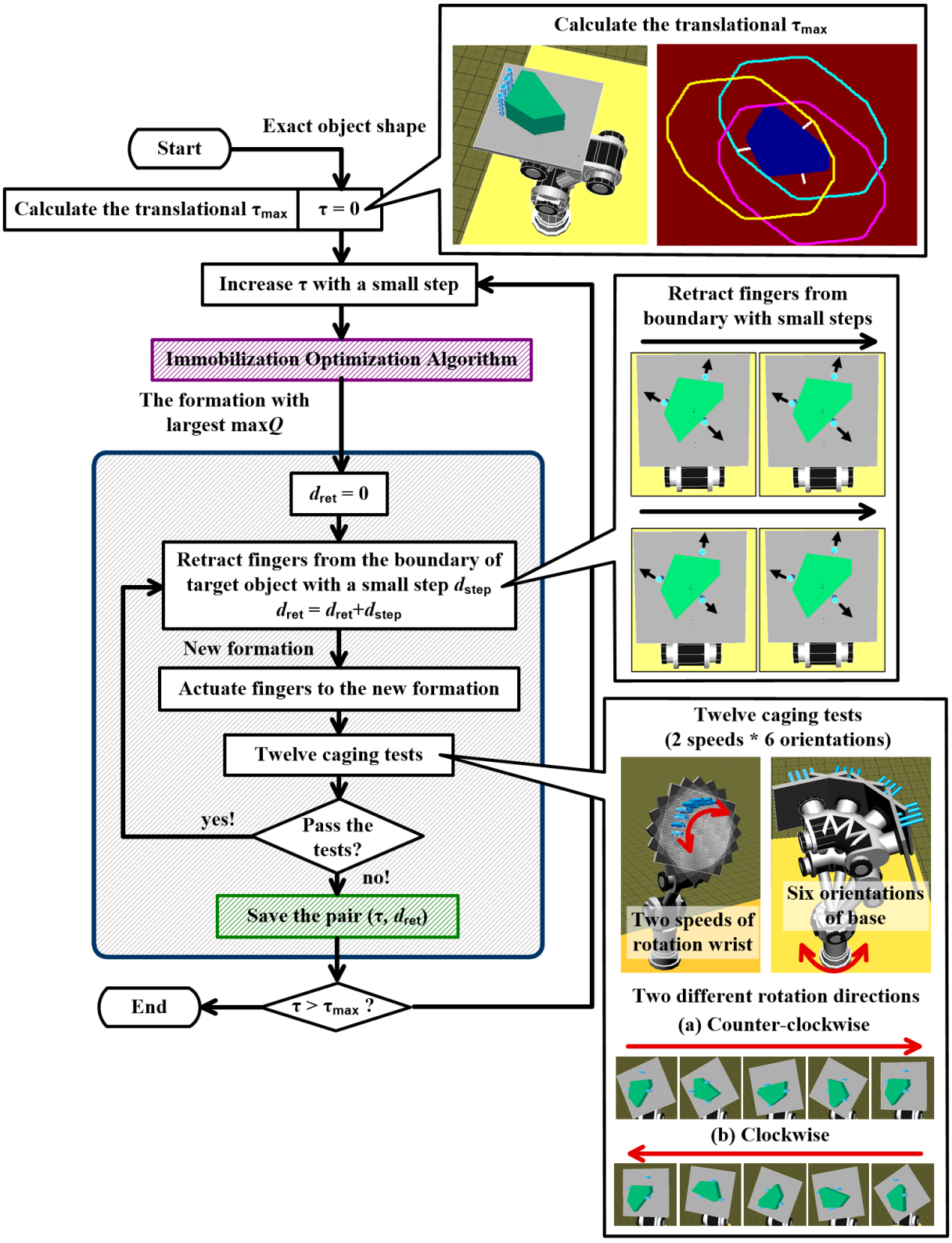


Figure 3.34: Choosing the proper parameters.

step by step to check the maximum robustness of this optimized immobilization formation to caging breaking. In the beginning of this blue frame, I initialize d_{ret}, namely the distance of retraction, with 0 and increase it by d_{step} in each loop. The maximum d_{ret} that fingers can retract from boundary of target objects without breaking caging is the maximum robustness of the optimized immobilization formation. Therefore, the result of this blue frame would be a pair where the first element is an optimized immobilization formation while the second element is the maximum retraction distance d_{ret}. Since an optimization immobilization formation corresponds to a certain parameter settings of τ. This pair can be written as (τ, d_{ret}). During the simulation this pair will be saved for further analysis. The green box in the blue frame of Fig.3.34 shows the saving process of the pair.

Given a target object, the procedure in Fig.3.34 can output a series of (τ, d_{ret}) pairs. We can analyze this pair and find how to set τ to get the maximum retraction distance d_{ret}.

There are two details that need further consideration in this procedure. The first one is how to define one step, say how to choose τ_{step} and d_{ret}. They should not be set to a fixed value since they depends on the shape and size of target objects. For example, d_{ret} of a $100.0cm \times 100.0cm$ rectangular object is surely different from a $radius = 10.0cm$ circular object. If we choose fixed values, the pair (τ, d_{ret}) won't be able to imply the robustness of caging. We need to choose τ_{step} and d_{ret} with respect to a certain reference. The translational $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\ldots,\boldsymbol{f_{n_f}}\}}$ of a target object is a good reference candidate. We name the translational $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\ldots,\boldsymbol{f_{n_f}}\}}$ of a target object the translational robustness and denote it with τ_{max}. This τ_{max}, as we analyzed before, can be calculated quickly. It at the same time encodes geometric information of target objects. Consequently, I prefer choosing it as the reference.

This τ_{max} should be calculated with respect to $\tau = 0$. The calculation is shown in the beginning of the flowchart in Fig.3.34. Note that only the τ_{max} at $\tau = 0$ encodes the geometric information of target objects and is calculated as the reference since this is the translational caging for a target object. When $\tau \neq 0$, the translational robustness is no larger than τ_{max}. Given a target object, we can pre-calculate τ_{max} and prepare it for later reference in defining τ_{step} and d_{ret}. In my simulation, the τ_{step} and d_{ret} are chosen to be $0.05\tau_{\mathsf{max}}$. That is to say, at a ith outer looping step, τ is set to $0 + i \cdot 0.05\tau_{\mathsf{max}}$. At the jth inner looping step, d_{ret} is set to $0 + j \cdot 0.05\tau_{\mathsf{max}}$. Each of the pair ($\tau$, d_{ret}) would be a pair of values with respect to reference τ_{step}, namely ($0 + i \cdot 0.05\tau_{\mathsf{max}}$, $0 + j \cdot 0.05\tau_{\mathsf{max}}$).

The second detail that needs further consideration is the retraction direction when performing d_{ret}=$0 + j \cdot 0.05\tau_{\mathsf{max}}$. In order to ensure the result of faster robust caging algorithm as robust as possible, we can give an intuitive answer. That is to retract fingers along a direction that can break caging as quickly as possible. We name this direction the "Breaking" direction. If the fingers can be retracted with quite large distance along the "Breaking" direction, it potentially can be retracted longer along the other directions and the result of the faster robust caging algorithm would be more convincing. However, I wonder if another direction is really worse comparing with the "breaking" direction. Therefore, in my simulation, I manually define another retraction direction for comparison. The figures in the middle dialog box of Fig.3.34 shows the other retraction direction. In this case, each

finger is retracted from the surface of target objects along normal direction of the surface. I name this retraction direction the "normal" direction. The "breaking" direction can be calculated by exploring $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\ldots,\boldsymbol{f_{n_f}}\}}$. Recall expression (3.26), the $Q^{\mathsf{T}}_{\{\boldsymbol{f_1},\boldsymbol{f_2},\boldsymbol{f_3}\}}$ is the minimum distance of all $d^{\mathsf{T}}_{\boldsymbol{f_1}}, d^{\mathsf{T}}_{\boldsymbol{f_2}}, \ldots, d^{\mathsf{T}}_{\boldsymbol{f_{n_v}}}$. Each $d^{\mathsf{T}}_{\boldsymbol{f_i}}$ denotes the minimum distance from the finger f_i to the translational caging boundary. This minimum distance has a direction which means the "short-cut" to breaking. Retracting fingers along this "short-cut" direction would break caging quickly. Therefore, I use it as the "breaking" direction. The white segments in the upper dialog box of Fig.3.34 show these "breaking" directions. I save the (τ, d_{ret}) pairs by both retractions along "normal" directions and "breaking" directions for analysis.

Here is a summary of the detailed techniques used in the simulation and in choosing parameters.

τ	τ is used to filter out the translational caging formations.
τ_{max}	τ_{max} is the translational robustness of a target object.
d_{ret}	d_{ret} is the distance that fingers are retracted from target objects.
Normal	Normal is one retraction direction. It is along the normals of object boundary.
Breaking	Breaking is another retraction direction. It is along the direction of $d^{\mathsf{T}}_{\boldsymbol{f_i}}$.
(τ, d_{ret})	(τ, d_{ret}) is a pair of value with respect to τ_{max}. This pair implies the maximum retraction distance d_{ret} with respect to τ.

In the following part I will show the saved (τ, d_{ret}) of different objects and analyze them. I divide the five objects into two categories. One is the symmetric "box" and "cylinders" and the other one is the other three general polygons.

The symmetric "box" and "cylinders"

Firstly, let us look at the results with the two symmetric "Box" and "Cylinder" shown in Fig.3.33. The reason why I separate the symmetric objects from the other polygons is there is no need to change τ for symmetric objects. Any τ results into the same pair. We can say τ is not available in symmetric cases. The third column of Fig.3.35 shows the "N/A" status of τ.

			Maximum Retraction		
Targets	τ_{max}	τ	**Normal**	**Breaking**	**Least Finger Number**
Cynlinder	**1.6*cm***	**N/A**	**0.45** τ_{max}	**0.45**	**4**
Box	**5.3*cm***	**N/A**	**0.35** τ_{max}	**0.35**	**3**

Figure 3.35: Parameter results of the two symmetric objects.

The least number of fingers to cage a "Cylinder" object is three while the least number of fingers to cage a "Box" object is four. The result is the same as our analysis in section 2.2. The last column of Fig.3.35 shows the number. The fourth and fifth columns in Fig.3.35 shows the retraction thresholds along different directions. Since τ is not applicable, there is only one maximum d_{ret}. Any τ results into the same d_{ret} shown in the fourth and fifth columns. Starting from 0.0, I increase τ step by step with $0.05\tau_{\text{max}}$ step length. The caging maintains until $0.35\tau_{\text{max}}$ for "Cylinder" while until $0.45\tau_{\text{max}}$ for "Box". Say, our proposal can offer a $0.35\times1.6cm$=$0.6cm$ threshold for caging "Cylinder" while offer a $0.45\times5.3cm$=$2.4cm$ threshold for caging "Box". Different choices of τ give the same results.

Retracting along different directions are the same in Fig.3.35. It is different from my expectation that "breaking" direction has a smaller retraction distance. This is reasonable for "Box" object since the "Box" object is special that the two directions share the same vector. It is a little confusing for the "Cylinder" object. One possibility is that the "Cylinder"' s robustness is too small to discern the difference. The difference between retraction along "Normal" direction and retraction along "Breaking" direction can be neglected safely. Fig.3.36 shows the details of the two retraction directions and the caged "Cylinder" with different retraction distance. The white segments in (a-1) and (a-2) of Fig.3.36 demonstrate the two different retraction directions of the "Cylinder" object. Fig.3.36(b-1) shows one caging status during the retraction loop where τ=$0.1\tau_{\text{max}}$. Fig.3.36(b-2) shows the maximum retraction distance ($0.35\tau_{\text{max}}$). In Fig.3.36(b-2), a further retraction step shall break caging.

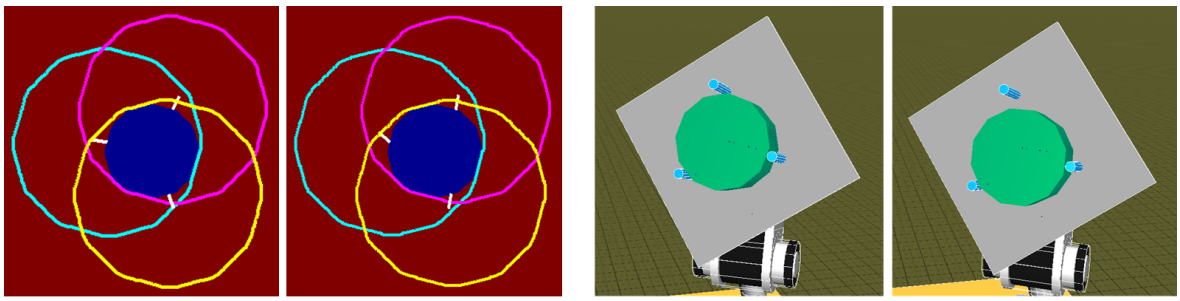
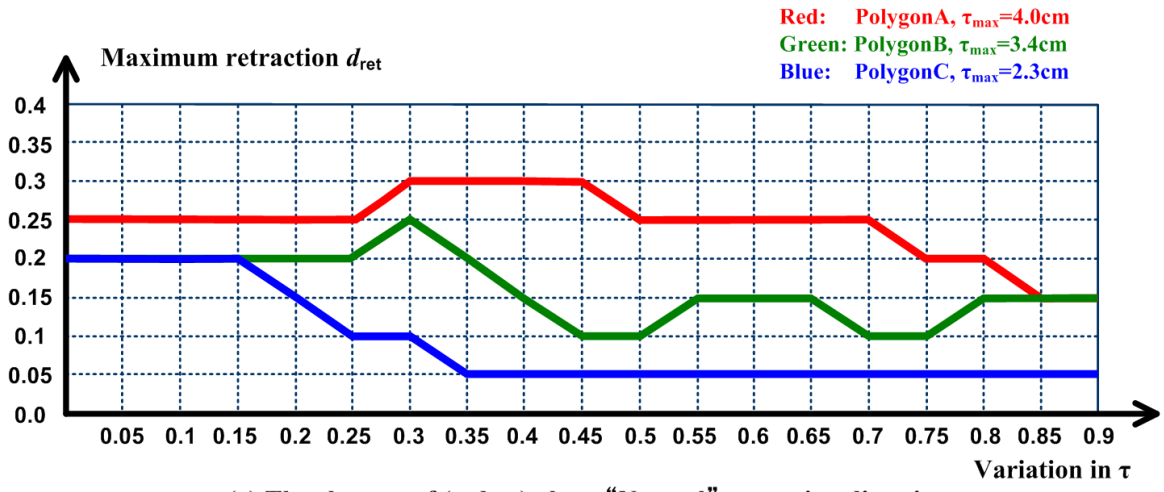


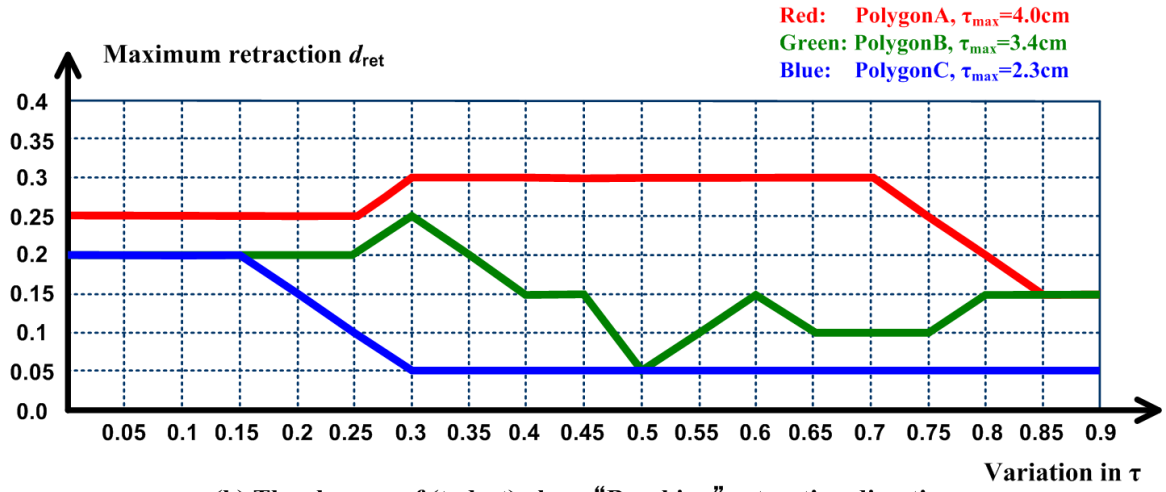
Figure 3.36: A retraction example.

General polygons

Unlike the symmetric objects, τ is applicable to general polygons and each of the general polygons "Polygon A", "Polygon B" and "Polygon C" in Fig.3.33 corresponds to a series of (τ, d_{ret}) pairs. The (τ, d_{ret}) of the three polygons are shown in Fig.3.37 with curves of different colors. The horizontal coordinate of Fig.3.37 denotes the variation in τ while the vertical coordinate of Fig.3.37 denotes the maximum retraction distance before caging breaking.



(a) The changes of (t, dret) along "Normal" retraction direction.



(b) The changes of (t, dret) along "Breaking" retraction direction.

Figure 3.37: Parameter results of the three polygon objects.

Both the horizontal an vertical coordinates in Fig.3.37 are shown with respect to reference τ_{max}. They are a certain proportion of τ_{max}. Take the red curve of Fig.3.37(a) for example. This is the result of "Polygon A" when we retract fingers along "Normal" direction. The maximum retraction distance of this curve appears at τ=$0.3\tau_{\mathsf{max}}$. The τ_{max} of "Polygon A" is shown in the upper-right text of Fig.3.37(a). It is $4.0cm$. Therefore, the maximum retraction distance of this curve appears at τ=$0.3\tau_{\mathsf{max}}$=$1.2cm$ and its value is τ=$0.3\tau_{\mathsf{max}}$=$1.2cm$. That is to say, we have a pair ($1.2cm$, $1.2cm$) for this "Polygon A". The white point in Fig.3.37(a) denotes this ($1.2cm$, $1.2cm$) pair.

Now let us analyze these curves in detail.

a) The choice of τ. The pair (τ, d_{ret}) with maximum d_{ret} along the curves of "Polygon A" and "Polygon B" both appear at τ=$0.3\tau_{\mathsf{max}}$. Therefore, I prefer choosing τ=$0.3\tau_{\mathsf{max}}$ as the filtering parameter of translational constraints. However, this seems quite different on "Polygon C". The curve of "Polygon C", no mater along which direction the fingers are retracted, is always going monotonically. There's not significant superiority around $0.3\tau_{\mathsf{max}}$. This is because this "Polygon C" object is too long or thin and caging is not suitable to cage this kind of objects.

b) Retraction/shrinking direction. We can compare the performance of different retraction directions by comparing the results of Fig.3.37(a) and Fig.3.37(b). The conclusion is retraction along either direction can both be flexible enough. There is not too much difference between the two different directions. The breaking direction seems to be a better choice on "Polygon A" while a worse choice on "Polygon B" and "Polygon C". The reason is probably the same as symmetric objects. That is, the robustness is not very large and it doesn't result into significant difference. Retraction along "Normal" direction and retraction along "Breaking" direction have nealry the same performance and we could choose either safely.

Thin objects like "Polygon C" is not suitable for caging, but what can we do if we really want to cage it? The answer is to increase the number of fingers. This relates to the problem of when to increase fingers or how many fingers do we need to cage a target object.

I propose to deal this problem by measuring the maximum retraction distance. Take "Polygon C" for example. The three-finger maximum retraction distance of "Polygon C" is unsatisfying. It is $0.2\tau_{\mathsf{max}}$=$0.46cm$. We can introduce an extra finger to enlarge this maximum retraction distance. Fig.3.38 shows the (τ, d_{ret}) curve with respect to four fingers.

As we can see in the upper-right text of Fig.3.38, τ_{max} increases greatly from $2.3cm$ to $3.6cm$ after introducing an extra finger and the maximum retraction distance grows from $0.2 \times 2.3cm$=$0.46cm$ to $0.25 \times 3.6cm$=$1.08cm$. It is interesting that this maximum retraction distance appears around τ=$0.3\tau_{\mathsf{max}}$ which is the preferred value of "Polygon A" and "Polygon B". Actually, if we choose a different τ, the result formation would be quite different. Fig.3.39 shows the difference. The formations with τ=$0.1\tau_{\mathsf{max}}$, τ=$0.2\tau_{\mathsf{max}}$, τ=$0.4\tau_{\mathsf{max}}$ and τ=$0.5\tau_{\mathsf{max}}$ cannot offer as much robustness as the formation as τ=$0.3\tau_{\mathsf{max}}$. Their maximum retraction distances are $0.15\tau_{\mathsf{max}}$, $0.15\tau_{\mathsf{max}}$, $0.15\tau_{\mathsf{max}}$ and $0.11\tau_{\mathsf{max}}$ respectively. We can also find evaluate the performance of each formation in Fig.3.39 by observation and prediction. Comparing with the third figure where τ=$0.3\tau_{\mathsf{max}}$, the first result biases toward the fatter

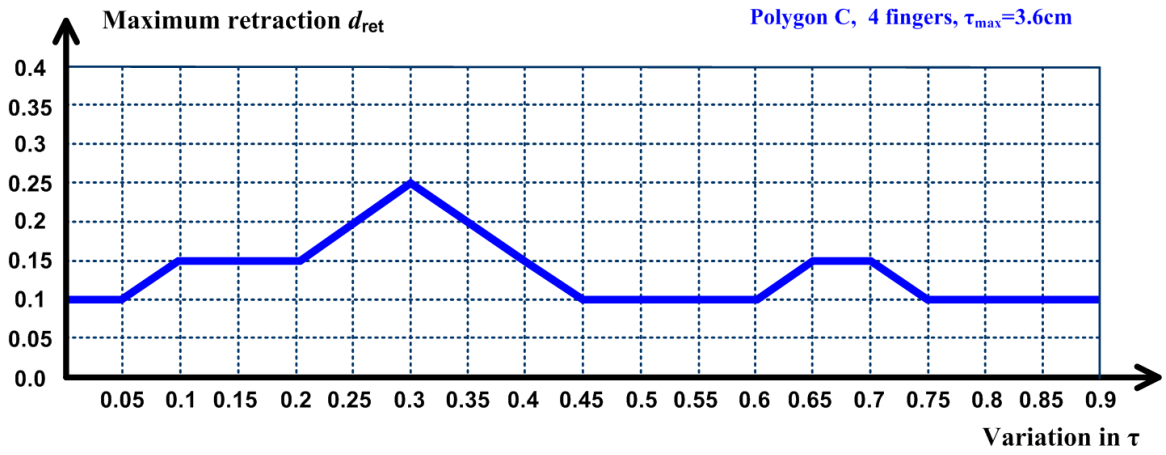


Figure 3.38: Parameter results of "Polygon C" with four fingers.

end of "Polygon C". This bias may cause certain problems. The second figure involves two closely arranged fingers. Due to those two closely arranged fingers, the formation in the second figure deteriorates into three finger case. The fourth and fifth figure both has a large gap between fingers. The fourth one has a large gap between the lower two fingers while the fifth one has a large gap between the upper two fingers. When there is uncertainty, the target object may drop out easily from those gaps. The third one where τ=0.3τ_{max} is a satisfying result.

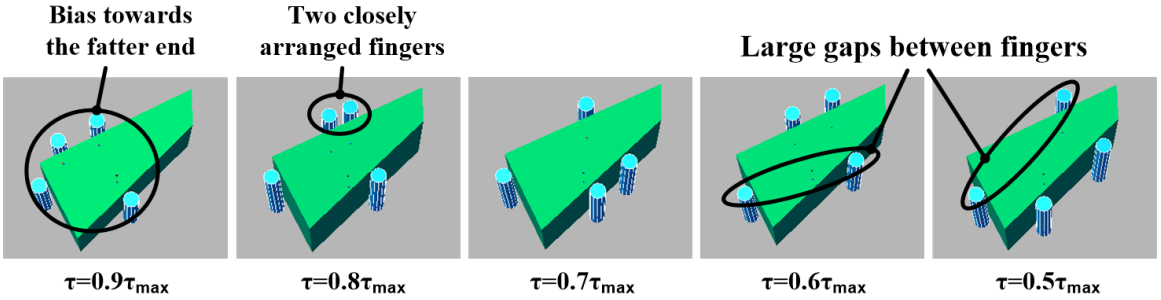


Figure 3.39: Different results due to different parameter settings.

By the way, I would like to explain more about the "Polygon C" object. As the target object becomes thinner and thinner, it may require more and more fingers to get a satisfying retraction distance. In the extreme case where target becomes a long thin stick (for instance, a pen), we may never cage it and have to refer to friction for help. This is the common sense in our daily life.

After the first group of simulation, we can make clear the answer to problems (1), (3) and (4) of the faster robust caging algorithm.

(1) How to retract fingers? The answer is we can either retract fingers along normal direction or breaking direction. This is no significant difference between them.

(3) How to choose the selecting parameter τ? The answer is to choose τ=$0.3\tau_{\mathsf{max}}$. However, we should ensure that the maximum retraction distance is larger than a certain threshold which can be defined by users. If we hope to have a caging with at least $0.5cm$ robustness, it is advisable to use a threshold of $1.0cm$. This is because when the maximum retraction distance is larger than $1.0cm$, we can safely retract fingers with $0.5cm$. After retraction, the fingers have both $0.5cm$ robustness against collision with target boundary and $d_{\mathsf{max}} - 0.5cm$ robustness against caging breaking.

(4) how to decide the number of fingers? The answer is to decide finger number by maximum retraction distance and a user-defined threshold. If the maximum retraction distance is smaller than the user-defined threshold, we should introduce extra fingers to increase the maximum retraction distance so that caging could be more robust.

Now we can mix up everything and summarize the complete faster robust caging algorithm. It is shown in Fig.3.40. This figure illustrates in detail how to set those parameters and how to decide the number of fingers. Especially, it employs the normal direction as the retraction direction. There is no remaining problems in the summarized algorithm. However, I agree that the parameter settings of this complete version suffer from doubts. For example, readers may consider that these empirical settings are summarized from five objects. The number five is too small. Actually, testing various object is dull due to the manual step-by-step tests. For instance, each object have to go through $(\tau_{\mathsf{max}}/0.05\tau_{\mathsf{max}})\times 12n_d$=$1200n_d$ tests and we have to perform $6000n_d$ caging tests for five objects. Here n_d denotes the number of retraction steps in each inner loop. Therefore, only five objects are employed in Group I. The five target objects are not selected arbitrarily. They are representative enough to represent "convex" target objects. In case of more doubts about these parameter settings and at the same time in order to answer the problem (2). I will use the algorithm to test its performance in the next group of simulation.

3.3.4.2 Group II – Evaluating performance of faster robust caging

The second group of simulation aims at evaluating the performance of the algorithm and parameter settings shown in Fig.3.40. As we know, uncertainties come from two aspects, namely (1) uncertainty from perception devices and (2) uncertainty from actuation of fingers. These two aspects are demonstrated in Fig.3.40. I emulate these noises by adding Gaussian noises to vertices of target objects and I suppose that the noises model both uncertainty in perception devices and uncertainty from actuation. Noises from actuation is integrated into noises of perception in this case. Specifically, for each vertex $(v_x,\ v_y)$ of the groundtruth shape, a random noise $(\delta v_x,\ \delta v_y)$ is generated with respect to a Gaussian distribution $\mathsf{N}(0, \sigma^2)$ and added to $(v_x,\ v_y)$. The noisy shape $v_{i_\mathsf{x}} + \delta v_{i_\mathsf{x}}, v_{i_\mathsf{y}} + \delta v_{i_\mathsf{y}},\ i = 1, 2, \ldots, n_v$ is taken as the noisy shape input into the faster robust caging algorithm. Then the algorithm works exactly following the flowchart in Fig.3.40. The noisy shape is used to calculate a robust caging formation while the groundtruth shape is used for caging test. We need to calculate

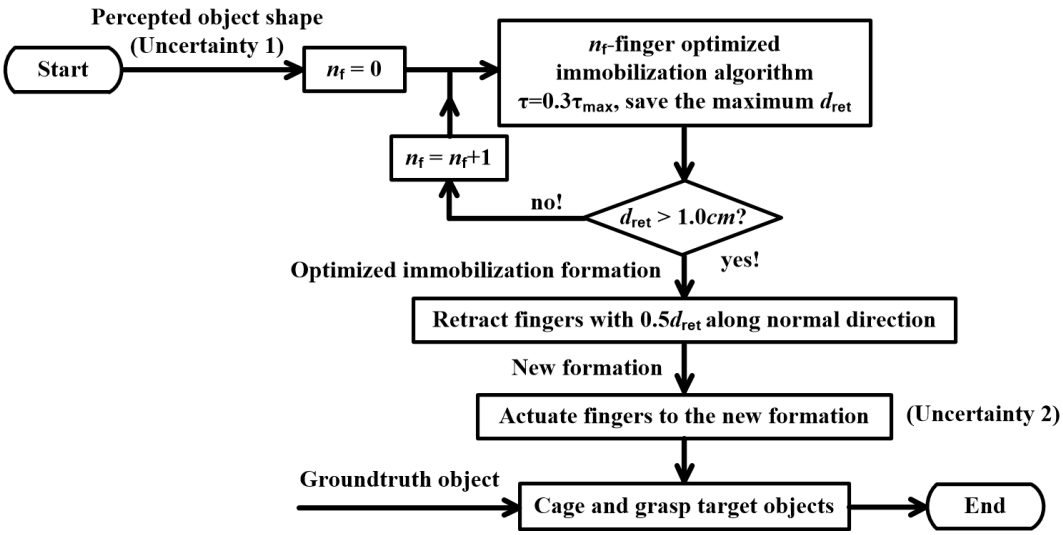


Figure 3.40: The final faster robust caging algorithm with empirical parameter settings.

a robust caging formation for each Gaussian noise and evaluate whether this Gaussian noise can be endured by formation.

The following figure shows the procedure of adding Gaussian noises to the ground-truth shape.

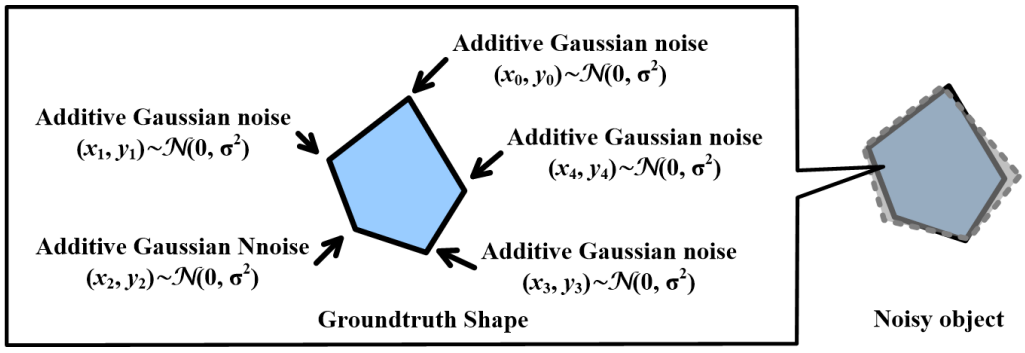


Figure 3.41: Adding noises to groundtruth shapes to evaluate the robustness of caging.

The performance of faster robust caging under Gaussian noises are shown in Fig.3.42. The noisy shapes in this figure are selected and rearranged to better compare performance the robustness. The first column of Fig.3.42 denotes the meaning of each row and which object in Fig.3.33 these shapes correspond to. It also shows how many fingers are needed to ensure $d_{ret} > 1.0cm$. The second column of Fig.3.42 denotes the groundtruth shapes of target object. The other column shows the changes of shapes with different Gaussian noise and whether the noisy shapes could be caged. If a noisy object can pass the cage test, its "result" row would be "success". Or else the "result" row could be "collision" or "breaking"

which corresponds to the two possibilities of failure. The "collision" failure means the fingers collides with target objects and the "breaking" failure means caging test fails and the target object drop out of the formation. The results are according to $\tau = 0.7\tau_{\mathsf{max}}$ with $0.5d_{\mathsf{ret}}$ retraction.

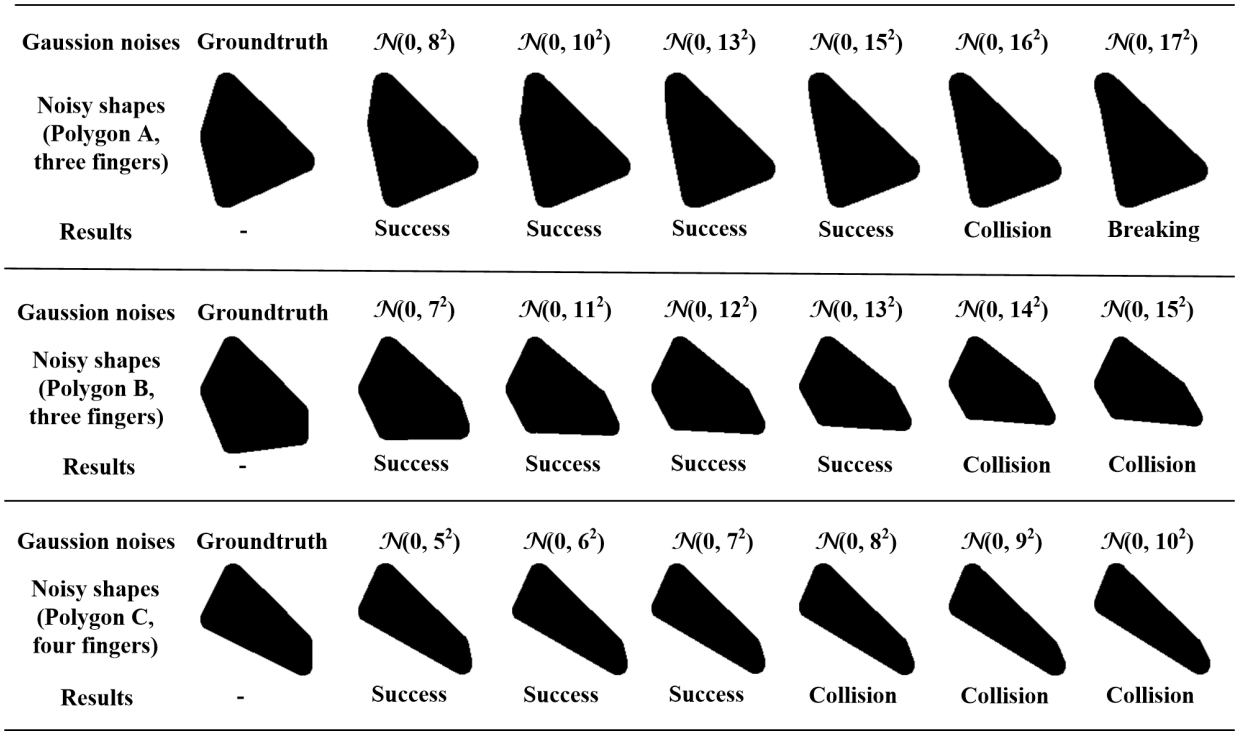


Figure 3.42: Performance of the faster robust caging under Gaussian noises.

As we can see from Fig.3.42, it is safe to endure more than $\mathsf{N}(0,\ 1.3^2)$ and $\mathsf{N}(0,\ 1.5^2)$ Gaussian noises with "Polygon A" and "Polygon B" with three fingers. After introducing an extra finger, it is also safe to endure more than $\mathsf{N}(0,\ 0.7^2)$ Gaussian noises on "Polygon C". These results show that my proposal is potentially good to work with Swiss Ranger. The parameter settings and faster robust caging algorithm could be a promising way to calculate robust caging formations for convex target objects. I will show some applications of this algorithm in the next chapter.

Chapter 4

Applications I – Distributed Agents

The chapter includes two applications of the faster robust caging algorithm. The first one is our distributed end-effector which has been conceptually illustrated in Fig.1.1 and Fig.1.2. This application uses both KINECT and Swiss Ranger to perceive target objects. We will see the comparison of their performance and the feasibility of my algorithm. The second one is multi-robot co-operation. This application was widely studied by Sudsang, Wang and Pereira, etc. My work here is different from them and it uses the faster robust caging algorithm to calculate caging positions of each robots. The second application uses a V100:R2 OptiTrack [NaturalPoint, 2013] to track and control the motion of each mobile robot. We will see how the algorithm work with it. Note that the "graping by caging" part of this algorithm is not used in the applications. That is because the "grasping by caging" part requires local sensors to perceive the contacts between agents and target objects. Local sensors are not available to the applications of this thesis. However, readers may see the "grasping by caging" procedure in Fig.3.30. In that case, "grasping" is a simple demonstration, it is blind and suffers from the danger of squashing target objects.

4.1 Caging on the Distributed End-effector

4.1.1 Details of the end-effector

4.1.1.1 Hardware implementation

The same as the conceptual illustration in Fig.1.1 and Fig.1.2, the Distributed End-effector is composed of an $x-y-\theta$ actuator and several distributed fingers. The $x-y-\theta$ actuator works as the palm of a robotic hand while the distributed fingers work as the fingers of a robotic hand. Fig.4.1 shows the implemented $x-y-\theta$ actuator. It has two motors to control x and y translation, one motor to control θ rotation and an extra motor to control the insertion of pins. Readers may refer to the connecting module of Fig.6.14 and related texts to better understand the θ motor and pin-insertion motor.

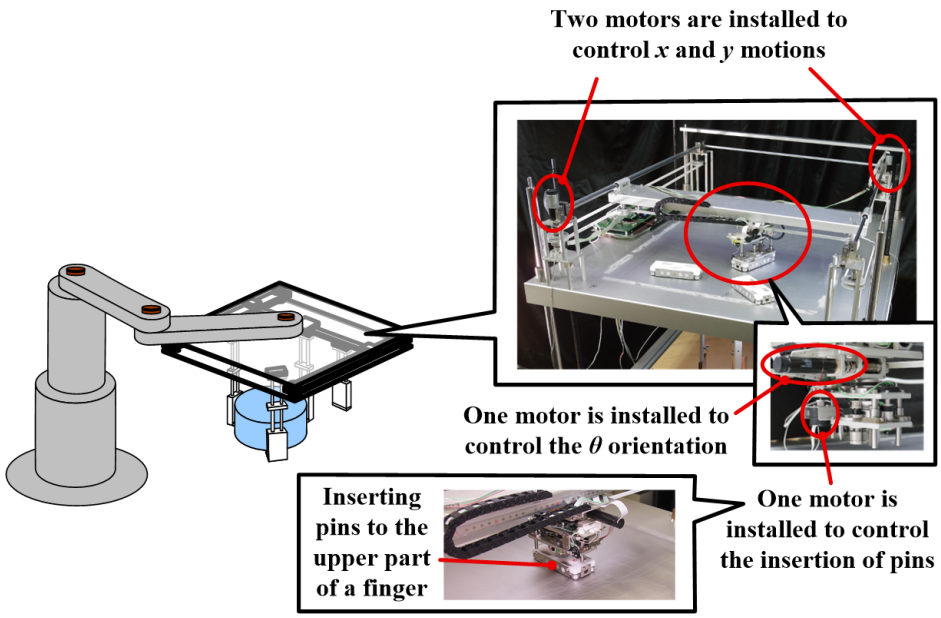


Figure 4.1: The implemented $x - y - \theta$ actuator and the roles of four motors.

The $x-y-\theta$ actuator actuates a finger with three steps. In the first step, it attaches itself to a finger by inserting pins to the fingers. Then, it translates and rotates the attached finger to the position calculated by the faster robust caging algorithm. After that, the $x - y - \theta$ actuator detach itself from the upper part of the finger and finish the actuation. Readers may review Fig.1.2 to recall this procedure. Fig.4.2 shows the details of one distributed finger. It involves a prismatic module which stretches fingers down and a nail module which will be inserted into the bottom of target objects.

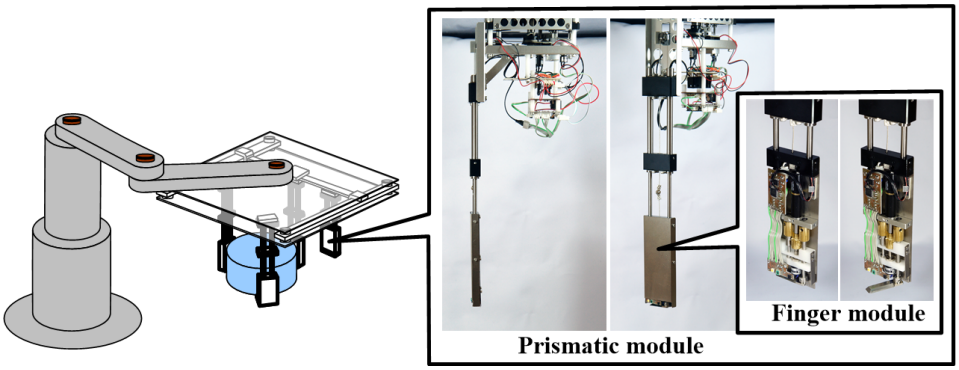


Figure 4.2: The implemented one distributed finger.

Since there are not enough room to install a manipulator shown in the left part of Fig.4.1 and Fig.4.2 to move this end-effector, I use some tricks to test the performance of

my algorithm. Rather than moving the end-effector, I install a sliding plate under the end-effector and move the sliding plate instead. The lower-left dialog box of Fig.4.3 shows the sliding plate. At the corners of this sliding plate, four markers are installed to help calibrate perception devices which are installed on the top. The perception devices involve a KINECT and a Swiss Ranger, they are emphasized in the upper-left dialog box of Fig.4.3.

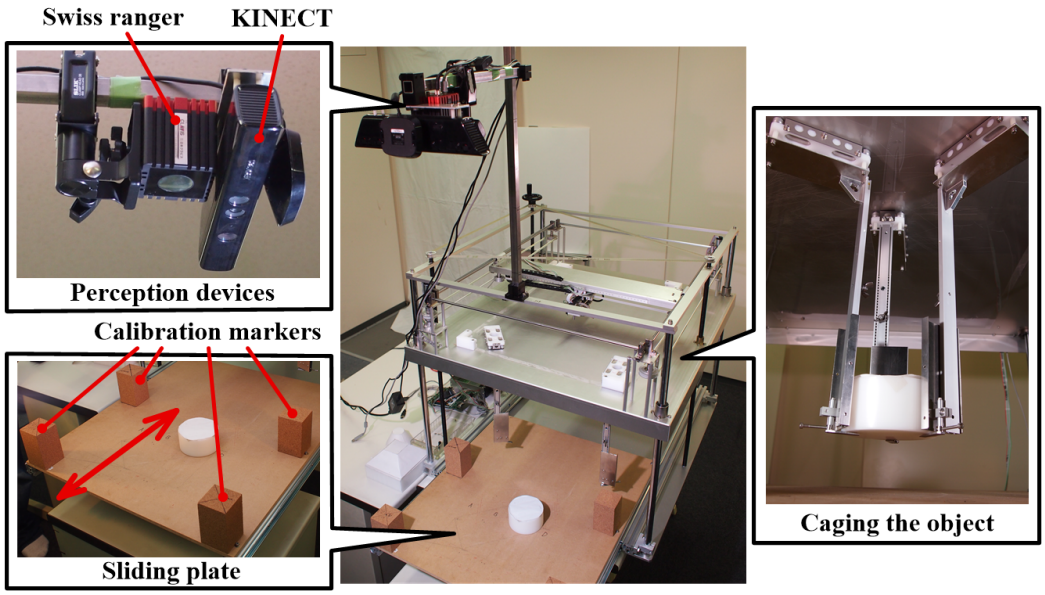


Figure 4.3: The sliding plate and the perception devices.

The right dialog box of Fig.4.3 shows an action where the distributed end-effector cages a cylindrical target object. Note that in this application I use some dummy fingers instead of the complete version shown in Fig.4.2. These dummy fingers have the same mechanical structure as Fig.4.2 except that they are not equipped with motors. These dummy fingers save costs of our implementation as well as being enough to demonstrate the faster caging algorithm.

4.1.1.2 Software integration

In the lower level, the end-effector is controlled by a AVR Atmega2560 micro-controller which can perform basic commands from computers. On the one hand, the micro-controller encapsulates low-level controls like proportional-integral-derivative algorithms and deals with the low-level protocols like formats of encoder data. On the other hand, it receives commands from computers through serial communications and interrupts and controls motors according to those commands.

In the higher level, the end-effector is controlled by MATLAB2011b running on a MacBook. Fig.4.4 shows the MATLAB interface of the high-level controller and its relationship with the low-level micro-controller.

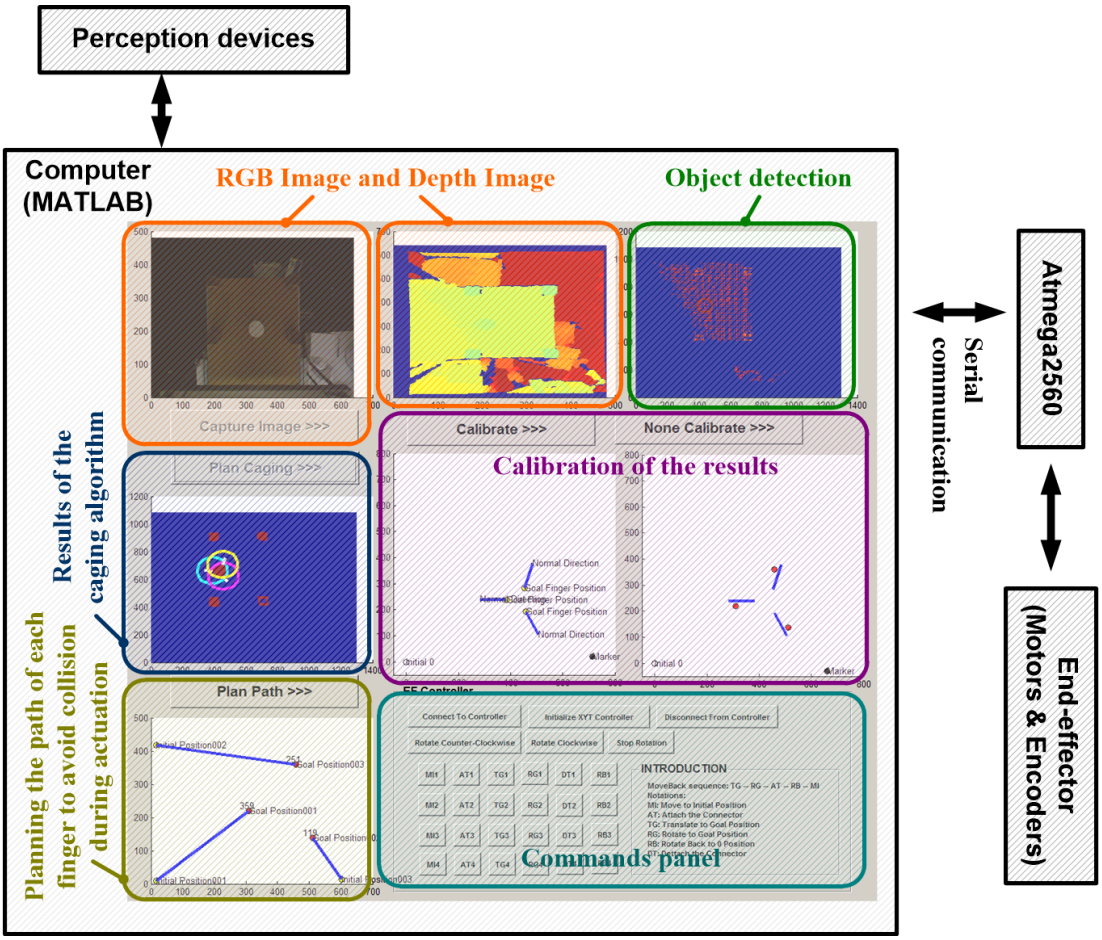


Figure 4.4: Connections between the high-level computer and the low-level micro-controller.

It can be seen from Fig.4.4 that the high-level computer performs the following tasks consequently. (1) Processing the information collected from perception devices. This task is shown by the orange boxes and the green box in Fig.4.4. (2) Calculating an optimized caging formation. This task is shown by the blue box in Fig.4.4. (3) Calibrating the perception devices. This task is shown by the purple box in Fig.4.4. (4) Planning actuation paths of each finger. This task is shown by the yellow box in Fig.4.4. After that, the high-level computer encode the results of those tasks into commands and send the commands the micro-controller through serial communication. This is shown by the cyan box (command panel) in Fig.4.4. The micro-controller actuates the motors on the end-effector according to the received command to perform caging.

4.1.2 Demonstration and analysis

We have reviewed the hardware implementation and software integration in foregoing texts. Next, it is time to see the results of the faster robust caging algorithm. We will see two topics in this sub-section. One is comparison between KINECT and Swiss Ranger and the other one is the demonstration with different convex objects.

4.1.2.1 Comparison between KINECT and Swiss Ranger

I tested two different perception devices, namely the KINECT and the Swiss Ranger. The two perception devices have different precisions so that we can better see the robustness of the faster robust caging algorithm.

KINECT and Swiss Ranger represent two different types of depth cameras. The KINECT calculates the offset of the dot patterns from their factory calibrated ideal positions. Depth of a certain point is evaluated by the offset. The KINECT camera is composed of an IR source, an IR camera and a RGB camera. Its resolution is 640×480. The Swiss Ranger measures changes in the light phase at each image sensor signal. The phase difference divided by period length indicates a portion of maximum measurement, namely the depth value. The Swiss Ranger is composed of a light source and an image sensor. Its resolution is 176×144. Although the Swiss Ranger is lower in resolution, it is higher in precision and of course higher in price.

Precision of the depth camera, especially the KINECT camera, depends on its distance to the scenarios. Therefore, I install the two cameras to four different positions to compare their precisions. The four different positions are shown in the upper-left part of Fig.4.5. I measure the distance between two markers shown in the upper-right part of Fig.4.5 and compare the measured results with ground truth value. Groundtruth value of the distance between markers is $510mm$. However, the measured results differ conspicuously. They are shown in the lower bar-graph of Fig.4.5.

The Swiss Ranger has relative higher precision. Its error is less than $10mm$. The KINECT, however, is much worse and introduces an error of nearly $50mm$. The precision of devices introduces limitation to their caging applications. By comparing the caging regions of objects in Fig.3.33, we can roughly have a reference value that the robustness of caging is about $\frac{1}{10}$ of object diameter [1]. Note that this is a rough value. It cannot be used to evaluate robustness but it can be used to check whether the perception devices are suitable to some caging applications. The Swiss Ranger is potentially fine to work with objects with target objects whose diameters are no less than $100mm$. The KINECT is much worse. It may fail to work as a suitable perception device unless the diameter of target objects are larger than $500mm$.

Fig.4.6 shows the results with the cylinder object shown in Fig.3.33. The diameter of this cylinder object is $100mm$ and the height of this cylinder object is $50mm$. The caging results at those four different heights in the upper-left part of Fig.4.5 are shown in Fig.4.6(a), (b),

[1] Object diameter can be considered as the diameter of largest outer circle that covers the object

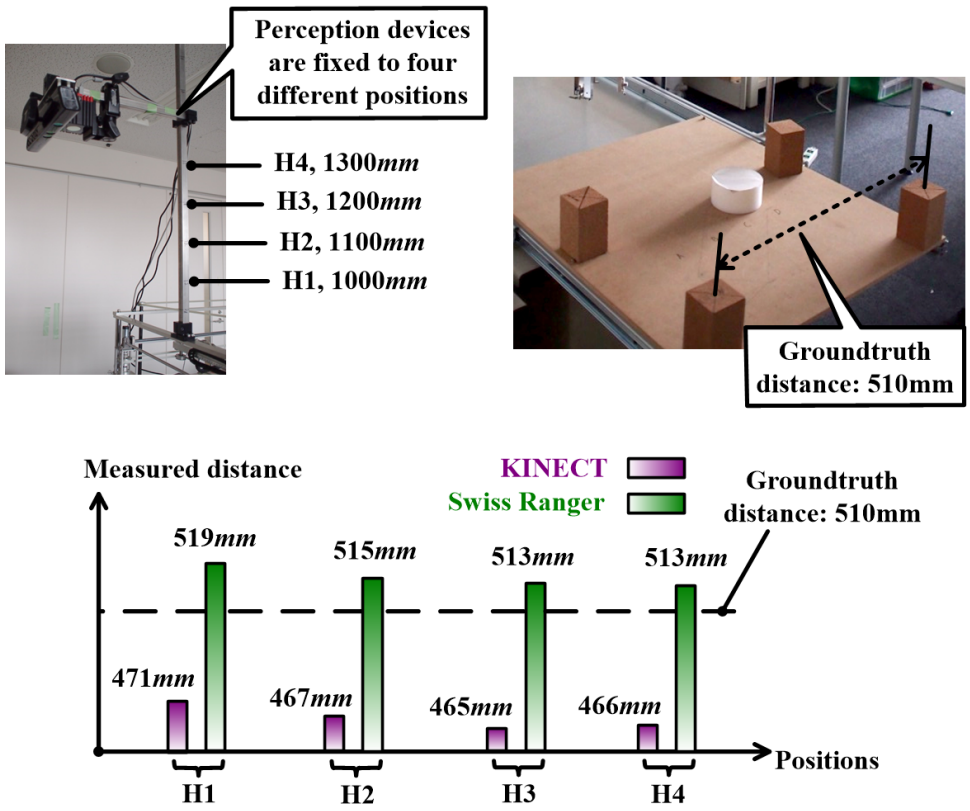


Figure 4.5: Perception errors of KINECT and Swiss Ranger.

(c) and (d) respectively. Both Swiss Ranger and KINECT are involved at each height. The ideal caging results, namely the caging results calculated with perfect shape and position information, are rendered in red while the caging results calculated with cloud points from the perception devices are rendered in green. The Swiss Ranger, especially when its height is higher than H2=1100mm, is robust enough to our faster robust caging algorithm. On the contrary, the KINECT fails due to dramatic offset.

4.1.2.2 Demonstration with various target objects

According to the results of Fig.4.6, I choose the Swiss Ranger and fixed it at height H3=1200mm to provide objects information to the distributed end-effector. The faster robust caging algorithm is carried out on different objects shown in Fig.4.7. It involves a cylinder, a frustum, a trapezoid and a set of reconfigurable cuboids. In total, there are 7 convex objects tested. These objects are not as complex as those used in the simulation, however, they are more common in real world. Most of the objects in real world are of these shapes. Moreover, the trapezoid and the frustum are more challenging due to their inclined side surfaces. That is because the inclined side surfaces cause ambiguity in depth and consequently increase the

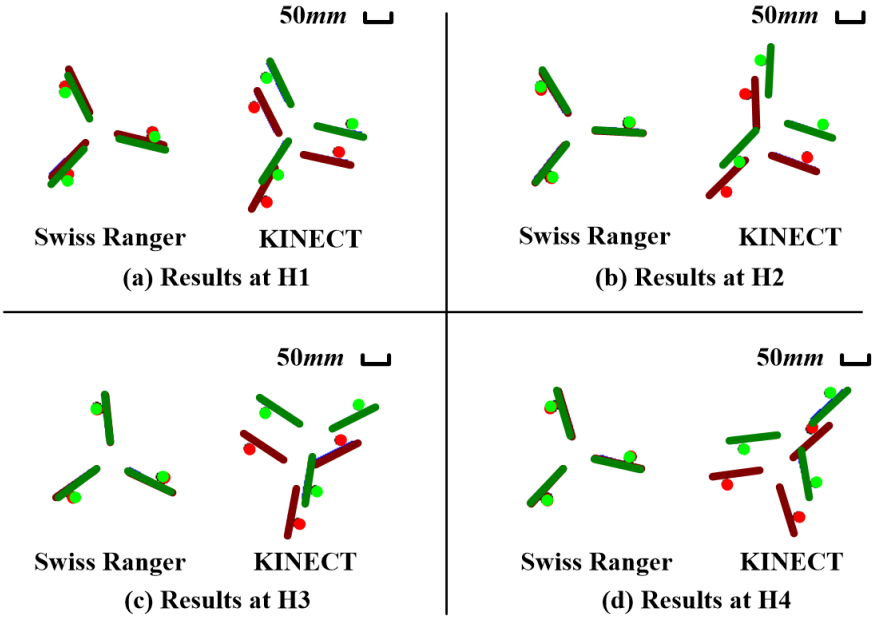


Figure 4.6: Comparing the caging results of different devices.

noises of boundary detection.

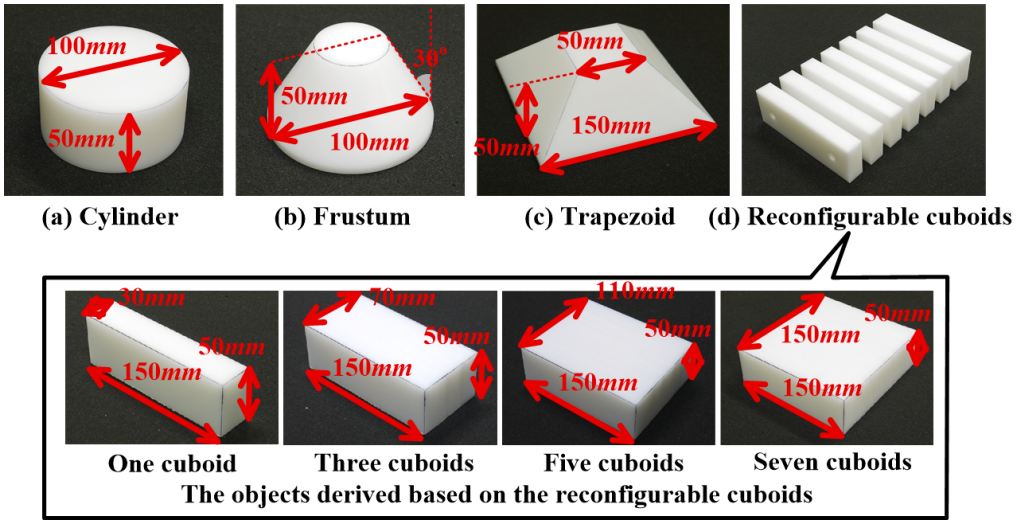


Figure 4.7: The objects used in demonstrating the distributed end-effector.

Fig.4.8 exemplifies a caging process with the circular object. The high-level perception and planning has been shown in Fig.4.3. Fig.4.8 is the low-level control of each distributed finger. The last figure of Fig.4.8 is the caging state of the cylinder object. The caging fingers

offer robustness to collision while maintains a loose closure. Without caging, the distributed fingers may (1) squash target objects or (2) lose target objects.

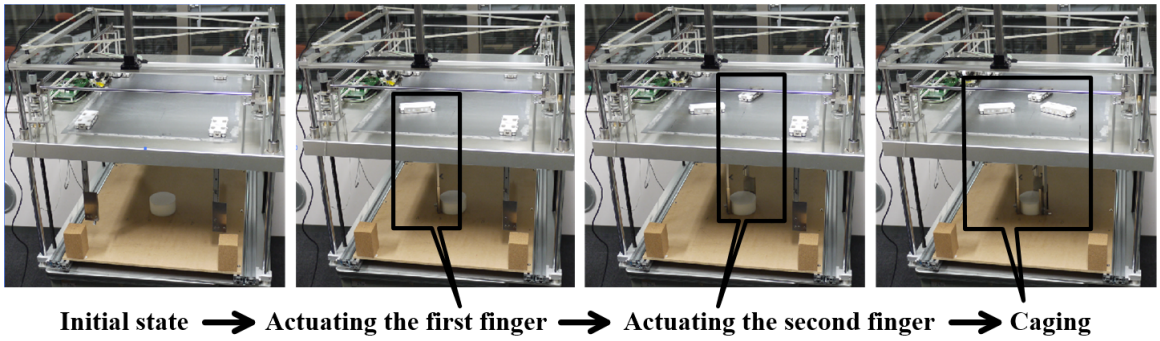


Figure 4.8: The caging process of the circular object.

Fig.4.9 shows in detail the results of each object. Since the frustum and trapezoid have inclined side surfaces, their boundaries are more ambiguous to the Swiss Ranger and there are more noise. In most cases, the noise results into smaller perception results. Smaller perception results cause the fingers to be nearer to target objects. If there is no caging, the nearer fingers would surely squash the objects. The results in figure Fig.4.9(b) and (c) validates this analysis. As the perceived objects become smaller, the gaps between fingers and objects become smaller. If the faster robust caging is not employed and we do not retract fingers from the boundaries of percepted objects, the fingers would sure squash into the target object. The small gap is more obvious in Fig.4.9(c). That is because the inclination of the trapezoid is even larger so that the perceived boundary is even smaller. In the case shown in Fig.4.9(c), there is nearly no gap between fingers and target objects. The operation is on the boundary of failure. The gaps of the other objects are robust enough to guarantee successful caging. They either constrains objects and avoids collision.

All of these six objects can be successfully caged by the distributed end-effector. An exception is the one-cuboid object. Fig.4.10 shows the status when the end-effector fails. The algorithm can find an optimized caging configuration, however, the real-world nail fail to reach beneath the bottom. Actually, this one-cuboid object is like the "Polygon C" object which was analyzed in the simulation of section 3.3.4.1. The one-cuboid object is too thin and it is not suitable for caging. However, if we do want to safely cage it, we may increase the number of fingers.

4.2 Caging on Multi-robot Co-operative Transporation

Multi-robot cooperative transportation is tightly coupled cooperation which aims to finish a complex transportation task with several simple robots. Classical works in this realm

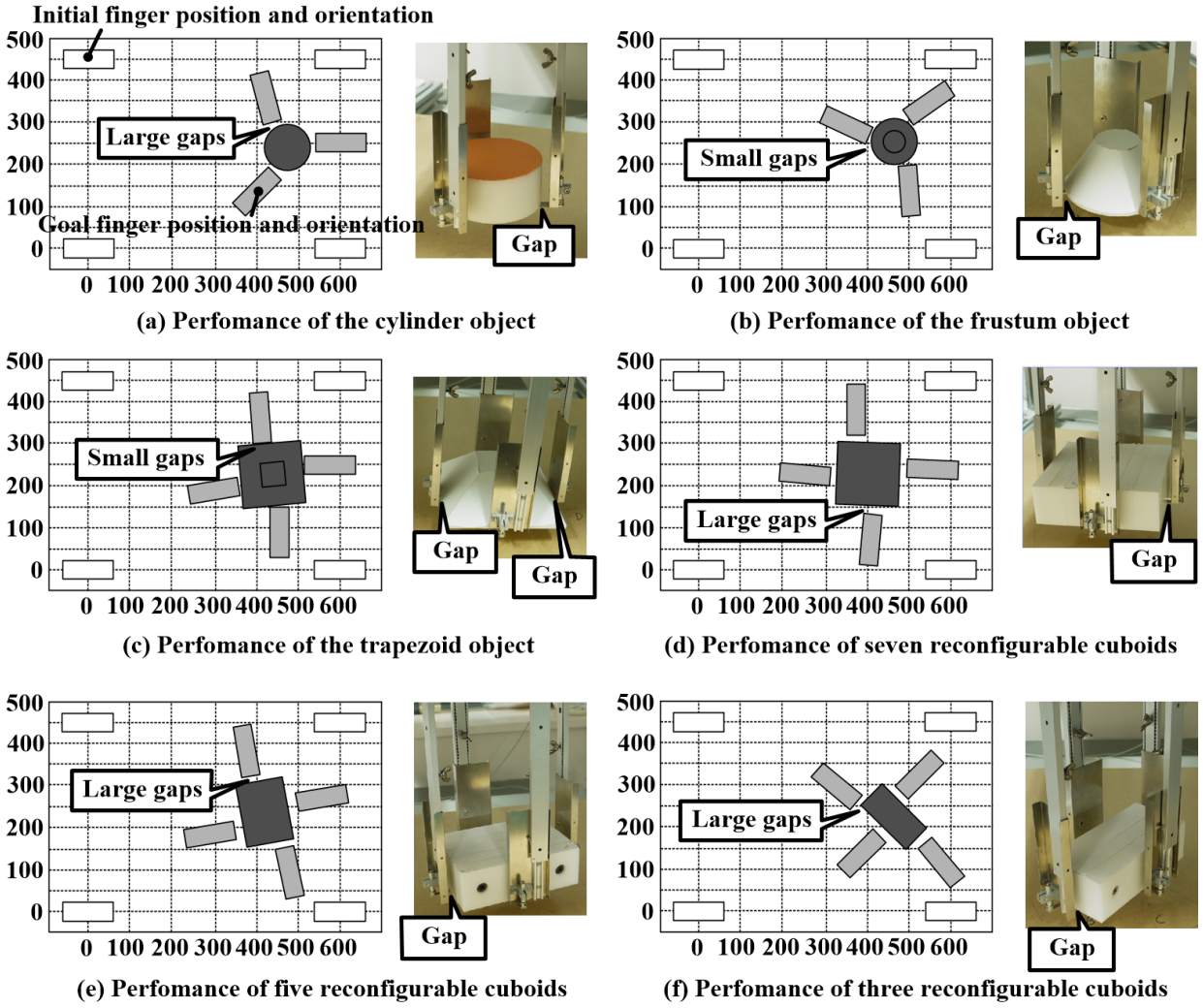


Figure 4.9: Detailed caging results of each object.

relates to task allocation, application-specific control analysis and sensor fusion. Although there are lots of publications in these aspects, I recommend readers refer to [Parker, 1998], [Gerkey and Mataric, 2002], [Montemayor and Wen, 2005] and [Cheng et al., 2008] for a rough review. They are representative works of the last decade. The delicate analysis and design, like prehensile manipulation in the realm of robotic manipulators, are usually target-specific or robot-specific. This makes it difficult to implement general and intelligent systems. Many researchers introduce caging into this field to alleviate the toughness in force and timing design. I have shown a review of the multi-robot cooperation research based on caging in Chapter 2.

Although caging-based multi-robot cooperative transportation is promising, the foregoing works are not satisfying. They either employ redundant robots to ensure successful caging or

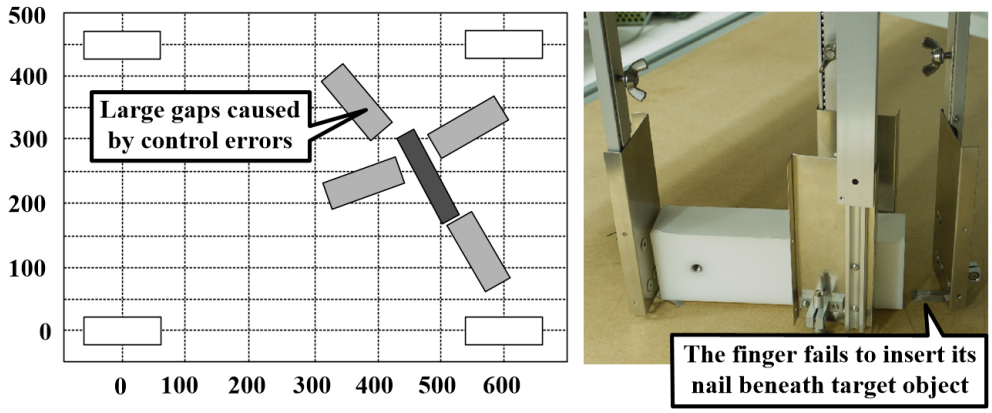


Figure 4.10: The one-cuboid object is not suitable to four-finger caging.

employ fixed number of robots due to the limitation of their algorithms. Different from those works, we can take the merits of the algorithms shown in Chapter 3 and use the least number of robots or proper number of robots as well as maintain large caging robustness. Fig.4.11(a) compares the difference between classical multi-robot cooperation and caging-based multi-robot cooperation while Fig.4.11(b) compares the advantages of my work with previous caging-based solutions. Here references [Rus, 1997] and [Sudsang et al., 2002] fall in category previous work I while references [Pereira et al., 2004], [Wang et al., 2005], [Fink et al., 2008] and [Cappelleri et al., 2011] fall in category previous work II. My work uses 3 plus x robots where x depends on the requirements on quality. In the least case, it could be 3. In most other cases, 4 would be enough. My work maintains the advantages of caging-based cooperation (see the red texts of Fig.4.11(a)) while reduce the number of robots as much as possible. Its robustness to uncertainty is between previous work I and previous work II (see the last row of Fig.4.11(b)). In summary, my work owns superiority in the following aspects. (1) Least number of robots. (2) Application independent. (3) Robust to low control quality. (4) Implicit force control. Let us see its details in following contents.

4.2.1 Simulation

Firstly, I carry out some simulations by using Webots robot simulation software. Like the caging tests with pole fingers in section 3.3.4, the Open Dynamic Engine (ODE) embedded in Webots offers us a powerful tool to test cooperative transportation with least number of robots and minimum caging.

Fig.4.12 shows the scene of our experiments. The task defined in this scene is to transport the target object cooperatively from initial position to goal region at the other side of the slope. The inclination of slope at either side is $0.2rad$. Friction coefficients of target objects are set to 0, indicating that target objects may move freely in the cage formed by robots. The free motion from 0 coefficient brings ultimate challenge and ensures exhaustive tests

	Classical cooperatoin	Caging-based cooperation
Advantages	**Small robot number** **Precise manipulation**	**Application independent** **Low requirements on control** **Implicit control**
Disadvantages	**Application dependent** **High requirements on control** **Active control**	**Large robot number** **Loose manipulation**

(a) Comparison of classical and caging-based multi-robot cooperation

	Previous work I	Previous work II	Our work
Robot number	**3**	**≥ 4**	**3+x, x depends on requirements of robustnesss**
Assumption	**Convex target objects**	**Redundant robots**	**Convex target objects**
Robustness	**$Q_1 \leq Q_3 \leq Q_2$, Q_1, Q_2 and Q_3 indicate the robustness of previous work I, II and our work respectively.**		

(b) Comparison of our caging-based multi-robot cooperation with previous work

Figure 4.11: Advantages of my work comparing with other works.

against caging. Each robot is run as an independent process so that they result into random errors like multiple robots in real world. Four different kinds of target objects are employed in our experiments. Their dimensions are shown in Fig.4.12(a), Fig.4.12(b), Fig.4.12(c) and Fig.4.12(d). The core algorithm for robust caging works with the Minkowski sum of target shape and robot dimension. The target shape is its projection on ground. In that case, the ball object suffers more from breaking. Dimensions of the four mobile robots are the same and their height is $280mm$. It is lower than boundary of ball projection ($500mm$). Therefore, ball objects suffer more from breaking. We will revisit the problems of ball object later. According to the results in Chapter 3.3.4.1, I set the filtering parameter τ into $0.3\tau_{\mathsf{max}}$ and retract robots from the surface of target object with $0.1\tau_{\mathsf{max}}$.

Fig.4.13 shows my strategy in formation control. It is a leader-follower solution. In each *formation control step*, a leader robot is chosen. The follower robots locomote following the leader as well as maintain the whole robot formation, namely maintain the relative distances between robots. Note that this implementation may not guarantee maximum safety since we do not want to incorporate explicit specification of motion orders, directions and leaders. However, each robot could move with certain offset from its formation owing to the robustness of optimized caging. Motion of each robot is decided by decentralized planners. I use potential field planner to calculate the dragging and repelling force between current positions

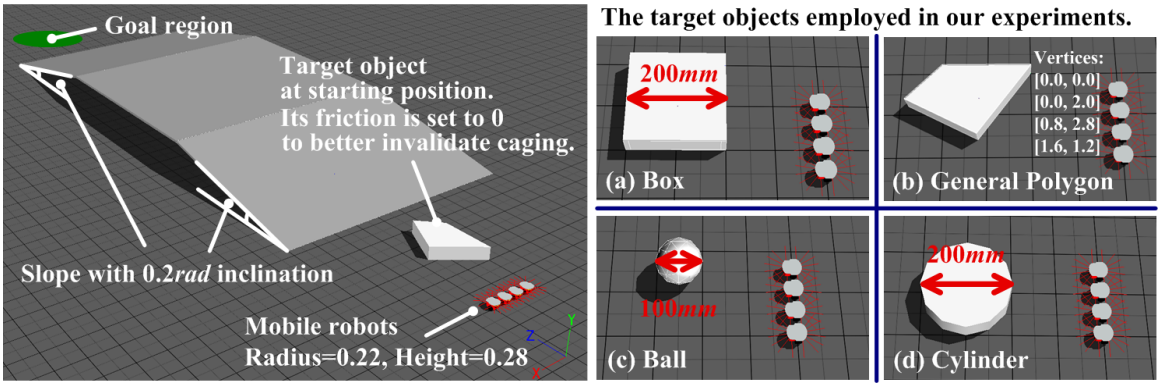


Figure 4.12: Simulation scene and the target objects.

and goal positions. Readers may refer to Chapter 4 of reference [Choset et al., 2005] for more details o potential field methods and how to conquer problems caused by local minima. In this procedure, each robot could be driven along any direction. Surely, if the motion of robots at one time step are too drastic, jam may appear. However, I expect that robust caging could avoid jam and endure certain formation deviation appeared in the locomotion of each single robot since it offers satisfying robustness to caging breaking and consequently enough tolerance to control errors. In real applications, of course practitioners could attain better performance by defining motion orders, locomotion directions and specific leader.

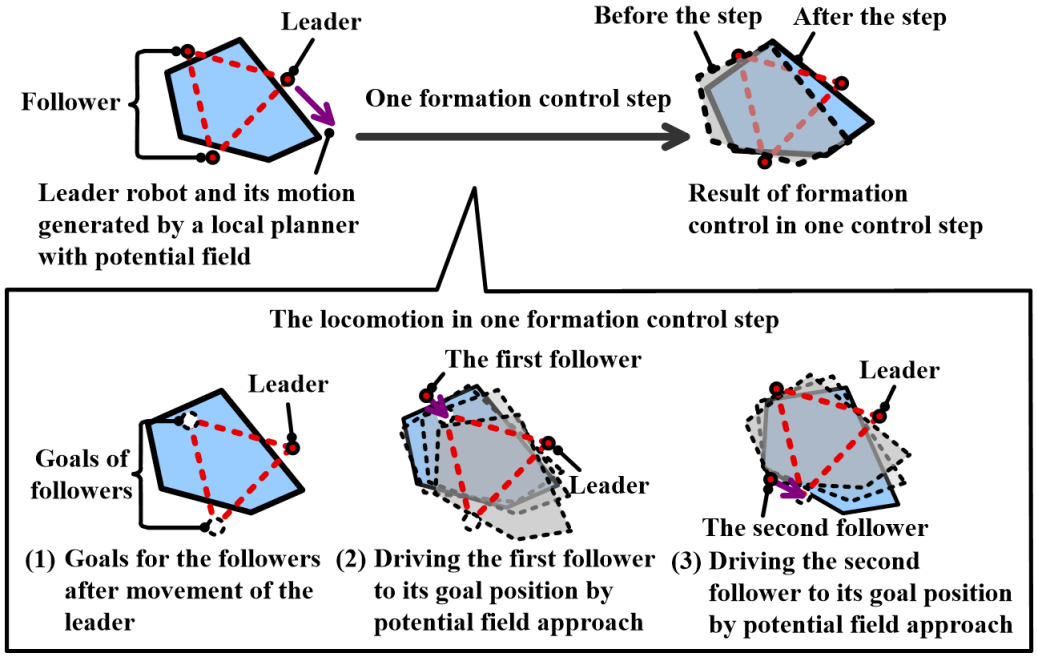


Figure 4.13: One control step of the leader-follower control strategy.

4.2.1.1 Transportation by formations control

Fig.4.14 and Fig.4.15 respectively shows the tolerance of optimized caging to errors of formation control in manipulating the object in Fig.4.12(a) and the object in Fig.4.12(b). The same as our expectation, at least four robots are needed for the object in Fig.4.12(a) while at least three robots are required for the object in Fig.4.12(b). The left parts of Fig.4.14 and Fig.4.15 show the formation formed by robots in transportation. The other parts show the variation in inter-robot distance during transportation. Initial inter-robot distance is emphasized with blue color in these parts. The data in Fig.4.14 and Fig.4.15 are taken from the first 70 *formation control steps* (see the horizontal axis). Although the distance varies dramatically (see the vertical axis), our optimized caging transportation can offer great tolerance and perform robustly. The length of a single *formation control step* is set to 64*milisecond*, namely the formation control strategy shown in Fig.4.14 and Fig.4.15 are performed every 64*milisecond*.

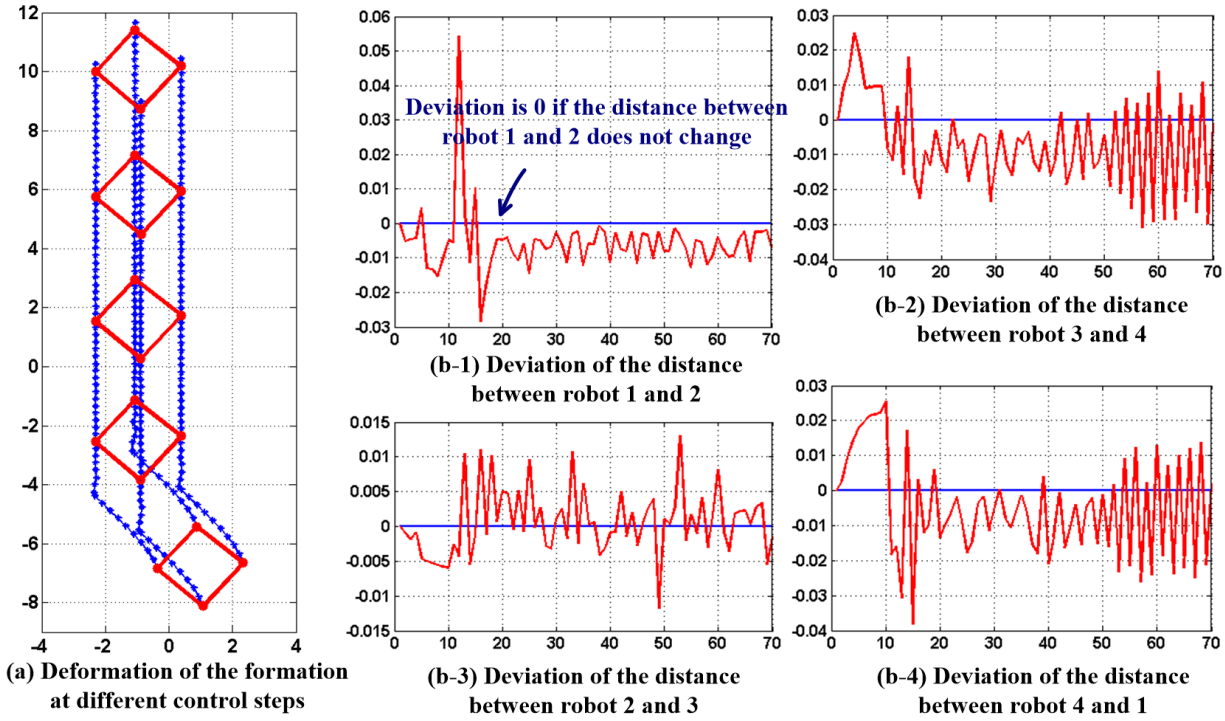


Figure 4.14: Simulation results of the object in Fig.4.12(a).

When the length of a formation control step changes, the uncertainty changes accordingly. A longer control step usually results into larger uncertainty. Fig.4.16 demonstrates the changes of uncertainty in formation control with different time steps. Like our intuition,

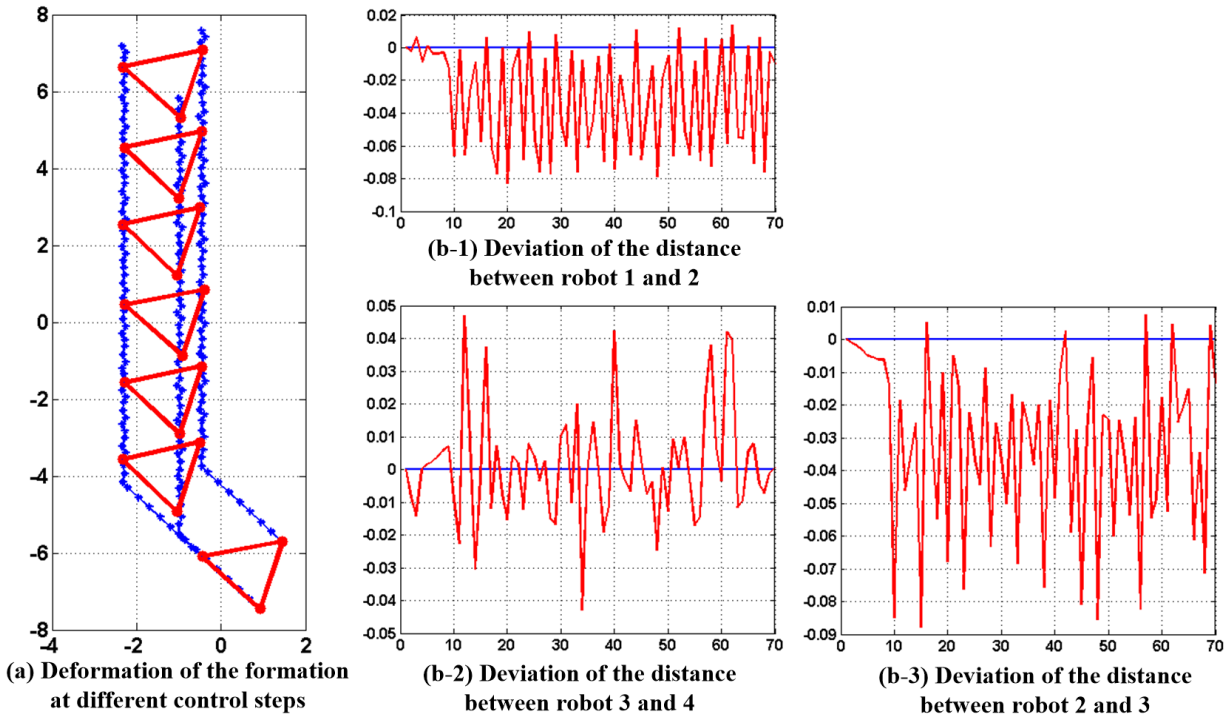


Figure 4.15: Simulation results of the object in Fig.4.12(b).

as the length of one time step increases, the uncertainty or the deviation errors increase accordingly. Although larger deviation errors appear as the length of time step increases, our robust caging algorithm can endure them. With the objects in Fig.4.12(a) and (b), our caging algorithm can be robust to all the three control steps of Fig.4.16. The optimized caging formation from my algorithm can reduce robot number as much as possible as well as offer satisfying robustness.

The ball object, namely the object in Fig.4.12(a), is a special case as its size is relatively small comparing with robots and different from its projection on the ground. In our experiments, only formation control at every $64 miliseconds$ could guarantee successful transportation. Actually, circular objects with small dimension could be most challenging to formation control and could be vulnerable to caging breaking. After changing the sphere into a cylinder with larger size (radius = $1000mm$, see object (d)), the robots can perform successful transportation at not only $64 milisecond$ but also $128 milisecond$ intervals. However, the ball object cannot be successfully caged and transported when control step becomes as long as $256 milisecond$.

Fig.4.17 shows some frames extracted from the video clip that records the 3-robot caging transportation of object (b). Specifically, in Fig.4.17(a), (b) and (c) the robots are driven to

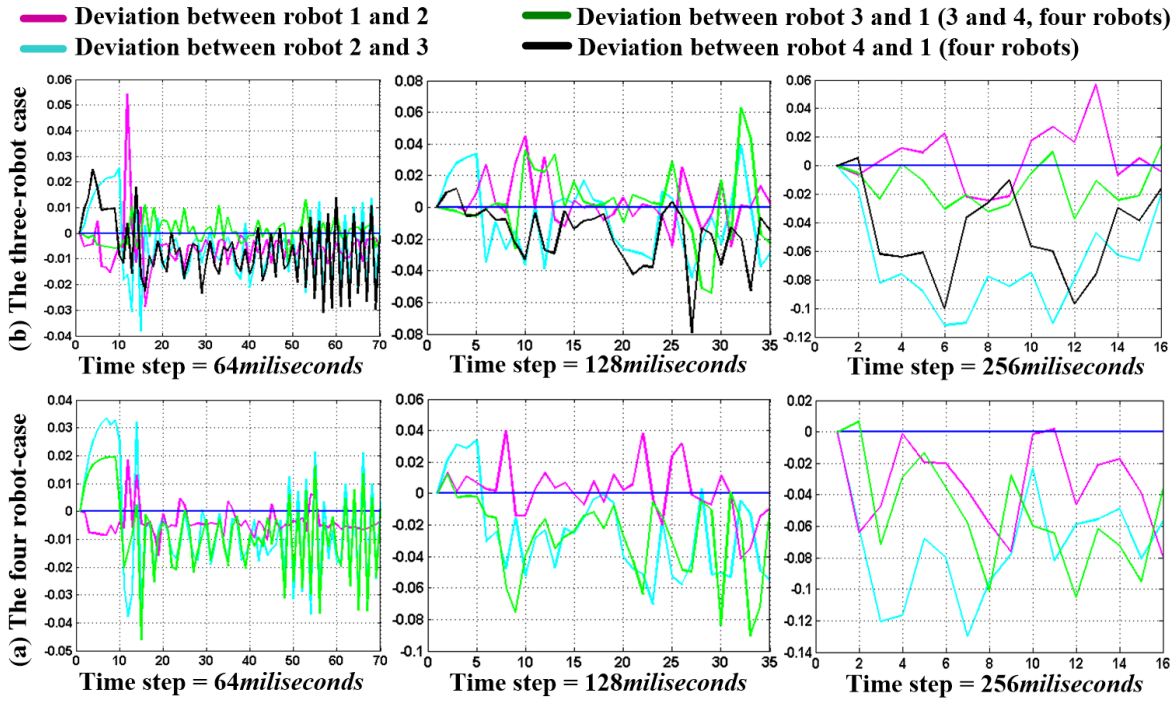


Figure 4.16: Changes of uncertainty in formation control with different time steps.

the optimized caging positions while in Fig.4.17(d), (e) and (f) the formation formed by the robots cooperatively transport target object from initial position to goal position. Readers may refer to the video clips related to this thesis (http://goo.gl/3ddrNn) for more details.

4.2.1.2 Choosing proper robot number

Besides the robustness in uncertain formation control, our algorithm could suggest proper number of robots to obtain larger tolerance to formation control errors. As has been discussed in the end of section 3.3.4.1, the fourth question, namely how to decide the number of fingers, can be decided by the maximum retraction distance or a user-defined threshold. Consequently, we can add redundant robots when the robustness is not satisfying. After adding an extra mobile robot, both transportation of object (c), the ball, and object(d), the cylinder, may endure formation control uncertainty caused by 256*miliseconds* formation control step. Fig.4.18 demonstrates the advantages of introducing an extra mobile robot by comparing the results of 3-robot and 4-robot caging of object (c). Control uncertainty caused by 256*miliseconds* formation control step breaks 3-robot caging. On the contrary, 4-robot caging can be robust all the time.

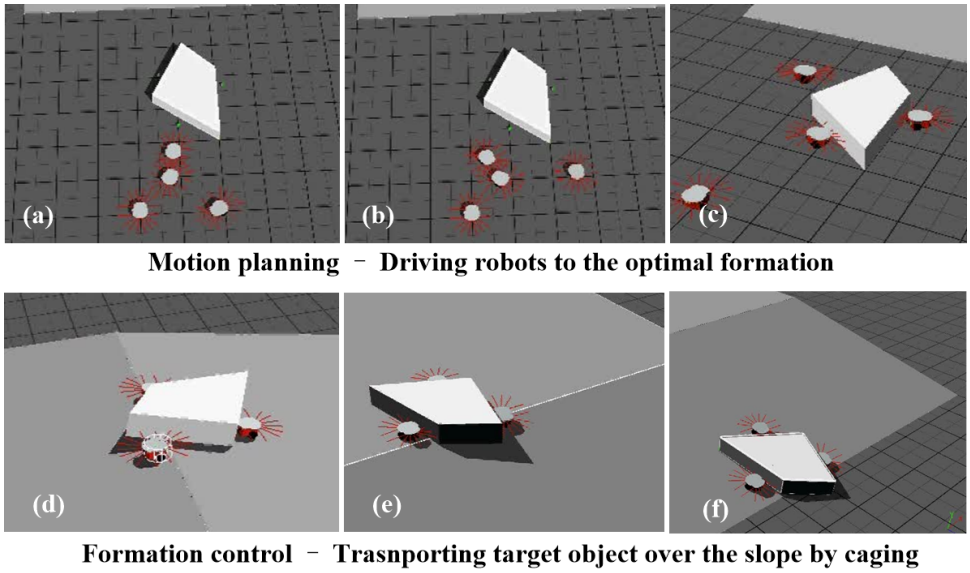


Figure 4.17: Frames extracted from the caging and transportation video.

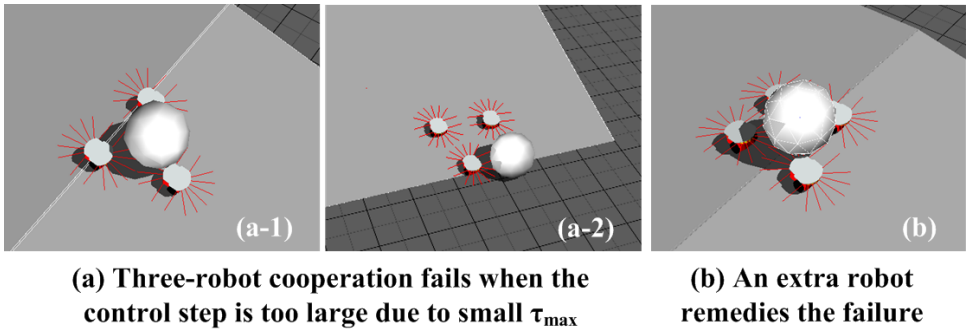


Figure 4.18: Choosing the least or proper number of robots by considering requirements of robustness.

4.2.2 Implementation with real robots

Besides the simulation, we further implement the algorithm with real mobile robots. The mobile robots are hobby commercial products named Beauto Rover. They are produced by Vstone Co. Ltd [Vstone, 2013]. We use the V100:R2 OptiTrack as perception devices and track the orientation and position of both target object and mobile robots. Moreover, in order to better evaluate the performance of the implementation, we install three caster wheels at the bottom of the target object to reduce frictions and enable free motions. Fig.4.19 shows the target objects, the mobile robots, the installation of OptiTrack markers and the workspace scene.

Like the scene in simulation, we build up a slope in the middle to induce ultimate challenge

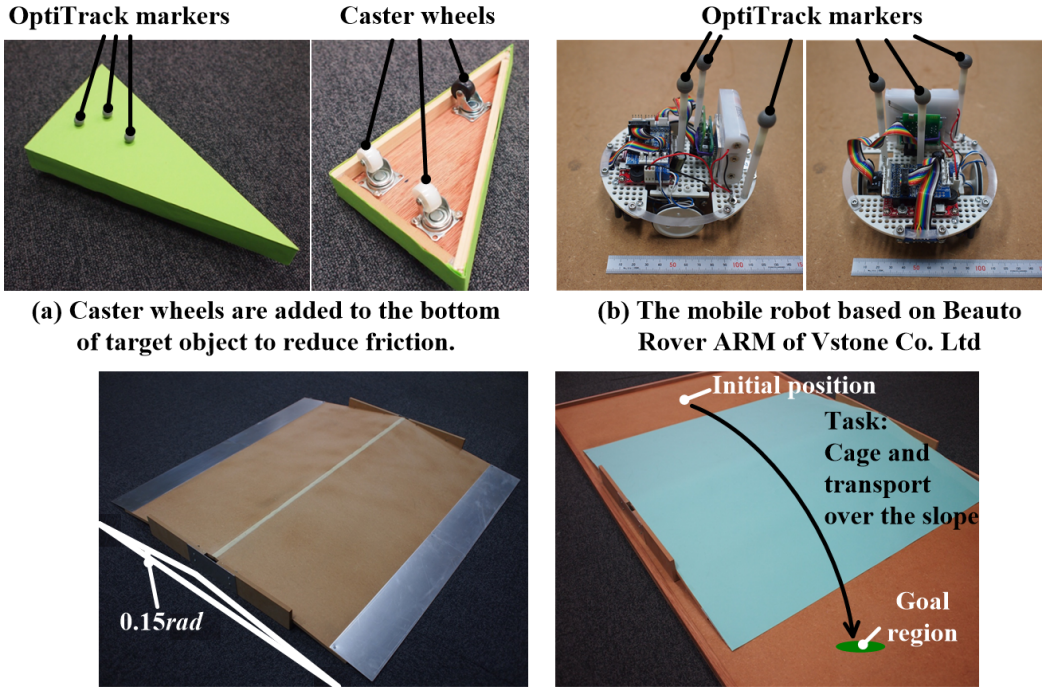


Figure 4.19: The real-world target objects, mobile robots and workspace scene.

to caging. Fig.4.19(c) illustrates this slope. Note that we did not use exactly the same $0.2rad$ inclination but alternatively use $0.15rad$ due to the limitation of Beauto Rover. Again, like the scene in simulation, the aim of this implementation is to transport target objects from an initial position to a goal region beyond the slope.

4.2.2.1 Software integration

In the lower level, each mobile robot is controlled by an integrated ARM board installed on its body. In the higher level, a control laptop computer communicates with those robots and control robot formation by using OptiTrack. The control computer controls those mobile robots through blue-tooth communication. Each robot has its own encoder installed on motors for feedback control. Fig.4.20 shows the whole procedure of one *formation control step*. This exactly corresponds to the *formation control step* discussed in the simulation section.

Fig.4.21 shows the interface that performs the co-operative transportation task. It integrates the following functions. (1) Caging optimization, namely finding an optimized formation of fingers that can cage the target object. This is done by the button enclosed by orange frame in Fig.4.21 (2) Motion planning, namely actuating the mobile robots into their goal

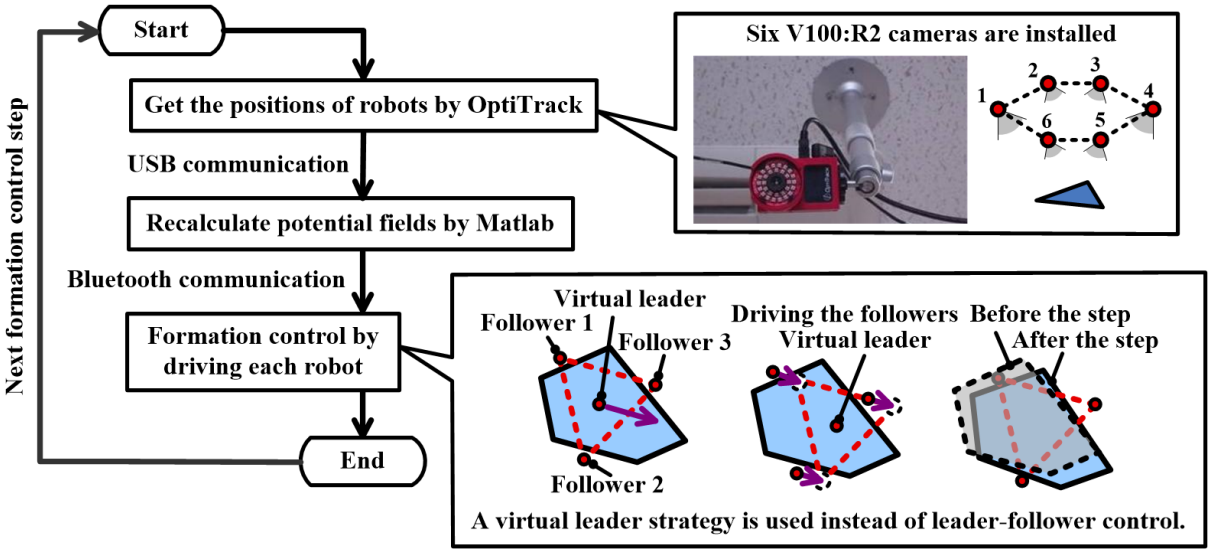


Figure 4.20: One formation control step of the real-world implementation.

positions without colliding with each other. This is done by the button enclosed by green frame in Fig.4.21. Users can also set parameters like the length of formation control step by using those input boxes. Specifically, we use time-based cell decomposition for motion planning of multiple robots. There are many books that introduces the cell decomposition algorithm, readers may refer to Chapter 6 of reference [Choset et al., 2005] or Chapter 6 of reference [LaVelle, 2006] for more information. Like the implementation in simulation, we use potential field approach to calculate the dragging and repelling force of each robot and locomote them.

4.2.2.2 Formation control

In the simulation, the formation of robots is controlled by a leader-follower strategy shown in Fig.4.13. This strategy, however, results into large uncertainty in real world. That is because the strategy essentially ensures the precision of the leader robot while gives up the precision of the follower robot. It causes large uncertainty to the precision of the followers. In simulation, the uncertainty is not evidential since many factors are neglected. However, when dealing with real robots, there is quite large deviation and the leader-follower formation control fail to guarantee caging.

Consequently, we choose another formation control strategy. The formation control strategy is named virtual leader and it is illustrated in the lower frame box of Fig.4.20. In contrast of the leader-formation control strategy, virtual leader control select the center of robot formation as the virtual leader. At each formation control step, it aims at ensuring the precision of a virtual leader rather than a certain robot. In that case, control uncertainty is distributed into each robot and a single robot would not suffer from large errors.

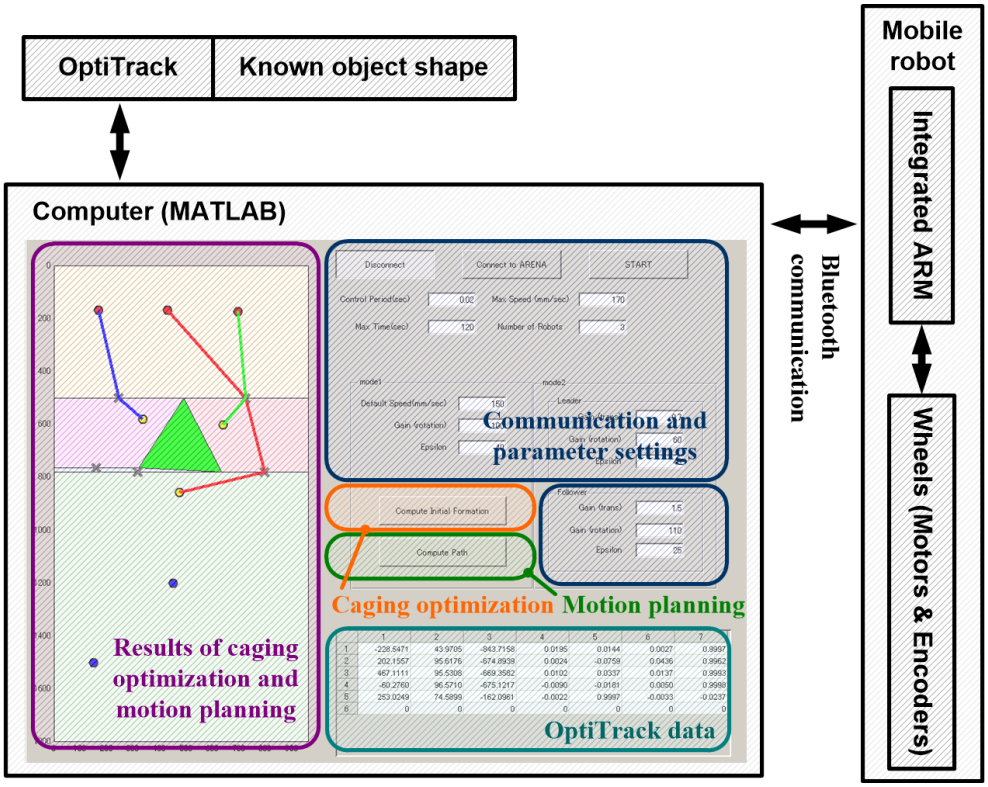


Figure 4.21: Connections between the high-level computer and the low-level mobile robot.

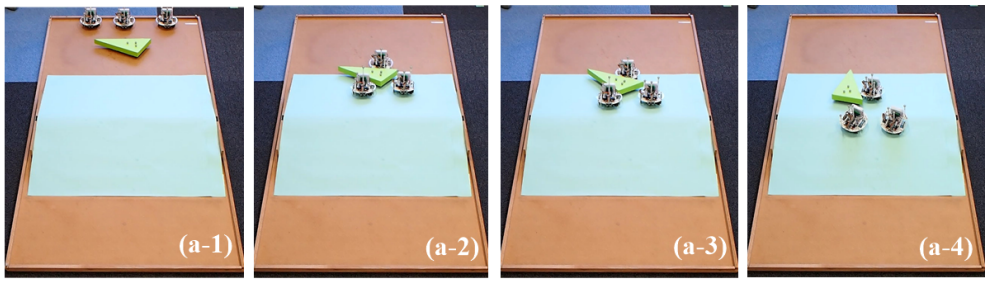
Fig.4.22 shows the errors caused by control uncertainty with the two different formation control strategies. The deviation of leader-follower formation control strategy could be larger than $50mm$ while the deviation of virtual leader formation control is always smaller than $20mm$. The virtual leader strategy is a better strategy and we use it in the real-world implementation. Optimized caging with three mobile robots is robust enough to endure the uncertainty caused by virtual leader formation control while cannot endure the uncertainty caused by leader-follower formation control.

	Standard variation (*mm*)	**Maximum (*mm*)**
Leader-follower strategy	**14.67**	**64.54**
Virtual leader strategy	**7.75**	**17.21**

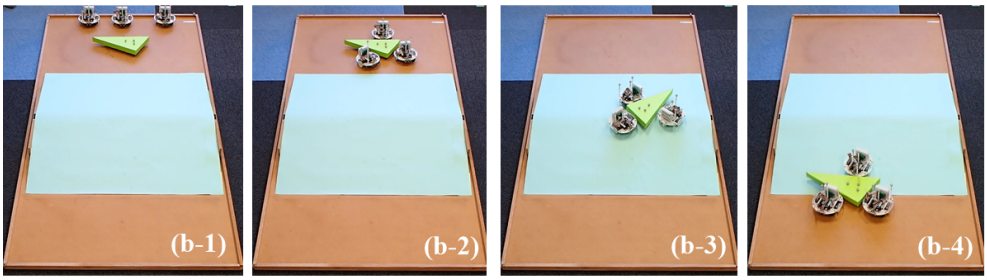
Figure 4.22: The errors caused by different strategies.

Fig.4.23 shows the extracted frames from a video clip related to this thesis (http://goo.gl/3ddrNn). In Fig.4.23(a), we use leader-follower formation control strategy. In this case, the robots fail to cage the target object when crossing over the slope. In Fig.4.23(b), we use the virtual

leader formation control strategy. In this case, the robots can always cage target object and perform successful caging-based transportation. We carried out 5 times of experiments with the virtual follower strategy and all of them succeeded with the virtual leader formation control strategy.



(a) The optimal caging cannot endure uncertainty caused by leader-follower control.



(b) The optimal caging can endure uncertainty caused by virtual leader control.

Figure 4.23: Results of leader-follower control and virtual leader control.

4.2.2.3 Robustness to perception uncertainty

We have repeated a lot in foregoing texts that caging optimization is promising in dealing with two kinds of uncertainty – the perception uncertainty and the control uncertainty. The uncertainty caused by formation control is control uncertainty. The uncertainty caused by OptiTrack is perception uncertainty. Actually, the uncertainty from perception is quite small since the precision of V100:R2 OptiTrack is on average smaller than $1mm$. Therefore, we manually add noises to vertices of the target object to test its robustness to both uncertainty.

Fig.4.24 shows the noisy objects and their successful rates in our experiments. The noises are manually added to vertices by dragging them $23.55mm$ away from the geometric center of the target object. $23.55mm$ is three times of the standard variation of virtual leader strategy shown in Fig.4.22. We generate seven different noisy objects and perform five transportation experiments for each of them. During the experiments, the algorithm calculates an optimized caging formation by using the noisy object and performs transportation by using the ground-truth object. The same as our expectation, when the aspect ratio of noisy object becomes smaller like objects Fig.4.24(a) and Fig.4.24(b), the successful rates decrease

accordingly. In all, we fail to get 100% successful rate in the presence of both control and perception uncertainty. I think this is because the dimension of target object is relatively small comparing with the dimension of robots and consequently the robustness of caging is low. We may need extra robots to obtain 100% successful rate in the presence of both uncertainty.

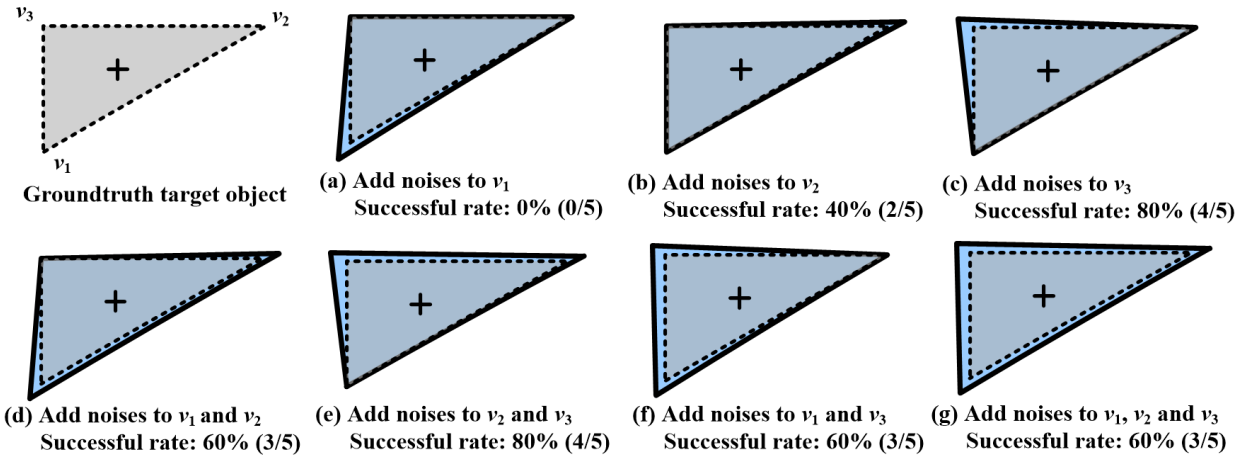


Figure 4.24: The noisy objects and their successful transportation rates.

Part III

Caging in $\mathcal{C}^{\mathsf{frm}}$ and Its Applications

Chapter 5

Caging in The Configuration Space of Fingers

5.1 Changing From $\mathcal{C}^{\mathsf{obj}}$ to $\mathcal{C}^{\mathsf{frm}}$

The simulations and real-world applications in second part of this thesis shows that performance of the faster robust caging algorithm in $\mathcal{C}^{\mathsf{obj}}$ is satisfying. However, it suffers from a fatal drawback. Say, it can only be applicable to convex objects. How can we extend it to many other objects? In order to solve this problem, we need to recall how our faster robust caging algorithm in $\mathcal{C}^{\mathsf{obj}}$ became limited to convex objects. The answer involves many fundamental techniques of the faster robust caging algorithm, for instance, $\mathcal{A}_{\mathsf{c}}$, immobilization optimization and translational caging. That means if we would like to extend the algorithm to various other objects, the whole algorithm should be redesigned. We should no longer employ immobilization optimization and no longer employ translational caging. That is difficult in $\mathcal{C}^{\mathsf{obj}}$ and therefore in the third part of this thesis, I re-consider the algorithm in **the configuration space of finger formation**. This **configuration space of finger formation** will be denoted by $\mathcal{C}^{\mathsf{frm}}$. I will show the details of it and the new algorithm in following contexts.

5.1.1 Consider a different center

I introduced $\mathcal{C}^{\mathsf{obj}}$ in section 2.1.2 when discussing about immobilization. In $\mathcal{C}^{\mathsf{obj}}$, the target object becomes a 3D point while the fingers becomes 3D obstacles. The caging test in $\mathcal{C}^{\mathsf{obj}}$ is actually testing whether a point is enclosed by obstacles. During this procedure, the target object is **the center of planning** and the configuration space encodes the position and orientation of the target object.

We can consider the **center** in a reverse way. Recall that there are two conditions for a caging test problem. I introduced it in the beginning of section 3.1. One condition is the center of $\mathcal{C}^{\mathsf{obj}}$, namely the target object. The other condition is the positions of fingers. Can

we take the position of fingers as the center and build a space that encodes these positions? The answer is positive. Fig.5.1 illustrates this reverse consideration.

When target object is the center, caging test means to see if the target object can escape from the formation of fingers. Fig.5.1(a) illustrates this idea. The correspondent configuration space $\mathcal{C}^{\mathsf{obj}}$ is a three-dimensional $\mathbb{R}^2 \times \mathcal{S}$ space. When the positions of fingers are the center, caging test means to see if the formation of fingers can escape, or go through, the target object. Fig.5.1(b) illustrates this idea. The correspondent configuration space is a $2n_f$-dimensional $\mathbb{R}^2 \times \mathbb{R}^2 \times \ldots \times \mathbb{R}^2 = \mathbb{R}^{2n_f}$ space where n_f is number of fingers. I name this space $\mathcal{C}^{\mathsf{fgr}}$ to emphasize its difference from $\mathcal{C}^{\mathsf{frm}}$.

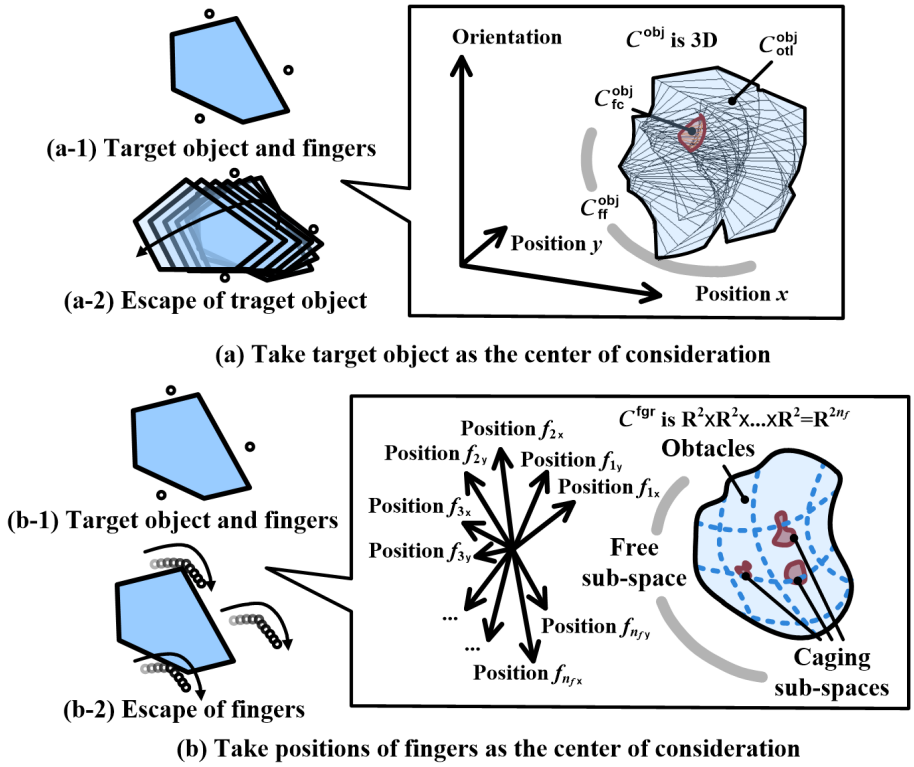


Figure 5.1: Comparison of different centers.

The bad news of taking positions of fingers as the center of consideration is it suffers from curse of dimensionality. As we can see from Fig.5.1, the dimension could be as high as $\mathbb{R}^{2n_f}$ with $2n_f$ fingers. This makes it difficult to model the obstacles in this space. Works like Rodriguez [Rodriguez et al., 2011] and Pippattanasomporn [Pipattanasomporn and Sudsang, 2011] are based on exploration of this space. However, they either take the space as a topological space which cannot be modeled or reduce the number of fingers n_f into 2 for easier analysis. That is not satisfying. There are two strategies to deal with the high dimensionality of $\mathcal{C}^{\mathsf{fgr}}$. One is to consider some techniques in other research fields that can work in high-dimension spaces. Probabilistic approaches in the research field of robotic motion planning

provides such kinds of techniques. These techniques involve but are not limited to Probabilistic Roadmap Method (PRM) [Kavraki et al., 1996] and Rapidly-Exploring Random Trees (RRT) [Lavalle and Kuffner, 2000]. Unfortunately, these techniques only guarantee weak probabilistic completeness ([Choset et al., 2005], pp.242 -246). Caging requires an object to be completely constrained by fingers. It should be depend on a certain probability. Therefore, this is not the best strategy. Another strategy is to reduce the dimension of $\mathbb{R}^{2n_f}$. I will take the second strategy and show the details in the next subsection.

5.1.2 The configuration space of finger formation

Let us review caging in Fig.5.1(b). When positions of fingers are the center of consideration, caging means the formation formed by the fingers cannot escape from target objects. That is to say, we do not need to consider the positions of every fingers. The positions of the formation and its orientation is enough for caging test. This conclusion reduces $\mathbb{R}^{2n_f}$ into a three-dimensional space. This space is different from both $\mathcal{C}^{\mathsf{obj}}$ and $\mathcal{C}^{\mathsf{fgr}}$. I name it $\mathcal{C}^{\mathsf{frm}}$ to indicate the difference. Fig.5.2 illustrates this $\mathcal{C}^{\mathsf{frm}}$ space. The planar two dimension of $\mathcal{C}^{\mathsf{frm}}$ denote the position of f_1 and the third dimension of $\mathcal{C}^{\mathsf{frm}}$ denotes the orientation of the whole formation. Consequently, a point in $\mathcal{C}^{\mathsf{frm}}$ represents the position and orientation of a fixed finger formation in work space.

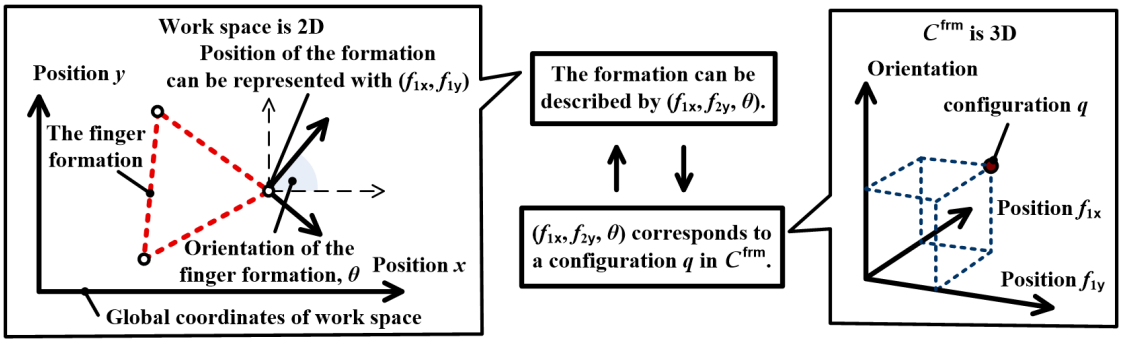
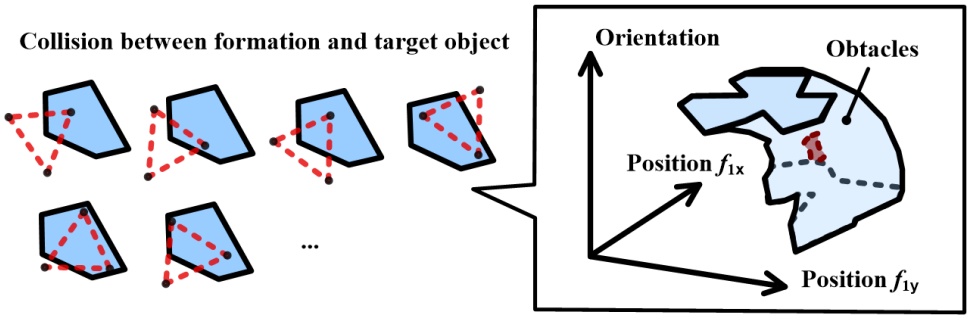


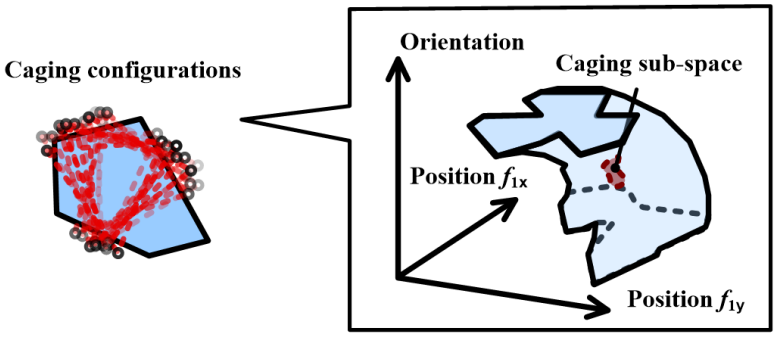
Figure 5.2: The configuration space of finger formation.

Unlike $\mathcal{C}^{\mathsf{obj}}$, the obstacle in $\mathcal{C}^{\mathsf{frm}}$ has no correspondence in $\mathcal{W}$ space. Take Fig.5.3(a) for instance. The obstacle in $\mathcal{C}^{\mathsf{frm}}$ is a compact set of configurations at which the formation collide with target objects. It does not correspond to a single finger like $\mathcal{F}_i$ but correspond to a whole formation. Caging means that (1) the formation does not collide with the target object and (2) the formation cannot go to an infinite configuration without colliding with the target object. In the $\mathcal{C}^{\mathsf{frm}}$, caging correspond to some caging sub-spaces. If a formation is at a configuration inside these caging sub-spaces, it fulfills the two conditions since (1) the caging sub-space is free and it is not obstructed and (2) current configuration cannot be connected to an infinite configuration unless it collides with the obstacles. Fig.5.3(b) illustrates a caging status and an caging sub-space. The caging sub-space in $\mathcal{C}^{\mathsf{frm}}$ is like the

caging sub-space in $\mathcal{C}^{\text{obj}}$. They do have certain relations. I will show their relations later after finishing solving the caging problems.



(a) The obstacles in C^{frm} denote a compact set of configuration points at which the finger formation collides with the target object



(b) All caging configurations belong to one or two compact regions in C^{frm}

Figure 5.3: Caging in the configuration space of finger formation.

$\mathcal{C}^{\text{frm}}$ has low dimensionality which makes it possible to be modeled. However, since the $\mathcal{C}^{\text{frm}}$ only encodes position and orientation of the whole formation, one modeled $\mathcal{C}^{\text{frm}}$ only corresponds to one finger formation. If the relative positions of fingers in a formation changes, the $\mathcal{C}^{\text{frm}}$ becomes totally different and we have remodel those caging sub-spaces and obstacles following Fig.5.3. That is a boring task. Moreover, when the target object changes, the obstacles should also be remodeled because checking whether formation and target objects collides depends on both finger formation and the shape of target objects. How to decouple the connection between $\mathcal{C}^{\text{frm}}$ and finger formation and decouple the connection between $\mathcal{C}^{\text{frm}}$ and obstacles become the bottleneck. I will start discussing these details from section 5.2. Before that, let us define the symbols that will be used in $\mathcal{C}^{\text{frm}}$ and compare $\mathcal{C}^{\text{obj}}$ and $\mathcal{C}^{\text{frm}}$.

The symbols that will be used in $\mathcal{C}^{\text{frm}}$ are as following.

$\mathcal{W}$	Work space. This is the same as $\mathcal{C}^{\text{obj}}$.
$\boldsymbol{\omega_i}$	A discretized grid of $\mathcal{W}$ space. Since $\mathcal{W}$ space is 2D, this grid can be represented by a coordinate $(\omega_{i_x}, \omega_{i_y})$ when granularity of discretization is small enough.

O The target object in $\mathcal{W}$ space. This is the same as $\mathcal{C}^{\mathsf{obj}}$.

$\boldsymbol{f_i}$ A point finger in $\mathcal{W}$ space. This is the same as $\mathcal{C}^{\mathsf{obj}}$. The $\boldsymbol{f_i}$ has two coordinate elements $\{f_{i_\mathsf{x}}, f_{i_\mathsf{y}}\}$. Mathematically, $\boldsymbol{f_i} = \boldsymbol{\omega_i}$ when granularity of discretization is small enough. They both denote a position in 2D plane.

F A formation of fingers in $\mathcal{W}$ space. It is a set of $\mathcal{W}$ space point finger $\mathsf{F} = \{f_i = \{f_{i_\mathsf{x}}, f_{i_\mathsf{y}}\} | i = 1, 2, \ldots, n_f\}$.

$\mathcal{C}^{\mathsf{frm}}$ The configuration space of finger formation.

$\boldsymbol{q}^{\mathbf{frm}}$ $\boldsymbol{q}^{\mathbf{frm}}$ is a configuration in $\mathcal{C}_{\mathsf{frm}}$. Since we have reduced the high dimensional $\mathcal{C}^{\mathsf{fgr}}$ into three-dimensional $\mathcal{C}^{\mathsf{frm}}$, $\boldsymbol{q}^{\mathbf{frm}} = \{q_\mathsf{x}^{\mathsf{frm}}, q_\mathsf{y}^{\mathsf{frm}}, q_\theta^{\mathsf{frm}}\}$ where the first two items denote the position of $\boldsymbol{f_1}$, namely $q_\mathsf{x}^{\mathsf{frm}} = f_{1_\mathsf{x}}$ and $q_\mathsf{y}^{\mathsf{frm}} = f_{1_\mathsf{y}}$. and the last item denotes the orientation of the formation.

$\mathsf{F}[\boldsymbol{q}^{\mathbf{frm}}]$ $\mathsf{F}[\boldsymbol{q}^{\mathbf{frm}}]$ denotes a finger formation F at configuration $\boldsymbol{q}^{\mathbf{frm}}$. Mathematically, it represents a set of 2D positions in $\mathcal{W}$ space occupied by F. For example, when $\mathsf{F} = \{\boldsymbol{f_i} = \{f_{i_\mathsf{x}}, f_{i_\mathsf{y}}\} | i = 1, 2, \ldots, n_f\}$, $\mathsf{F}[\boldsymbol{q}^{\mathbf{frm}}] = \{R^{(q_\theta^{\mathsf{frm}})} \cdot \{f_{j_\mathsf{x}} - f_{1_\mathsf{x}} + q_\mathsf{x}^{\mathsf{frm}}, f_{j_\mathsf{y}} - f_{1_\mathsf{y}} + q_\mathsf{y}^{\mathsf{frm}}, 1\} | 1 < j \leq n_f\}$. Note that this expression should be performed in with homogeneous coordinates with an augmented "1" at the end of $\{f_{j_\mathsf{x}} - f_{1_\mathsf{x}} + q_\mathsf{x}^{\mathsf{frm}}, f_{j_\mathsf{y}} - f_{1_\mathsf{y}} + q_\mathsf{y}^{\mathsf{frm}}\}$, namely $\{f_{j_\mathsf{x}} - f_{1_\mathsf{x}} + q_\mathsf{x}^{\mathsf{frm}}, f_{j_\mathsf{y}} - f_{1_\mathsf{y}} + q_\mathsf{y}^{\mathsf{frm}}, 1\}$ The rotation matrix $R^{(\theta)}$ follows the same definition as section 3.1.

$\mathcal{C}_{\mathsf{otl}}^{\mathsf{frm}}$ The obstacles in $\mathcal{C}^{\mathsf{frm}}$. Mathematically, it is some sub-spaces/compact sets of configurations $\boldsymbol{q}^{\mathbf{frm}}$ where $\mathsf{F}[\boldsymbol{q}^{\mathbf{frm}}] \cap \mathsf{O} \neq \emptyset$.

$\mathcal{C}_{\mathsf{free}}^{\mathsf{frm}}$ All free sub-spaces in $\mathcal{C}^{\mathsf{frm}}$. Mathematically, it is the complementary space of $\mathcal{C}_{\mathsf{otl}}^{\mathsf{frm}}$. Namely $\mathcal{C}_{\mathsf{free}}^{\mathsf{frm}} = \mathcal{C}^{\mathsf{frm}} \setminus \mathcal{C}_{\mathsf{otl}}^{\mathsf{frm}}$.

Readers may compare these notations with those defined in section 3.1 for comparison of $\mathcal{C}^{\mathsf{frm}}$ and $\mathcal{C}^{\mathsf{obj}}$. Recall that caging in $\mathcal{C}^{\mathsf{frm}}$ means two conditions and we can now express the two conditions formally with the defined notations alike the expression in $\mathcal{C}^{\mathsf{obj}}$. When caging is achieved, $\mathcal{C}_{\mathsf{free}}^{\mathsf{frm}}$ is divided into several disconnected sub-spaces. Most of the sub-spaces are enclosed by obstacles. The caging sub-space in Fig.5.3 is one enclosed example. However, it is a special case since it is the only caging sub-space. Generally, there would be always more than one caging sub-space. I denote these caging sub-spaces by $\mathcal{C}_{\mathsf{fc}}^{\mathsf{frm}} = \bigcup_{i=1}^{u} \mathcal{C}_{\mathsf{fc_i}}^{\mathsf{frm}}$ in accordance with the caging sub-spaces in $\mathcal{C}^{\mathsf{obj}}$. Note that $\mathcal{C}^{\mathsf{frm}}$ is inherently different from $\mathcal{C}^{\mathsf{obj}}$. object is convex, u may be still larger than 1. One other disconnected sub-spaces is the complementary of $\mathcal{C}_{\mathsf{fc}}^{\mathsf{frm}}$ which can be denoted by $\mathcal{C}_{\mathsf{ff}}^{\mathsf{frm}}$. In summary, $\mathcal{C}_{\mathsf{free}}^{\mathsf{frm}} = \mathcal{C}_{\mathsf{fc}}^{\mathsf{frm}} \cup \mathcal{C}_{\mathsf{ff}}^{\mathsf{frm}} = (\bigcup_{i=1}^{u} \mathcal{C}_{\mathsf{fc_i}}^{\mathsf{frm}}) \cup \mathcal{C}_{\mathsf{ff}}^{\mathsf{frm}}$, $\mathcal{C}_{\mathsf{fc}}^{\mathsf{frm}} \cap \mathcal{C}_{\mathsf{ff}}^{\mathsf{frm}} = \emptyset$. Whether O can be caged by a finger formation F can be validated by the following expression.

$$(\mathcal{C}_{\mathsf{free}}^{\mathsf{frm}} = (\bigcup_{i=1}^{u} \mathcal{C}_{\mathsf{fc_i}}^{\mathsf{frm}}) \cup \mathcal{C}_{\mathsf{ff}}^{\mathsf{frm}}) \wedge (\boldsymbol{q_0}^{\mathbf{frm}} \in \mathcal{C}_{\mathsf{fc_k}}^{\mathsf{frm}}) \wedge (|\mathcal{C}_{\mathsf{fc_k}}^{\mathsf{frm}}| > 1), 1 \leq k \leq u \tag{5.1}$$

This expression is nearly the same as expression (3.1). $|\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc_k}}|$ means the cardinality of $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc_k}}$. When $|\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc_k}}| > 1$, the target object is either in the state of caging or in the state of contact caging. When $|\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc_k}}| = 1$, the target object is in the state of immobilization. Note that $\boldsymbol{q}^{\mathsf{frm}}_{\boldsymbol{0}}$ in expression (5.1) is the initial configuration of F. It is not the initial configuration of O.

Everything seems to be following the same rule as $\mathcal{C}^{\mathsf{obj}}$. Unfortunately, we encounter the bottleneck when trying to model and calculate $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc}}$. Recall Fig.3.1 and Fig.3.2. We can model $\mathcal{F}_i$ in $\mathcal{C}^{\mathsf{obj}}$ with wireframe modeling since there are correspondence between each layer of $\mathcal{F}_i$ and orientation of the target object. We only need to consider one position, name the position $\boldsymbol{f_i}$. In contrast, there is no correspondence between each layer of $\mathcal{C}^{\mathsf{obj}}_{\mathsf{otl}}$ and orientation of the formation. That is because not only the orientation but also the position of the formation may change during collision with a target object. Fig.5.4 shows the bottleneck. I propose to solve this bottleneck with a space mapping technique widely used in motion planning. It belongs to **solid modeling** and it has the advantage of decoupling from specific shapes of targe objects. Let us view its details in the next few sections.

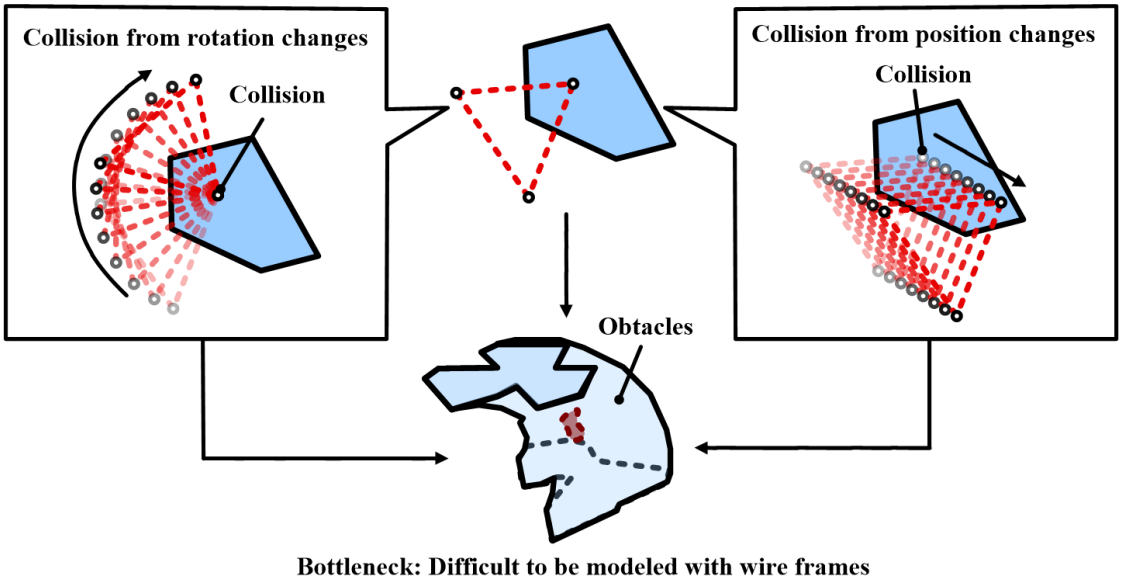


Figure 5.4: $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$ is difficult to be modeled with wireframe modeling.

5.2 Space Mapping

I borrow the $\mathcal{W}$-$\mathcal{C}$ vertex mapping and $\mathcal{W}-\mathcal{C}$ edge mapping algorithm proposed in [Leven and Hutchinson, 2002][Kallman and Mataric, 2004] and [Liu et al., 2010] to perform model $\mathcal{C}^{\mathsf{obj}}_{\mathsf{otl}}$. The $\mathcal{W}$-$\mathcal{C}$ vertex mapping and $\mathcal{W}-\mathcal{C}$ edge mapping algorithm help a robot to response quickly to avoid collision with changing obstacles. These mappings make it possible

to deal with changing obstacle configurations or changing obstacle shapes. It decouples planning from shapes of obstacles. That is one key factor to deal with the bottle neck. Therefore, I employ these mappings. Following the $\mathcal{W}$-$\mathcal{C}$ vertex mapping and $\mathcal{W}-\mathcal{C}$ edge mappings, I propose to pre-build a space mapping between discretized grids in 2D $\mathcal{W}$ space and discretized voxels in 3D $\mathcal{C}$ space to quickly rebuild $\mathcal{C}_{\mathsf{otl}}^{\mathsf{frm}}$ and recover $\mathcal{C}^{\mathsf{frm}}$. Since caging in $\mathcal{C}^{\mathsf{frm}}$ means exploiting the status of $\mathcal{C}_{\mathsf{otl}}^{\mathsf{frm}}$ and $\mathcal{C}^{\mathsf{frm}}$ and checking the existence of holes (expression (5.1)), quickly rebuild $\mathcal{C}_{\mathsf{otl}}^{\mathsf{frm}}$ essentially makes caging test in $\mathcal{C}_{\mathsf{otl}}^{\mathsf{frm}}$ efficient. We can perform 2D caging test quickly in $\mathcal{C}^{\mathsf{frm}}$ with space mapping.

5.2.1 $\mathcal{W}$-$\mathcal{C}$ vertex mapping and $\mathcal{W}$-$\mathcal{C}$ edge mapping

Firstly, let us review $\mathcal{W}$-$\mathcal{C}$ vertex mapping and $\mathcal{W}$-$\mathcal{C}$ edge mapping. This is not a simple review. It includes some of my own understanding and it is the supporting concepts to my space mapping proposal.

Traditional robotic motion planning research involves two steps. The first step is to recover the configuration space of a robot or build a roadmap in the configuration space of robot. Let us denote this "configuration space of robot" with $\mathcal{C}^{\mathsf{rbt}}$. Sometimes, the $\mathcal{C}^{\mathsf{rbt}}$ has ow dimensionality and it could be recovered completely. For instance, a two-joint manipulator's $\mathcal{C}^{\mathsf{rbt}}$ can be represented by its joint space which is two dimensional. The two dimensional space could be recovered completely. If a manipulator has a large number of joints, its $\mathcal{C}^{\mathsf{rbt}}$ would be high dimensional and therefore difficult to be recovered completely. In that case, researchers usually build a roadmap in the high dimensional space instead of recovering it completely. The $\mathcal{W}$-$\mathcal{C}$ vertex mapping and $\mathcal{W}$-$\mathcal{C}$ edge mapping can be employed to deal either the completely recovered space or the roadmap. Fig.5.5(a) gives an example based on a four-joint manipulator. The $\mathcal{C}^{\mathsf{rbt}}$, or joint space of this four-joint manipulator is four dimensional. This four dimensional space is shown in see Fig.5.5(b). Although we can recover this four-dimensional $\mathcal{C}^{\mathsf{rbt}}$ completely, we do not go that far since four dimension space is difficult to be visualized. Moreover, mappings with roadmaps are more general in motion planning comparing with mappings with the whole space since most robots have quite high dimensionality. Therefore, I will show a roadmap built in this four-dimensional $\mathcal{C}^{\mathsf{rbt}}$ and show the basic ideas of $\mathcal{W}$-$\mathcal{C}$ vertex mapping and $\mathcal{W}$-$\mathcal{C}$ edge mapping with this roadmap. The roadmap is a graph shown by blue segments in Fig.5.5(b). Let us denote this graph by $\mathbf{G}$. $\mathbf{G}$ is a graph composed of edges and vertices. Let us denote the edges with $\mathbf{E}$ and the vertices with $\mathbf{V}$. Then, $\mathbf{G}$=$\{\mathbf{E}, \mathbf{V}\}$. One vertex of $\mathbf{V}$ corresponds to a single configuration of the manipulator while one edge of $\mathbf{E}$ corresponds to a set of configurations. The dialogue frames in Fig.5.5 illustrates the correspondence. Here I use the symbol $\boldsymbol{v_i}$ and $\boldsymbol{e_i}$ to denote one vertex and one edge respectively. Namely, $\boldsymbol{v_i} \in \mathbf{V}$ and $\boldsymbol{e_i} \in \mathbf{E}$. $\boldsymbol{e_i}$ corresponds to many configurations so that $\boldsymbol{e_i}$=$\{\boldsymbol{e_{ij}}|j = 1, 2, \ldots, n_e\}$. Here n_e means the number of discretization along an edge. These notations and the discretization of one $\boldsymbol{e_i}$ are shown in Fig.5.5(c).

The second step involves two sub-steps. The first sub-step is to delete the edges and vertices that are obstructed by obstacles. This sub-step is called roadmap updating and is shown in Fig.5.6(a). The vertices $\boldsymbol{v_j}$, $\boldsymbol{v_k}$ and the edge $\boldsymbol{e_m}$ in Fig.5.6(a) are obstructed and

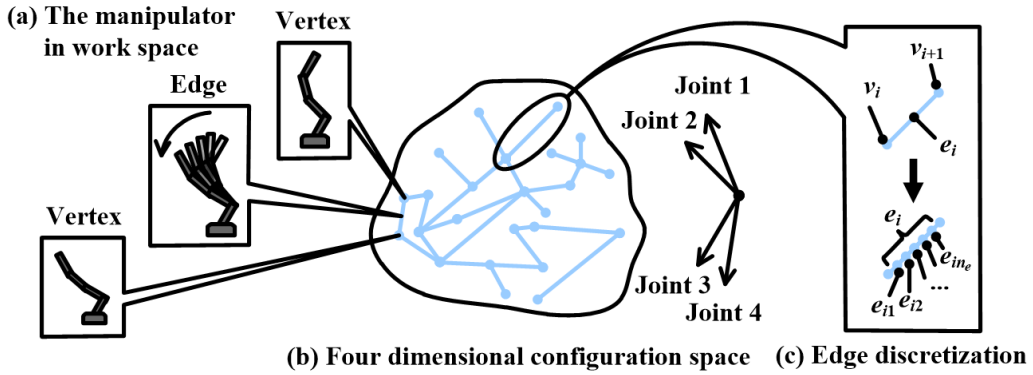


Figure 5.5: A four-joint manipulator and its roadmap in configuration space.

they are rendered with black color for comparison. Note that an edge $\boldsymbol{e_i}$ is considered to be obstructed as long as one discretized configuration $\boldsymbol{e_{ij}}$ is obstructed. The black edges and vertices will be deleted to update the roadmap. In this way, we can make sure that all the edges and vertices in the updated roadmap are reachable. The second sub-step is to connect the starting configuration and goal configuration of the robot to the updated roadmap and plan a path through the updated roadmap that can connect the starting configuration and goal configuration. Fig.5.6(b) shows the path planned in the second sub-step. It is rendered in red color. After planning the path, we can control the manipulator and move it from starting configuration to goal configuration without collision with obstacles. The right part of Fig.5.6(b) shows the moving manipulator along the path and how it avoids collision with obstacles. It is indeed a sequence of configurations that corresponds to every $\boldsymbol{v_i}$ and $\boldsymbol{e_{ij}}$ along the path.

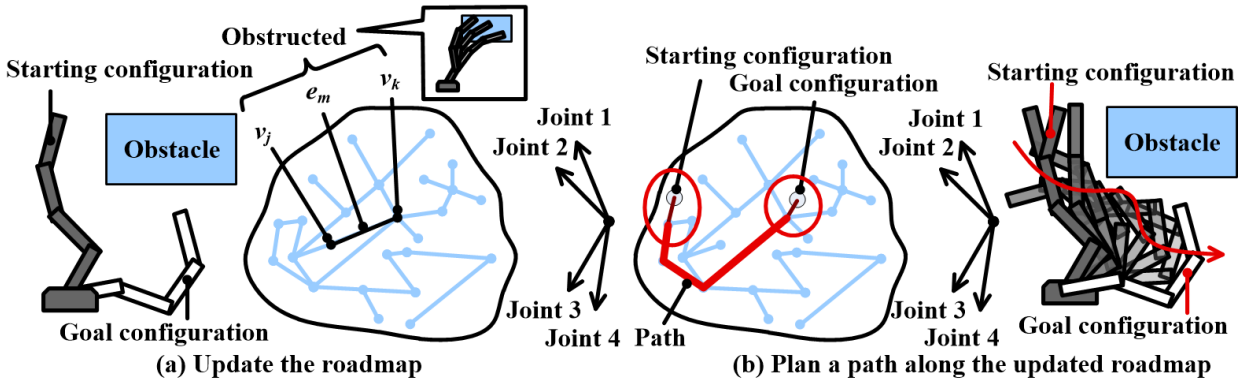


Figure 5.6: Updating and planning a path in the roadmap.

An important problem in these steps is how to update the roadmap. The updating procedure is to delete all obstructed edges and vertices which requires checking weather the robot at a certain configuration collides with the obstacles in $\mathcal{W}$ space. Checking weather

the robot at a certain configuration collides with the obstacles in $\mathcal{W}$ space is a traditional research topic in Computer Graphics, namely **Collision Detection**. Collision detection depends on configurations and geometric shapes of both robots and obstacles. Generally speaking, the configurations of manipulators are limited to the $\boldsymbol{v_i}$ and $\boldsymbol{e_{ij}}$ in $\mathbf{G}$ and the geometric shape of a robot is usually fixed. Nevertheless the configurations and geometric shapes of obstacles may change. As the configurations and geometric shapes of obstacles change, we have to perform collision detection repeatedly. That is a time-consuming task. In order to solve this problem and decouple planning algorithms from the configurations and geometric shapes of obstacles, researchers proposed the idea of $\mathcal{W}$-$\mathcal{C}$ vertex mapping and $\mathcal{W}$-$\mathcal{C}$ edge mapping.

The basic principle of $\mathcal{W}$-$\mathcal{C}$ vertex mapping and $\mathcal{W}$-$\mathcal{C}$ edge mapping is to divide the $\mathcal{W}$ space into **grids** and pre-build a mapping between these grids and the v_i and $\boldsymbol{e_{ij}}$ in $\mathbf{G}$. Note that I am explaining the mappings with 2D $\mathcal{W}$ space. If the space is 3D, it should be divided into **voxels**. Here I use the notation $\boldsymbol{\omega_k}$ to denote a grid in the 2D $\mathcal{W}$ space. Then, the two mappings can be calculated according to the expressions (5.2) and (5.3) respectively. In these expressions, notation R denotes a robot in $\mathcal{W}$ space. $\mathsf{R}[\boldsymbol{v_i}]$ or $\mathsf{R}[\boldsymbol{e_{ij}}]$ denote the grids occupied by R when the robot is at a configuration $\boldsymbol{v_i}$ or $\boldsymbol{e_{ij}}$. Note that $\boldsymbol{v_i}$ and $\boldsymbol{e_{ij}}$ are both configurations in $\mathcal{C}^{\mathsf{rbt}}$.

$$\begin{aligned} \Phi_v(\boldsymbol{\omega_k}) &= \{\boldsymbol{v}_i | (\boldsymbol{v}_i \in \mathbf{V}) \wedge (\mathsf{R}[\boldsymbol{v}_i] \cap \boldsymbol{\omega_k} \neq \emptyset)\} \\ \Omega(\boldsymbol{v}_i) &= \{\boldsymbol{\omega_k} | (\boldsymbol{\omega_k} \in \mathcal{W}) \wedge (\mathsf{R}[\boldsymbol{v}_i] \cap \boldsymbol{\omega_k} \neq \emptyset)\} \end{aligned} \tag{5.2}$$

$$\begin{aligned} \Phi_e(\boldsymbol{\omega_k}) &= \{\boldsymbol{e}_i | (\boldsymbol{e}_i \in \mathbf{E}) \wedge (\mathsf{R}[\boldsymbol{e}_{ij}] \cap \boldsymbol{\omega_k} \neq \emptyset), \forall \boldsymbol{e}_{ij} \in \boldsymbol{e}_i\} \\ \Omega(\boldsymbol{e}_i) &= \{\boldsymbol{\omega_k} | (\boldsymbol{\omega_k} \in \mathcal{W}) \wedge (\mathsf{R}[\boldsymbol{e}_{ij}] \cap \boldsymbol{\omega_k} \neq \emptyset), \forall \boldsymbol{e}_{ij} \in \boldsymbol{e}_i\} \end{aligned} \tag{5.3}$$

Each mapping is calculated redundantly with two functions, namely $\Phi()$ and $\Omega()$. The $\Phi()$ function saves all the vertices and edges that corresponds to a $\boldsymbol{\omega_k}$. The $\Phi_v(\boldsymbol{\omega_k})$ in expression (5.2) and the $\Phi_e(\boldsymbol{\omega_k})$ in expression (5.3) denote them. The $\Omega()$ function saves all the grids that correspond to $\boldsymbol{v_i}$ or $\boldsymbol{e_i}$ of a $\mathbf{G}$. The $\Omega(\boldsymbol{v}_i)$ in expression (5.2) and the $\Omega(\boldsymbol{e}_i)$ in expression (5.3) denote them.

The mapping can be **pre-built off line** before being used in real time. Therefore, it cost little resources and offers the following benefits.

- **(1):** Given a $\boldsymbol{v}_i$ or an $\boldsymbol{e}_i$, the mapping can quickly tell us the $\boldsymbol{\omega_k}$s that $\mathsf{R}[\boldsymbol{v}_i]$ or $\mathsf{R}[\boldsymbol{e}_i]$ overlaps according to $\Omega(\boldsymbol{v}_i)$ or $\Omega(\boldsymbol{e}_i)$.
- **(2):** Given an $\boldsymbol{\omega_k}$, the mapping can quickly tell us the $\boldsymbol{v}_i$s or $\boldsymbol{e}_i$s at which $\mathsf{R}[\boldsymbol{v}_i]$ or $\mathsf{R}[\boldsymbol{e}_i]$ overlaps with the given $\boldsymbol{\omega_k}$ according to $\Phi_v(\boldsymbol{\omega_k})$ or $\Phi_e(\boldsymbol{\omega_k})$.

Fig.5.7 illustrates the details of $\mathcal{W}$-$\mathcal{C}$ mappings. Fig.5.7(a) is the $\Omega()$ function which maps a configuration of the roadmap to a set of $\boldsymbol{\omega_k}$ in $\mathcal{W}$ space. Fig.5.7(b) is the $\Phi()$ function which maps a grid in $\mathcal{W}$ space to a set of vertices and edges in $\mathbf{G}$.

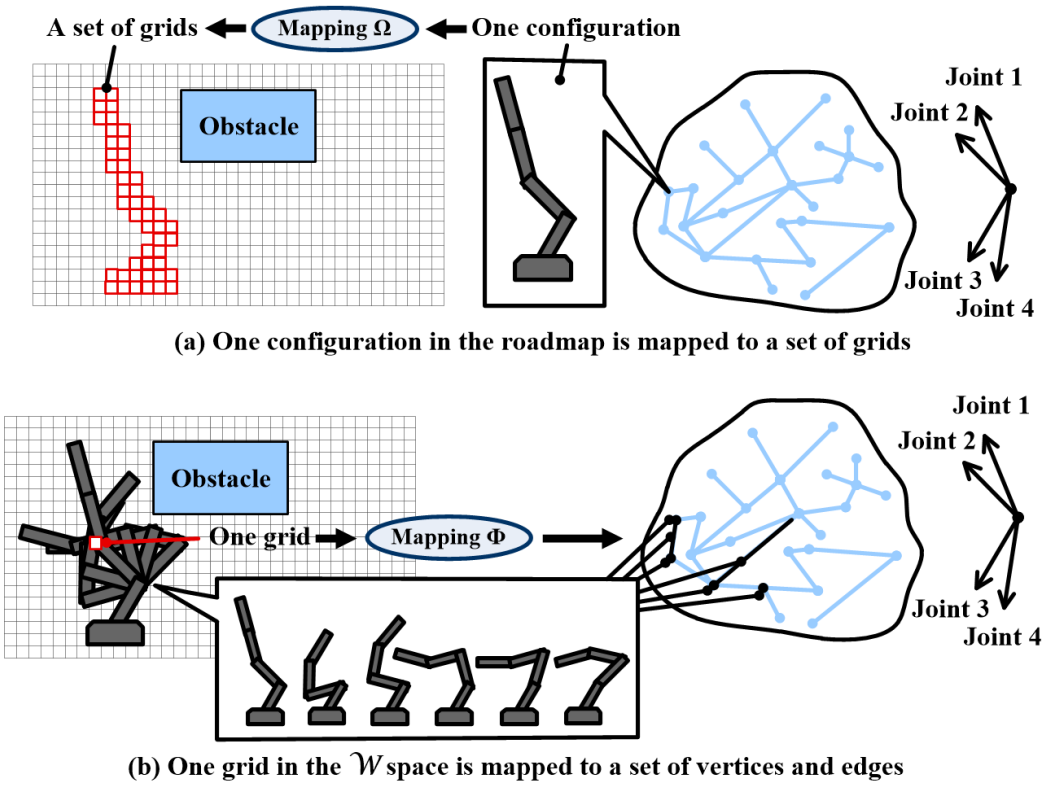


Figure 5.7: Illustration of the functions in expression (5.3) and expression (5.2).

When obstacles change, perception devices can get the $\mathcal{W}$ space grids $\boldsymbol{\omega_k}$s occupied by the obstacles. Then the algorithm check whether $\boldsymbol{v}_i$s or $\boldsymbol{e}_i$s of the $\mathbf{G}$ are obstructed by looking up $\Phi_v(\boldsymbol{\omega_k})$ and $\Phi_e(\boldsymbol{\omega_k})$ following benefit **(2)**. Note that the $\Omega()$ functions are not used explicitly here. In this way, the algorithm does not need to perform collision detection repeatedly to updated $\mathbf{G}$ since detailed obstacle information is encoded by the grids $\boldsymbol{\omega_k}$s and an unique space mapping. It only needs to look up the pre-built mapping to update roadmap $\mathbf{G}$. This procedure is efficient and enables a robot to re-plan paths and avoid collision in real time.

Fig.5.8 illustrates this procedure. Fig.5.8(a) updates the roadmap by considering the grids occupied by the obstacle and by looking up the mappings of those grids. This is shown in the left part of Fig.5.8(a). Then, a path is planned based on the updated roadmap in the right part of Fig.5.8(a). The procedure is efficient since we do not need to perform on-line calculations like collision detection. When configurations or shapes of the obstacle changes, we can still quickly update and plan paths by detecting the occupied grids and by looking up the mappings of those occupied grids. Fig.5.8(b) and Fig.5.8(c) illustrate the case where the obstacle is at a different configuration and the case where the obstacle has a different geometric shape respectively. The $\mathcal{W}$-$\mathcal{C}$ mappings successfully decouples planning algorithms from specific obstacles and enables real-time computation.

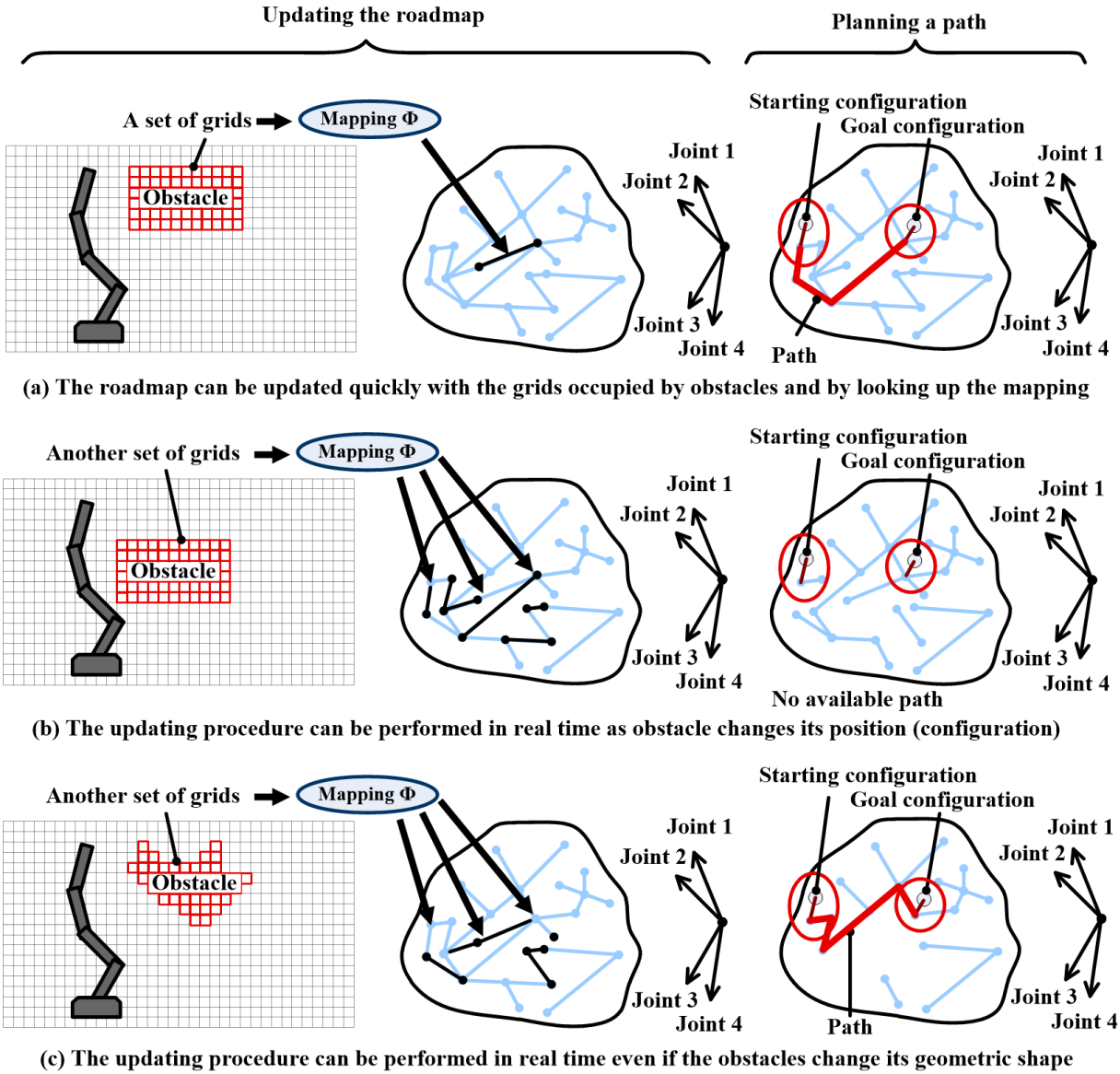


Figure 5.8: $\mathcal{W}$-$\mathcal{C}$ mappings enables real-time motion planning.

If this mapping technique can be employed in $\mathcal{C}^{\mathsf{frm}}$, we may solve the bottleneck of $\mathcal{C}^{\mathsf{frm}}$. On the one hand, we can pre-build lots of off-line mappings for different formations. These off-line mapping can be pre-built off line and can be employed efficiently to update many different $\mathcal{C}^{\mathsf{frm}}$s in necessary cases. On the other hand, each $\mathcal{C}^{\mathsf{frm}}$ can be used to deal with unknown target objects efficiently since target objects have been decoupled from algorithms. I am going explain how to use this mapping technique in the next subsection.

5.2.2 Space mapping for caging test

5.2.2.1 From motion planning to caging test

The aim of motion planning is to plan a path in $\mathcal{C}^{\mathsf{rbt}}$ along which a robot can avoid collision with obstacles. The aim of caging test in $\mathcal{C}^{\mathsf{frm}}$ is to check whether there exist an caging subspace. There seems to be no relationship between them. However, if we view the second aim in another form, we can find their similarity. The aim of caging test in $\mathcal{C}^{\mathsf{frm}}$ is to check whether there exist a path in $\mathcal{C}^{\mathsf{frm}}$ along which a formation of fingers can escape into infinity from the target object without collision. We can find that there are some some correspondences between these two problems. A robot in $\mathcal{W}$ space becomes a formation of fingers while the obstacles become the target object. Fig.5.9 shows their correspondence. Especially, these two correspondences are marked with "Correspondence 1" and "Correspondence 2" in Fig.5.9. Note that both the two $\mathcal{W}$ spaces in Fig.5.9 are discretized into grids for mapping.

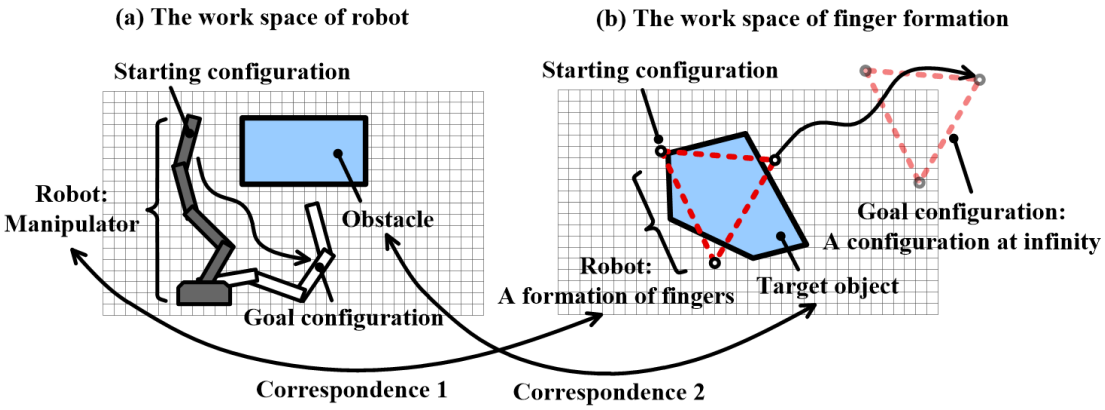
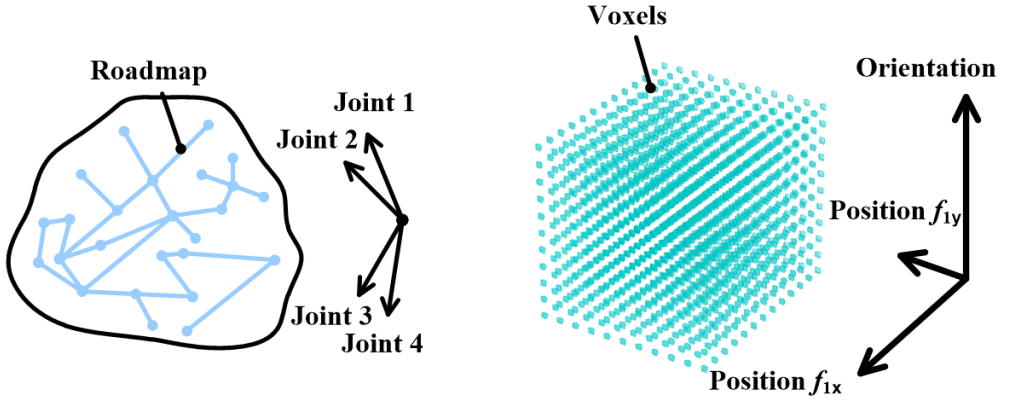


Figure 5.9: Correspondences between motion planning and caging test.

The discretization shows that a robot at a certain configuration is discretized into a compact set of grids while a formation of fingers at a certain configuration is discretized into several unconnected grids. Since we discuss the problem with point fingers. The number of discretized grids of a formation of fingers is the same as the number of fingers in this formation. That is to say, $\boldsymbol{f_i} = \{f_{i_x}, f_{i_y}\}$ is actually the same thing as $\boldsymbol{\omega_j} = \{\omega_{j_x}, \omega_{j_y}\}$. They both indicate a position on 2D plane. Formally, it can be expressed as following.

$$\mathsf{F}[\boldsymbol{q}^{\mathsf{frm}}] = \{R^{(q_\theta^{\mathsf{frm}})} \cdot \{f_{j_\mathsf{x}} - f_{1_\mathsf{x}} + q_\mathsf{x}^{\mathsf{frm}}, f_{j_\mathsf{y}} - f_{1_\mathsf{y}} + q_\mathsf{y}^{\mathsf{frm}}, 1\} | 1 < j \leq n_f\} = \{\omega_{k_\mathsf{x}}, \omega_{k_\mathsf{y}}\} | 1 < k \leq n_f\} \tag{5.4}$$

The $\mathcal{C}$ spaces of the two problems, unlike their correspondences in $\mathcal{W}$ spaces, are quite different. That is because motion planning aims at finding a path. Therefore, we do not need to update the whole $\mathcal{C}^{\mathsf{rbt}}$. A roadmap of the space could be enough to find a path. In contrast, the caging test problem aims at checking whether there exists a path. That means we need to update the whole $\mathcal{C}^{\mathsf{frm}}$ and check the existence. Luckily the $\mathcal{C}^{\mathsf{frm}}$ is limited to a single formation and it is three dimensional. We can recover and update the whole three dimensional space by discretizing it into voxels. Fig.5.10 compares discretization of the two $\mathcal{C}$ spaces.



(a) The motion planning in C^{rbt} aims at finding a path. Therefore, discretizing it into a roadmap could be enough.

(b) The caging test in C^{frm} aims at checking the existence of paths. We discretize it into voxels to ensure completeness.

Figure 5.10: Discretization of $\mathcal{C}^{\mathsf{rbt}}$ and $\mathcal{C}^{\mathsf{frm}}$.

The $\mathcal{W}$-$\mathcal{C}$ vertex mapping and $\mathcal{W}$-$\mathcal{C}$ edge mapping in expression (5.2) and (5.3) become a whole space mapping which is acctually the correspondence between grids in $\mathcal{W}$ space and voxels in $\mathcal{C}^{\mathsf{frm}}$. Expression (5.5) shows the space mapping for caging test. Here each voxel of $\mathcal{C}^{\mathsf{frm}}$ is represented by a configuration, namely $\mathbf{q}^{\mathsf{frm}}$.

$$\begin{aligned} \Phi(\omega_k) &= \{\mathbf{q}^{\mathsf{frm}} | (\mathbf{q}^{\mathsf{frm}} \in \mathcal{C}^{\mathsf{frm}}) \wedge (\mathsf{F}[\mathbf{q}^{\mathsf{frm}}] \cap \omega_k \neq \emptyset)\} \\ \Omega(\mathbf{q}^{\mathsf{frm}}) &= \{\omega_k | (\omega_k \in \mathcal{W}) \wedge (\mathsf{F}[\mathbf{q}^{\mathsf{frm}}] \cap \omega_k \neq \emptyset)\} \end{aligned} \tag{5.5}$$

Fig.5.11 shows the mapping between one voxel/one configuration of $\mathcal{C}^{\mathsf{frm}}$ and the $\mathcal{W}$ space grids. It is the $\Omega()$ function.

The other function, $\Phi()$ is a little complicated. Given a grid, there are two possibilities of obstruction. One possibility is translational obstruction. In this possibility the orientation of a finger formation does not change and each finger overlaps with the grid. There are n_f

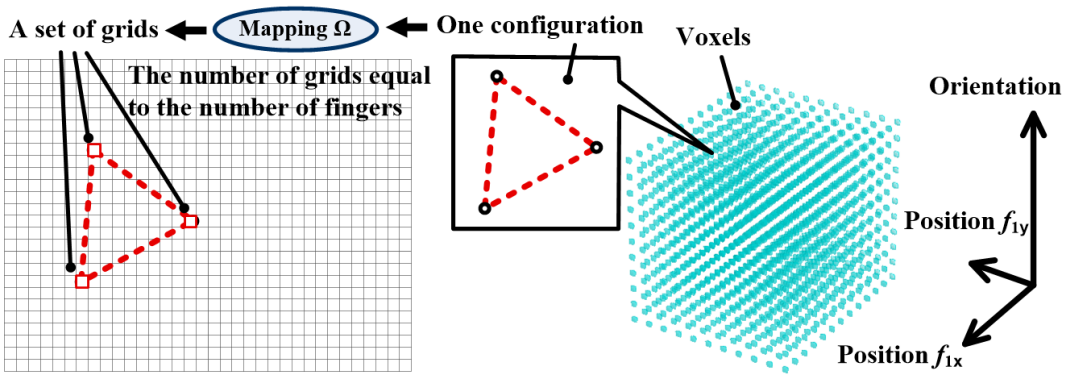


Figure 5.11: Illustration of the Ω() function of space mapping for caging test.

different collisions. The other possibility is rotational obstruction. In this possibility the orientation of a finger formation changes and a finger overlaps with the grid continuously during rotation. Fig.5.12 demonstrates the two possibilities of obstruction in $\mathcal{W}$ space with the three-finger formation. Since the configuration of a finger formation is decided by the position of $\boldsymbol{f_1}$ and the orientation of the whole formation, The two possibilities of obstruction is mapped to some voxels in $\mathcal{C}^{\mathsf{frm}}$. These voxels are shown in Fig.5.13. A gird in the left part of Fig.5.13 is mapped into a set of helical voxels in the right part of Fig.5.13 and it indicates the Φ() function.

We can pre-build and store the mappings between every grids in $\mathcal{W}$ space and the voxels in $\mathcal{C}^{\mathsf{frm}}$ and reuse it in on-line procedures to efficiently perform caging test. Specifically, the perception devices can get the $\mathcal{W}$ space grids $\boldsymbol{\omega_k}$s occupied by the target objects. Then the algorithm recover and update the $\mathcal{C}^{\mathsf{frm}}$ by referring to the pre-built space mappings. This procedure only uses the Φ() function while the Ω() function is not used explicitly. In this way, the algorithm does not need to re-calculate the wire-frame model according to specific geometric shapes of target objects. Detailed target object information is encoded by and the grids $\boldsymbol{\omega_k}$s and the space mapping. We only needs to look up the pre-built mapping to update $\mathcal{C}^{\mathsf{frm}}$ with **solid modelling**. This procedure is efficient and enables efficient caging tests.

Fig.5.14 illustrates this procedure. Fig.5.14(a) recovers and updates $\mathcal{C}^{\mathsf{frm}}$ by considering the grids occupied by the target object and by looking up the mappings of those grids. The grids occupied by the target object are shown in the left part of Fig.5.14(a). When $\mathcal{C}^{\mathsf{frm}}$ is updated, caging test can be quickly performed by checking the existence or isolation of certain sub-spaces through labelling the voxels. The right part of Fig.5.14(a) shows the result of labeling a certain object. The mapped voxels of those occupied grids are labeled with "cyan" color in the right part of Fig.5.14(a) while the caging sub-spaces are labeled with "red" color. Since the finger formation cannot cage the target object in Fig.5.14(a). There is no "red" sub-spaces. When configurations or shapes of the targe object changes, we can quickly update $\mathcal{C}^{\mathsf{frm}}$ and re-perform caging test by detecting the occupied grids and by

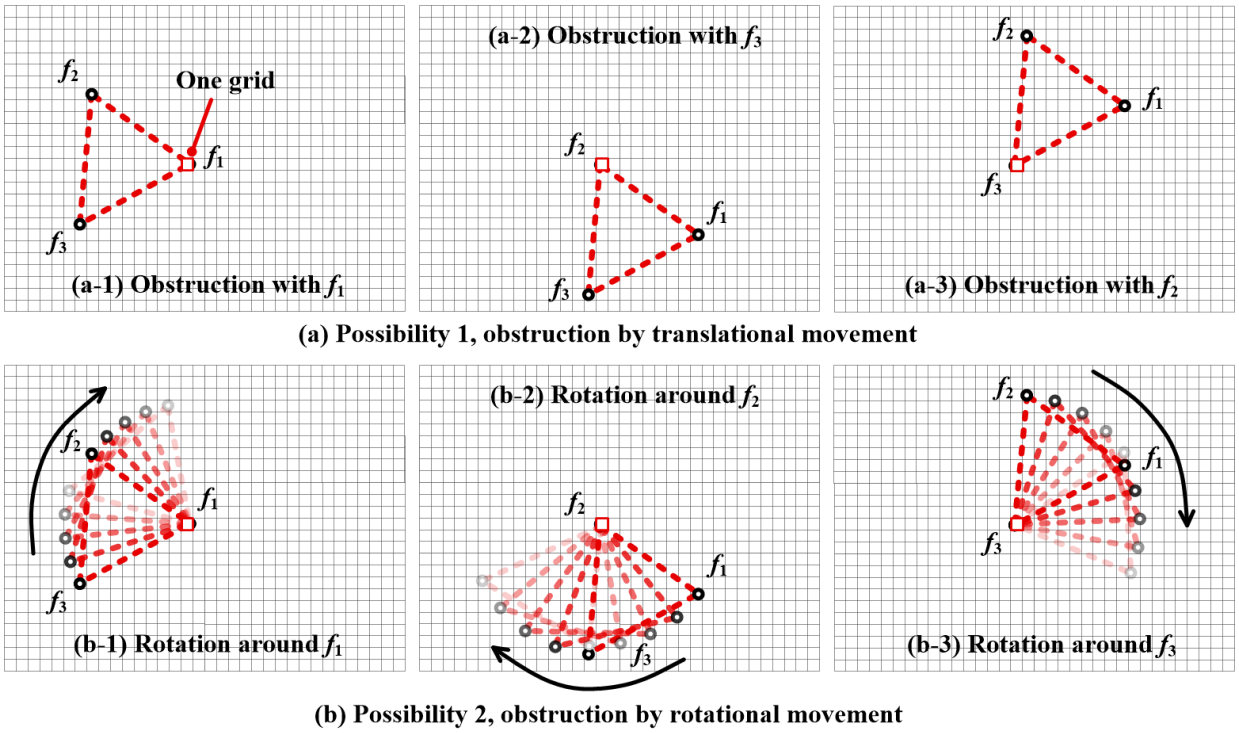


Figure 5.12: A grid in $\mathcal{W}$ space may obstruct with fingers in two ways.

looking up the mappings of those occupied grids again. Fig.5.14(b) illustrate the case where the target object has a different geometric shape. In this case, the finger formation could cage the given target objects because there are caging sub-spaces and the initial configuration of the finger formation is inside one of them. The space mapping for caging test successfully decouples caging tests from specific target objects and enables real-time caging test.

Note that the target object in Fig.5.14(b) is not convex. This is another advantage of using the space mapping and decoupling specific target objects from algorithms. We will see later that the space mapping-based algorithm is not only applicable to both convex and concave target objects but also applicable to target objects with hollow holes. The major disadvantage of space mapping is we have to pre-build different mappings for different finger formations. We will explore more about this disadvantage soon but firstly let us summarize the mapping algorithm formally and analyze its performance.

5.2.2.2 The mapping algorithm and analysis

Fig.5.13 showed that the mapping of a grid in $\mathcal{W}$ space is a set of helical voxels in $\mathcal{C}^{\mathsf{frm}}$. In order to formally present the algorithm of calculating this mapping, we need to define the granularity of discretization along orientation axis. Here I would like to keep coherent with the discretization along orientation in $\mathcal{C}^{\mathsf{obj}}$ and use $2m + 1$ as the granularity of [-π, π). Accordingly, the algorithm of calculating space mapping is as the pseudo code in

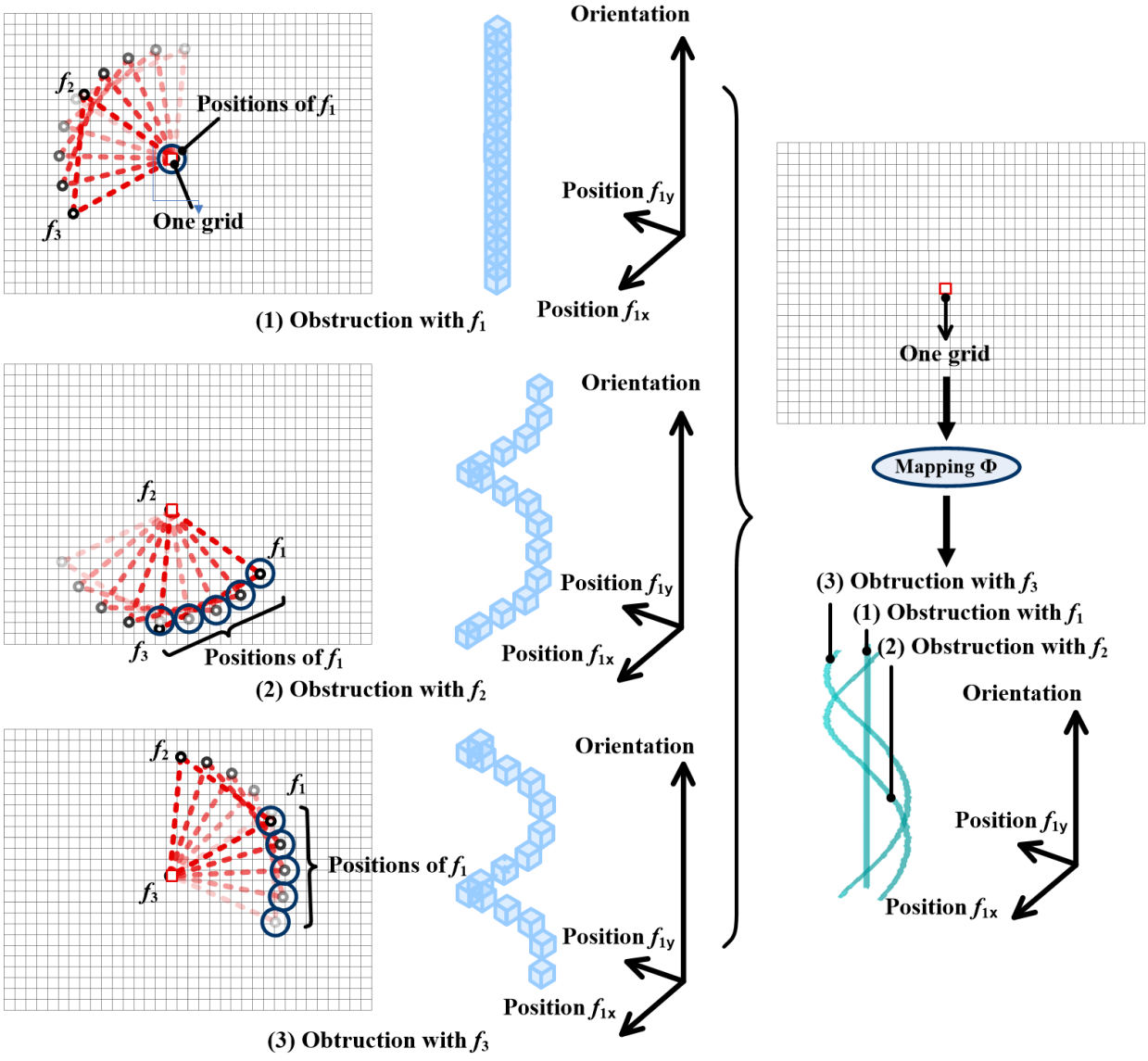


Figure 5.13: A grid in $\mathcal{W}$ space corresponds to some helical voxels in $\mathcal{C}^{\mathsf{frm}}$.

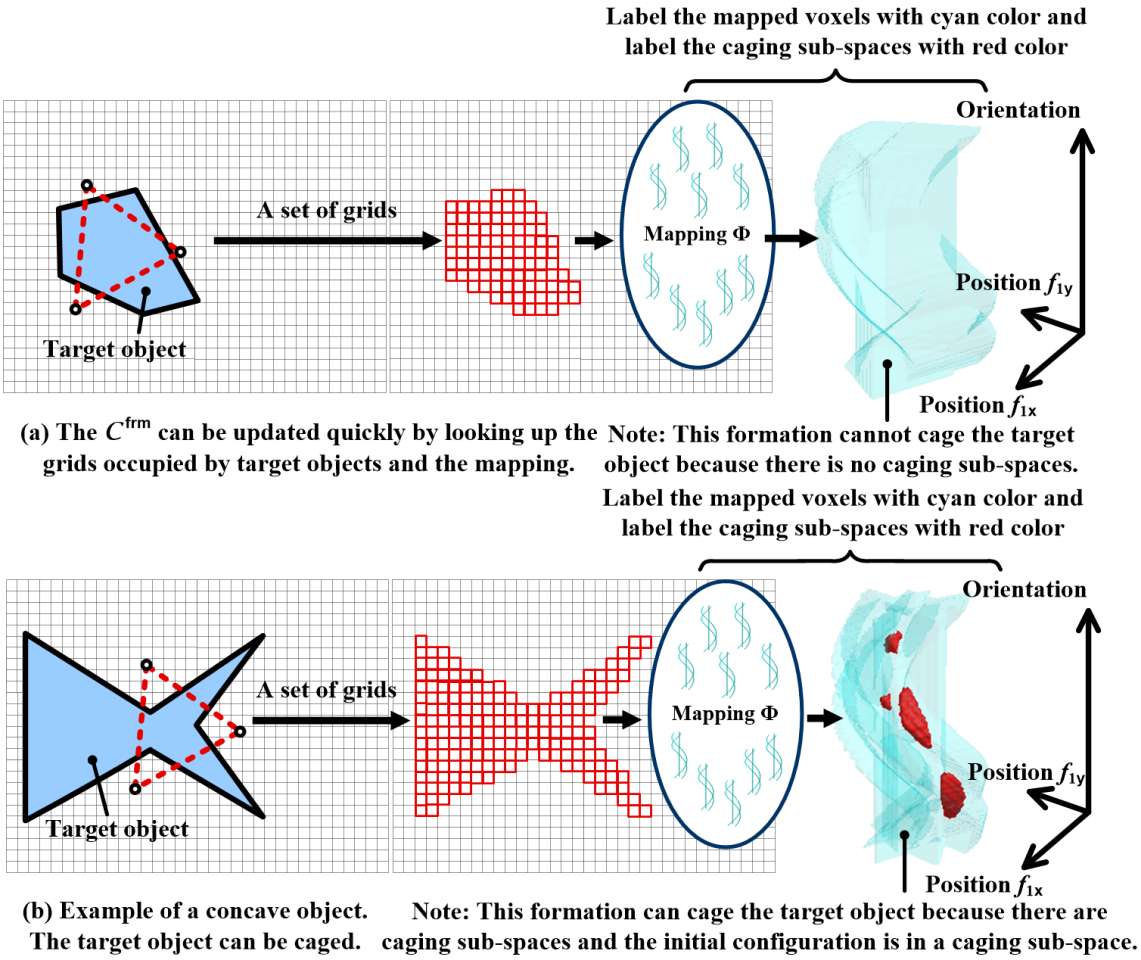


Figure 5.14: Caging test by using the space mappings.

Alg.3. I recommend readers refer to Fig.5.13 to better understand this algorithm. Here the notation n_g indicates the number of discretized grids of the 2D $\mathcal{W}$ space. Note that the translation and rotation obstruction illustrated in Fig.5.12 are not explicitly calculated in Alg.3. Instead, I use $\{f_{1_x} - f_{j_x} + f_{1_x}, f_{1_x} - f_{j_y} + f_{1_x}\}$ to calculate the positions of $\boldsymbol{f_1}$ and the occupied voxels when rotation center is changed to finger $\boldsymbol{f_j}$, $j \neq 1$. Fig.5.15 illustrates this conversion. The positions of $\boldsymbol{f_1}$ in Fig.5.13(b),(c) or Fig.5.15(b),(c) are not calculate explicitly. They are converted to the same rotation in Fig.5.15(a) and implicitly calculated by $\{f_{1_x} - f_{j_x} + f_{1_x}, f_{1_x} - f_{j_y} + f_{1_x}\}$. This conversion saves us from repeated translation and rotation of the finger formation. This conversion is low-level details of the algorithm implementation but it could simplify the complexity of programs.

Results of this algorithm would be a set of mappings where each mapping in the set records the correspondence between one grid and some voxels. This set of mappings is denoted by $\Phi(\omega)$ in Alg.3 while each element of this set is denoted by $\Phi(\omega_k)$.

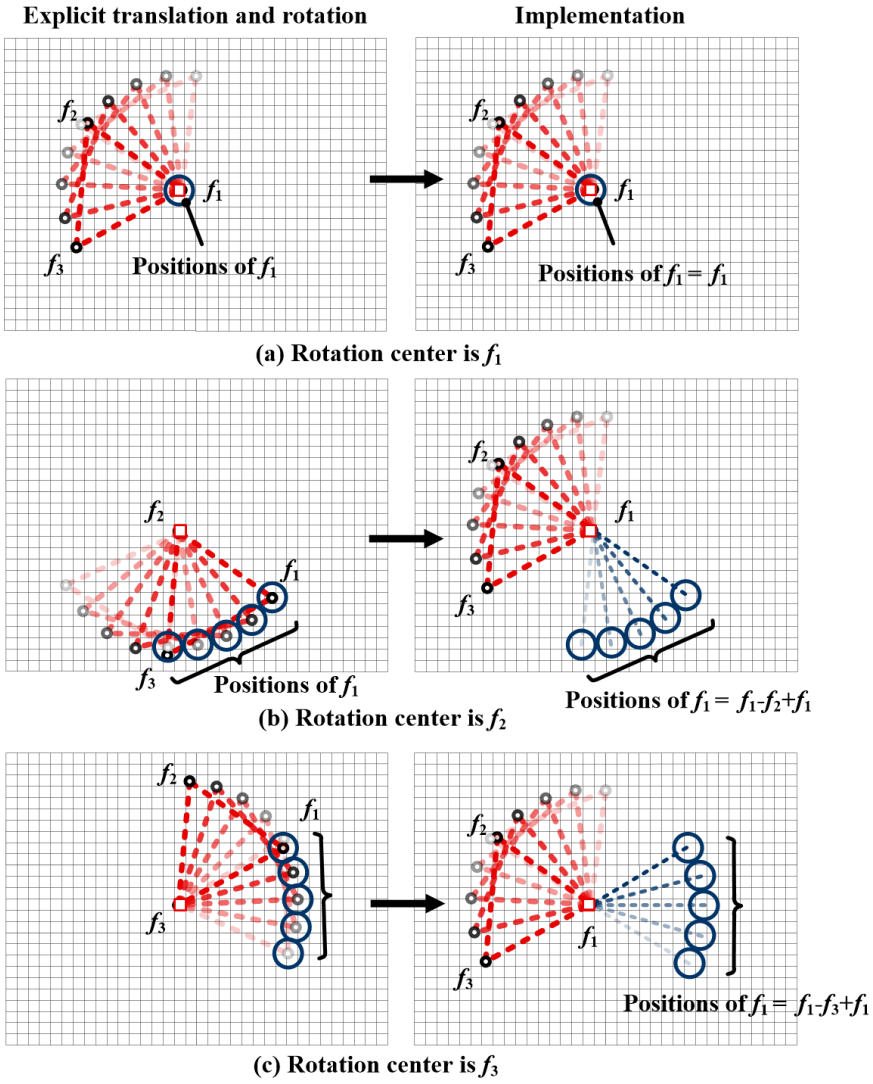


Figure 5.15: Converting translation and rotation to $\{f_{1_x} - f_{j_x} + f_{1_x}, f_{1_x} - f_{j_y} + f_{1_x}\}$.

Algorithm 3: Pre-building the space mapping.

Data: ω_k, n_g, F
Result: $\Phi(\omega)$=$\{\Phi(\omega_k)|k = 1, 2, \ldots, n_g\}$

```
1  begin
2      Φ(ω_k) ← ∅
3      for k ∈ {1 : 1 : n_g} do
4          for i ∈ {−m : 1 : +m} do
5              θ^frm=(i/(2m + 1)) ∗ 2π
6              q_k^frm ← {ω_kx, ω_ky, θ^frm}
7              foreach f_j ∈ F[q_k^frm] do
8                  if j==1 then
9                      q_j^frm ← {f_jx, f_jy, θ^frm}
10                 else
11                     q_j^frm ← {f_1x − f_jx + f_1x, f_1x − f_jy + f_1x, θ^frm}
12                 addToSet(Φ(ω_k), q_j^frm)
13             end
14         end
15         appendToMapping(Φ(ω), Φ(ω_k))
16     end
17     return Φ(ω)
18 end
```

The time complexity of this algorithm is $O((2m + 1) \cdot n_g \cdot n_f) = O(m \cdot n_g)$ since the number of fingers n_f has in most cases small value. Although $O(m \cdot n_g)$ is asymptotic to 2nd-order curves, it costs lots of computational resources. This is because n_g is the number of discretized grids on 2D plane. It could be 10000 even if the plane is roughly divided into 100×100 grids. This is one drawback. Nevertheless, since the mappings could be computed off line, this is not an important issue.

The storage complexity of this algorithm is also $O((2m + 1) \cdot n_g \cdot n_f) = O(m \cdot n_g)$. Each grid on $\mathcal{W}$ space corresponds to $(2m + 1) \cdot n_f$ voxels in $\mathcal{C}^{\mathsf{frm}}$. n_f is neglected since it has always small value. The storage complexity is troublesome because even if it is pre-built off line, we have to spare enough memory to record the mapping.

I implement the space mapping algorithm in Alg.3 with different n_g and different m to better view the time and storage complexity. It is based on an Intel i7 M620 CPU with MATLAB 2010b. The result is shown in Fig.5.16.

As n_g increases, both time and storage costs increase dramatically since both of their complexity are $O(m \cdot n_g)$. It heavily depends on n_g. Suppose we choose n_g=150×150=22500, then the time cost and storage cost of one finger formation would be $1.11s$ and 3.26MB respectively. This value is acceptable. However, mapping of one finger formation is never enough. We need to build the mappings for many different formations. Suppose that we pre-build the

n_g	m ($2m$+1)	Time (in second)	Storage size (in megabytes)
70x70=4900	18 (2x18+1=37)	0.26s	0.31MB
100x100=10000	18 (2x18+1=37)	0.46s	0.78MB
100x100=10000	36 (2x36+1=73)	0.51s	1.04MB
120x120=14400	36 (2x36+1=73)	0.75s	1.79MB
150x150=22500	36 (2x36+1=73)	1.11s	3.26MB
200x200=40000	36 (2x36+1=73)	2.12s	7.28MB
300x300=90000	36 (2x36+1=73)	4.03s	19.75MB
400x400=160000	36 (2x36+1=73)	7.35s	39.29MB

Figure 5.16: Time and storage cost of space mapping for caging test.

mappings for 1000 finger formations. Then, the total time cost of the off line mapping procedure would be $1.11\times1000s$=$1110s$. The total storage cost would be 3.26MB$\times$1000=3.26GB. That is no longer acceptable. Moreover, the cost would be much higher if we use finer discretization. Consequently, it is necessary to make certain improvements to lower various costs of the space mapping algorithm.

5.3 Further Improvements

5.3.1 Improve space mapping by shifting

Space mapping can help us perform caging test against a fixed formation rapidly. However, it has two disadvantages. One is we have to pre-build and record the mapping of every grids in $\mathcal{W}$ space. This drawbacks make the time and storage cost of Alg.3 as high as $O(m \cdot n_g)$. It can only be pre-build off line and it is unavailable to be built in real time. The second draw back is we have to pre-build many mappings in order to recover and update the $\mathcal{C}^{\mathsf{frm}}$ of different finger formations. The storage cost could be $O(m \cdot n_g \cdot n_{\mathsf{F}})$ with n_{F} denoting the number of formations. Can we improve it further? The answer is yes.

Actually, each ω_k in $\mathcal{W}$ space corresponds to the same pattern of voxels. All mapping of a given finger formation has the same helical pattern shown in Fig.5.13 except for the difference in positions. That is because the two horizontal coordinates of the 3D $\mathcal{C}^{\mathsf{frm}}$ are defined as the position of $\boldsymbol{f_1}$. Given a grid ω_k and a finger formation, its correspondent helical voxels are as following. Note that the expressions $f_{1_\mathsf{x}} - f_{2_\mathsf{x}} + \omega_{i_\mathsf{x}}$ and $f_{1_\mathsf{y}} - f_{2_\mathsf{y}} + \omega_{i_\mathsf{x}}$ in this expression (5.6) are exactly the same thing as Fig.5.15 and the expressions $\{f_{1_\mathsf{x}} - f_{j_\mathsf{x}} + f_{1_\mathsf{x}}, f_{1_\mathsf{x}} - f_{j_\mathsf{y}} + f_{1_\mathsf{x}}\}$.

This conversion saves us from repeated translation and rotation of the finger formation. in Alg.3. They are just in different forms. $\{f_{1_x} - f_{j_x} + f_{1_x}, f_{1_x} - f_{j_y} + f_{1_x}\}$. This conversion saves us from repeated translation and rotation of the finger formation. in Fig.5.15 and Alg.3 uses two f_{1_x} because they are a finger from formation $\mathsf{F}[\boldsymbol{q}_k^{\mathsf{frm}}]$. Expresson (5.6) uses f_{1_x} and ω_{k_x} because all fingers are from the initial orientation $\mathsf{F}[\boldsymbol{q}_0^{\mathsf{frm}}]$. $\mathsf{F}[\boldsymbol{q}_k^{\mathsf{frm}}]$ is not explicitly expressed.

$$\begin{aligned}\Phi(\omega_k) &= \{\mathbf{q}^{\mathsf{frm}} | (\mathbf{q}^{\mathsf{frm}} \in \mathcal{C}^{\mathsf{frm}}) \wedge (\mathsf{F}[\mathbf{q}^{\mathsf{frm}}] \cap \omega_k \neq \emptyset)\} \\ &= \{\{\omega_{k_x}, \omega_{k_y}, (j/(2m+1)) * 2\pi\}, \\ &\quad \{f_{1_x} - f_{2_x} + \omega_{k_x}, f_{1_y} - f_{2_y} + \omega_{k_y}, (j/(2m+1)) * 2\pi\}, \ldots, \\ &\quad \{f_{1_x} - f_{n_{f_x}} + \omega_{k_x}, f_{1_y} - f_{n_{f_y}} + \omega_{k_y}, (j/(2m+1)) * 2\pi\} | -m \leq j \leq +m\}\end{aligned} \tag{5.6}$$

Given another grid ω_i and a finger formation, its correspondent helical voxels are as following.

$$\begin{aligned}\Phi(\omega_i) =& \{\mathbf{q}^{\mathsf{frm}} | (\mathbf{q}^{\mathsf{frm}} \in \mathcal{C}^{\mathsf{frm}}) \wedge (\mathsf{F}[\mathbf{q}^{\mathsf{frm}}] \cap \omega_i \neq \emptyset)\} \\ =& \{\{\omega_{i_x}, \omega_{i_y}, (j/(2m+1)) * 2\pi\}, \\ &\{f_{1_x} - f_{2_x} + \omega_{i_x}, f_{1_y} - f_{2_y} + \omega_{i_y}, (j/(2m+1)) * 2\pi\}, \ldots, \\ &\{f_{1_x} - f_{n_{f_x}} + \omega_{i_x}, f_{1_y} - f_{n_{f_y}} + \omega_{i_y}, (j/(2m+1)) * 2\pi\} | -m \leq j \leq +m\}\end{aligned} \tag{5.7}$$

We can see that $\Phi(\omega_k)$ and $\Phi(\omega_i)$ follow exactly the same helical pattern except the difference in horizontal coordinates. That is to say,

$$\Phi(\omega_k) = \Phi(\omega_i) + \{\omega_{k_x - \omega_{i_x}}, \omega_{k_y - \omega_{i_y}}, 0\} \tag{5.8}$$

Consequently, we do not need to calculate the mappings of every grid. We can firstly calculate the mapping of a reference grid, say ω_1. Then, the mappings of another grid, say ω_j, could be obtain by translating the mapped voxels in $\Phi(\omega_1)$ by expression (5.8). I name this translating procedure "shifting" and name the space mapping algorithm with "shifting" the *improved space mapping*.

The shifting procedure reduces storage cost of one finger formation from $O(m \cdot n_g)$ to $O(m \cdot 1)$=$O(m)$ and the storage cost of n_{F} formations into $O(n_{\mathsf{F}} \cdot m)$. This saves lots of storage resources. At the same time, it does loose much efficiency. In the *improved space mapping*, the total time cost is the time cost of calculating the mapping of one reference grid plus the time cost of shifting. Calculating the mapping of one grid costs $O(m)$. It has the same complexity of storage cost. Shifting depends on the number of grids occupied by target objects. Surely we do not need to shift to any voxels on the horizontal plane. Simply shifting to the grids occupied by the target object would be enough to recover and update the whole $\mathcal{C}^{\mathsf{frm}}$. Suppose the target object takes up n_o grids, then the cost of shifting would be $O(n_o)$. Reducing the mapping of every grids in the plane to those occupied by the target object is like the idea of lazy evaluation in motion planning [Bohlin and Kavraki, 2000]. It

lazily postpones the mapping until necessary grids instead of actively maps all of them. That is the main advantage of shifting.

Now let us compare the original space mapping and improved mapping and shifting. In order to make clear their difference, I use *raw space mapping* to denote the space mapping proposed in section 5.2.2.2 and use *improved space mapping* to denote the space mapping with shifting. In *raw space mapping*, the mapping of all grids in $\mathcal{W}$ space are pre-built off line while in *improved space mapping*, only the mapping of one grid in $\mathcal{W}$ space is pre-built off line. The mapping of the other obstructed grids are lazily shifted during execution. Fig.5.17 shows the time and storage costs of these two mappings.

	m ($2m$+1)	Raw space mapping	Improved space mapping	
Time cost	Mapping	$O(m \cdot n_g)$	$O(m)$	Pre-built off line
	Shifting	N/A	$O(n_o)$	On-line calculation
	Labeling	$O(m \cdot n_o)$	$O(m \cdot n_o)$	
Storage cost		$O(m \cdot n_g)$	$O(m)$	

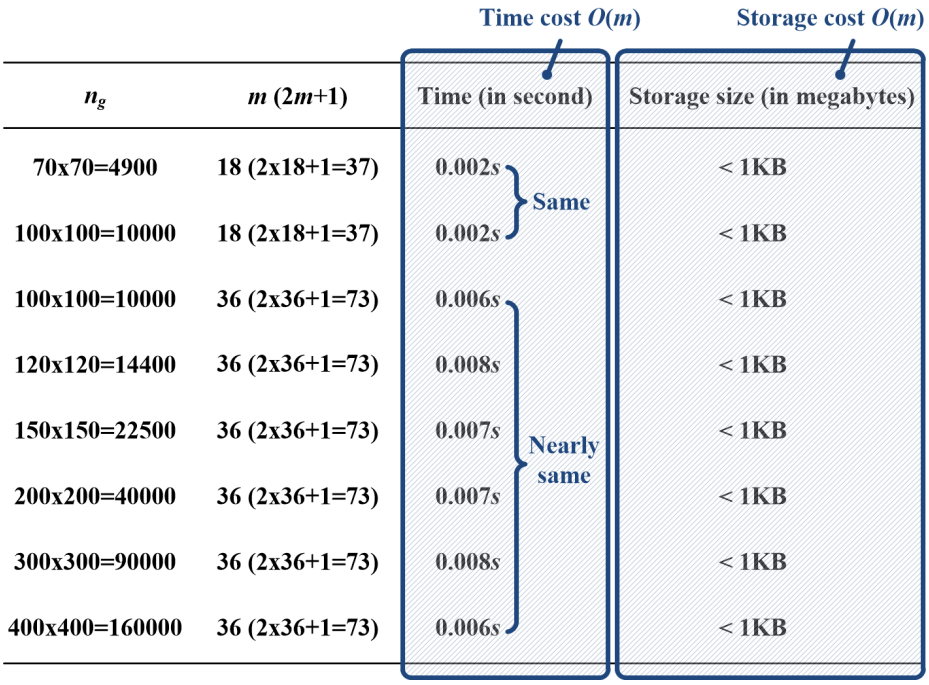
Figure 5.17: Comparison of cost between *raw space mapping* and *improved space mapping*.

As shown in Fig.5.17, the total time cost of *raw space mapping* is $O(m \cdot n_g)$(off line) while the total time cost of improved mapping is $O(m)$ (off line) plus $O(n_o)$ (on-line shifting). Here n_o denotes the number of grids occupied by target objects. Note that n_o is smaller than n_g since the occupied grids belong to a sub-region of $\mathcal{W}$ space.

It is true that if we only consider about on-line cost, *improved space mapping* is larger than *raw space mapping*. *Raw space mapping* costs $0 + O(m \cdot n_o) = O(m \cdot n_o)$ in on-line procedures while *improved space mapping* costs $O(n_o) + O(m \cdot n_o)$. It has an extra cost from on-line shifting. However, since the target object is always much smaller than $\mathcal{W}$ space, n_o cannot be very large and we can expect that there is little loss in the on-line procedure of *improved space mapping*.

I implement the *improved space mapping* algorithm with different n_g and different m to better view its time and storage complexity. This implementation uses the same settings as *raw space mapping* in Fig.5.16. Its result is shown in Fig.5.18. Readers may compare Fig.5.16 and Fig.5.18 to better compare the their performances.

The result is coherent with our analysis that the time cost and storage cost only depend on m. When $2m + 1$ is 36, time cost is $0.002s$. When $2m + 1$ changes to 72, time cost changes to 0.006 $0.008s$. The storage cost is always smaller than 1KB and it is a satisfying cost. There is no direct relationship between n_g and the costs. It is different from where the cost is $O(m \cdot n_g)$ which heavily depends on n_g. In the *improved space mapping*, all costs depend on m. Suppose that n_g is chosen to be $150 \times 150 = 22500$, then, the time cost and storage cost of one finger formation is as low as $0.007s$ and less than 1KB respectively.



n_g	m ($2m$+1)	Time (in second)	Storage size (in megabytes)
70x70=4900	18 (2x18+1=37)	0.002s	< 1KB
100x100=10000	18 (2x18+1=37)	0.002s	< 1KB
100x100=10000	36 (2x36+1=73)	0.006s	< 1KB
120x120=14400	36 (2x36+1=73)	0.008s	< 1KB
150x150=22500	36 (2x36+1=73)	0.007s	< 1KB
200x200=40000	36 (2x36+1=73)	0.007s	< 1KB
300x300=90000	36 (2x36+1=73)	0.008s	< 1KB
400x400=160000	36 (2x36+1=73)	0.006s	< 1KB

Figure 5.18: Time and storage cost of improved space mapping for caging test.

Even if we pre-build the mappings for 1000 finger formations. The total time cost would be $0.007\times1000s=7s$ and the total storage cost would be smaller than 1KB×1000=1MB. That is a satisfying value.

5.3.2 Caging test with the improved space mapping

5.3.2.1 Algorithm flow and analysis

The shifting is an on-line procedure. When the grids occupied by the target objects are detected, caging test can be performed by recover and update $\mathcal{C}^{\mathsf{frm}}$ through the improved space mapping. Specifically, caging test algorithm labels all the voxels in the mapping of all occupied grids with "cyan" color. These "cyan" voxels become elements of $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$. Then, the labeled $\mathcal{C}^{\mathsf{frm}}$ are labeled again according to their states and connectivity. The $\mathcal{C}^{\mathsf{frm}}$ may be separated into several different components which follow our analysis in the beginning of this Chapter. The $\bigcup_{i=1}^{u} \mathcal{C}^{\mathsf{frm}}_{\mathsf{fc}_i}$ components would be labeled with "red" color while the $\mathcal{C}^{\mathsf{frm}}_{\mathsf{ff}}$ component would not be labeled. This is exactly the same as Fig.5.14. If one component fulfills expression (5.1), namely it is (1) in unobstructed state and enclosed by a surrounding obstructed component, (2) initial configuration of the given formation is inside that component. Otherwise, vice versa.

Fig.5.19 shows details of caging test with improved space mapping. Comparing with the

original space mapping in Fig.5.14, the flow chart in Fig.5.19 involves the on-line shifting procedure which is emphasized with shadowed green box.

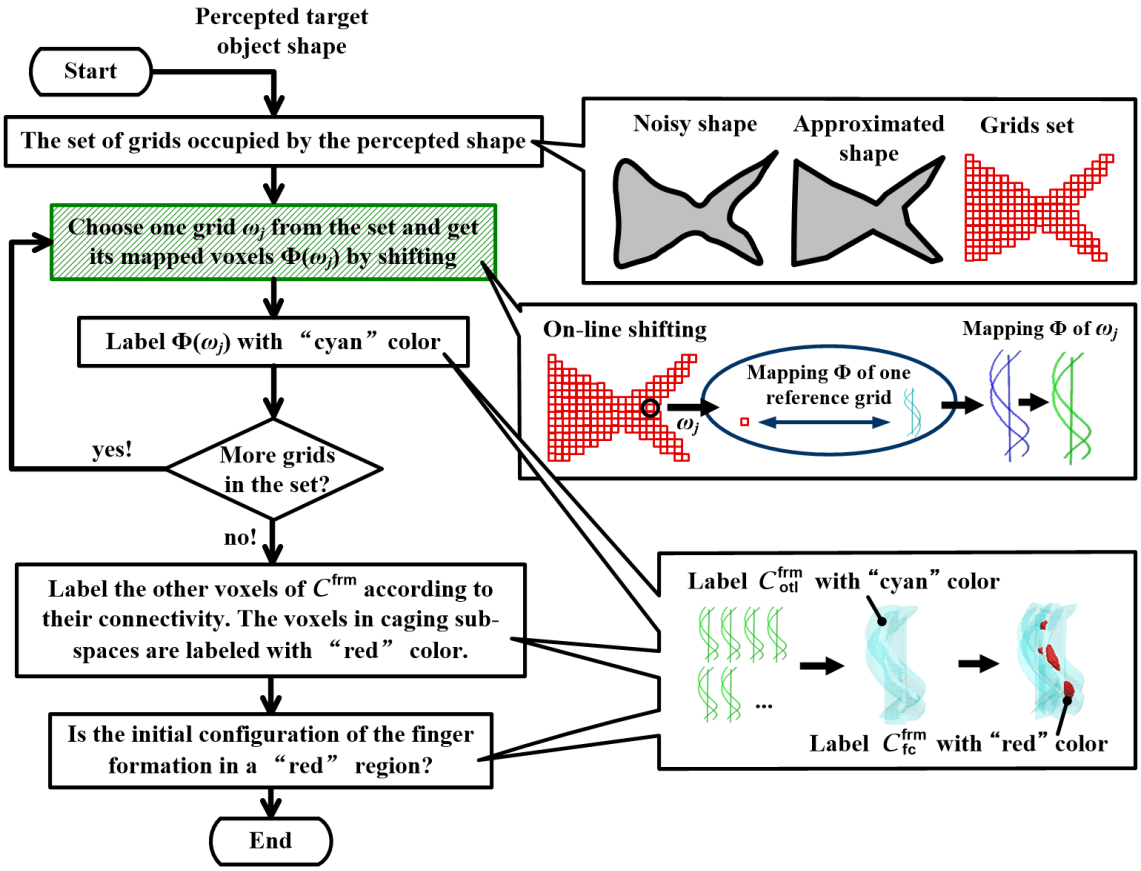


Figure 5.19: The flowchart of caging test with improved space mapping.

Note that the labeling procedure, especially the one which labels "red" sub-spaces out of $\mathcal{C}^{\text{frm}}$, could be most time-consuming. My algorithm scans every grids inside the sub-space enclosed by the "cyan" voxels. It costs $O(n_f \cdot m \cdot n_o)$=$O(m \cdot n_o)$ when the number of fingers is constant.

Since both mappings have the same cost of labeling, there are not too much difference in their time cost. The most significant merit of *improved space mapping* is that *improved space mapping* has much smaller of storage cost. It is $O(m)$ and it is much smaller than the $O(m \cdot n_g)$ of *raw space mapping*. This merit reduces the storage cost of mapping and makes it possible to pre-build mappings for many different formations.

Actually, besides the small storage cost, *improved space mapping* has another merit. That is, we can make everything on-line in *improved space mapping*. As shown in Fig.5.17, the off-line procedure of *improved space mapping* is $O(m)$. Even if we do it on line, the total time cost is as small as $O(m + n_o)$. It only has 1st-order complexity and it is an acceptable

value form on-line processing. The merit of making everything on line is key to different formations. **It helps to decouple caging test algorithms from pre-built mappings.** Given a finger formation and a target object, we no longer need the pre-building procedure and we can test whether the given finger formation can cage the target object with total time cost of $O(m + n_o + m \cdot n_o)$=$O(m \cdot n_o)$.

The results of Fig.5.18 is coherent with our analysis. Since the time cost is smaller than $0.01s$, we can use the *improved space mapping* as an on-line proedure. *Improved space mapping* significantly reduces storage cost while maintains time efficiency. It can not only decouple caging test algorithms from specific target objects but also decouple caging test algorithms from specific finger formations. Using *improved space mapping*, we can perform rapid caging test against many various finger formations and various target objects without worrying about time and storage cost.

5.3.2.2 Implementation with three representative finger formations

Based on the flowchart introduced Fig.5.19, I performed caging test with three different formations and some different target objects. The input of this problem are (1) one object and (2) three finger formations. The output of this problem is which formation can cage the object. The three formations are based on some eigen-shapes of a Barrett hand. The idea of eigen-shape is developed by Prof. Peter K. Allen and his students. It is published in a series of papers [Miller et al., 2003][Ciocarlie et al., 2007][Ciocarlie and Allen, 2009]. It is also referred to by some another active researchers in robotic grasping [Ekvall and Kragic, 2007] [Jonathan Weisz, 2012]. Especially, [Jonathan Weisz, 2012] used the alike idea to simplify the mechanical complexity of a robotic hand and it an interesting exploration into robotic design. Eigen-shape claims that most shapes of a dexterous hand are redundant to real applications. Only some essential shapes are useful and they name them eigen-shapes. The eigen-shapes are considered to be able to cover most grasping tasks. If we use these eigen-shapes to cage target objects, we may save ourselves from recovering, updating and labeling thousands of $\mathcal{C}^{\mathsf{frm}}$. We only need to perform caging test with the eigen-shapes, namely some given finger formations. The three formations that will be used in the following part are based on the idea of eigen-shape. It is the three most useful finger formations of a Barrett hand. Fig.5.20 illustrates them and their mappings with respect to one reference grid ω_1.

The positions of each finger and dimensions of each formation have been attached below the left figures of Fig.5.20(a), (b) and (c). Note that in Fig.5.20, the mapping is built by setting n_g=150×150 and m=36 ($2m + 1$=73).

Implementation I: Correspondence between $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc}}$ and $\mathcal{W}$

Before presenting the results of various objects, let us first take a target object shown in Fig.5.21(a) for example and see the correspondences between $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc}}$ and $\mathcal{W}$ space. Here, we use the two-finger formation F_1 shown in Fig.5.20(a) to recover and update the $\mathcal{C}^{\mathsf{frm}}$ of a concave target object. The $\mathcal{W}$ space is discretized into 150×150 grids while the $\mathcal{C}^{\mathsf{frm}}$ is discretized into $150 \times 150 \times 73$ voxels. I will use these settings continuously in later parts.

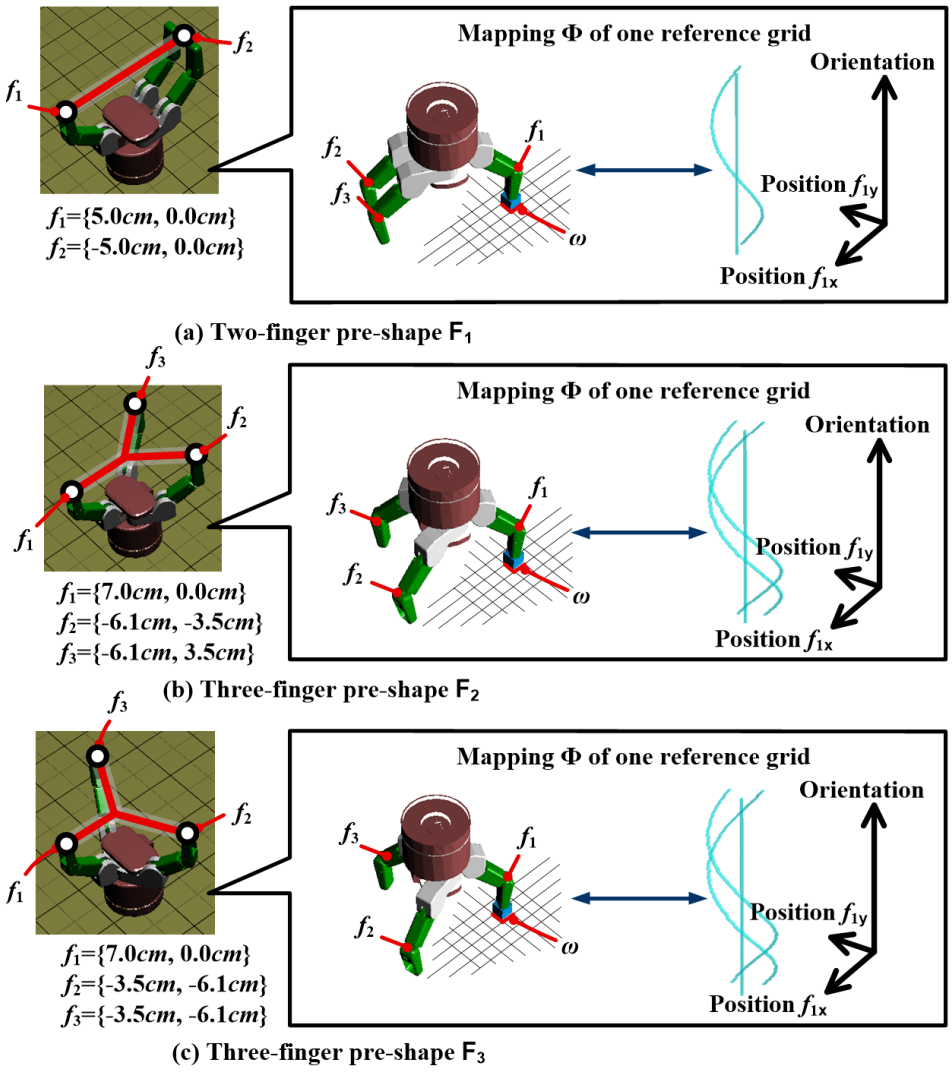


Figure 5.20: The three eigen-formations of Barrett hand and their mappings.

There is no specific reason why I chose these settings. We can set them to any satisfying value as long as they do not result into high cost in the labeling procedure.

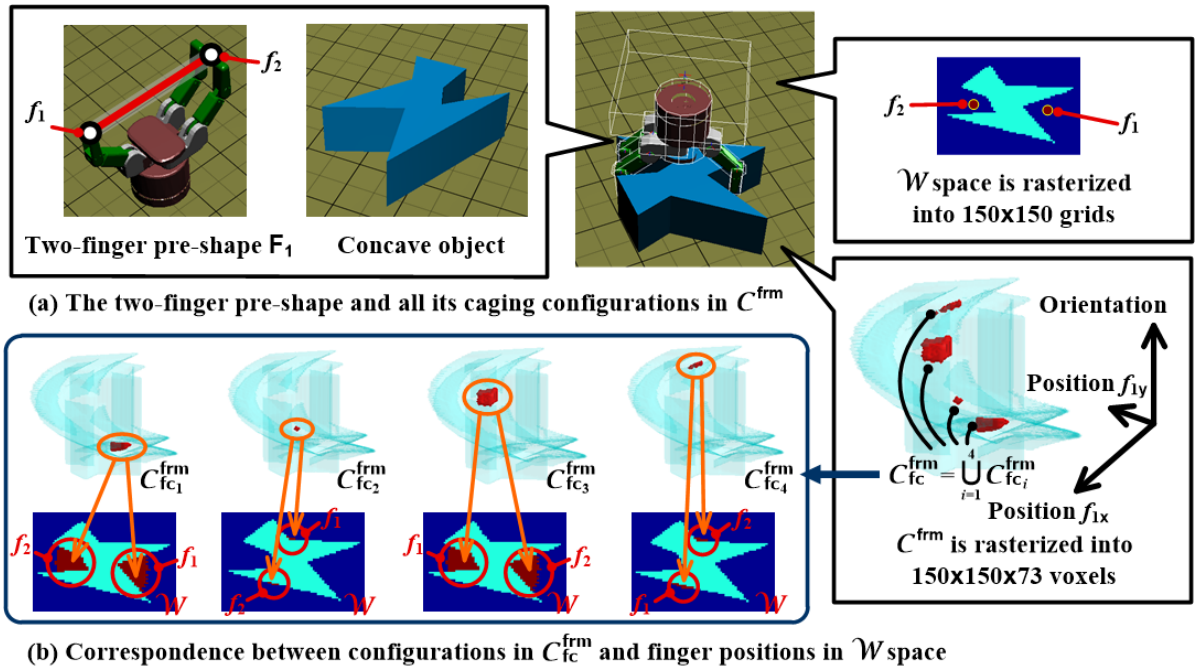


Figure 5.21: Caging test with improved space mapping – Concave object.

The middle part of Fig.5.21(a) shows a case where the formation of fingers is applied to a Barrett hand and it is actuated to cage the target object. In this case, the two finger positions with respect to the target object is shown in the upper-right dialog box of Fig.5.21. Those two red points denote the positions of the two fingers while the cyan sub-space denotes the grids occupied by the target object. Below this upper-right dialog box I shows the recovered and updated $\mathcal{C}^{\mathsf{frm}}$. The $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc}}$ of this $\mathcal{C}^{\mathsf{frm}}$ has four components, namely $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc}}=\bigcup_{i=1}^{4}\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc_i}}$. When the initial configuration of the formation is given, we can check whether it could cage the target object by testing whether it is in one of these $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc_i}}$.

Reversely speaking, this algorithm not only offers the possibility of testing whether an initial configuration of the formation could cage the target object but also offers the possibility of finding all configurations of the formation that can cage the target object. These "all configurations" are the configurations in these four $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc_i}}$. The details of them are shown in Fig.5.21(b).

Each configuration in a $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc_i}}$ corresponds to a set of two-finger formations. Each two-finger formation is rendered as two red points shown in the upper-right dialog box. Therefore, the correspondent finger positions of formation configurations in each $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc_i}}$ are rendered into red sub-spaces in the four lower figures of Fig.5.21(b). The number of voxels in $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc}_1}$, $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc}_2}$, $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc}_3}$, $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc}_4}$ are 478, 7, 393 and 22 respectively. That means there are totally 478+7+393+22=900

different caging configurations. They in $\mathcal{W}$ space are all the possible caging positions of the given finger formation.

In addition to the example in Fig.5.21, Fig.5.22 gives another example with the three-finger formation F_2 shown in Fig.5.20(b) and a convex target object. In this case, the $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc}}$ of this $\mathcal{C}^{\mathsf{frm}}$ has three components, namely $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc}}=\bigcup_{i=1}^{3}\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc_i}}$. Like Fig.]refcfrmexample, the correspondent finger positions of formation configurations in each $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc_i}}$ are rendered into red sub-spaces in the three lower figures in the right part of Fig.5.22. The number of voxels in $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc_1}}$, $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc_2}}$, $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc_3}}$ are 309, 168 and 29, respectively. That means there are totally 309+168+29=605 different caging configurations under settings n_g=150 × 150 and m=36 ($2m+1$=73). They in $\mathcal{W}$ space are all the possible caging positions of F_2.

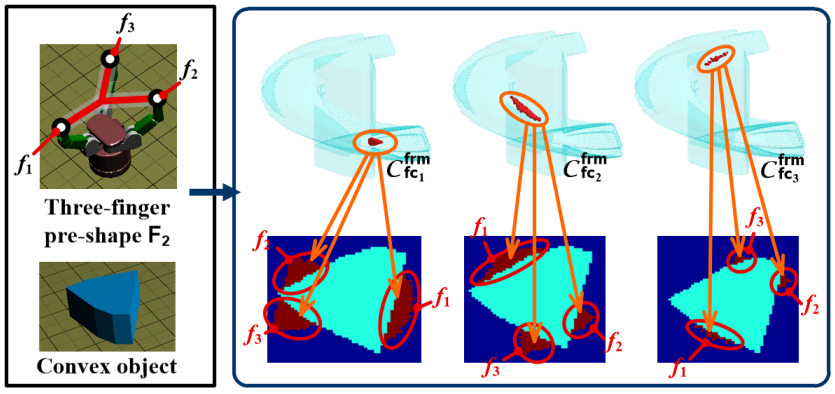


Figure 5.22: Caging test with improved space mapping – Convex object.

Implementation II: Performance on different objects

Next, let us compare the performance of the caging test algorithm with improved space mapping on different target objects. The target objects are tested in two ways. One is to perform caging tests with four different target objects and the three pre-defined finger formations while the other one is to perform caging tests with changing finger formation or changing target object shapes.

In the first group of caging test, the caging test with improved space mapping is performed on four different target objects. They involve O_1: a grippable concave object, O_2: a non-convex object, O_3: a convex polygon and O_4: a circular object. The rows of Fig.5.23 illustrate the shapes of these target objects, their occupied number of grids n_o and their time and storage costs.

Object O_1: {{10.0, 10.0}, {2.0, 0.0}, {10.0, -10.0}, {0, -4}, {-10.0, -10.0}, {-1.0, 0.0}, {-10.0, 10.0}, {10.0, 10.0}}

Object O_2: {{15.0, 15.0}, {4.0, 0.0}, {10.0, -10.0}, {-3.0, 0.0}, {15.0, 15.0}}

Object O_3: {{7.0, 7.0}, {9.0, 3.0}, {5.0, -4.0}, {0.0, -7.0}, {-5.0, -1.0}, {-3.0, 7.0}, {7.0, 7.0}}

Object O_4: *center* = {0.0, 0.0}, *radius* = 7.0

As is shown in Fig.5.23, the first finger formation F_1 shown in Fig.5.20(a) could cage O_1. The second finger formation F_2 shown in Fig.5.20(b) could cage O_1, O_3 and O_4. The third finger formation F_3 shown in Fig.5.20(c) could also cage O_1, O_3 and O_4.

The mapping cost of these caging tests are always smaller than $0.01s$. This is the same as our analysis in Fig.5.17 and Fig.5.18. It only depends on m and it has no relationship with n_o. Shifting cost is $O(m \cdot n_o)$, therefore it changes as n_o differs. Although shifting cost is larger than mapping cost, it can be done in less than $0.2s$. The most time consuming procedure is labeling. It costs more than $0.5s$ in the worse case which appears with F_2 and O_4. We can see from Fig.5.17 that with the parameter settings $150 \times 150 \times 73$, the total cost of mapping plus shifting plus labeling can be performed in less than $0.7s$. This is a satisfying value to performed on line.

Note that none of the given pre-shapes can cage O_2. However, in our intuition, as the inter-finger distance of 3-finger formation decreases, O_2 shall be caged. We should introduce more formations and perform more caging tests to get a successful finger formation that can cage O_2. We do this with the second group of caging test, namely performing caging tests with changing finger formation or changing target object shapes.

Fig.5.24 shows the changes of $\mathcal{C}^{\mathsf{frm}}$ with O_2 and a series of varying finger formations based on F_3. The object can be caged by formation variation 3 and formation variation 4 where their inter-finger distances become much smaller comparing with F_3.

We also performed caging test with a series of changing objects. Fig.5.25 shows the results. In this case, the target shape is deforming dynamically while the finger formation is fixed to pre-shape F_3. The caging test algorithm with improved space mapping is fast enough to response to the changing object shapes. The target objects in variation 1, variation 2 and variation 3 of Fig.5.25 can be caged by F_2 while the last two target objects cannot.

Implementation II: Real-world objects

Finally, I implement the algorithm with a miscellany of real-world objects for further appreciation. The inputs of this implementation are pictures of each target object. These pictures are shown in Fig.5.26. Readers can assume that the perception device in this case is a camera while the percepted target object shape is those pictures. The pictures are processed and approximated for caging tests.

Below each picture of Fig.5.26, I attached the result of caging tests. Note that in these results, there are two extra operations. (1) all hollow holes are filled up. For example, the target object #6 has a hollow hole. However, the approximated shape which is shown in the lower-left block of Fig.5.27 does not save this hollow hole. It is filled up. (2) Only finger formations F_2 and F_3 are tested. The two-finger formation F_1 is not used for caging tests here because it can cage none of these target objects when their hollow holes are filled up.

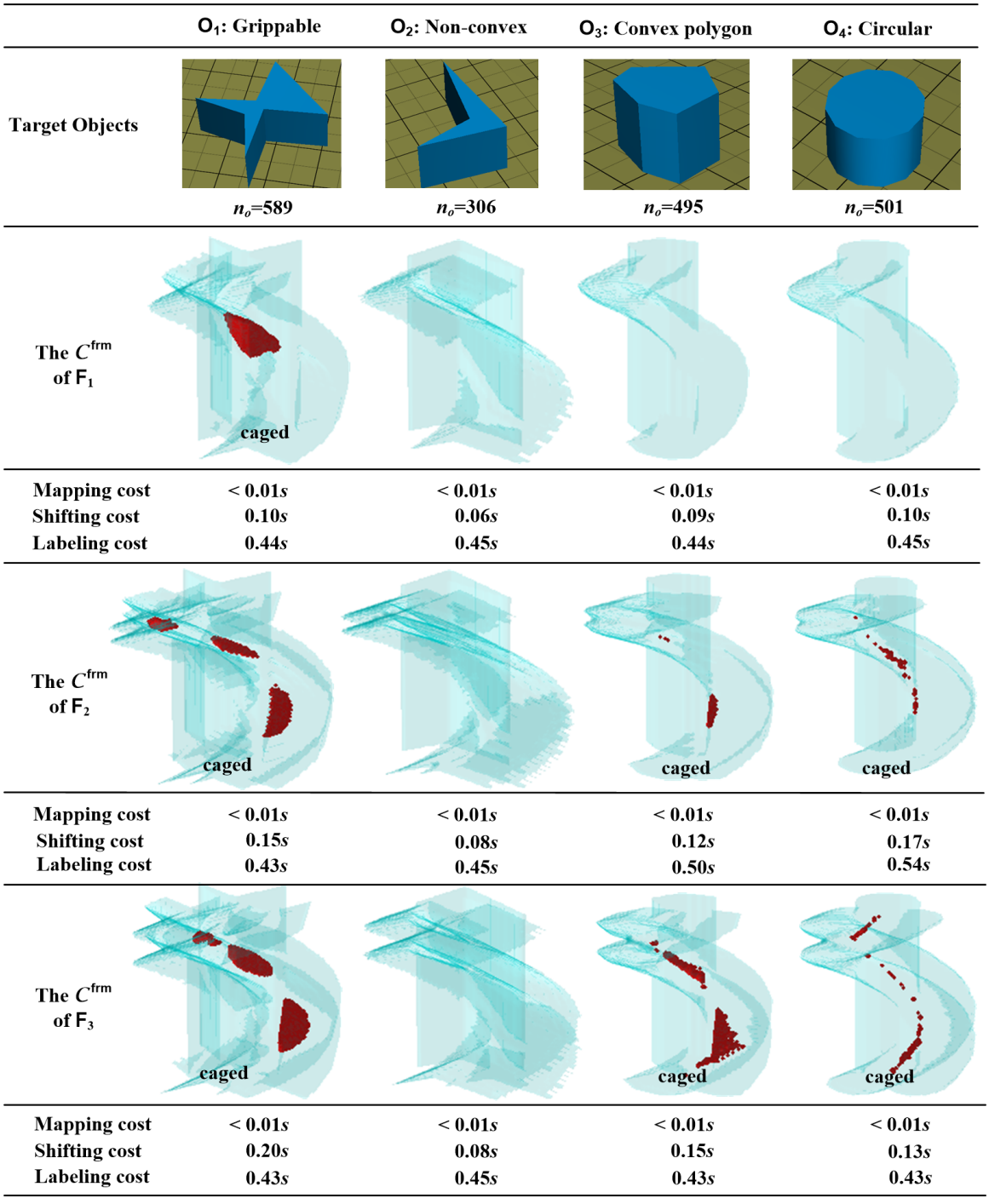


Figure 5.23: Caging results with different objects.

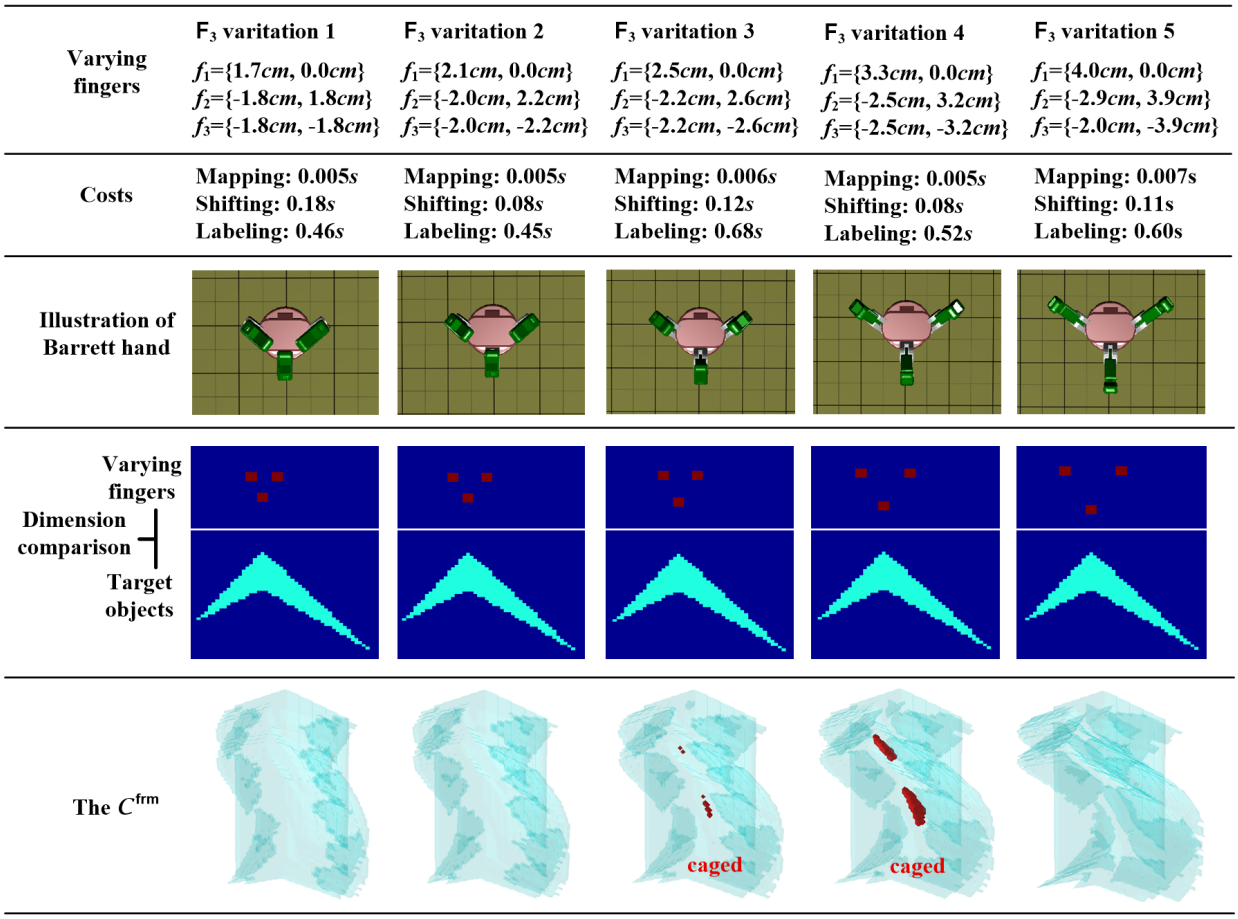


Figure 5.24: Caging test with improved space mapping – Changing formation.

Fig.5.27 shows in detail some results of Fig.5.26. In each block of Fig.5.26, I shows the dimension of the finger formation together with the dimension of the target object so that readers can better compare them. The correspondent finger positions in $\mathcal{W}$ space are rendered with yellow points. These yellow points are the same as the finger positions in lower part of Fig.5.21 and Fig.5.22. I rendered them with yellow points instead of red grids to better illustrate each finger in a formation. In each block of Fig.5.27, $\boldsymbol{f_1}$ is rendered with light yellow color while the other fingers are rendered with dark yellow. Note that only the first caging sub-space, namely $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc}_1}$, and their correspondent yellow finger positions are rendered in Fig.5.27.

Surely the caging test algorithm introduced here can work with hollow objects. Fig.5.28 shows some of the caging results of target object #6 with finger formation F_1 when the hollow hole of target object #6 is not filled up. In this case, F_1 can always cage target object #6. Note that in this figure, only the finger configurations inside the blue circle of Fig.5.28(a) are shown. The configurations inside the blue circle has three components. One is shown

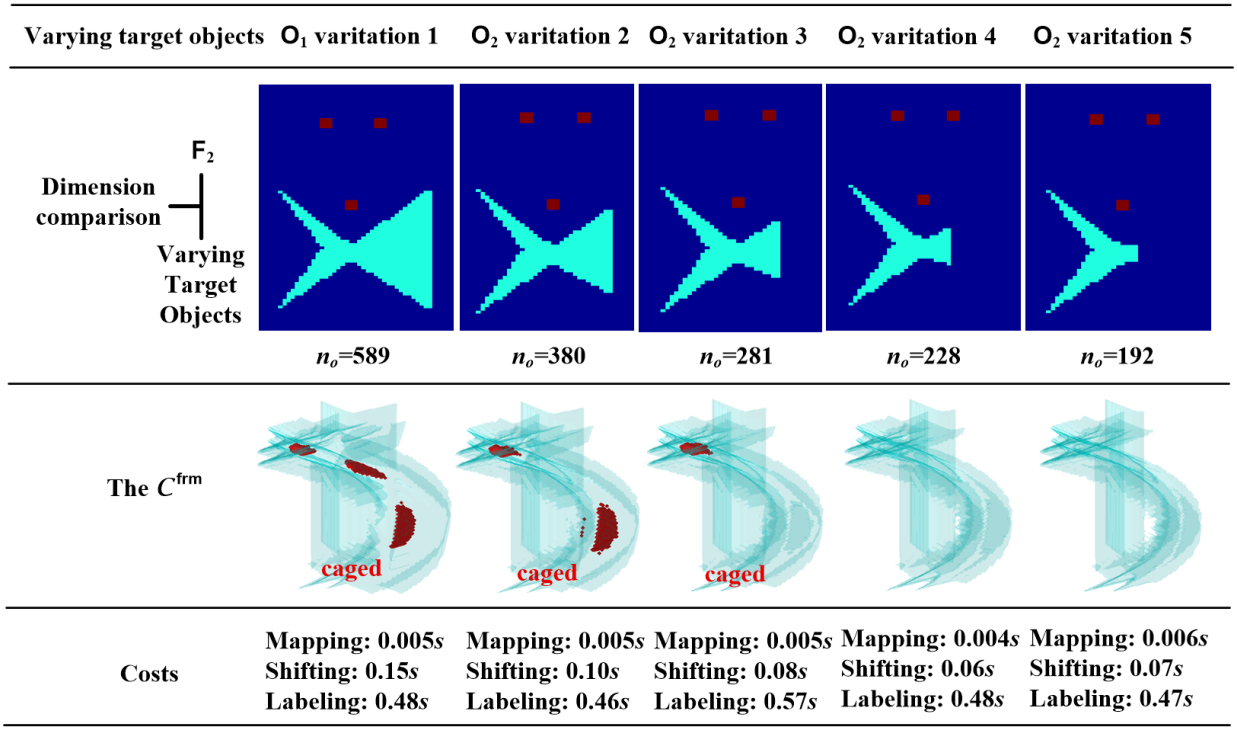


Figure 5.25: Caging test with improved space mapping – Changing target object.

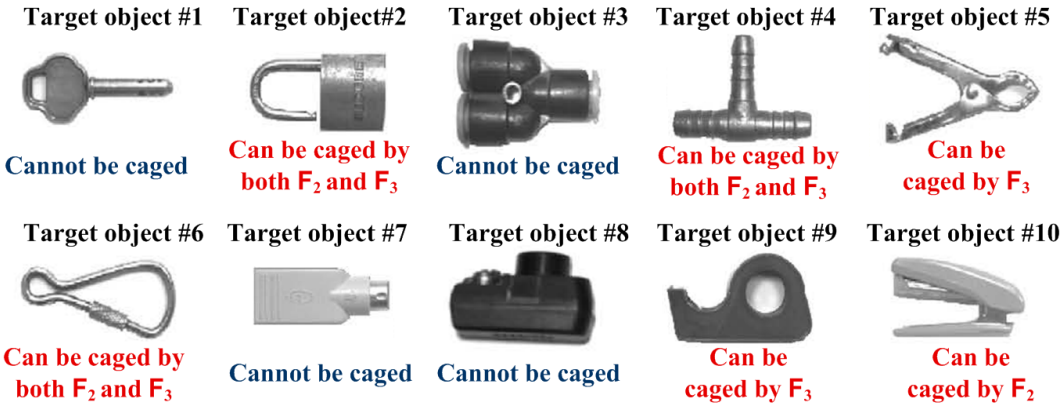


Figure 5.26: Real-world objects in pictures.

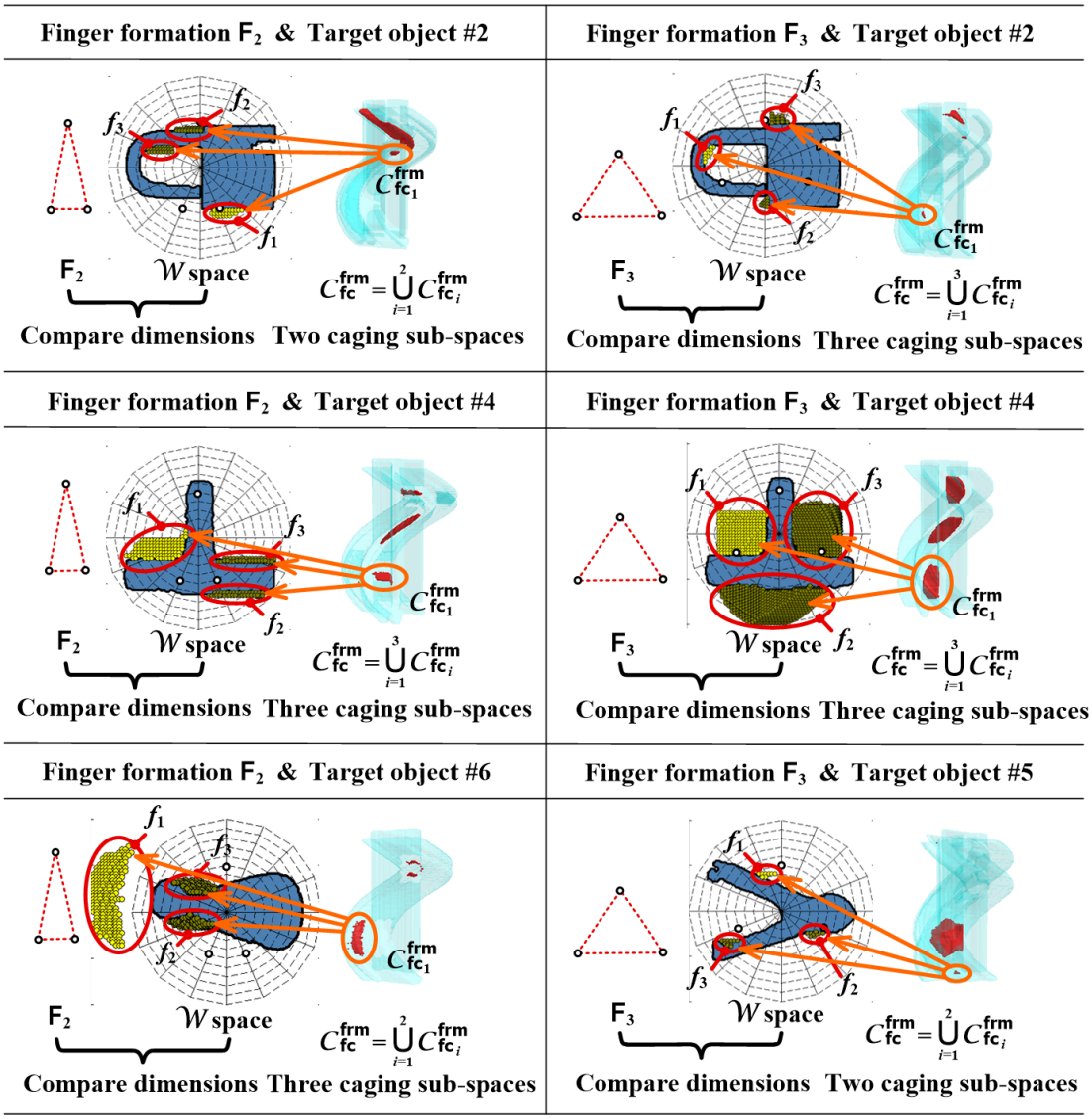


Figure 5.27: Details of some results in Fig.5.26.

in Fig.5.28(b-1), the other two are shown in Fig.5.28(b-2) and (b-3). Readers may notice that the components in Fig.5.28(b-2) and (b-3) should belong to the same component while they are divided into two parts in my results. That is because the orientation axis of $\mathcal{C}^{\mathsf{frm}}$ is set to $[-\pi, \pi)$. The lower coordinates and the upper coordinates of the orientation axis are actually connected with each other periodically and Fig.5.28(b-2) and (b-3) do belong to the same component. The $\mathcal{W}$ space of Fig.5.28(b-2) and (b-3) give a clearer illustration of the periodical connections.

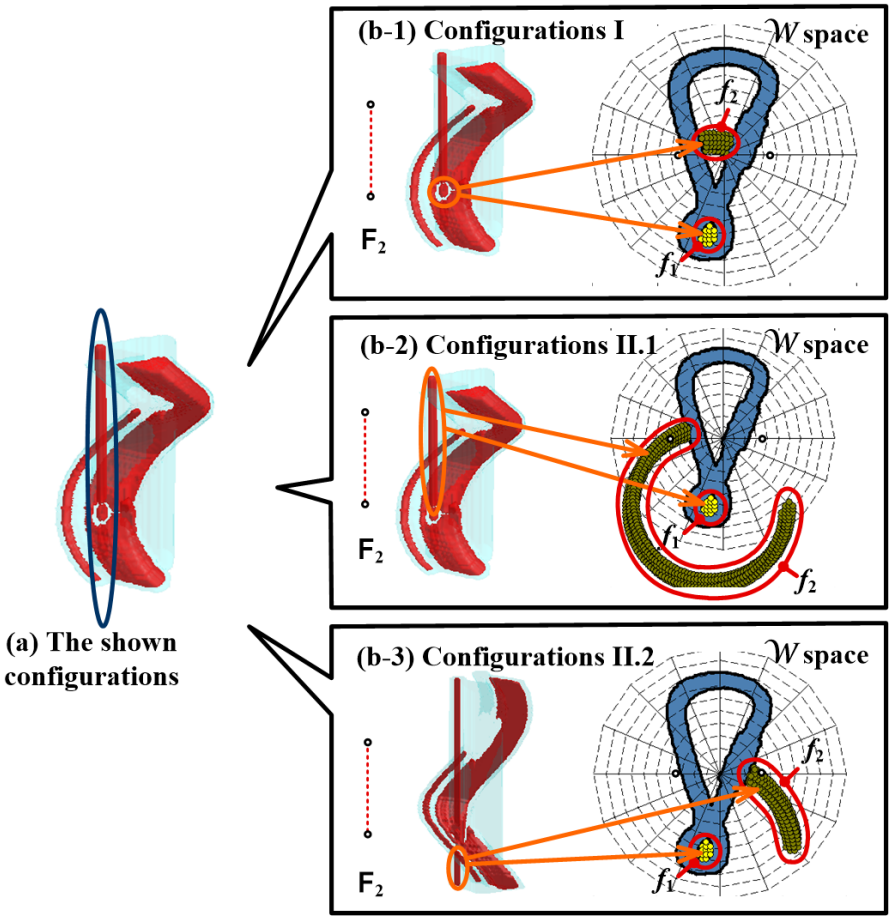


Figure 5.28: The result of target object #6 when hollow holes are taken into account.

Now we can have a satisfying solution to **caging test**. Nevertheless, that only the first step. We have discussed in the beginning of this thesis that besides **caging test** we need to (1) find a set of finger formations that could cage the target object and (2) develop a **robust caging** algorithm to find an optimized formation of fingers that could be most robust to endure uncertainties. It is delightful that we have got the finger formation sets. They are the red grids in lower part of Fig.5.21 and Fig.5.22 or the yellow points in Fig.5.27. These sets can be obtained efficiently and they can be further employed in step (2). Our next step

would be to the step (2), namely to find an optimized one from these sets. I will propose the algorithms in the next part.

5.4 Robustness of Caging in $\mathcal{C}^{\mathsf{frm}}$

5.4.1 Quality function and the robust caging algorithm

When we were finding an optimized three-finger caging in section 3.2.6, I proposed a two-step solution. Namely, (1) finding an optimized three-finger immobilization with the immobilization optimization algorithm and (2) retracting fingers to obtain certain robustness to avoid collisions. Following this idea, I propose to define the robustness of caging in $\mathcal{C}^{\mathsf{frm}}$ with two measurements. The first one is the "distance to escaping" while the second one is the "distance to collision".

In order to calculate the two measurements, we need to again analyze $\mathcal{C}^{\mathsf{frm}}$. We have known in expression (3.1) that this sub-space, when caging is obtained, can be divided into the free sub-space $\mathcal{C}^{\mathsf{frm}}_{\mathsf{ff}}$, the caging sub-space $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc}}$ and the obstacle sub-space $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$. When finger configurations fall into $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc}}$, they cage the target object. When finger configurations fall into $\mathcal{C}^{\mathsf{frm}}_{\mathsf{ff}}$, they neither cage nor collide with the target object. Fingers at configurations in this sub-space can go through or escape from the target object. When finger configurations fall into $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$, the fingers collide with target objects. Fig.5.29 illustrates these descriptions.

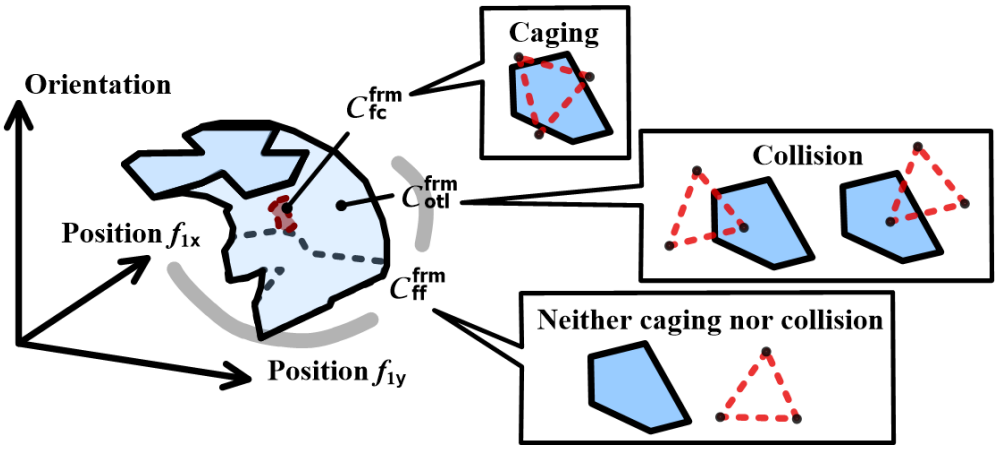


Figure 5.29: The meanings of configurations in different sub-spaces.

There is nothing new in Fig.5.29. It is simply a reverse view of the construction procedure. $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc}}$, $\mathcal{C}^{\mathsf{frm}}_{\mathsf{ff}}$ and $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$ have these meanings because we built them in this way. Readers may review section 5.1.2 to refresh it.

What really concern are critical configurations between these sub-spaces. The critical configurations, or namely the voxels on surfaces between the sub-spaces $\mathcal{C}^{\mathsf{frm}}_{\mathsf{fc}}$, $\mathcal{C}^{\mathsf{frm}}_{\mathsf{ff}}$ and $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$, are important to the measurement of robustness. Fig.5.30 shows the meanings of these

furfaces. Here I denote the surface between $\mathcal{C}_{\mathsf{fc}}^{\mathsf{frm}}$ and $\mathcal{C}_{\mathsf{otl}}^{\mathsf{frm}}$ with symbol $\mathcal{S}_{\mathsf{cln}}^{\mathsf{frm}}$ while denote the surface between $\mathcal{C}_{\mathsf{otl}}^{\mathsf{frm}}$ and $\mathcal{C}_{\mathsf{ff}}^{\mathsf{frm}}$ with symbol $\mathcal{S}_{\mathsf{esp}}^{\mathsf{frm}}$. Configurations in $\mathcal{C}_{\mathsf{fc}}^{\mathsf{frm}}$ means caging while configurations in $\mathcal{C}_{\mathsf{otl}}^{\mathsf{frm}}$ means collision. Therefore, configurations on the surface between $\mathcal{C}_{\mathsf{fc}}^{\mathsf{frm}}$ and $\mathcal{C}_{\mathsf{otl}}^{\mathsf{frm}}$, namely $\mathcal{S}_{\mathsf{cln}}^{\mathsf{frm}}$, means the critical condition of collision. Likewise, configurations in $\mathcal{C}_{\mathsf{otl}}^{\mathsf{frm}} \cup \mathcal{C}_{\mathsf{fc}}^{\mathsf{frm}}$ means either collision or caging while configurations in $\mathcal{C}_{\mathsf{ff}}^{\mathsf{frm}}$ means breaking of caging. Therefore, configurations on the surface between $\mathcal{C}_{\mathsf{otl}}^{\mathsf{frm}}$ and $\mathcal{C}_{\mathsf{ff}}^{\mathsf{frm}}$, namely $\mathcal{S}_{\mathsf{esp}}^{\mathsf{frm}}$, mean the critical condition of escaping.

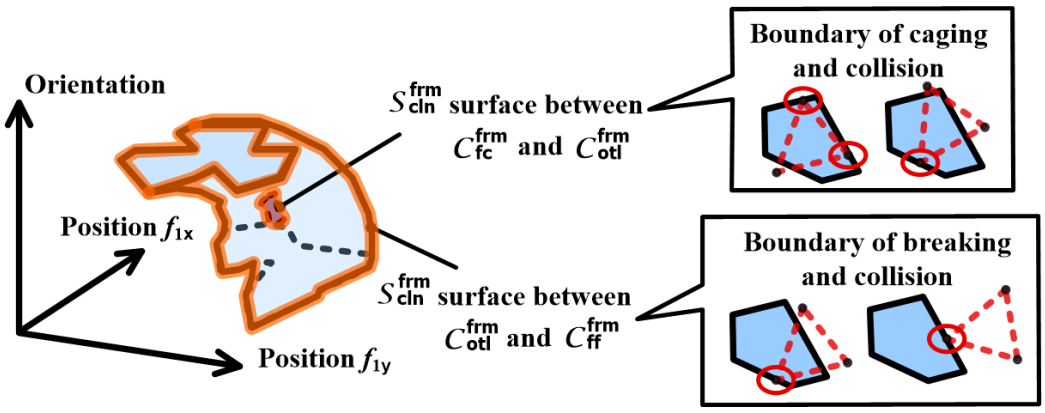


Figure 5.30: The meanings of configurations on critical surfaces.

Note that the case in Fig.5.30 is a quantitative illustration and there is only one caging sub-space. In real cases, there could be many $\mathcal{C}_{\mathsf{fc}_i}^{\mathsf{frm}}$ so that the surfaces between $\mathcal{C}_{\mathsf{fc}_i}^{\mathsf{frm}}$ and $\mathcal{C}_{\mathsf{otl}}^{\mathsf{frm}}$ changes from $\mathcal{S}_{\mathsf{cln}}^{\mathsf{frm}}$ to $\mathcal{S}_{\mathsf{cln}_i}^{\mathsf{frm}}$ correpondingly.

Since the two surfaces are critical conditions of collision and caging breaking, given a configuration $\boldsymbol{q}^{\mathsf{frm}}$, $\boldsymbol{q}^{\mathsf{frm}} \in \mathcal{C}_{\mathsf{fc}_i}^{\mathsf{frm}}$, we can calculate the two measurements by measuring its distance to the two surfaces. The following expressions show the two measurements. I denote them with δ_{cln} and δ_{esp}.

$$
\begin{aligned}
&\textit{for all } \mathbf{q}_{\mathsf{cln}_j}^{\mathsf{frm}} \in \mathcal{S}_{\mathsf{cln}_i}^{\mathsf{frm}} \textit{ and } \mathbf{q}_{\mathsf{esp}_k}^{\mathsf{frm}} \in \mathcal{S}_{\mathsf{esp}}^{\mathsf{frm}}, \\
&\delta_{\mathsf{cln}} = \mathsf{min}((\mathbf{q}^{\mathsf{frm}} - \mathbf{q}_{\mathsf{cln}_j}^{\mathsf{frm}})^{\mathrm{T}} \mathrm{W} (\mathbf{q}^{\mathsf{frm}} - \mathbf{q}_{\mathsf{cln}_j}^{\mathsf{frm}})) \\
&\delta_{\mathsf{esp}} = \mathsf{min}((\mathbf{q}^{\mathsf{frm}} - \mathbf{q}_{\mathsf{esp}_k}^{\mathsf{frm}})^{\mathrm{T}} \mathrm{W} (\mathbf{q}^{\mathsf{frm}} - \mathbf{q}_{\mathsf{esp}_k}^{\mathsf{frm}}))
\end{aligned}
\tag{5.9}
$$

In other words, given a configuration, we measure the "distance to collision" by calculating the minimum distance between this configuration and the surface of the caging sub-space that this configuration belongs to. That is where caging becomes collision. Likewise, given a configuration, we measure the "distance to escaping" by calculating the minimum distance between this configuration and the critical boundary to $\mathcal{C}_{\mathsf{ff}}^{\mathsf{frm}}$. That is where fingers escape.

Moreover, we employ a weighing matrix W in expressions (5.9) to indicate the complicate metrics of the two measurements. Of course, W should follow basic definition of metrics. How to define the metrics have no strict criterion and it remains an open problem. We

can even treat horizontal metrics and vertical metrics differently like the decomposition of translational caging and rotational caging in $\mathcal{C}^{\mathsf{obj}}$. In my implementation, I simply choose W as $diag(1,1,1)$. In that case, the measurement degenerates into simple Euclidean norms $||\mathbf{q}^{\mathsf{frm}} - \mathbf{q}^{\mathsf{frm}}_{\mathsf{cln}_j}||$ and $||\mathbf{q}^{\mathsf{frm}} - \mathbf{q}^{\mathsf{frm}}_{\mathsf{esp}_k}||$. For more pragmatic usage, W should be chosen according to mechanisms of specific robotic systems. For instance, a hand on a Cartesian robot should have higher weighing parameter on translational measurement than a hand on SCARA robot since Cartesian robots have lower tendency of rotation. The W parameter is inherent to a specific robotic system.

The next step after obtaining the two measurements is to combine them and define the quality function of robustness. This step also suffers from ambiguous parameter settings. Generally speaking, each measurement should has a weighing parameter like the w_{cln} and w_{esp} in expression (5.10).

$$Q^{\mathsf{frm}}_q = w_{\mathsf{cln}}\delta_{\mathsf{cln}} + w_{\mathsf{esp}}\delta_{\mathsf{esp}} \tag{5.10}$$

Different weighing parameters lead to different performance. In my implementation, I simply choose $w_{\mathsf{cln}} : w_{\mathsf{esp}}$=1 : 1. For more pragmatic usage, $w_{\mathsf{cln}} : w_{\mathsf{esp}}$ should be chosen according to requirements of applications. For instance, if we are performing a grasping task rather than multi-robot cooperation, we may concern more about collision. That is to avoid collision with target objects as much as possible during grasping even if there are certain perception errors. In that case, we may set a large value to w_{cln}. I will discuss in detail the importance of w_{cln} and w_{esp} in the implementation and analysis section.

Fig.5.31 illustrates the flowchart of finding a robust caging based on expression (5.10). The basic idea of this flow chart is to traverse all voxels in all caging sub-spaces, namely compare all the caging configurations in the caging sets to find an optimized one.

There are two remaining problems about this algorithm. The first one is its computational complexity. The second one is how to extend to multiple finger formations. These two problems have something in common. That is, we must make the cost of this algorithm as small as possible to have it work with as many finger formations as possible. However, the computational efficiency of this traversing algorithm is sometimes quite low. In the extreme case, it could be as high as $O(n_o \cdot m \cdot n_s)$ where n_s is the number of voxels on the surface of $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}} \cup \mathcal{C}^{\mathsf{frm}}_{\mathsf{fc}}$. n_s is nearly the same order as n_o and therefore the complexity could be written as $O(n_o \cdot n_o \cdot m)$ to avoid extra parameters. This is one drawback of this algorithm. Fortunately, the extreme case is rare. It only appears when most of the $n_o \cdot (2m+1)$ grids belong to caging sub-spaces. In most cases, the algorithm complexity is in the same order as $O(n_o \cdot m)$. Fig.5.32 gives the example of this rare and extreme case. It is actually the same hollow object #6 of last section. In this example, there are too many candidate caging configurations and it takes $122s$ to find a best configuration from this set. Readers may compare the other candidate sets in Fig.5.27. They are much smaller and costs less. It is easy to expect that the rare case happens only to a special series of concave or hollow objects. It can be decided in the following way. (1) Calculate the size of the target object. (2) Calculate the convex hull of the target object and calculate the size of this convex hull.

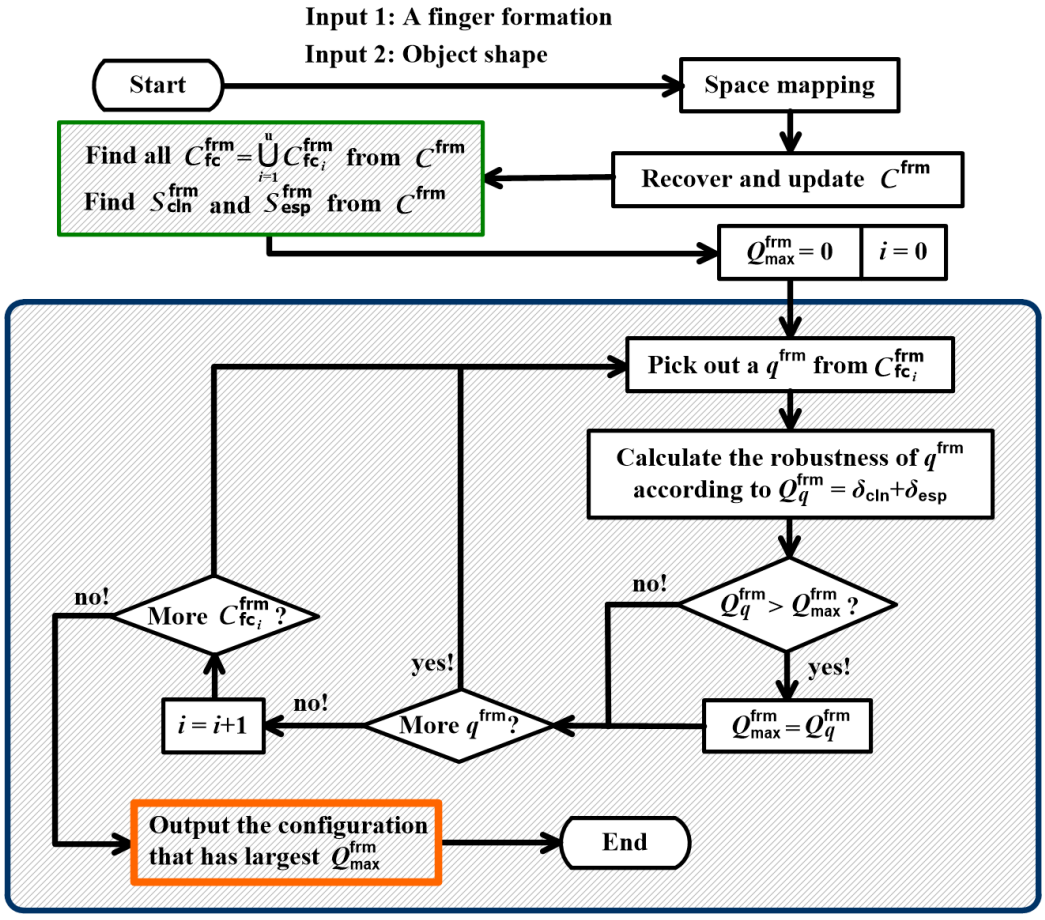


Figure 5.31: Flowchart of finding a robust caging configuration.

(3) Compare the two sizes, if the size of (1) is much smaller than that of (2). It might be time-consuming to find an optimized configuration. Note that the three steps are only rough estimation. It not suitable for pragmatic use.

In order to avoid the time-consuming exceptions, I introduce a threshold and to filter the cardinality of each $\mathcal{C}_{fc_i}^{frm}$. Firstly, I only choose the $\mathcal{C}_{fc_i}^{frm}$ with proper cardinality for calculation. That is to say, the green box of the flowchart in Fig.5.31 is changed into finding the $\mathcal{C}_{fc_i}^{frm}$ in a pre-defined range. Only the $\boldsymbol{q}^{frm}$ in those filtered $\mathcal{C}_{fc_i}^{frm}$ will saved as the candidate caging sets. If none of the cardinality of $\mathcal{C}_{fc_i}^{frm}$ is in the pre-defined range. The target object is supposed to be unsuitable for caging with the give formation. We may need to calculate with some other finger formations or reject ensuing tasks. This algorithm depends on the pre-defined range of set cardinality. If a practitioner would like to make complete evaluation of all caging candidates without worrying about time cost, he or she may set upper bound of the range to $+\infty$.

Fig.5.33 is the flowchart of a whole robust caging algorithm. It not only shows in detail

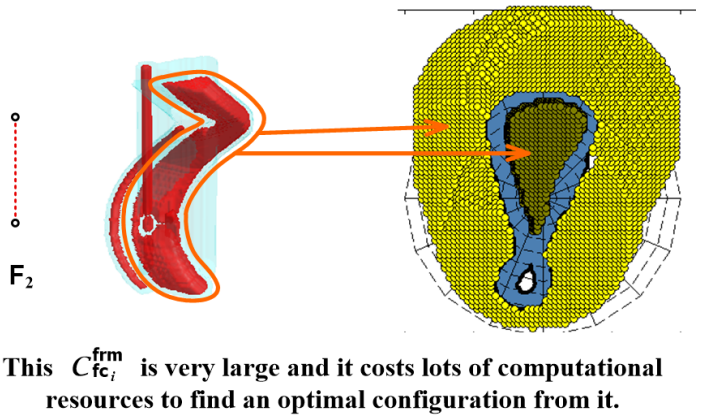


Figure 5.32: An extreme case which costs lots of computational resources.

the thresholds in finding one optimized configuration but also the whole procedure of deciding an optimized finger formation. The pre-defined range is named $(\tau_{\text{rng}_l}, \tau_{\text{rng}_r})$ in this figure. It is denoted by "Threshold 1" in Fig.5.33 and it can help avoid extreme high costs of resources. Besides this "Threshold 1", there is another "Threshold 2" in this flowchart. In the ideal case, the finger formation that has largest $\mathcal{Q}^{\text{frm}}$ will be chosen as an optimized finger formation while the $\boldsymbol{q}^{\text{frm}}$ that leads into this largest $\mathcal{Q}^{\text{frm}}$ will be chosen as the correspondent optimized formation configuration. However, the largest $\mathcal{Q}^{\text{frm}}$ might be too small and it is not suitable for caging. "Threshold 2" filters out the small $\mathcal{Q}^{\text{frm}}$. When the branch of "Threshold 2" goes to "no!", none of the finger formations can robustly cage the target object and the algorithm rejects ensuing tasks.

Readers might be interested in how to combine the caging test/caging optimization algorithms (or more generally caging planning algorithms) with the motion planning algorithms of manipulators. After all, a hand without arm (manipulator) is non-sense. However, this thesis is not dedicated to this combination.[1] It concentrates on the caging problems. If practitioners would like to combine, one potential solution is to insert rewards to the loops of Fig.5.33. The rewards could tune the selection of pre-defined finger formations and offer proper configurations to specific manipulators. In that way, the planning of caging could be extended to the planning of caging manipulation.

5.4.2 Implementations and analysis

Now let us see implementations of the algorithm shown in previous part. The implementations use the same platform and the same parameter settings as section 5.3.2.2. Since I have shown lots of results of convex objects in section 5.3.2.2, I won't discuss much about convex objects in this part. I will spend most of the texts here to concave and hollow objects and the performance of my quality function.

[1]This combination is actually another popular research topic named mobile manipulation.

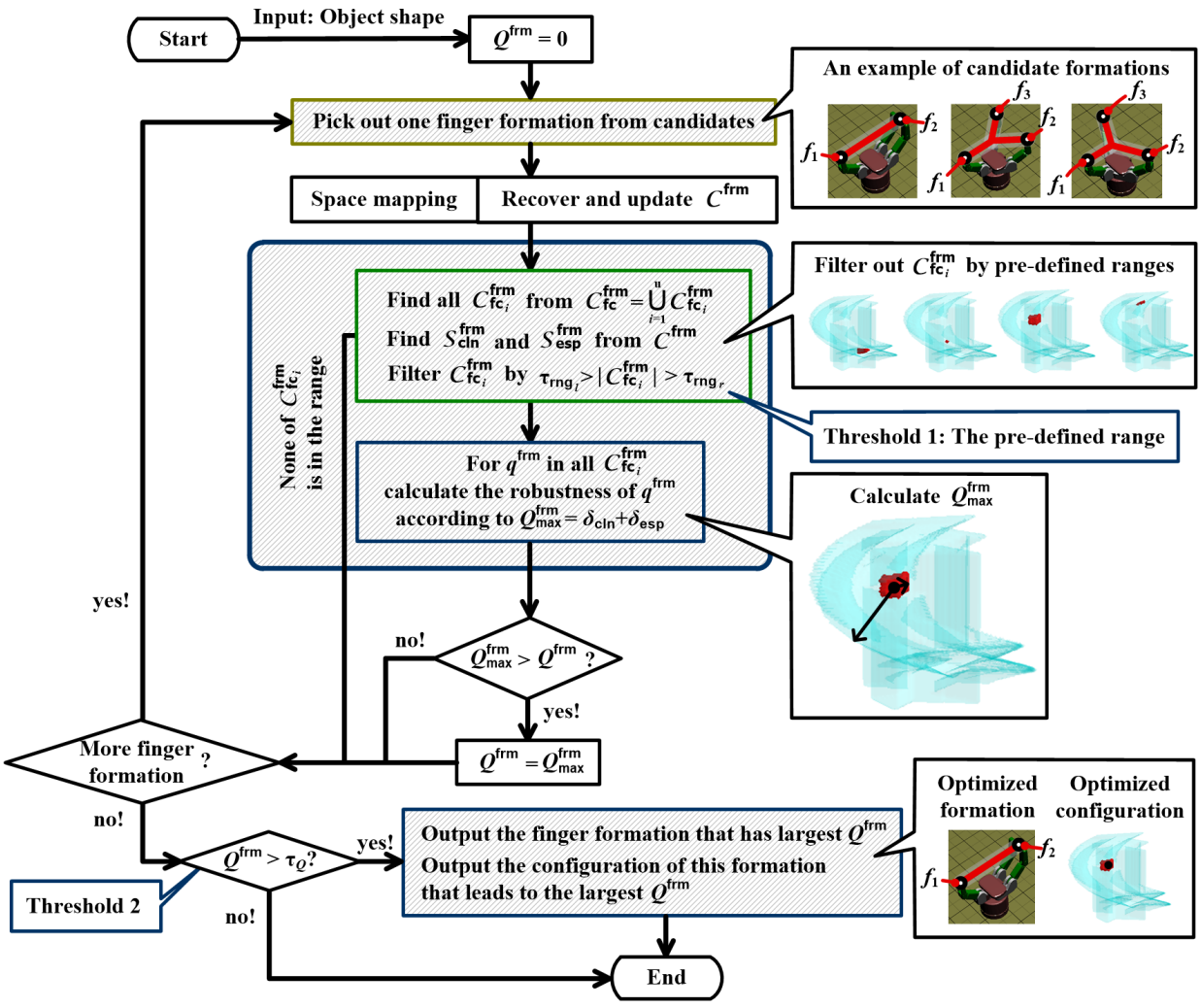


Figure 5.33: Flowchart of finding a robust caging (both formation and configuration).

5.4.2.1 Performance with object O_1 of Fig.5.23

Visualization of the results

Firstly, I will show the performance of the quality function with the concave object O_1 of Fig.5.23 and a series of two-finger formations. This object and the series of two-finger formations are chosen because they have better visual effects. The series of two-finger formations include seven $\mathcal{F}_i$. Their finger positions are shown in Fig.5.34.

Fig.5.35 shows the optimized $\boldsymbol{q}^{frm}$ and their correspondent $F_i[\boldsymbol{q}^{frm}]$ in $\mathcal{W}$ space of each formation. Here the thresholds showed in Fig.5.33 are set as following. $\tau_{rng_r} = 10$, $\tau_{rng_l} = +\infty$. τ_Q is not employed. That means we consider a caging sub-space to be a set of candidate caging configurations when it is composed of more than 10 grids. Note that the formations

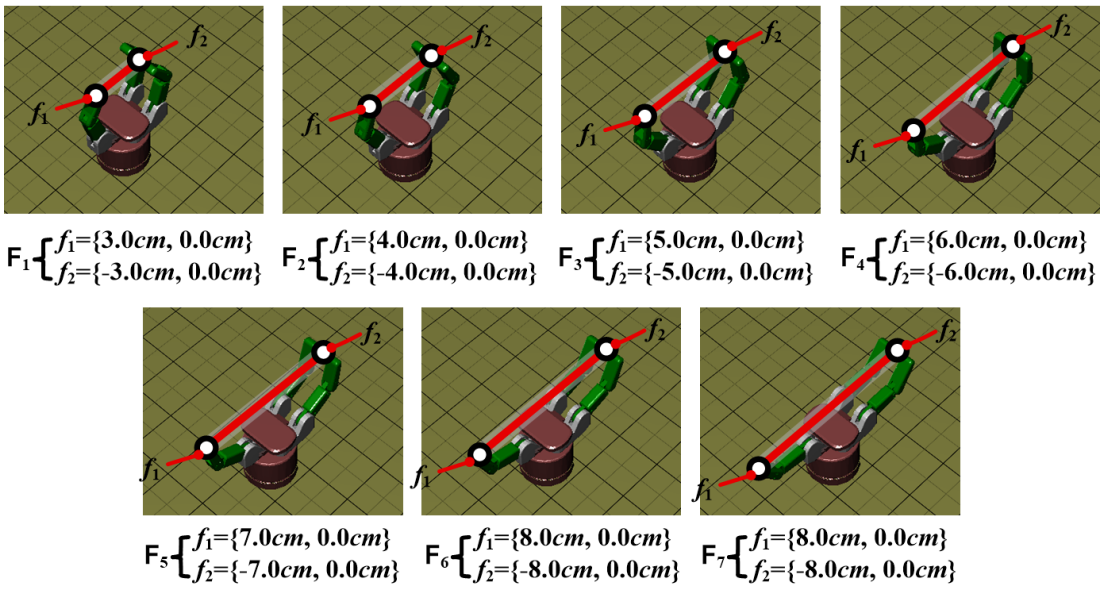


Figure 5.34: A series of seven two-finger formations.

$\mathcal{F}_4$ and $\mathcal{F}_7$ cannot cage the target object under these settings and they are not shown in this figure. Fig.5.36 shows the δ_{cln} and δ_{esp} of largest $\mathcal{Q}^{\mathsf{frm}}_{\mathsf{max}}$ of each finger formation. Like Fig.5.35, $\mathcal{F}_4$ and $\mathcal{F}_7$ cannot cage the target object and they are labeled as "caging breaking" in Fig.5.36. As can be seen from Fig.5.36, the third finger formation F_3 has the largest $\mathcal{Q}^{\mathsf{frm}}$ of all $\mathcal{Q}^{\mathsf{frm}}_{\mathsf{max}} = \delta_{\mathsf{cln}} + \delta_{\mathsf{delta}}$. Therefore, it is the result of robust caging in $\mathcal{C}^{\mathsf{frm}}$. We may better understand the performance of different F_i by referring to $\mathsf{F}_i[\boldsymbol{q}^{\mathsf{frm}}]$ in the $\mathcal{W}$ space of Fig.5.35. Comparing with the other $\mathsf{F}_i[\boldsymbol{q}^{\mathsf{frm}}]$, $\mathsf{F}_3[\boldsymbol{q}^{\mathsf{frm}}]$ ensures both large distance from object boundary (large distance from collision) and large distance from caging breaking. It has the potential of ensuring large robustness against perception and control noises and consequently indicates a good choice for caging or grasping by caging tasks.

Necessity of both δ_{cln} and δ_{esp}

Another problem I would like to discuss in analyzing the performance of the quality function is the necessity of both δ_{cln} and δ_{esp} for robust caging. I have shown in Fig.5.30 and expression (5.9) that δ_{cln} and δ_{esp} mean distance to collision and distance to caging breaking respectively. But are the real-world results in $\mathcal{W}$ space the same as our analysis? Fig.5.37 gives the answer.

Fig.5.37 compares the combined $\delta_{\mathsf{cln}} + \delta_{\mathsf{esp}}$ measurement with a single δ_{cln} measurement. Both δ_{cln} and δ_{esp} play important roles in optimization. Without δ_{cln}, the eigen-caging at an optimized configuration may fall on (collide with) surface of the target object. Without δ_{esp}, there will be no bias towards bulky parts of target objects. The lower row of figures in Fig.5.37 shows the case of a single δ_{esp} measurement. Comparing with the combined measurement (upper row), the most robust $\mathsf{F}_i[\boldsymbol{q}^{\mathsf{frm}}]$ in lower row does not have bias towards

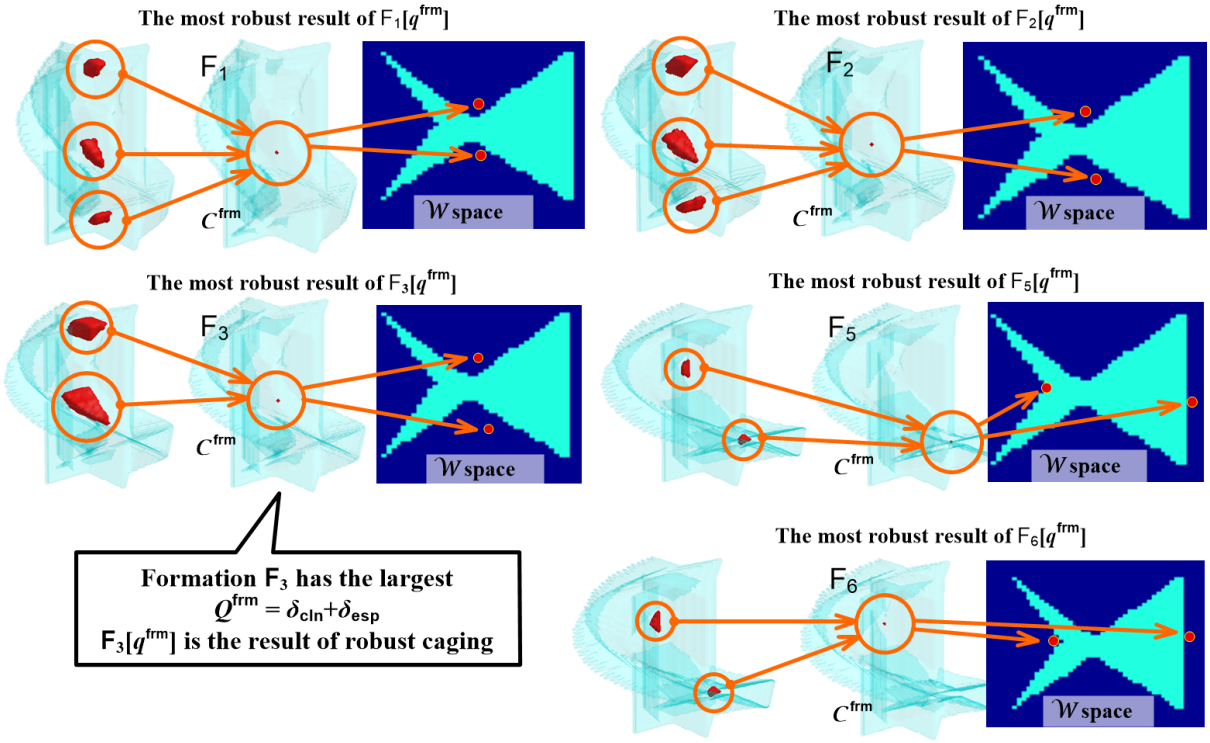


Figure 5.35: Optimized configurations of the seven formations.

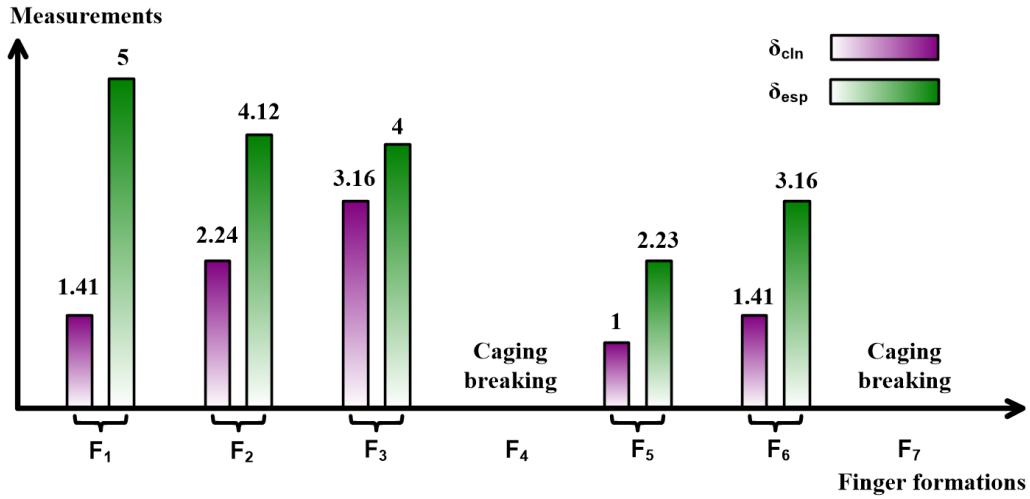


Figure 5.36: The δ_{cln} and δ_{esp} of maximum $\mathcal{Q}^{frm}_{max}$ of each finger formation.

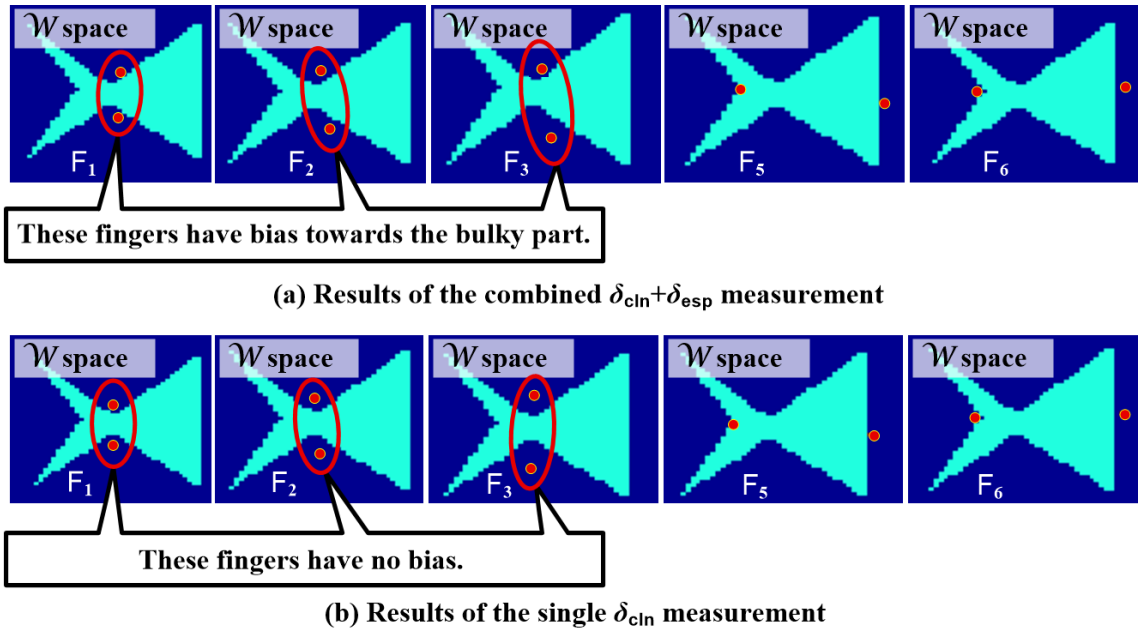


Figure 5.37: The importance of both δ_{cln} and δ_{esp}.

the bulky parts of target objects. This is exactly the same as our expectation. Configurations near the bulky part of the target object have larger δ_{esp}. If we take into account both δ_{esp} and δ_{cln}, the most robust $\mathsf{F}_i[\boldsymbol{q}^{\mathsf{frm}}]$ would have a bias towards the bulky parts like the upper row of Fig.5.37. If δ_{esp} is not considered, there would be no bias and the most robust $\mathsf{F}_i[\boldsymbol{q}^{\mathsf{frm}}]$ would be a configuration that is equally offset target object boundaries like the lower row of Fig.5.37. Caging and grasping the bulky part of an object is much safer and therefore the combined measurement of δ_{cln} and δ_{esp} is promising.

Results of simulation

Besides the qualitative analysis, I further evaluate the robustness and control noise by adding Gaussian noise to vertices of the groundtruth objects like section 3.41. Meanwhile, whether a finger formation cages the target objects during simulation is checked by rotating the end-effector, namely the twelve tests in introduced in the beginning of section 3.3.4. The difference here is we no longer need to retract fingers to see the robustness. Instead, we have already get the optimized results with the robust caging algorithm of $\mathcal{C}^{\mathsf{frm}}$, the left job is to record the largest endurable Gaussian noises and compare them.

The details are as following. In the first place, we calculate an optimized $\mathsf{F}_i[\boldsymbol{q}^{\mathsf{frm}}]$ of the noisy object and actuate the robotic hand into $\mathsf{F}_i[\boldsymbol{q}^{\mathsf{frm}}]$. Then, we evaluate the performance of $\mathsf{F}[\boldsymbol{q}]$ on the groundtruth object. This is the same as what we did in Fig.3.40. However, note that we do not change the number of fingers and retract fingers this time since finger formations are fixed to F_i while $\mathsf{F}_i[\boldsymbol{q}^{\mathsf{frm}}]$ is already offset object boundary due to δ_{cln}. Fig.5.38 shows the maximum $\mathcal{N}(0, \sigma^2)$ of Gaussian noise that the seven two-finger formations in

Fig.5.34 can endure. Here, we re-plot the bar graph sub-figure of Fig.5.36 in sum form $(\delta_{\mathsf{cln}} + \delta_{\mathsf{esp}})$ for convenience. Like our analysis, $\mathsf{F}_3[\boldsymbol{q}^{\mathsf{frm}}]$, which has the largest $(\delta_{\mathsf{bkg}} + \delta_{\mathsf{cln}})$, can endure the largest Gaussian noise.

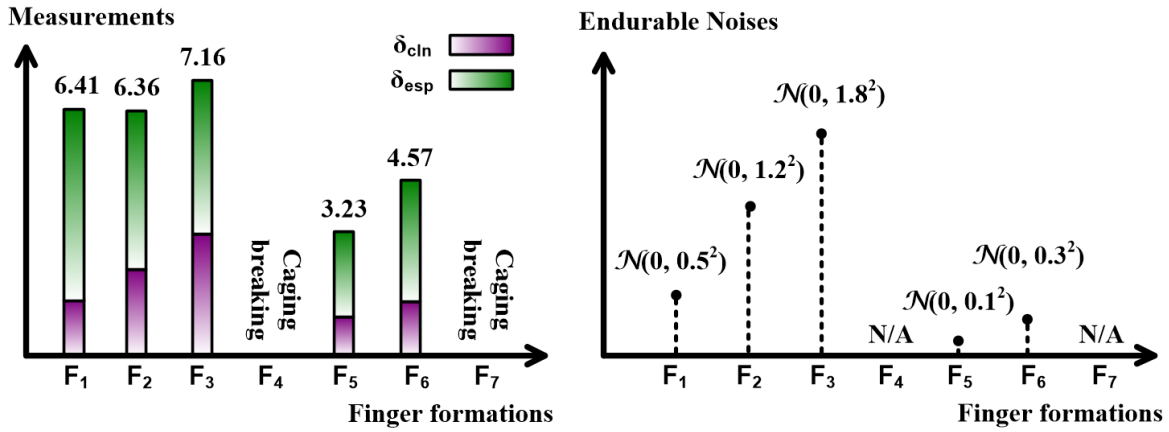


Figure 5.38: The $(\delta_{\mathsf{cln}} + \delta_{\mathsf{esp}})$ of each finger formation and their correspondent endurable noise.

5.4.2.2 Performance on other objects

In this part we show the results of some other examples. Note that in this part I just would like to illustrate the changes of $\mathcal{C}^{\mathsf{frm}}$ with respect to different finger numbers and objects so that I wont take the bother to show their exact dimensions.

Firstly, let us see the result of a convex semi-circular object with finger formations of three, four and seven fingers. The results are shown in Fig.5.39.

Three fingers cannot cage the semi-circular object so that Fig.5.39(a) doesn't have inner red holes. Our algorithm will reject this three-finger formation. Fig.5.39(b) and (c) demonstrate the relationship between finger number and robustness. Redundant fingers can be more robust. Comparing with four fingers which is the least number of fingers that are required to cage a semi-circular object, seven fingers offer more voxels in caging sub-spaces and hence offer more robustness. We can choose the preferable number of fingers by tuning the threshold $\tau_{\mathcal{Q}}$ which was shown in Fig.5.33.

Then, let us see performance of the algorithm on a hollow concave object. It is shown in Fig.5.40.

The first and second rows of Fig.5.40 are the result with a large hollow hole. In these two rows, I recover and update $\mathcal{C}^{\mathsf{frm}}$ and calculate the most robust caging with respect to a one-finger formation (first row) and a two-finger formation (second row) respectively. The third and fourth rows show the results with a smaller inner hole. In these two rows, I recover and update $\mathcal{C}^{\mathsf{frm}}$ and calculate the most robust caging with respect to the same two-finger formation as the second row (third row) and another two-finger formation which has larger

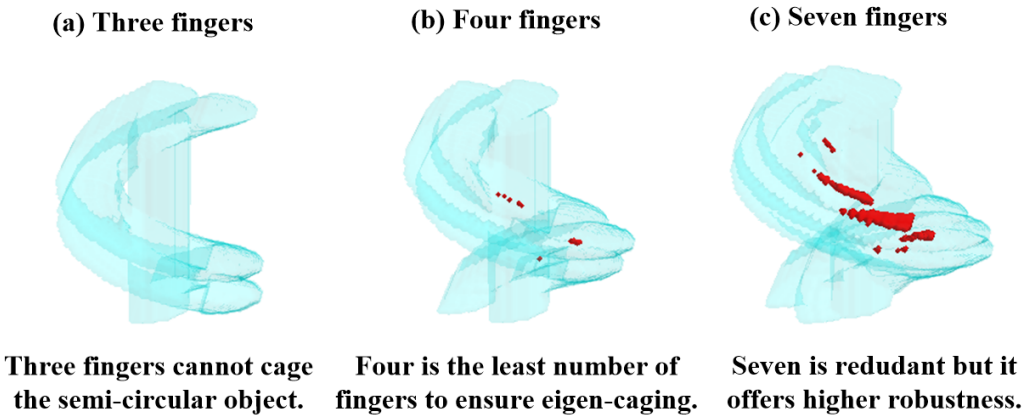


Figure 5.39: The results of a semi-circular object with different number of fingers.

inter-finger distance (fourth row). The most robust $\mathsf{F}_i[\boldsymbol{q}^{\mathsf{frm}}]$ for "grasping by caging" and their correspondent configuration voxels are shown in the "The most robust $\mathsf{F}_i[\boldsymbol{q}^{\mathsf{frm}}]$ in $\mathcal{W}$ space" column and the "The most robust $\boldsymbol{q}$" column respectively. When the hollow hole is large enough, our algorithm prefer inserting a finger into the hole to ensure caging (see the first and second row of Fig.5.40). When the hollow hole is small, our algorithm avoids inserting a finger to prevent collision (the third row of Fig.5.40). However, if the inter-finger distance of the eigen-shape is too large so that it cannot cage the object without inserting into the hole (see the fourth row of Fig.5.40. In that case, there are only two caging subspaces and the eigen-shape can only cage the object by inserting one finger into the hollow hole.), our algorithm returns the configuration with one finger in the hole.

Thirdly, in Fig.5.41, let us see the result of the caging optimization on the real-world objects in Fig.5.26. Here we show the results of Target object #5 and Target object #6 since they have better visual effects. Note that the upper bound τ_{rng_l} is set to $+\infty$, therefore the case of Fig.5.32 is calculated. These results can be auxiliary materials to better understand the simulation results in previous texts.

5.4.2.3 Grasping by caging

Like section 3.3.3.2, we can also perform "grasping by caging" by shrinking fingers of a formation. In that case, "Grasping by caging" would be an additional step at the end of Fig.5.33. The shrinking strategy could be exactly the same as the one introduced in the last part of section 3.3.3.2 since we only would like to discuss **shrinking caging** here.

Fig.5.42, Fig.5.43 and Fig.5.44 show the frame series of grasping by caging based on the results of the O_1 object in section 5.4.2.1, the frame series of grasping by caging based on the hollow concave object with a large hollow hole and the frame series of grasping by caging based on the hollow concave object with a small hollow hole in section 5.4.2.2, respectively.

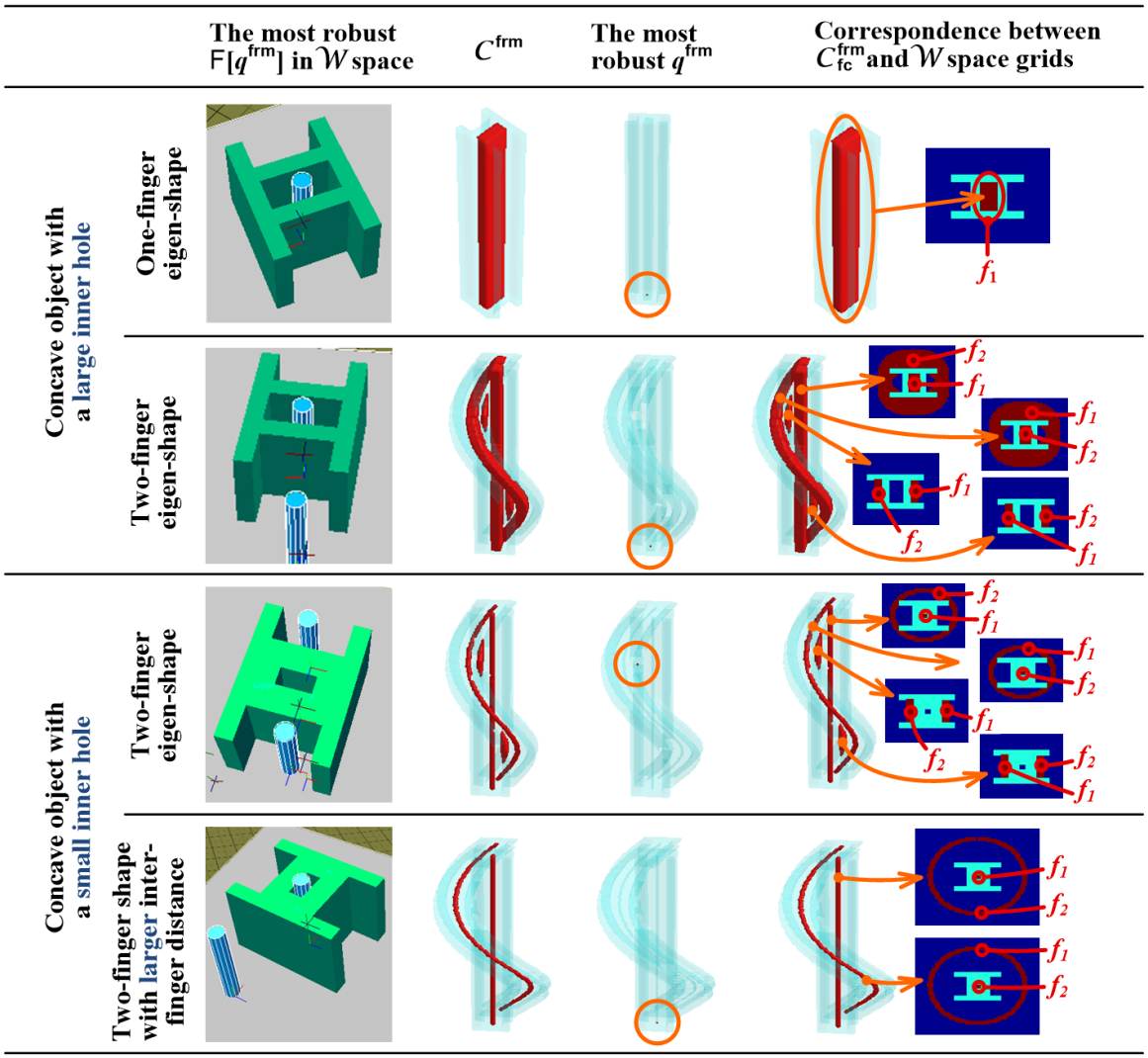


Figure 5.40: The results of a hollow concave object.

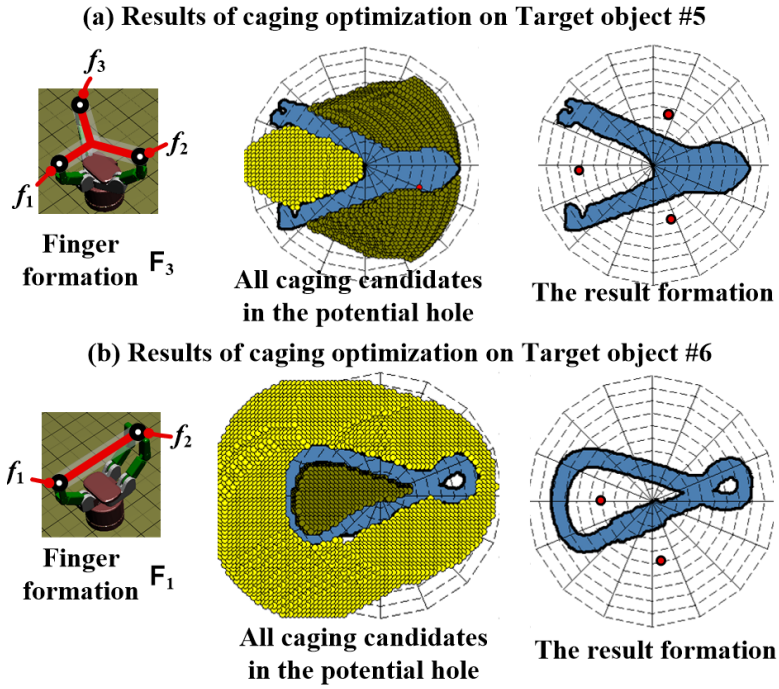


Figure 5.41: The results of Target object #5 and #6 in Fig.5.26.

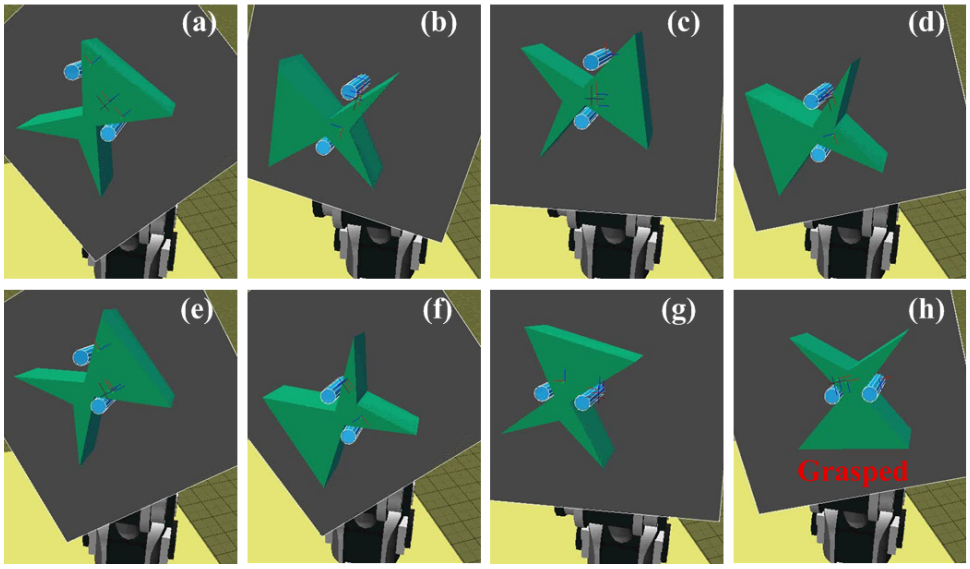


Figure 5.42: The "Grasping by Caging" procedure of O_1.

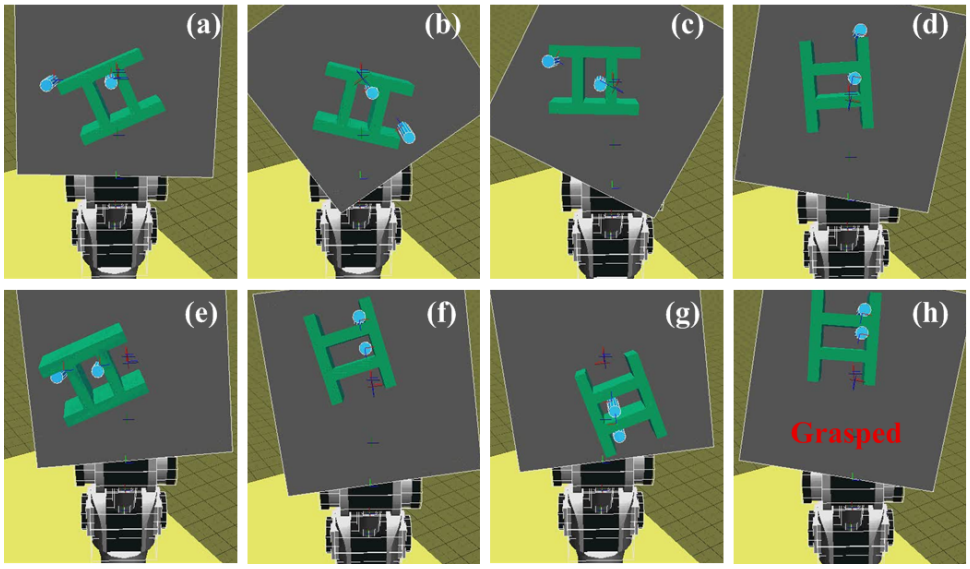


Figure 5.43: The "Grasping by Caging" procedure of the hollow concave object with a large hollow hole.

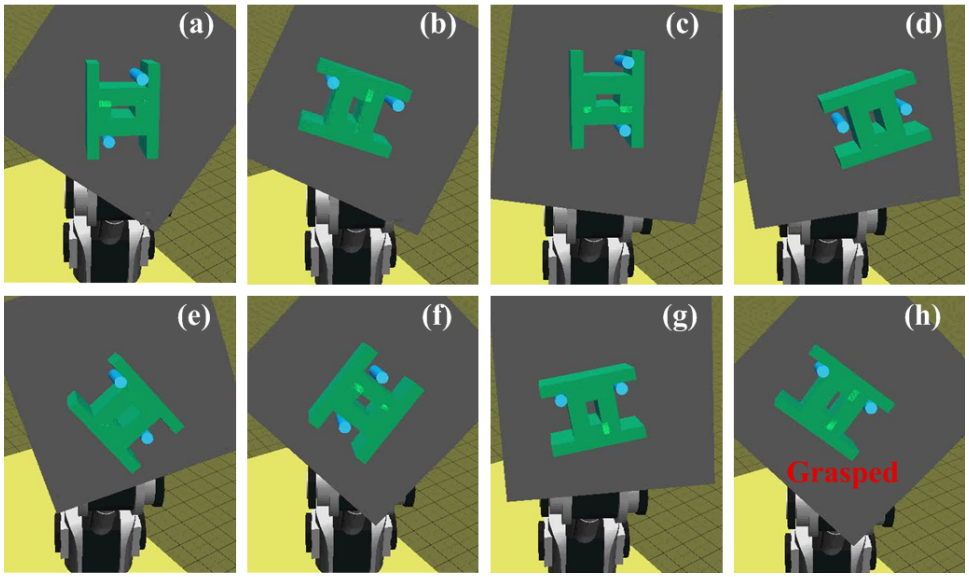


Figure 5.44: The "Grasping by Caging" procedure of the hollow concave object with a small hollow hole.

Chapter 6

Applications II – Hand Design

Caging can offer robustness to uncertainties in grasping. If a robotic hand is designed based on the idea of caging, it would probably work well with noisy perception devices and low-quality control. In this chapter we will see an application which design ands implements a gripping hand based on the caging algorithm in $\mathcal{C}^{\mathsf{frm}}$. The gripping hand is concise and offers a low-cost alternative to co-operate with noisy data and low-quality control. This chapter includes two sections. In the design section, I will show how I successfully simplify the number of actuators into one by quantitatively analyzing finger formations with caging tests conducted on both random objects and objects from MPEG-7 shape database. Following the simplified one-actuator design I will in the implementation section show the implementation and demonstration of a gripping hand by modifying a SCHUNK RH707 gripper and carried out experiments with a manipulator built on the Neuronics Katana arm. The one-actuator gripping hand could work with the Swiss Ranger and both convex and concave objects. It demonstrates the merits of caging, especially the advantages of caging in $\mathcal{C}^{\mathsf{frm}}$ over $\mathcal{C}^{\mathsf{obj}}$.

6.1 Designing a Gripping Hand by Using Caging

Although lots of theories have been developed in the research field of manipulation and grasping. These theories involve not only some of the previously discussed topics like form/force closure, caging and their optimization but also some of undisclosed topics like enveloping [Trinkle et al., 1988]. However, a large gap exists between these theories and real-world designs and applications. For example, robotic hand dimensions and finger numbers are neither designed according to mathematical formulae of form/force closure nor designed according to perception devices. They are, in most cases, decided by (1) purpose of usage, (2) biomimetic study or (3) mechanical constraints and empirical experiences. In this part, I propose the design a gripping hand according to the theory of caging. The hand has only one actuator. It is concise, low-cost and owns all merits from caging (like robustness to uncertainties).

6.1.1 Retrospecting the hand design

Firstly, let us retrospect the contemporary works of hand design. There are two problems regarding the design of a robotic hand. The first problem is its complexity. Let us compare the following three representative examples — (a) The Schunk JGZ industrial gripper [SCHUNK, 2013], (b) the Barrett Hand [Barrett Technology, 2013] and (c) the Robonaut Hand [Bridgwater et al., 2012]. Note that there are many alike candidates whereas I take these three for instance. The three hands differ significantly in DoFs (Degree of Freedoms), actuation types and purpose of usage. The JGZ gripper has one DoF. It is fully actuated and designed for industrial usage. The Barrett Hand has four DoFs. It is under-actuated and designed to manipulate versatile objects. The Robonaut Hand has twelve DoFs which mimics a human hand. It is dexterous and designed for tele-operation. These hands are designed either according to their usage, biomimetic study or empirical experiences. Comparing with the design strategies of these hands, I would like to take into account of caging and design a hand that is both high in generality and low in DoFs.

Design belongs to the **mechanism** aspect discussed in the beginning of this thesis. But we do need to take into account both **sensing** and **mechanics** as well. Reference [Pollard, 2010] offers a good summary of hand design and the most related works to my case are [Zhang and Goldberg, 2001], [Dollar and Howe, 2010] and [Hammond et al., 2012]. Robotic hands in these works are designed according to "constraining" models. Specially, the SDM hand presented in [Dollar and Howe, 2010] follows principles of enveloping [Trinkle et al., 1988][Dollar and Howe, 2005] and won great success in grasping in unstructured environments. [Hammond et al., 2012] discusses in detail how to reduce motor number and designs an under-actuated robotic hand. Like these works, I also simplify and design our gripping hand based on a "constraining" model. The "constraining" model is caging. By performing caging tests on random objects and objects from MPEG-7 shape data base libraries, I find an optimized actuator and finger setting that have highest successful caging rate. The optimized actuator and finger setting help to reduce the number of actuators into one. At the same time, it owns all merits from caging and endows us the potential to perform safe and robust grasping.

The second problem is integration with perception devices. We have reviewed the popular works that detect objects and synthesize grasping in section 3.3.3.2. In all the techniques and devices employed by those works, database matching is effective in grasping known objects but it is not as satisfying with unmodeled targets. RGB camera is affordable and applied to many industrial systems. However it suffers a lot from unstructured environments. Depth sensors can be summarized into two categories, namely scanners and rangers. Scanners have high precision as well as high costs. Rangers are much cheaper. Examples of rangers involve the ToF-based (Time of Flight) Swiss Ranger or structure light-based KINECT. These two devices were discussed in detail in section 4.1.2.1. With the help of caging, the hand is expected to work with the Swiss Ranger. Noises of the Swiss Ranger are +/-10mm in depth and +/-7mm in horizontal plane. I believe if the hand could work with the Swiss Ranger, it is suitable to most applications.

6.1.2 A basic design based on qualitative analysis

There are some design candidates that are in well accordance with the caging theory. In section 2.2 we have known that **the number of fingers that are sufficient to cage an object in n_{dim} space is $n_{\mathsf{dim}}+1$ to $2n_{\mathsf{dim}}$**. That is to say, we should install at least 2×2 = 4 fingers to cage any 2D shape. It is true that the performance becomes better if there are more fingers. However, I prefer the least number 4 since I would like to reduce the complexity as much as possible.

After deciding the number of fingers, the remaining problem is how to install the four fingers and how to actuate them. One example is the distributed end-effector discussed in the first subsection of Chapter 4. In that design, fingers are attached, actuated and detached sequentially by the single x-y-θ actuator and only three motors are required. Although the design lowers system cost, it introduces a time-consuming attaching-actuating-detaching procedure which slows down operation. Unlike that design, I will in this part consider the installation of actuators by quantitative evaluation with caging algorithms.

Fig.6.1 shows the candidate installations of actuators. Note that I do not consider the shapes of fingers here and they are therefore rendered as simply poles. The first candidate, Fig.6.1(a), is the most intuitive installation. It endows distributed control to each finger and requires as many as eight actuators. The distributed end-effector is actually a variation of it. The last candidate, Fig.6.1(d), drives four fingers simultaneously. It requires only one actuator and the SDM hand [Dollar and Howe, 2010] follows its principle. The last candidate fully ensure equal inter-finger distances which adds strong bias to **rotational constraints** (recall section 3.3.1). However, it has no flexibility for **translational caging**. The SDM hand solved this inflexibility problem by fabricating delicate under-actuated fingers (say, delicate shapes of fingers). In our case which aims at a concise "gripping" hand, or namely a hand with pole-like fingers, Fig.6.1(b) and Fig.6.1(c) are better choices.

The difference between Fig.6.1(b) and Fig.6.1(c) are their levels of biases towards equal inter-finger distances. Fig.6.1(b) has higher flexibility in position control and it holds more bias towards **translational caging**. Nevertheless, three actuators complicate the gripping system. I prefer choosing the two-actuator candidate Fig.6.1(c) as **the basic design**. Fig.6.2 shows in detail of how this basic design works. Each actuator in this installation drives two pairs of fingers and either caging or grasping by caging can be performed by this design.

The basic design is based on qualitative analysis. We would like find some quantitative supports to demonstrate its advantages. In the next section, we will quantitatively analyze this basic design with the caging algorithm in $\mathcal{C}^{\mathsf{frm}}$ and various objects and see if it has high successful rates in caging. If the basic design has high successful caging rates, it is considered to be a satisfying candidate to manipulation with caging and inherits the merits of caging with the help of caging or grasping by caging algorithms.

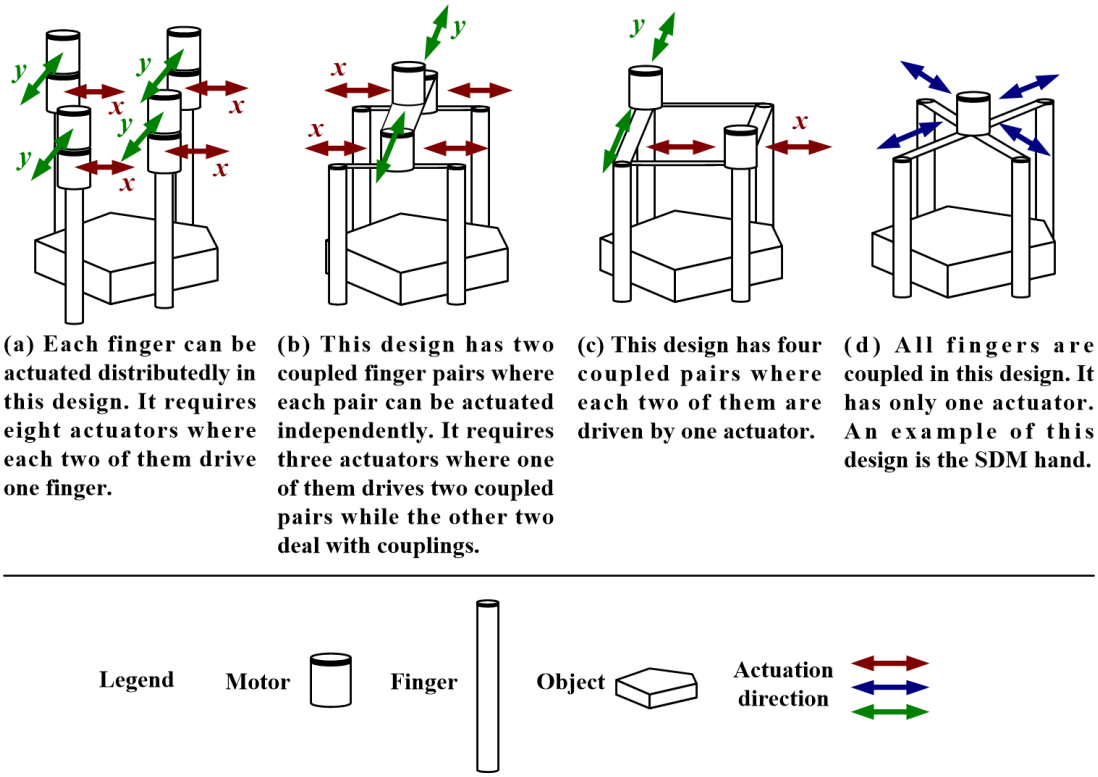


Figure 6.1: Four candidate installations of actuators.

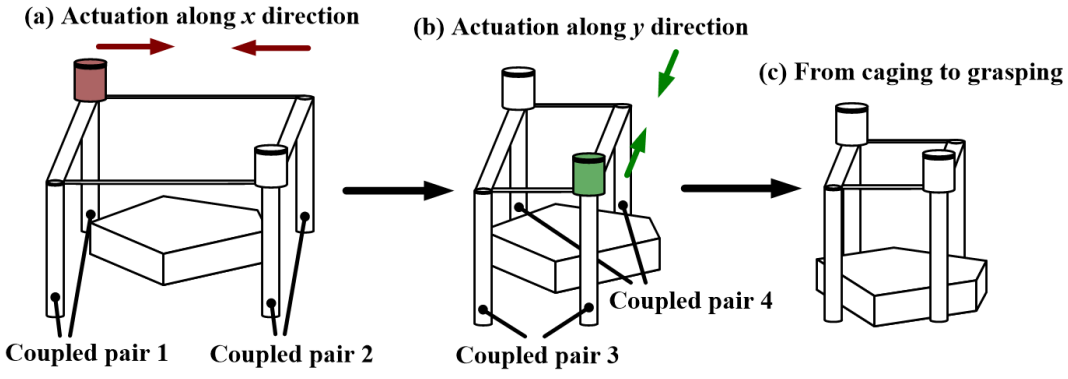


Figure 6.2: The basic design and one of its caging or grasping by caging procedure.

6.1.3 Quantitative analysis of the basic design

6.1.3.1 Objects for quantitative analysis

In order to find some quantitative supports for the basic design, I employ an object generator to randomly generate some shapes and quantitatively evaluate the performance of the basic design with these shapes and caging tests.

It is difficult for a random generator to cover any 2D shapes but we try to enlarge its coverage as much as possible. This is done by setting a parameter n_s, namely the number of sectors. See bold segments in Fig.6.3(a) for details. The object generator generates random shapes inside a background circle decided by different n_s parameters. The figures in in Fig.6.3(b) exemplify some randomized objects. Three groups of randomized objects are generated according to different parameter settings. Readers may refer to Alg.4 to better understand the roles of n_s. The algorithm randomize a position along radial direction at each sector. This position is saved as one vertex of the target object. Final list of target object vertice indices are returned as $\mathsf{P}_{\mathsf{bdry}}$.

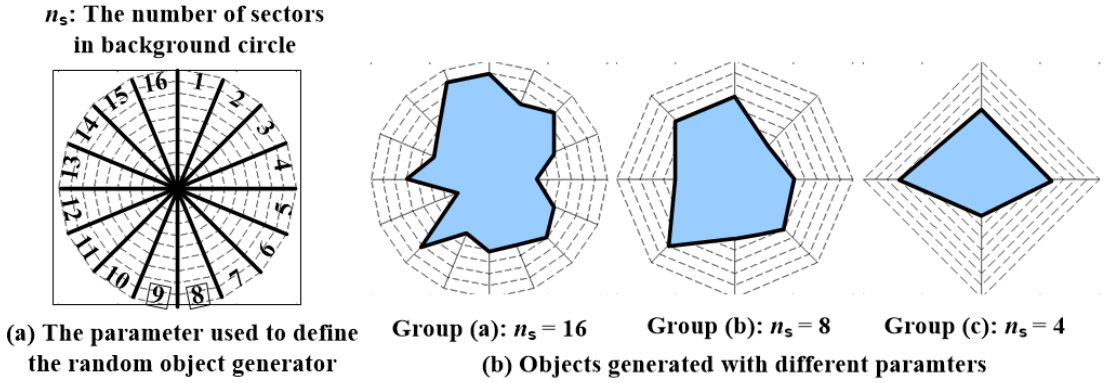


Figure 6.3: The random object generator.

This object generator is subject to the following limitations. (1) It cannot generate shapes with inner holes. This limitation is acceptable since I would like to constrain the caging into squeezing caging. (2) It may require thousands of randomization before reaching a convincing conclusion. In order to conquer the second limitation, we generate objects by three groups. Each group is randomized according to different n_s. Their details are as following. **Group (a): n_s=16.** We expect the random shapes generated in this group may be either smooth (small probability) or with sharp prostrusion (high probability). Shapes in this group should be, in most cases, easy to be caged owing to their protrusions. **Group (b): n_s=8.** The random shapes generated in this group has higher bias towards smooth objects and general polytopes while have less bias towards protrusion. We expect that shapes in this group become more difficult to be caged comparing with Group (a). **Group (c): n_s=4.** The random shapes in this group help to fill up the loss of Group(a) and Group (b). For instance, it has high probability of generating quadrilaterals and trilaterals which

Algorithm 4: The random object generator

Data: n_s
Result: P_{bdry}
begin
  $P_{bdry} \leftarrow \emptyset$
  for $i \in \{0 : n_a\}$ **do**
    /*Randomly select a position along radial direction*/
    $p_i \leftarrow$randomize a number between 0 and 9
    $P_{bdry} \leftarrow P_{bdry} \bigcap p_i$
  end
  return P_{bdry}
end

are hardly generated in Group(a) and Group(b). Shapes in Group (c) should be easier to be caged comparing with Group (b) as their inner angles become sharper. We expect that comparing with a single-group generator, generating shapes by these three groups with different parameter settings could offer convincing conclusions with fewer randomizations.

Besides the random object generator, I further evaluate the performance of the basic design with objects extracted from the MPEG-7 shape library (see Fig.6.4). Shapes in the MPEG-7 library are based on real-world objects, they are more realistic comparing with our random generator. These objects can further confirm the performance of our basic design.

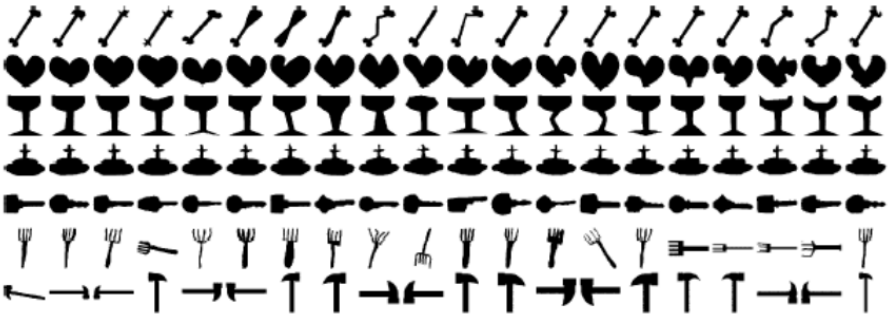


Figure 6.4: Some objects from the MPEG-7 shape library.

In total, we perform caging tests on 1000 shapes from Group (a), 1000 shapes from Group (b), 1000 shapes from Group (c) and 1100 shapes from the MPEG-7 shape library.

6.1.3.2 Results and analysis

The caging test algorithm in $\mathcal{C}^{frm}$ is limited to one finger formation and one object. It is not applicable to testing the performance of a hand which could form into infinite number of

formations. I propose to solve this problem by defining 20 candidate formations and assume these 20 formations could be representative and cover the infinite number of cases. Fig.6.5 shows the 20 formations. It involves five columns of pair 1 and pair 2 along x axis and four rows of pair 3 and pair 4 along y axis. Readers may refer to the texts in Fig.6.2 to recall the pairs 1, 2, 3 and 4. Note that since x and y axes of the basic design are symmetric, 5 *columns* $\times$ 4 *rows* complements 4 *rows* $\times$ 5 *columns*. In this way, using 5×4=20 formations is more efficient than using symmetric multiplications like 5×5 or 4×4. Note that the background of the object generator is illustrated together with the finger positions in Fig.6.5(b). Readers may compare the dimensions of the twenty formations and the size of randomly generated objects by referring to it.

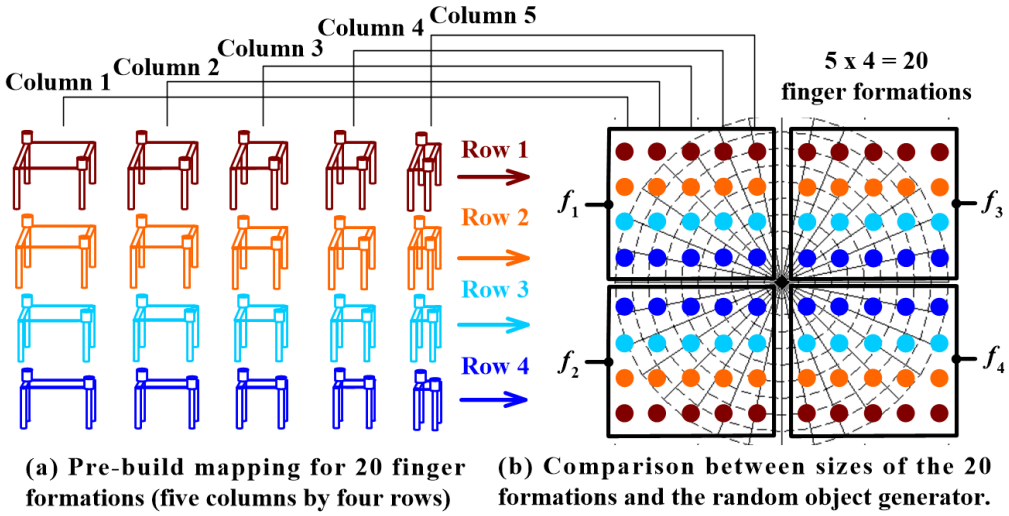


Figure 6.5: The 20 representative finger formations of the basic design.

I quantitatively evaluate the 20 formations of **the basic design** with the 3000 random objects and 1100 MPEG-7 shapes and the caging test algorithm in $\mathcal{C}^{\mathsf{frm}}$. That is to say, we carry out (1000+1000+1000+1100)*20 = 82000 caging tests. This is possible owning to the rapidness and completeness of improved space mapping. Readers may review section 5.3.2.1 to recall the details of caging test in $\mathcal{C}^{\mathsf{frm}}$.

Fig.6.6(a) shows the total successful caging rate of the 20 formations. In this result, if an object can be caged by any of the 20 formation, then the basic design is assumed to be able to cage that object. The result here is obtained by setting n_g=150×150, m=72, τ_{rng_r}=25 and τ_{rng_r}=$+\infty$.

The result indicates that the basic design can cage objects with more than 90% successful rate and we can draw a conclusion that most "normal" objects, either they have convex, concave or smooth boundaries, can be caged by the basic design. Exceptions are those tiny or thin cases shown in Fig.6.6(b). The basic design is not suitable to cage those objects. Actually, those objects are not suitable for general caging. An intuitive example is caging

an eel in fishing. Eels cannot be captured by general fishing net and fish men use special net to cage them.

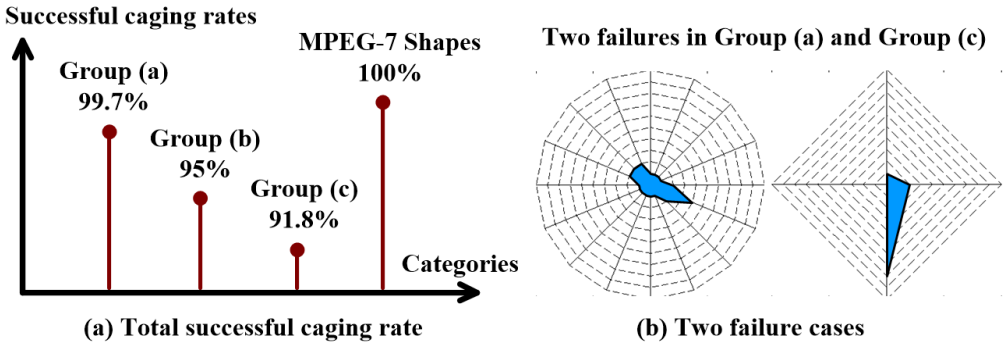


Figure 6.6: The total successful caging rate and two examples of failure.

Note that the objects from MPEG-7 shape library is resized to fit into the background of random object generator. That means the objects from MPEG-7 library cannot be too small. That could be the reason why they can always be caged with 100% successful rate. Objects from our random generator are harder to be caged comparing with MPEG-7 shapes.

More than 90% is quantitatively good performance. However, I wonder if it can be further simplified. Dollar's SDM hand has only one motor, whereas the basic design requires two. In the next section I will further analyze the total successful rate and check if the basic design can be further simplified.

6.1.3.3 Further simplification

The results in Fig.Fig.6.6(a) are the total rates of 20 formations. In another word, the basic design is considered to be able to cage an object as long as a single one from the 20 formation can cage it. This is reasonable as the basic design has two actuators and can be actuated into any of the 20 formations.

If we would like to further simplify the basic design, the most intuitive way is to to delete one actuator. However, deleting one actuator changes the 20 formations. For example, when the x-actuator, namely the red one in Fig.6.2, is deleted, the basic design can no longer be actuated from one column to another. That means the $5\times4 = 20$ formations become a single column of 4 formations. When the y-actuator, namely the green one in Fig.6.2, is deleted, the basic design can no longer be actuated from one row to another. That means the $5\times4 = 20$ formations become a single row of 5 formations.

Suppose we delete the y-actuator for simplification. Note that deleting the x-actuator works in the same way as x axis and y axis are symmetric. Here I delete y because we discretized the formations into 5×4. Deleting the actuator along y direction leaves 5 formations along x direction. The resolution is larger. If it were 4×5, it would be a better choice to delete the actuator along x direction. After deleting the y-actuator, the basic design can

no longer be actuated from one row to another and we can only keep a single row. In that case, the designing problem becomes which row should we retain to ensure high successful caging rates. This problem could be solved by further analyzing the total successful rate in Fig.6.6(a). Fig.6.7 shows the further analysis of Fig.6.6(a). In this figure, the decomposed rates of total results are illustrated. Successful caging rates of each row are shown respectively in the left part. They indicate the successful caging rates of each row of the 20 formations on the three random groups and each row is rendered with different color bars. The right figure is a copy of the left part of Fig.6.6(a) for easy comparison.

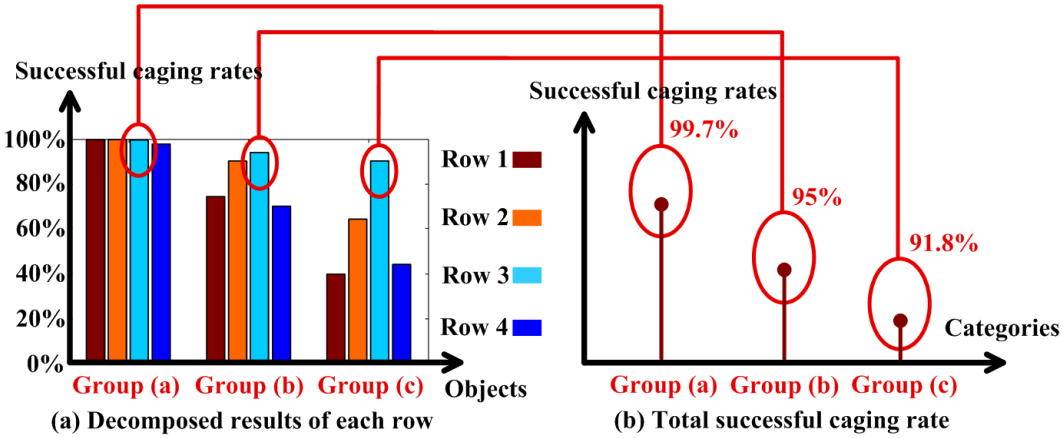


Figure 6.7: The successful caging rates of each row.

As is denoted in Fig.6.7, the third row of formations, namely the row with cyan color, has highest successful caging rates. It is also the key row of the 20 formations. Readers can compare successful caging rates of the "cyan" bars with total successful caging rates in the left part of Fig.6.7 for better comprehension. Successful rates of the "cyan" row on the three groups of random objects, namely the "cyan" bar in the left part of Fig.6.7, are nearly the same as the total successful rates of all 20 formations. That is to say, **the randomized shapes are mainly caged by finger formations in the "cyan" row and we can delete the other rows without much loss of successful rates**.

Consequently, we can get the following simplification rule. **The basic design can be further simplified by fixing one actuator without much loss of successful caging rates. The inter-finger distance of the fixed actuator should be around the "cyan" row.** Fig.6.8 shows the idea. After simplification, only the actuator along x axis remains.

Let us retrospect this simplified design and compare it with Fig.6.1(d). They both have only one actuator. But is the simplified design really better? A confirming conclusion can be drawn by deeper review of Fig.6.8. Fig.6.9 shows in depth the deeper details of the analysis in Fig.6.7. In this figure, not only the successful caging rates of each row but also the successful caging rates of each formation are illustrated. The finger formation with higher successful caging rates have larger circle sizes. Successful caging rates of the simplified design are roughly the sum of "cyan" circles. I use the word "roughly" because there is redundancy

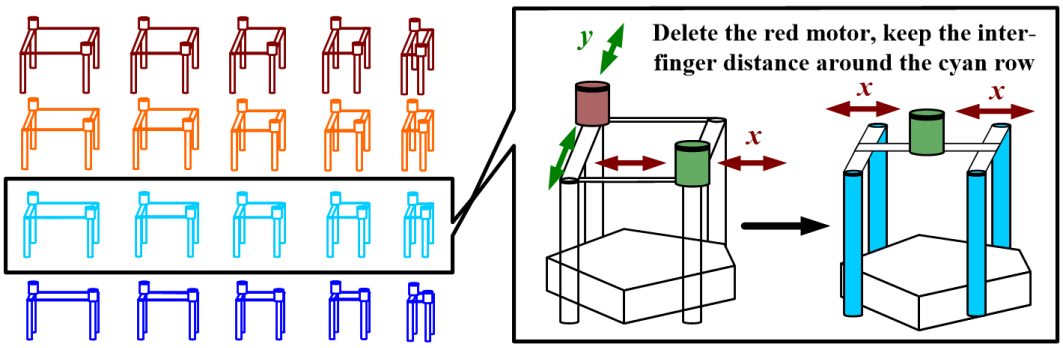


Figure 6.8: Further simplification of the basic design.

in addition. This is a rough analysis. In contrast, successful caging rates of the design in Fig.6.1(d) is roughly the sum of diagonal circles. Note that like the simplified design, there is redundancy in addition. The simplified design has larger sum of circle sizes and higher successful caging rate. Therefore, it is indeed a better design comparing with Fig.6.1(d).

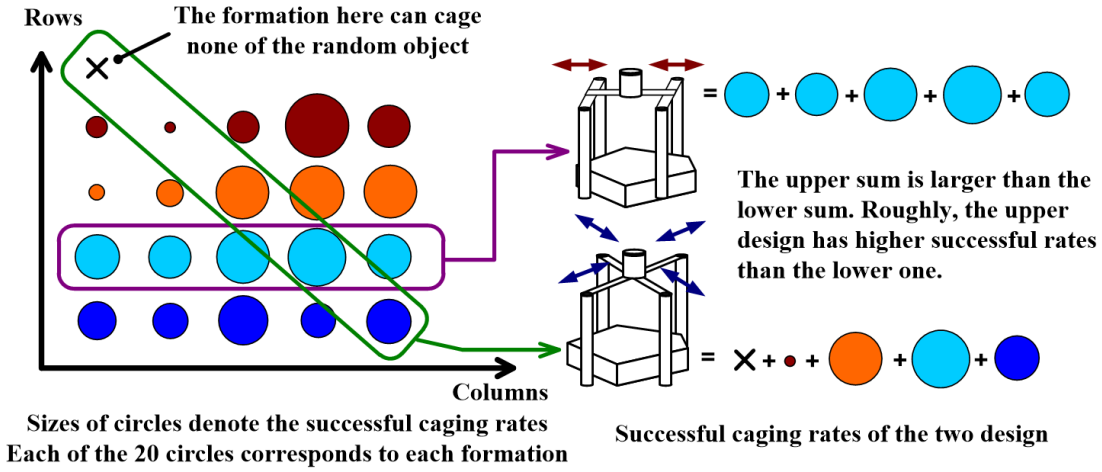


Figure 6.9: Comparison of the simplified design and the design in Fig.6.1(d).

6.2 Implementation of the Design

Now, we can implement a one-actuator gripping hand by using the simplification rule. The implementation would be concise as well as effective. It could benefit from all merits of caging. We will be more confident in that after the following parts through applications like dealing with noisy perception devices and low-quality control.

Firstly, let us digitalize the "cyan" row. According to ratios of the formation rows in Fig.6.5, I set fingers as following. For an object in a background circle of diameter 8, we

choose 3 as its fixed inter-finger distance. That is to say, the "cyan" row is digitalized into a 8:3 ratio. Then, we can implement a gripping hand based on this ratio. Fig.6.10(a) illustrate settings of fingers and their installation on a RH707 hand. Here, the range of x axis actuation is set between $10mm$ and $80mm$ while the fixed inter-finger distance along y axis is fixed to $30mm$ to maintain the 8:3 ratio. **There is no special mechanisms in this implementation. It is nothing more than a simple "gripping" hand. However, the 8:3 ratio, which is based on lots of caging tests in last section, changes the essence of "gripping". According to the foregoing analysis, this implementation should be able to cage or grasp by caging objects inside a $80mm$-diameter background circle with more than 90% successful caging rate. Moreover, it would be robust to co-operate with noisy devices and low-quality control.**

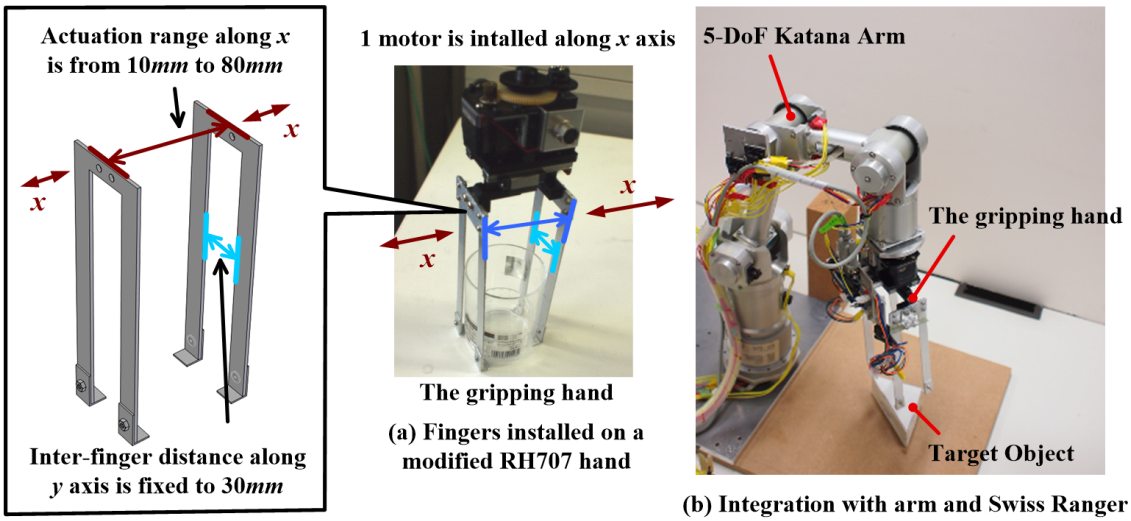


Figure 6.10: Implementation of the "gripping" hand and its integration with the Katana Arm.

The implemented hand is integrated with a 5-DoF Katana Arm and the same SR4000 Swiss Ranger as Chapter 4 to perform various caging and grasping by caging tasks. Fig.6.10(b) shows an overview of the integration. Perception device, namely the Swiss Ranger, is installed on top of the arm and it is not shown in this figure. The Swiss Ranger is installed at a height of $1200mm$ from the operation plate. As was discussed in Chapter 4, the Swiss Ranger suffers from perception noises and perceives top-view 2D shapes of target objects with $10mm$ noise. Fig.6.11 demonstrates the noises. It compares an ideal object shape and its correspondent approximated polytope based on point clouds perceived from the Swiss Ranger. The ideal shape should be a triangle. However, The perceived polytope becomes a quadrilateral. The perceived polytope differs a lot from the ideal shape. Explicitly calculating force/form closures based on this noisy polytope is quite tough and causes danger like squashing in "picking up" tasks. Fig.6.12 shows the failure case when caging is not employed.

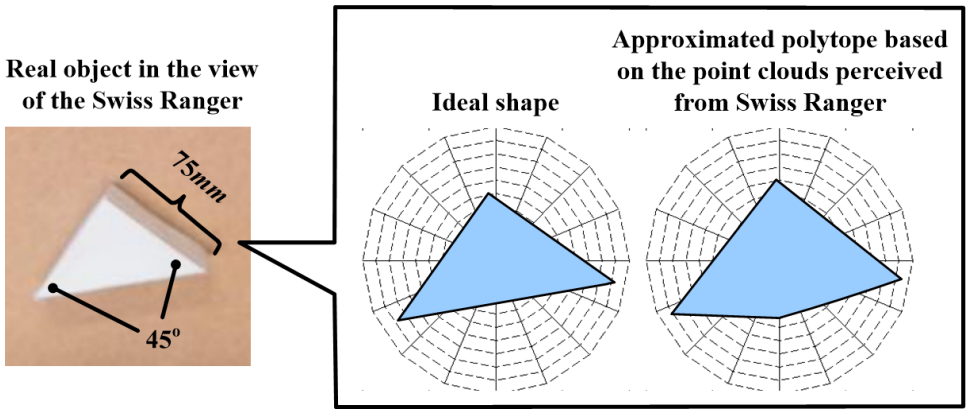
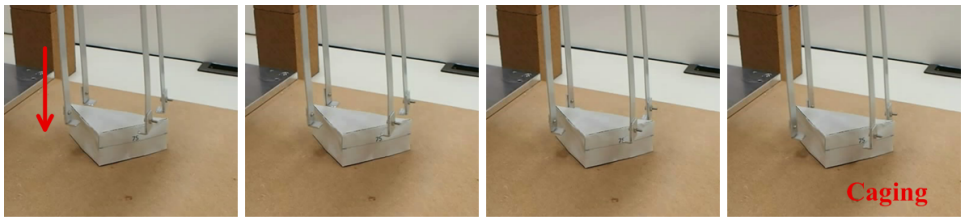
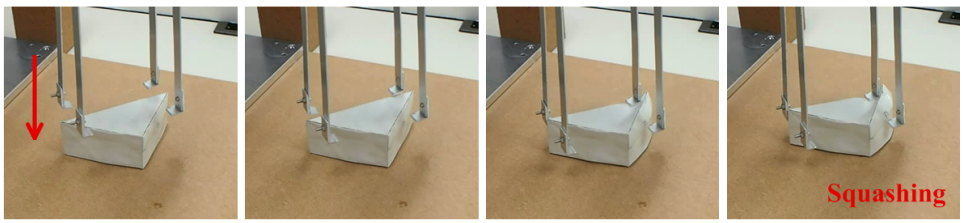


Figure 6.11: Comparison between the approximated shape and the ideal shape.



(a) The target object can be successfully caged with the implementation based on the simplified design and the robust caging algorithms



(b) The finger may squash the target object or fail to cage the target object due to perception errors if caging was not employed.

Figure 6.12: Comparison between successful caging and failure caused by perception noises.

The implemented hand should collaborate together with the robust caging algorithm of last Chapter to deal with various noises. Algorithm flow of robust caging has been shown in Fig.5.31. Here in the following part I repeatedly summarize the whole procedure of a "grasping by caging" task performed by the hand. Readers may collaboratively review Fig.5.31 to better understand the procedure.

Performing the "grasping by caging" task with the implemented hand requires the following four steps. For convenience and better comprehension. I illustrate each step with figures in Fig.6.13. Readers may refer to it in case of confusion.

Step 1 Get the cloud points from the Swiss Ranger and detect the target objects. In the first step, the basic shape of target object is extracted by using the cloud points collected from Swiss Ranger.

Step 2 Calibrate the object position and simplify the extracted shape into a polytope. In the second step, the extracted shape is approximated into a polytope and calibrated with respect to the calibration marker denoted in Fig.6.10(b).

Step 3 Rebuild and update the $\mathcal{C}^{\mathsf{frm}}$ space with the approximated polytope of **Step 2** and find the candidate caging configurations.

Step 4 Find an optimized caging configuration from the candidates and actuate the manipulator. This optimization procedure is done by measuring expression (5.10).

Like the results in Fig.5.27, Fig.5.28 and Fig.5.29, the yellow points in **Step 3** and **Step 4-1** of Fig.6.13 show the caging finger formations of the gripping hand. They correspond to configurations in all configuration voxels in the isolated sub-space in the middle part of Fig.6.13, **Step 4-1**. Any one of the caging formations could cage the object and we can choose an optimized one from the them as the pre-grasp caging formation. We can calculate the optimized caging formation for the hand on line and pick up objects robustly in a "grasping by caging" way.

The **Step 4-2** of Fig.6.13, or Fig.6.14(b), shows the actuation of a "grasping by caging" procedure of the triangle object. The original RH707 hand accepts only the open control and close control. It cannot close to a certain position. I install a photo-interrupter to the RH707 hand as the encoder of close control. The photo-interrupter interrupts at five gaps so that the modified hand could close to five different positions that correspond the five different columns shown in Fig.6.5. The five different closing positions are shown in Fig.6.14(a). When the robust caging algorithm find an optimized caging formation, the Katana Arm moves the hand to the optimized position and orientation. Then, the hand opens itself to the optimized photo-interrupter stop. When performing "grasping by caging" task, the hand closes itself to the neighbour photo-interrupter stop to ensure enough friction or enough force to insertion of finger tips and pick up target objects (There is an extra neighbour inside the fifth stop). Note that this closing procedure is not safe. It is totally blind as there is no local sensors on fingers.

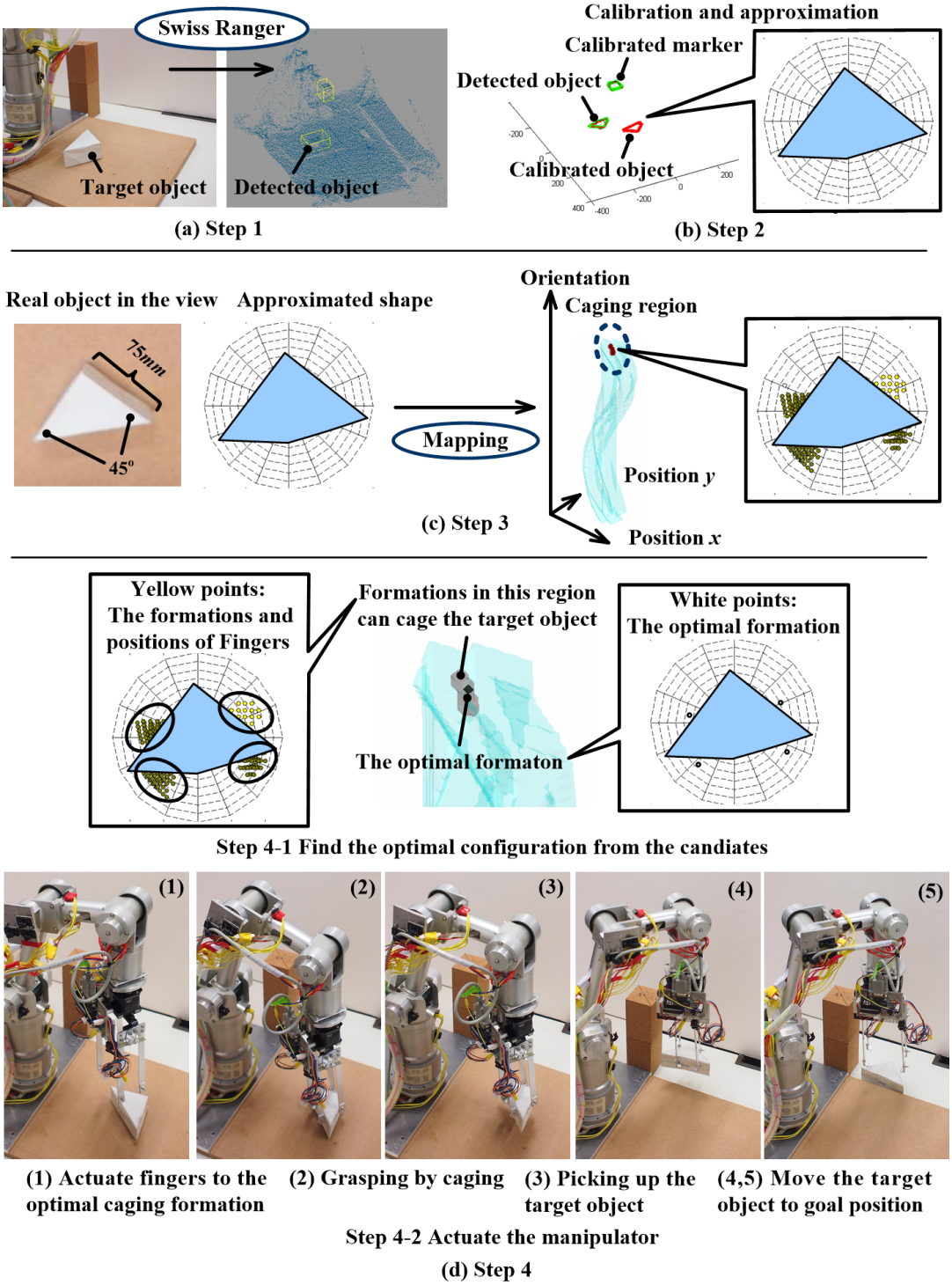


Figure 6.13: The four steps to perform a "grasping by caging" task with the implemented hand.

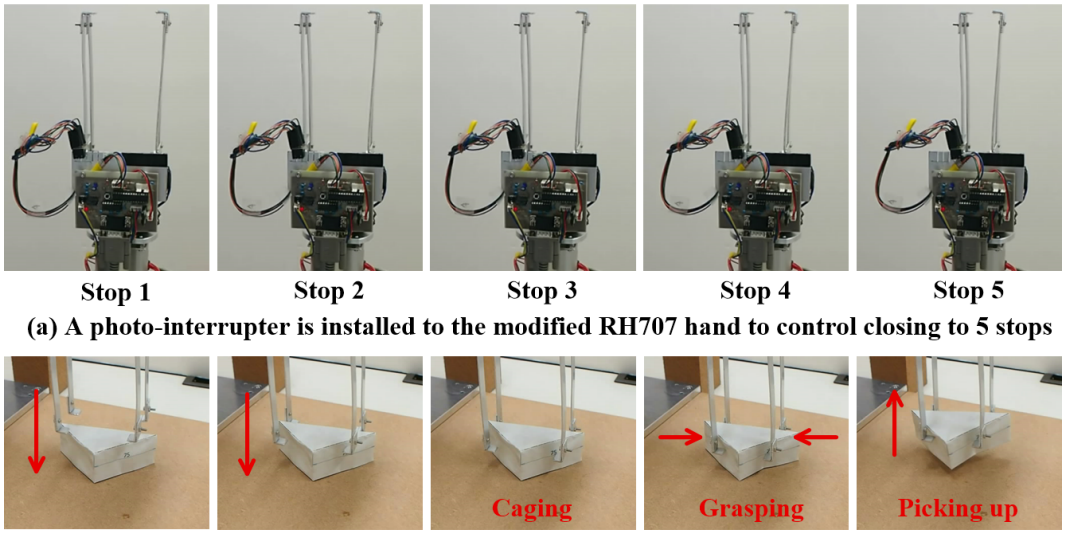


Figure 6.14: Five stops of the modified RH707 hand and the "grasping by caging" procedure of a triangle object.

Besides the triangular object, I make demonstrations with several other objects to show the superiority of the design and the robust caging algorithm. Fig.6.15 shows the details of the other objects. More complete demonstrations have been compiled into a video attachment accompanying this thesis. The video not only includes caging and grasping by caging tasks of various objects like boxes, octagons and concave polytopes but also includes lots of comparisons with other designs and failures. I strongly recommend readers refer to the video to better comprehend the design, especially the fingers settings and caging.

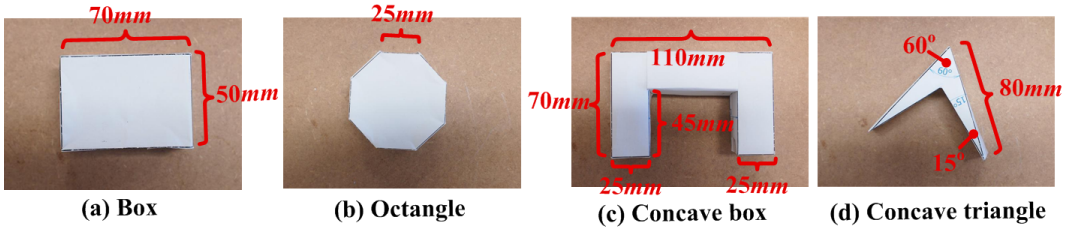


Figure 6.15: The other four objects used in the demonstration.

Part IV

In-depth $\mathcal{C}^{\mathsf{obj}}$ and $\mathcal{C}^{\mathsf{frm}}$

Chapter 7

In-depth analysis of $\mathcal{C}^{\mathsf{obj}}$ and $\mathcal{C}^{\mathsf{frm}}$

7.1 The Relationship Between $\mathcal{C}^{\mathsf{obj}}$ and $\mathcal{C}^{\mathsf{frm}}$

We have explored the caging problem in two spaces. The first one is the configuration space of target object, namely $\mathcal{C}^{\mathsf{obj}}$. The second one is the configuration space of finger formation $\mathcal{C}^{\mathsf{frm}}$. These two spaces are actually equal to each other with a linear transformation. Let us analyze their relationship in this section. The symbols used here are the same as those used in previous contexts of this thesis.

7.1.1 The expressions of $\mathcal{C}^{\mathsf{obj}}_{\mathsf{otl}}$

In order to compare these two spaces, we discretize $\mathcal{C}^{\mathsf{obj}}$ into voxels like $\mathcal{C}^{\mathsf{frm}}$ and express $\mathcal{C}^{\mathsf{obj}}_{\mathsf{otl}}$ and $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$ as two sets of voxels and compare the relationship between them.

Firstly, I am going to deduce the expressions of $\mathcal{C}^{\mathsf{obj}}_{\mathsf{otl}}$. According to Fig.3.1, we need to choose a pivot point to build the correspondence between a $\mathcal{W}$ space object to a $\mathcal{C}^{\mathsf{obj}}$ configuration. This pivot point can be chosen arbitrarily so that I use the first coordinate of target object $v_{1_\mathsf{x}}, v_{1_\mathsf{y}}$ as the pivot point[1]. Consequently, the initial configuration of a target object is $\boldsymbol{q}^{\mathsf{obj}}_0 = \{q^{\mathsf{obj}}_{0_\mathsf{x}}, q^{\mathsf{obj}}_{0_\mathsf{y}}, q^{\mathsf{obj}}_{0_\theta}\} = \{v_{1_\mathsf{x}}, v_{1_\mathsf{y}}, 0\theta\}$. After one counter-clockwise rotation (CCW) layer, the configuration becomes $\boldsymbol{q}^{\mathsf{obj}}_1 = \{v_{1_\mathsf{x}}, v_{1_\mathsf{y}}, 1\theta\}$. Likewise, after one clockwise rotation layer (CW) the configuration becomes $\boldsymbol{q}^{\mathsf{obj}}_{-1} = \{v_{1_\mathsf{x}}, v_{1_\mathsf{y}}, -1\theta\}$.

If we denote all vertices of the target object with $\mathsf{V} = \{\{v_{1_\mathsf{x}}, v_{1_\mathsf{y}}\}, \{v_{2_\mathsf{x}}, v_{2_\mathsf{y}}\}, \ldots, \{v_{n_{v\mathsf{x}}}, v_{n_{v\mathsf{y}}}\}\}$, then the $\partial\mathsf{O}[\{v_{1\mathsf{x}}, v_{1\mathsf{y}}, 0\theta\}]$ (see Fig.3.1) can be expressed as following.

$$\begin{aligned} T^{(-v_{1_\mathsf{x}}, -v_{1_\mathsf{y}})}\mathsf{V} =& \{T^{(-v_{1_\mathsf{x}}, -v_{1_\mathsf{y}})}v_1, T^{(-v_{1_\mathsf{x}}, -v_{1_\mathsf{y}})}v_2, \ldots, T^{(-v_{1_\mathsf{x}}, -v_{1_\mathsf{y}})}v_m\} \\ =& \{v_1 - v_1, v_2 - v_1, \ldots, v_m - v_1\} \end{aligned} \tag{7.1}$$

[1]This is different from the contexts in Chapter 3. That is because in Chapter 3, I chose a pivot point different from fingers to make clear the illustrations. In contrast, I choose $v_{1_\mathsf{x}}, v_{1_\mathsf{y}}$ in this section to make it coherent with our deduction of $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$.

$$\begin{aligned}\partial\mathsf{O}[\{v_{1\mathsf{x}}, v_{1\mathsf{y}}, 0\theta\}] &= R^{(0\theta)}T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}\mathsf{V} \\ &=\{R^{(0\theta)}T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}\boldsymbol{v_1}, R^{(0\theta)}T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}\boldsymbol{v_2}, \ldots, R^{(0\theta)}T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}\boldsymbol{v_{n_v}}\}\end{aligned} \tag{7.2}$$

Here, $T^{(\delta_x,\delta_y)}$ means the translation matrix with respect to a shifting pair (δ_x, δ_y). The shifting pair indicates the shift between two grids of $\mathcal{W}$ space, namely $(\delta_x, \delta_y) = \omega_i - \omega_j = (\omega_{i_\mathsf{x}} - \omega_{j_\mathsf{x}}, \omega_{i_\mathsf{y}} - \omega_{j_\mathsf{y}})$. $T^{(\delta_x,\delta_y)}$ can be written as a matrix in the following form.

$$T^{(\delta_x,\delta_y)} = T^{(\omega_i-\omega_j)} = \begin{bmatrix} 1 & 0 & \omega_{i_\mathsf{x}} - \omega_{j_\mathsf{x}} \\ 0 & 1 & \omega_{i_\mathsf{y}} - \omega_{j_\mathsf{y}} \\ 0 & 0 & 1 \end{bmatrix} \tag{7.3}$$

The expression (7.1) moves rotation center to the pivot point $\boldsymbol{v}_1$. Then, expression (7.2) rotates the target object vertices according to its orientation. Since expression (7.2) aims to express $\mathsf{O}[\{v_{1\mathsf{x}}, v_{1\mathsf{y}}, 0\theta\}]$, the rotation matrix is set to $R^{(0\theta)}$. At another layer, say layer 1, expression (7.2) would become $\partial\mathsf{O}[\{v_{1\mathsf{x}}, v_{1\mathsf{y}}, 0\theta\}] = R^{(1\theta)}T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}\mathsf{V}$.

Based on these expressions and the expressions (3.2), (3.3), (3.4) and (3.6) in Chapter 3, we can express $\mathcal{F}_1[q^{\mathsf{obj}}_{i_\theta}]$ as expression (7.4). Note that there is nothing new here. Expression (7.4) is actually exactly the same as expression (3.6) except that I change $\partial\mathsf{O}[\{f_{1\mathsf{x}}, f_{1\mathsf{y}}, q^{\mathsf{obj}}_{j_\theta}\}]$ into the form of vertices.

$$\mathcal{F}_1[q^{\mathsf{obj}}_{i_\theta}] = T^{(f_{1\mathsf{x}},f_{1\mathsf{y}})}R^{\pi}R^{(i\theta)}T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}\mathsf{V} \tag{7.4}$$

Consequently, the whole $\mathcal{F}_1$ can be expressed as the union of all layers, namely $\bigcup_{i=-m}^{m}\mathcal{F}_1[q^{\mathsf{obj}}_{i_\theta}]$. Expression (7.5) shows it. Like the relationship between expression (7.4) and expression (3.6), the expression (7.5) is exactly the same as expression (3.7).

$$\bigcup_{i=-m}^{m} T^{(f_{1\mathsf{x}},f_{1\mathsf{y}})}R^{\pi}R^{(i\theta)}T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}\mathsf{V} \tag{7.5}$$

Likewise, the $\mathcal{F}^{\mathsf{obj}}_j$ of any other finger f_j is composed of

$$\bigcup_{i=-m}^{m} T^{(f_{j\mathsf{x}},f_{j\mathsf{y}})}R^{\pi}R^{(i\theta)}T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}\mathsf{V},\ j = 2, 3, \ldots, m \tag{7.6}$$

In summary, $\mathcal{C}^{\mathsf{obj}}_{\mathsf{otl}}$ is the set of voxels enclosed by expression (7.5) and (7.6).

7.1.2 The expressions of $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$

Then, let us deduce the expressions of $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$. According to Fig.5.2, we denote the configuration of finger formation by the coordinates of its first finger and the orientation of the whole finger formation. Therefore, the initial configuration of a finger formation is $\boldsymbol{q}^{\mathsf{frm}}_0 = \{f_{1\mathsf{x}}, f_{1\mathsf{y}}, 0\theta\}$. In accordance with the rules in rotating a target object, after one CCW rotation layer, the

configuration of finger formation rotates reversely and becomes $\{f_{1_\mathsf{x}}, f_{1_\mathsf{y}}, 1\theta\}$. After one CW rotation step, it becomes $\{f_{1_\mathsf{x}}, f_{1_\mathsf{y}}, -1\theta\}$. Like $\partial\mathsf{O}[\{v_{1_\mathsf{x}}, v_{1_\mathsf{y}}, 0\theta\}]$, the $\mathsf{F}[\{f_{1_\mathsf{x}}, f_{1_\mathsf{y}}, 0\theta\}]$ can be expressed by the following two expressions where the first one is to move rotation center to the first finger position while the second one is to rotate the finger formation according to its orientation.

$$\begin{aligned} T^{(-f_{1_\mathsf{x}}, -f_{1_\mathsf{y}})}\mathsf{F} =& \{T^{(-f_{1_\mathsf{x}}, -f_{1_\mathsf{y}})} f_1, T^{(-f_{1_\mathsf{x}}, -f_{1_\mathsf{y}})} f_2, \ldots, T^{(-f_{1_\mathsf{x}}, -f_{1_\mathsf{y}})} f_{n_f}\} \\ =& \{\boldsymbol{f}_1 - \boldsymbol{f}_1, \boldsymbol{f}_2 - \boldsymbol{f}_1, \ldots, \boldsymbol{f}_{n_f} - \boldsymbol{f}_1\} \end{aligned} \tag{7.7}$$

$$\begin{aligned} &\mathsf{F}[\{f_{1_\mathsf{x}}, f_{1_\mathsf{y}}, 0\theta\}] = R^{(0\theta)} T^{(-f_{1_\mathsf{x}}, -f_{1_\mathsf{y}})}\mathsf{F} \\ =& \{R^{(0\theta)} T^{(-f_{1_\mathsf{x}}, -f_{1_\mathsf{y}})} \boldsymbol{f}_1, R^{(0\theta)} T^{(-f_{1_\mathsf{x}}, -f_{1_\mathsf{y}})} \boldsymbol{f}_2, \ldots, R^{(0\theta)} T^{(-f_{1_\mathsf{x}}, -f_{1_\mathsf{y}})} \boldsymbol{f}_{n_f}\} \end{aligned} \tag{7.8}$$

At another layer, say layer 1, expression (7.8) would become $\mathsf{F}[\{f_{1_\mathsf{x}}, f_{1_\mathsf{y}}, 0\theta\}] = R^{(1\theta)} T^{(-f_{1_\mathsf{x}}, -f_{1_\mathsf{y}})}\mathsf{F}$.

We can also express $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$ according to specific fingers. This procedure has been discussed when we were building the helical pattern in Fig.5.12 and Fig.5.13. The first component of $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$ involves the configurations (namely voxels) that fulfills the following two conditions. (1) The formation rotate around finger $\boldsymbol{f}_1$. (2) When the formation is at those configurations, $\boldsymbol{f}_1$ in inside the target object, namely it is inside V. Here the first condition ensures that this component is correspondent with $\boldsymbol{f}_1$ while the second condition ensures that the configurations belong to $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$. This first component can therefore be expressed as following.

$$\bigcup_{i=-m}^{m} \mathsf{V} \tag{7.9}$$

The other components which correspond to the other fingers besides $\boldsymbol{f}_1$ are a bit complicated. Take a finger $\boldsymbol{f}_j$ for example. Like $\boldsymbol{f}_1$, its correspondent component of $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$ involves the configurations that fulfills the following two conditions. (1) The formation rotate around finger $\boldsymbol{f}_j$. (2) When the formation is at those configurations, $\boldsymbol{f}_j$ in inside the target object, namely it is inside V. The second condition, according to Fig.5.15, can be converted to the rotation around $\boldsymbol{f}_j$ plus the translation $\boldsymbol{f}_1 - \boldsymbol{f}_j$ (It was $\boldsymbol{f}_1 - \boldsymbol{f}_j + \boldsymbol{f}_1$ in Fig.5.15 because we only consider the grid that was occupied by $\boldsymbol{f}_1$.). Therefore, the component that correspond to a finger $\boldsymbol{f}_j$ can be expressed by the following expression.

$$\bigcup_{i=-m}^{m} \begin{bmatrix} 1 & 0 & \\ 0 & 1 & R^{(-i\theta)} \\ 0 & 0 & \end{bmatrix} \begin{bmatrix} f_{1_\mathsf{x}} - f_{j_\mathsf{x}} \\ f_{1_\mathsf{y}} - f_{j_\mathsf{y}} \\ 1 \end{bmatrix} \mathsf{V},\ j = 2, 3, \ldots, n_f \tag{7.10}$$

Consequently, $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$ is the set of voxels enclosed by expression (7.9) and (7.10).

7.1.3 The relationship between $\mathcal{C}^{\mathsf{obj}}_{\mathsf{otl}}$ and $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$

Now let us compare the relationship between $\mathcal{C}^{\mathsf{obj}}_{\mathsf{otl}}$ and $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$ and make clear their relationship.

Consider the expression between a layer in $\mathcal{F}_i^{\mathsf{obj}}$ and a layer in the first component of $\mathcal{F}^{\mathsf{frm}}$,

$$\bigcup_{i=-m}^{m} T^{(f_{1\mathsf{x}},f_{1\mathsf{y}})} R^{\pi} R^{(i\theta)} T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})} \mathsf{V} \quad \text{and} \quad \bigcup_{i=-m}^{m} \mathsf{V} \tag{7.11}$$

They essentially differ with a linear transformation $T^{(f_{1\mathsf{x}},f_{1\mathsf{y}})} R^{\pi} R^{(i\theta)} T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}$. If the expressions

$$\begin{gathered} \bigcup_{i=-m}^{m} T^{(f_{j\mathsf{x}},f_{j\mathsf{y}})} R^{\pi} R^{(i\theta)} T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})} \mathsf{V},\ j = 2,3,\ldots,m \\ \text{and} \\ \bigcup_{i=-m}^{m} \begin{bmatrix} 1 & 0 & \\ 0 & 1 & R^{(-i\theta)} \\ 0 & 0 & \end{bmatrix} \begin{bmatrix} f_{1\mathsf{x}} - f_{j\mathsf{x}} \\ f_{1\mathsf{y}} - f_{j\mathsf{y}} \\ 1 \end{bmatrix} \mathsf{V},\ j = 2,3,\ldots,n_f \end{gathered} \tag{7.12}$$

share the same difference, we can draw a conclusion that $\mathcal{C}_{\mathsf{otl}}^{\mathsf{obj}}$ and $\mathcal{C}_{\mathsf{otl}}^{\mathsf{frm}}$ can be converted in layer level by a linear transformation $T^{(f_{1\mathsf{x}},f_{1\mathsf{y}})} R^{\pi} R^{(i\theta)} T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}$.

By left-multiplying $\begin{bmatrix} 1 & 0 & \\ 0 & 1 & R^{(-i\theta)} \\ 0 & 0 & \end{bmatrix} \begin{bmatrix} f_{1\mathsf{x}} - f_{j\mathsf{x}} \\ f_{1\mathsf{y}} - f_{j\mathsf{y}} \\ 1 \end{bmatrix} \mathsf{V}$ with the difference $T^{(f_{1\mathsf{x}},f_{1\mathsf{y}})} R^{\pi} R^{(i\theta)} T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}$, we can get

$$\begin{aligned} & T^{(f_{1\mathsf{x}},f_{1\mathsf{y}})} R^{\pi} R^{(i\theta)} T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})} \begin{bmatrix} 1 & 0 & \\ 0 & 1 & R^{(-i\theta)} \\ 0 & 0 & \end{bmatrix} \begin{bmatrix} f_{1\mathsf{x}} - f_{j\mathsf{x}} \\ f_{1\mathsf{y}} - f_{j\mathsf{y}} \\ 1 \end{bmatrix} \mathsf{V} \\ =& T^{(f_{1\mathsf{x}},f_{1\mathsf{y}})} R^{\pi} R^{(i\theta)} \begin{bmatrix} 1 & 0 & \\ 0 & 1 & R^{(-i\theta)} \\ 0 & 0 & \end{bmatrix} \begin{bmatrix} f_{1\mathsf{x}} - f_{j\mathsf{x}} \\ f_{1\mathsf{y}} - f_{j\mathsf{y}} \\ 1 \end{bmatrix} T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})} \mathsf{V} \end{aligned} \tag{7.13}$$

since $\begin{bmatrix} 1 & 0 & \\ 0 & 1 & R^{(-i\theta)} \\ 0 & 0 & \end{bmatrix} \begin{bmatrix} f_{1\mathsf{x}} - f_{j\mathsf{x}} \\ f_{1\mathsf{y}} - f_{j\mathsf{y}} \\ 1 \end{bmatrix}$ is a translation matrix in the form of $T^(\delta x, \delta y)$ and it is inter-changeable with $T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}$.

Considering that $\forall\phi,\ \forall\delta x$ and $\forall\delta y$

$$R^{\phi} T^{(\delta x,\delta y)} = \begin{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} R^{\phi} \begin{bmatrix} \delta x \\ \delta y \\ 1 \end{bmatrix} \end{bmatrix} R^{\phi} \tag{7.14}$$

expression (7.13) becomes

$$
\begin{aligned}
&T^{(f_{1\mathsf{x}},f_{1\mathsf{y}})}R^{\pi}R^{(i\theta)}\begin{bmatrix}1&0&\\0&1&R^{(-i\theta)}\\0&0&\end{bmatrix}\begin{bmatrix}f_{1_\mathsf{x}}-f_{j_\mathsf{x}}\\f_{1_\mathsf{y}}-f_{j_\mathsf{y}}\\1\end{bmatrix}\Bigg]T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}\mathsf{V}\\
=&T^{(f_{1\mathsf{x}},f_{1\mathsf{y}})}R^{(\pi+i\theta)}\begin{bmatrix}1&0&\\0&1&R^{(-i\theta)}\\0&0&\end{bmatrix}\begin{bmatrix}f_{1_\mathsf{x}}-f_{j_\mathsf{x}}\\f_{1_\mathsf{y}}-f_{j_\mathsf{y}}\\1\end{bmatrix}\Bigg]T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}\mathsf{V}\\
=&T^{(f_{1\mathsf{x}},f_{1\mathsf{y}})}\begin{bmatrix}1&0&\\0&1&R^{(\pi+i\theta)}R^{(-i\theta)}\\0&0&\end{bmatrix}\begin{bmatrix}f_{1_\mathsf{x}}-f_{j_\mathsf{x}}\\f_{1_\mathsf{y}}-f_{j_\mathsf{y}}\\1\end{bmatrix}\Bigg]R^{(\pi+i\theta)}T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}\mathsf{V}\\
=&T^{(f_{1\mathsf{x}},f_{1\mathsf{y}})}\begin{bmatrix}1&1&\\0&0&T^{(f_{j\mathsf{x}}-f_{1\mathsf{x}},f_{j\mathsf{y}}-f_{1\mathsf{y}})}\\0&0&\end{bmatrix}\begin{bmatrix}0\\0\\1\end{bmatrix}\Bigg]R^{(\pi+i\theta)}T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}\mathsf{V}
\end{aligned}
\tag{7.15}
$$

Since the first and second column of the translation matrix $T^(\delta x, \delta y)$ are $[1\ 0\ 0]'$ and $[0\ 1\ 0]'$ respectively,

$$
\begin{aligned}
&T^{(f_{1\mathsf{x}},f_{1\mathsf{y}})}\begin{bmatrix}1&1&\\0&0&T^{(f_{j\mathsf{x}}-f_{1\mathsf{x}},f_{j\mathsf{y}}-f_{1\mathsf{y}})}\\0&0&\end{bmatrix}\begin{bmatrix}0\\0\\1\end{bmatrix}\Bigg]R^{(\pi+i\theta)}T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}\mathsf{V}\\
=&T^{(f_{1\mathsf{x}},f_{1\mathsf{y}})}\left[T^{(f_{j\mathsf{x}}-f_{1\mathsf{x}},f_{j\mathsf{y}}-f_{1\mathsf{y}})}\begin{bmatrix}1\\0\\0\end{bmatrix}T^{(f_{j\mathsf{x}}-f_{1\mathsf{x}},f_{j\mathsf{y}}-f_{1\mathsf{y}})}\begin{bmatrix}1\\0\\0\end{bmatrix}T^{(f_{j\mathsf{x}}-f_{1\mathsf{x}},f_{j\mathsf{y}}-f_{1\mathsf{y}})}\begin{bmatrix}0\\0\\1\end{bmatrix}\right]\\
&R^{(\pi+i\theta)}T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}\mathsf{V}\\
=&T^{(f_{1\mathsf{x}},f_{1\mathsf{y}})}T^{(f_{j\mathsf{x}}-f_{1\mathsf{x}},f_{j\mathsf{y}}-f_{1\mathsf{y}})}R^{(\pi+i\theta)}T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}\mathsf{V}\\
=&T^{(f_{j\mathsf{x}},f_{j\mathsf{y}})}R^{(\pi+i\theta)}T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}\mathsf{V}\\
=&T^{(f_{j\mathsf{x}},f_{j\mathsf{y}})}R^{\pi}R^{(i\theta)}T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}\mathsf{V}
\end{aligned}
\tag{7.16}
$$

The result is exactly the same as expression (7.6) so that we are confirmed that at each layer $\mathcal{C}^{\mathsf{obj}}_{\mathsf{otl}}$ and $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$ differ with a linear transformation $T^{(f_{1\mathsf{x}},f_{1\mathsf{y}})}R^{\pi}R^{(i\theta)}T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}$. Fig.7.1 graphically illustrates the relationship between layers of the two spaces. The dashed polygon shape (one slice of $\mathcal{C}^{frm}$) in the right part of Fig.7.1 can be converted to the dashed polygon shape (one slice of $\mathcal{C}^{obj}$) in the left part of Fig.7.1 by multiplying a linear transformation matrix $T^{(f_{1\mathsf{x}},f_{1\mathsf{y}})}R^{\pi}R^{(i\theta)}T^{(-v_{1\mathsf{x}},-v_{1\mathsf{y}})}$.

This conclusion indicates that the difference of the two spaces are their metrics which are essential to caging robustness. In $\mathcal{C}^{\mathsf{obj}}$, our metrics aim to measure the distance from caging sub-space to free sub-space (see the left part of Fig.7.2). However, due to high computational cost, the metrics were difficult to be employed explicitly so that I decomposed the constraints into translational caging together with a rotational component for approximation. In $\mathcal{C}^{\mathsf{frm}}$, the metrics become clear and we combine the distances to critical surfaces as the quality

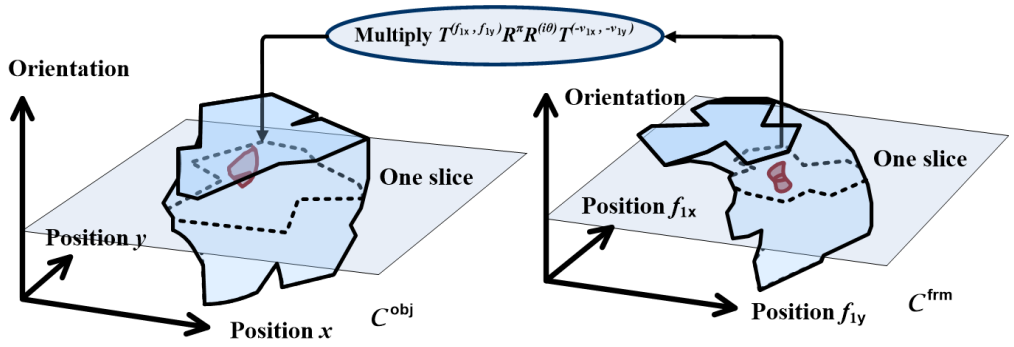


Figure 7.1: The slices in $\mathcal{C}^{frm}$ and $\mathcal{C}^{obj}$ can be converted to each other by a linear transformation.

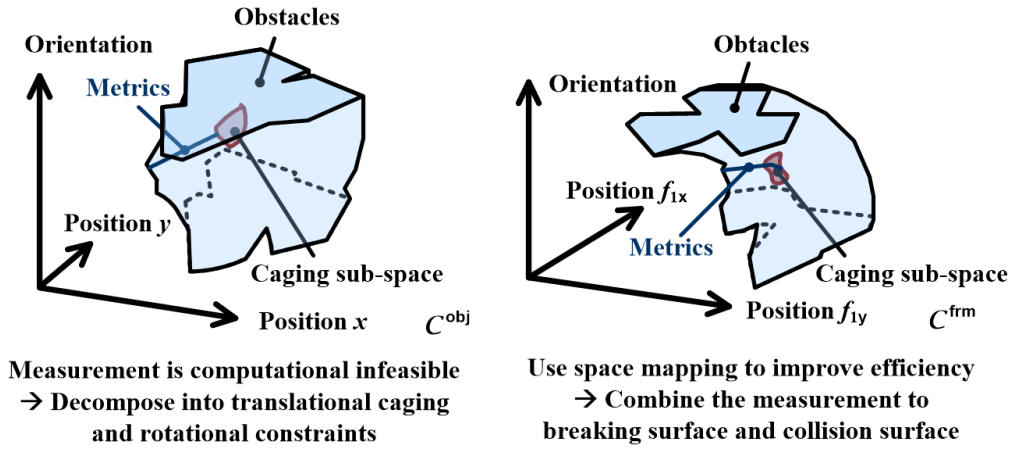


Figure 7.2: The metrics of the two spaces are essentially different.

function (see the right part Fig.7.2). If we analyze the relationship between the metrics in these two spaces in detail, it could be in the following form. Note that there is no transformation along rotational axes, namely the transformation between $\mathcal{C}^{\text{frm}}$ and $\mathcal{C}^{\text{obj}}$ are independent of the rotation. Take a 3-D vector $\overrightarrow{u_{ij}} = \begin{bmatrix} u_{i_x} - u_{j_x} \\ u_{i_y} - u_{j_y} \\ \theta_i - \theta_j \end{bmatrix}$ in $\mathcal{C}^{\text{frm}}$ for example. A metric with weighting matrix W in $\mathcal{C}^{\text{frm}}$ can be expressed by

$$\overrightarrow{u_{ij}}' W \overrightarrow{u_{ij}} \tag{7.17}$$

In $\mathcal{C}^{\text{obj}}$, it becomes the following expression.

$$\begin{array}{l} \left[\left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \theta_i - \theta_j \end{array} \right] T^{(f_{1\mathsf{x}}, f_{1\mathsf{y}})} R^{\pi} R^{(i\theta)} T^{(-v_{1\mathsf{x}}, -v_{1\mathsf{y}})} \left[\begin{array}{c} u_{i\mathsf{x}} - u_{j\mathsf{x}} \\ u_{i\mathsf{y}} - u_{j\mathsf{y}} \\ 1 \end{array} \right] \right]' \\ W \left[\left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \theta_i - \theta_j \end{array} \right] T^{(f_{1\mathsf{x}}, f_{1\mathsf{y}})} R^{\pi} R^{(i\theta)} T^{(-v_{1\mathsf{x}}, -v_{1\mathsf{y}})} \left[\begin{array}{c} u_{i\mathsf{x}} - u_{j\mathsf{x}} \\ u_{i\mathsf{y}} - u_{j\mathsf{y}} \\ 1 \end{array} \right] \right] \end{array} \tag{7.18}$$

With the same weighting matrix W, the metrics in these two spaces are quite different from each other. **The metrics heavily relate to the quality functions and the robustness of caging. From this viewpoint, $\mathcal{C}^{\mathsf{frm}}$, comparing with $\mathcal{C}^{\mathsf{obj}}$, offers a more intuitive way to choose satisfying metrics.**

7.2 Choosing Proper Algorithms

Now we have explored the caging algorithms to solve the caging test problem and the caging optimization problem in two spaces. One is the configuration space of target object, namely $\mathcal{C}^{\mathsf{obj}}$. The other one is the configuration space of finger formation, namely $\mathcal{C}^{\mathsf{obj}}$. Both the two spaces have some advantages and disadvantages. They can be summarized as Fig.7.3.

Figure 7.3: Comparing the advantages and disadvantages of the algorithms in $\mathcal{C}^{\mathsf{obj}}$ and $\mathcal{C}^{\mathsf{frm}}$.

There are two limitations to the algorithms in $\mathcal{C}^{\mathsf{obj}}$. Firstly, they are limited to convex objects. Then, some of their parameters like combination of Q^{T} and Q^{R} and the retraction of fingers are chosen empirically. However, the algorithms in $\mathcal{C}^{\mathsf{obj}}$ are suitable to be used in fully distributed applications like our distributed end-effector or the multi-robot co-operative transportation. That is because in these applications, each finger or mobile robot can be distributedly actuated to any position and there is no apparent eigen-shapes. Their potential finger formations or mobile robot formations are infinite. It is difficult to recover and update every $\mathcal{C}^{\mathsf{frm}}$ of those infinite formations. $\mathcal{C}^{\mathsf{obj}}$ is therefore more suitable than $\mathcal{C}^{\mathsf{frm}}$.

Comparing with $\mathcal{C}^{\mathsf{obj}}$, $\mathcal{C}^{\mathsf{frm}}$ is not limited to convex objects. It can be applied to arbitrary 2D objects, including either convex, concave objects or even objects with hollow holes. Moreover, we do not need the empirical parameters like $\mathcal{C}^{\mathsf{obj}}$. The algorithms in $\mathcal{C}^{\mathsf{frm}}$ is complete to discretization resolution. That is to say, the algorithms can always solve the caging problems as long as the resolution of grids is high enough. In contrast, the algorithms in $\mathcal{C}^{\mathsf{obj}}$ is not complete since it may fail and reject to cage certain objects due to empirical parameter settings. However, the algorithms in $\mathcal{C}^{\mathsf{frm}}$ requires us to maintain, say recover and update, a $\mathcal{C}^{\mathsf{frm}}$ for each finger formation. That means the number of formations is limited. Although the maintenance time of one $\mathcal{C}^{\mathsf{frm}}$ has been improved greatly by shifting pre-built mapping structures, the algorithms in $\mathcal{C}^{\mathsf{frm}}$ are most suitable to robotic hands with eigen-shapes or

robotic hands that can be represented by some pre-defined formations. It cannot cover as many formations as the algorithms in $\mathcal{C}^{\mathsf{obj}}$.

I proved in the last section that when the orientation is fixed, $\mathcal{C}^{\mathsf{obj}}_{\mathsf{otl}}$ and $\mathcal{C}^{\mathsf{frm}}_{\mathsf{otl}}$ can be converted to each other by a linear transformation. Namely, using $\mathcal{C}^{\mathsf{frm}}$ instead of $\mathcal{C}^{\mathsf{obj}}$ is essentially using different metrics. The metrics heavily relate to the quality functions and the robustness of caging. From that viewpoint, $\mathcal{C}^{\mathsf{frm}}$, comparing with $\mathcal{C}^{\mathsf{obj}}$, offers a more intuitive way to choose satisfying metrics.

At this point, we may have a question like this. How can we choose between them or can we combine them, preserving the merits and making up the weakness? I maintain that one may not smoothly combine the algorithms in these two spaces since they are essentially different in metrics. However, both $\mathcal{C}^{\mathsf{frm}}$ and $\mathcal{C}^{\mathsf{obj}}$ have disadvantages and advantages. I recommend using $\mathcal{C}^{\mathsf{obj}}$ and $\mathcal{C}^{\mathsf{frm}}$ separately according to mechanical structure of robots and tasks. If all capture points are distributed and target objects are convex, I recommend perform caging planning with the algorithms in $\mathcal{C}^{\mathsf{obj}}$. If capture points are kinematically constrained or target objects have various shapes, I recommend performing caging planning with the algorithms in $\mathcal{C}^{\mathsf{frm}}$. Fig.7.4 shows this idea with concrete illustration.

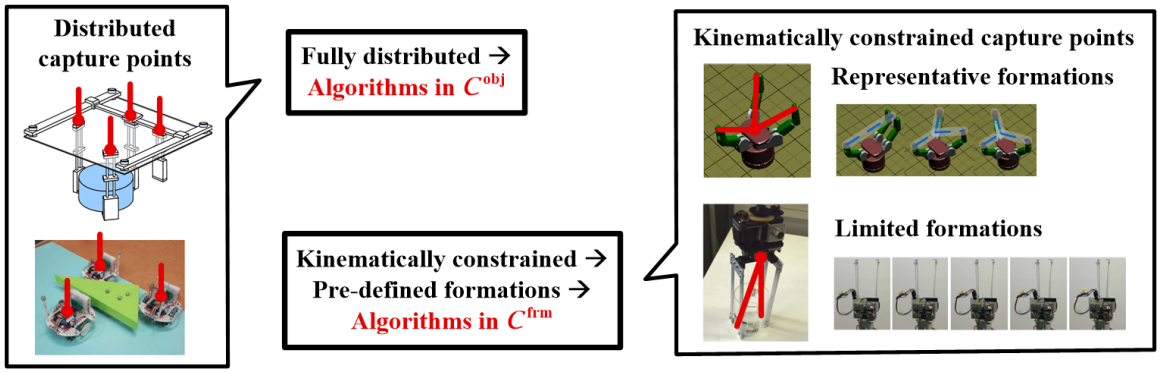


Figure 7.4: Choosing the proper algorithms according to real applications.

Chapter 8

Conclusions and Future Works

8.1 Summary of the Contents

This thesis proposes caging planning algorithms by using the configuration space of target object ($\mathcal{C}^{\mathsf{obj}}$) and the configuration space of finger formation ($\mathcal{C}^{\mathsf{frm}}$). It concludes that caging is the loose form of grasping so that it can be used to deal with uncertainty caused by perception and control. The effectiveness of the algorithms are demonstrated with simulation and real-world platforms.

The thesis is composed of eight chapters. Besides this 8th chapter, the other 7 chapters are as following.

Chapter 1 introduces the basic concepts of robotic manipulation and the basic structure of an end-effector collaboratively developed by me and some students in my group. Along with the introduction, this chapter explains how I started this research in caging and why I am confident that caging is promising. This chapter proposes that caging can be used to deal with uncertainty and can be a good supporting tool for the end-effector. It further defines the caging test and caging optimization problems and gives an overview of thesis organization. Specifically, caging test examines whether caging is attained or not while caging optimization finds a robust finger formation against uncertainty.

Chapter 2 reviews the traditional research topics in grasping closure. It makes clear that caging is the general form of grasping by connecting caging to such traditional grasping research topics as force closure, form closure and immobilization based on an intensive review of contemporary works in grasping closure. In detail, Fig.2.14 visualizes the shift between caging and traditional concepts in grasping in $\mathcal{C}^{\mathsf{obj}}$.

Chapters 3∼6 propose several algorithms to the caging problems in two different configuration spaces $\mathcal{C}^{\mathsf{obj}}$ and $\mathcal{C}^{\mathsf{frm}}$ (Chapter 3 and 5), and apply the algorithms to real robotic platforms (Chapter 4 and 6). Specifically, Chapter 3 proposes the caging algorithms by evaluating the regions in which fingers guarantee caging (caging region) based on the re-implementation of a previous work which aims to find the caging region of a third finger given a 2D convex target object and two boundary contacts. Then, it explores how to

push though the limitations of given fingers and extend the algorithms to this proposal to general cases. The solution is fixing the two boundary contacts alternatively and reducing the computational complexity of that algorithm by decomposing caging into translational constraints and rotational constraints. The proposed algorithms solve the caging problems of 2D convex objects. Their ability to endure uncertainty is demonstrated with simulation and real-world platforms like the distributed end-effector and multi-robot cooperative transportation in Chapter 4. Chapter 5 explores algorithms in $\mathcal{C}^{\mathsf{frm}}$ instead of $\mathcal{C}^{\mathsf{obj}}$. Since the center of $\mathcal{C}^{\mathsf{obj}}$ is the target object, it is inherently affected by object shapes. Unlike $\mathcal{C}^{\mathsf{obj}}$, $\mathcal{C}^{\mathsf{frm}}$ is the configuration space of finger formation and is free from the affection of object shapes. It therefore can be more flexible to various objects, including convex, concave and even hollow ones. In detail, this chapter firstly compares $\mathcal{C}^{\mathsf{obj}}$ and $\mathcal{C}^{\mathsf{frm}}$. It discusses both the advantages of $\mathcal{C}^{\mathsf{frm}}$ and the difficulties of implementing caging algorithms in $\mathcal{C}^{\mathsf{frm}}$ and proposes to overcome the difficulties by introducing the space mapping idea. Raw space mapping and especially its faster version, the improved space mapping, make it possible to update the whole space of $\mathcal{C}^{\mathsf{frm}}$ completely and efficiently. Based on the updated $\mathcal{C}^{\mathsf{frm}}$, the algorithm can quickly find the caging candidates in the updated $\mathcal{C}^{\mathsf{frm}}$, locate the optimal caging configuration and solve the caging problems. The caging algorithms in $\mathcal{C}^{\mathsf{frm}}$ are applied to the design and implementation of a gripping hand in Chapter 6. The algorithms play role in both design and implementation procedures. During design, the algorithms are employed to simplify and evaluate deign models. During implementation, the algorithms are employed to control the hand to cage and grasp objects. The design and implementation demonstrate advantages of the algorithms.

Chapter 7 explains the relationships of the algorithms and discusses how to select them. It proves that at different orientations $\mathcal{C}^{\mathsf{frm}}$ is the linear transformation of $\mathcal{C}^{\mathsf{obj}}$. Consequently, the metrics used in $\mathcal{C}^{\mathsf{frm}}$ and $\mathcal{C}^{\mathsf{obj}}$ are different. Both the two tools and their correspondent algorithms have reasons to exist. They therefore should be treated in parallel and selected according to requirements of specific applications. The algorithms in $\mathcal{C}^{\mathsf{obj}}$ are suitable to fully distributed capture points but they are limited by object shapes since they are inherently affected by target objects. In contrast, the algorithms in $\mathcal{C}^{\mathsf{frm}}$ are suitable to any 2D object shape but they are subject to limited number of finger formations since they are inherently affected by formations.

8.2 Contributions

The contributions of the thesis can be summarized in Fig.8.1.

As is shown in Fig.8.1, this thesis contributes in theoretical, algorithmic and application aspects. In theoretical contributions, it initially explains the relationship between caging and traditional research in grasping closure. Namely, caging is the extension of immobilization. In algorithmic contributions, it initially uses caging to deal with uncertainty. The thesis on the one hand proposes some rapid algorithms to deal with caging test while on the other hand proposes the caging optimization problem and a series of solutions in both $\mathcal{C}^{\mathsf{obj}}$ and

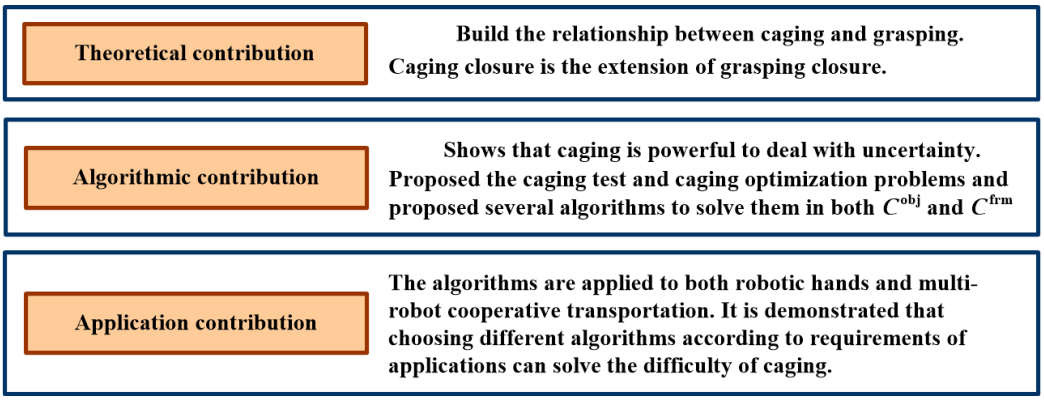


Figure 8.1: Contributions of the thesis.

$\mathcal{C}^{\mathsf{frm}}$. In application contributions, it applies the algorithms to robotic hands and multi-robot cooperative transportation. The applications based on the caging algorithms can not only robustly deal with various uncertainty but also help choose least or proper number of fingers or mobile robots.

The algorithms in $\mathcal{C}^{\mathsf{obj}}$ are limited to convex objects. However, they are suitable to be used in fully distributed applications like our distributed end-effector or the multi-robot co-operative transportation. That is because in these applications, each finger or mobile robot can be distributedly actuated to any position and there is no apparent eigen-shapes. Their potential finger formations or mobile robot formations are infinite. It is difficult to recover and update every $\mathcal{C}^{\mathsf{frm}}$ of those infinite formations.

Comparing with $\mathcal{C}^{\mathsf{obj}}$, $\mathcal{C}^{\mathsf{frm}}$ is not limited to convex objects. It can be applied to convex, concave objects and even objects with hollow holes. However, we have to maintain, say recover and update, a $\mathcal{C}^{\mathsf{frm}}$ for each finger formation. That means the number of formations is limited. Although the maintenance time of one $\mathcal{C}^{\mathsf{frm}}$ has been improved greatly by shifting pre-built mapping structures, it is most suitable to robotic hands with eigen-shapes and it cannot cover as many formations as the algorithms in $\mathcal{C}^{\mathsf{obj}}$.

Both the algorithms in $\mathcal{C}^{\mathsf{obj}}$ and $\mathcal{C}^{\mathsf{frm}}$ have their advantages and disadvantages. They actually correspond to different solutions of **geometric modeling**. The algorithm in $\mathcal{C}^{\mathsf{obj}}$ uses **wireframe modeling** while the algorithm in $\mathcal{C}^{\mathsf{frm}}$ uses **solid modeling**. Both modeling technology plays important roles in **geometric modeling** and either algorithms in $\mathcal{C}^{\mathsf{obj}}$ and $\mathcal{C}^{\mathsf{frm}}$ should exist. I demonstrate their relationship and explain how to choose properly between them in Chapter 7. Nevertheless, reader may have found that we can also use the solid modeling technology of $\mathcal{C}^{\mathsf{frm}}$ in $\mathcal{C}^{\mathsf{obj}}$. That is possible. However, that would increase the cost of evaluating one formation in $\mathcal{C}^{\mathsf{obj}}$, making it lose its advantages. The solid modeling technology is working efficiently in $\mathcal{C}^{\mathsf{frm}}$ because the formations and grids imply an unique mapping structure. Simple shifting and addition computation could help recover and update $\mathcal{C}^{\mathsf{frm}}$. The center of $\mathcal{C}^{\mathsf{obj}}$ is the target object and there is no underlying unique mapping structure.

The concepts and algorithms proposed in this thesis can efficiently solve 2D caging problems in the presence of uncertainty. I believe that caging is a promising tool to deal with perception and control uncertainty and would like to explore more about this tool in both theory and application aspects in the future. Some potential future directions can be found in the following texts.

8.3 Further Development of Caging Algorithms and Prospective Application Fields

The future work could involve two aspects. One is further development of the caging algorithms. The other is using it to challenge the difficulties caused by uncertainties in robotics.

8.3.1 Further development of caging algorithms

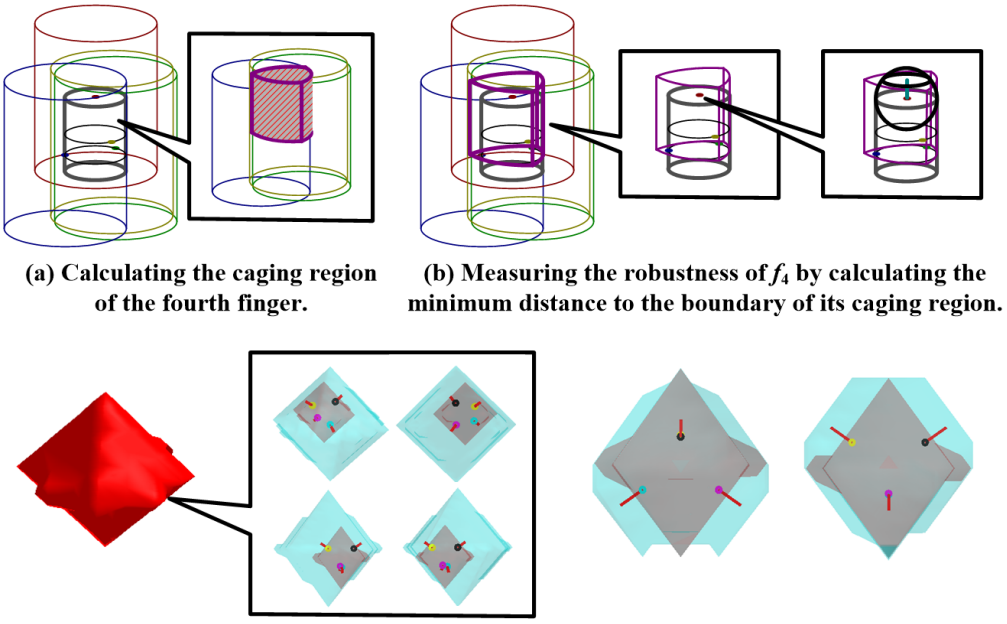
I have repeated lots of times that neither the algorithms in $\mathcal{C}^{\mathsf{frm}}$ nor the algorithms $\mathcal{C}^{\mathsf{obj}}$ are perfect. If one can develop an algorithm that can quickly find an optimized caging with no limitation in the number of finger formations and the shape of target objects, that would be an impressive contribution to caging. However, I wonder whether that impressive algorithm exists with current computers.

Actually, like many researchers in robotics, I believe in data warehouse and cloud computing rather than exact and perfect algorithms. The intelligence of machines should not be limited to on-board computing, but rely to (1) grid computing and (2) large-scale database. Of course, the robots based on data warehouse and cloud computing are limited to those supporting resources. Nevertheless, I believe this solution would succeed. Resources on the cloud shall one day make robots more intelligent than human beings and of course, solves caging problems. In the future, giant corporations would run their own supporting data warehouse and cloud computing resources and offer service to their terminal robots. Robots would be no more than thin hardware clients. Actually, we can already see some publications that are working into this direction. In this year's top conference on robotics, namely ICRA2013, Ben from Prof. Goldberg's group at the University of California, Berkeley published a robot grasping work by using google object recognition engine [Kehoe et al., 2013]. It is motivating and I will continue tracking their publications to see their ensuing steps.

Another future development of the algorithms is how to deal with 2.5D objects. In this thesis, all discussion concentrates on 2D objects. We did not discuss about 2.5D or 3D objects. Actually, the algorithms in $\mathcal{C}^{\mathsf{obj}}$ can be extended to 3D objects intuitively. Fig.8.2 illustrates the extension.

Fig.8.2(a) shows the extension of translational caging on 3D object with a simple cylinder object. I use this simple cylinder because it is more readable comparing with other complicated objects. Given three fingers f_1, f_2 and f_3, the translational caging region of f_4 could be rendered as the intersection of the $\mathcal{F}_{\mathcal{F}}$ of f_1, f_2 and f_3. This is exactly the same as the case of calculating the translational caging region of a third finger with 2D objects. Readers

may compare the shadowed region in Fig.8.2(a) and Fig.3.8 to understand their similarity. After calculating the caging region of f_4, we can measure the robustness of f_4 by referring to its distance to the boundary of its caging region. The distance of the cylinder object is shown in Fig.8.2(b). Then, like Fig.3.23, we can evaluate the robustness of a finger formation by alternatively fixing adjacent fingers and by recording the smallest robustness of a single finger. The remaining steps are exactly the same as 2D cases shown in Fig.3.26. Namely adding rotational constraints and retract fingers. This extension, of course, is still limited to convex shapes. I implemented the extension with a regular octahedron. Fig.8.2(c) shows the result of my implementation. Since regular octahedron is regular, the optimized fingers locate exactly at the center of each surface. Readers may refer the points and segments in Fig.8.2(c) for details. Here the points denote point fingers while the segments denotes surface normals.



(a) Calculating the caging region of the fourth finger.

(b) Measuring the robustness of f_4 by calculating the minimum distance to the boundary of its caging region.

(c) Results of the extension on a 3D regular octahedron.

Figure 8.2: The algorithms in $\mathcal{C}^{\text{obj}}$ can be extended to 3D objects intuitively.

This extension and implementation were neither further explored nor included in the major contents of this thesis because I think caging 3D objects with point fingers is impractical. General robotic hands, even if they are dexterous, can hardly fulfill the kinematic requirements of an optimized caging formations. However, discussing about 2.5D objects deserves efforts. Dealing with 2.5D objects means not only considering about top-view 2D shapes, but also consider the convexity and concavity of side views. Considering side views requires installing local sensors onto capture points. Some students in our group are working on that aspect. They installed infrared sensor arrays to the inner finger surfaces of the distributed end-effector and measure the shape of target object side views. Considering about side views

and dealing with 2.5D objects would not only stop target objects from escaping the finger formation but also stop target objects from "falling out of" the finger formation. Developing those algorithms would be a promising direction.

8.3.2 Perspective application fields

The second aspect of the future work is to improve the robotic applications which are hindered by the problems caused by uncertainties. An optimized caging offers robustness to both caging breaking and collision with target objects, and consequently offers robustness to uncertainties caused by perception and control. I believe that using the caging test and caging optimization algorithms developed in this thesis would greatly improve those hindered applications.

One example, which came into my view when I was exploring the problems caused by uncertainty, is the filed of micro/nano manipulation. When the scale is between $1\mu m$-$100\mu m$, people will name it micro-manipulation. When the scale is lower than $1\mu m$, people name it nano-manipulation. The manipulation in micro/nano world suffers from lots of uncertainties. On the one hand, perception in the micro/nano world is quite difficult. There is no existing method to simultaneously image and manipulate atoms. On the other hand, surface forces (e.g. electrostatic force, surface tension, van der Waals, casimir, etc) in the micro/nano world play dominating roles over gravity or inertia forces. It is difficult to model these forces mathematically. They are uncertain and deteriorate traditional control. Therefore, pure geometric solution like caging would probably play a promising role.

Like the macro-world applications in this thesis, some of the micro/nano manipulation systems are fully distributed while others are not. For example, manipulating objects with optical tweezers or micro-robots usually fall in the first category while manipulating objects with probe-tips or micro-grippers usually fall into the second category. Here are some works that relate to these categories. For instance, the bubble robots in [Hu et al., 2011] are examples of micro-robots. David's work [Cappelleri et al., 2012] and some earlier work of our group [Sato, 1996] are examples of probe-tips. The electro-thermally activated cell manipulator [Chronis and Lee, 2005] is an example of micro-manipulation. In the following part, I would like to discuss in detail about optical tweezers.

Optical tweezers [Grier, 2003][Onda and Arai, 2011] use two highly focused laser beams to trap very small crystal beads. That is possible owning to the attractive and repulsive forces caused by optic photons. Each bead can be viewed as a distributed capture point and researchers use many beads controlled by optical tweezers to co-operatively manipulate target objects in the micro-world. Like the multi-robot co-operative application in chapter 4, the optical tweezer-based secondary micro-manipulation suffers from uncertainty caused by formation control. Moreover, since controlling multiple beams requires more laser beams or more complicated control of spatial real-time modulator to modulate a single beam, researchers prefer less number of beads. Therefore, I strongly believe caging algorithms would play important roles in this field.

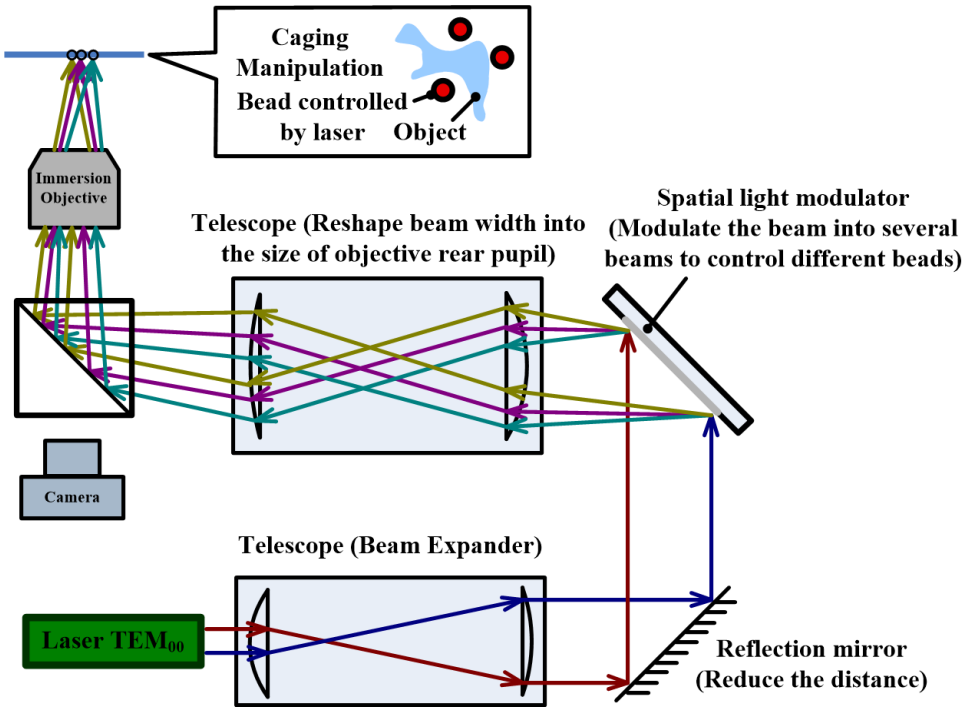


Figure 8.3: A typical micro-manipulation system (optical tweezers).

Fig.8.3 illustrates a typical micro-manipulation system. When spatial real-time modulators are employed, the number of formations of beads is usually limited to pre-programmed modulating patterns. In that case, the caging algorithms in $\mathcal{C}^{\mathsf{frm}}$ is preferable. When spatial real-time modulators are not employed, independent beads are distributedly controlled by independent laser beams. In that case, the caging algorithms in $\mathcal{C}^{\mathsf{obj}}$ is preferable. I am going to organize a workshop on "caging and its application in grasping/multi-agent cooperation" in November, 2013 during the International Conference on Intelligent Robots and Systems and discuss the possibility of these ideas with some researchers in micro/nano manipulation fields.

Another example is using caging to deal with in-hand manipulation. In-hand manipulation means to hold and move an object with one hand. It requires that the object should be constrained into the hand while being manipulated. The handle project [Handle Project, 2013] gives an excellent collection of state-of-the-art researches in this field.

However, state-of-the-art researches in in-hand manipulation aims at the manipulation of dexterous robotic hands which offer the possibility of rigid analysis of forces and force closures. In another word, the robots hold and move objects in hand by using forces. Using forces lead to the major difficulty of in-hand manipulation. That is, the state-of-the-art in-hand manipulation is hindered by the uncertainty from perception and control. Caging can be used to deal with uncertainty. Therefore, we have a perspective future research direction that can we do in-hand manipulation without forces by caging? If that is possible, we can

not only get merits from avoiding explicit force and control analysis but also do in-hand manipulation with simple hands [Mason et al., 2012] and reduce platform costs. Caging-based in-hand manipulation would greatly decrease the cost of deployment and real-world applications. This is a passionating topic and I am going to study further into this aspect in my postdoctoral research.

Caging is not only from the idea of bird cage but also from lots of nature creative (see the Venus flytrap plant[Wikipedia, 2013b][1]). I like caging, I believe in caging and I would like to spread caging to related research and applications. As the last sentence of my thesis, I would like to re-emphasize the merits of caging – Caging offers robustness to uncertainty.

[1]Plant suffers from perception/control uncertainty comparing with animals, they employ caging to complement the drawbacks caused by the uncertainty.

Bibliography

[Allen et al., 2012] Allen, T., Burdick, J., and Rimon, E. (2012). Two-fingered Caging of Polygons Via Contact-Space Graph Search. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 4183–4189.

[Barrett Technology, 2013] Barrett Technology (Last updated: 2013). The Barrett Hand. Last accessed: 2013. http://web.barrett.com/support/\{B}arrett{H}and_{D}ocumentation/{BH}8-280_{D}atasheet.pdf.

[Berenson, 2011] Berenson, D. (2011). *Constrained Manipulation Planning*. PhD thesis, Carnegie Mellon University.

[Berenson et al., 2007] Berenson, D., Diankov, R., Nishiwaki, K., Kagami, S., and Kuffner, J. (2007). Grasp Planning in Complex Scenes. In *Proceedings of IEEE International Conference on Humanoid Robots*.

[Berenson and Srinivasa, 2008] Berenson, D. and Srinivasa, S. (2008). Grasp Synthesis in Cluttered Environments for Dexterous Hands. In *Proceedings of IEEE International Conference on Humanoid Robots*.

[Berenson et al., 2009a] Berenson, D., Srinivasa, S., Ferguson, D., Collet Romea, A., and Kuffner, J. (2009a). Manipulation Planning with Workspace Goal Regions. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1397–1403.

[Berenson et al., 2009b] Berenson, D., Srinivasa, S., and Kuffner, J. (2009b). Addressing Pose Uncertainty in Manipulation Planning Using Task Space Regions. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1419–1425.

[Berenson et al., 2011] Berenson, D., Srinivasa, S., and Kuffner, J. (2011). Task Space Regions: A Framework for Pose-Constrained Manipulation Planning. *The International Journal of Robotics Research*.

[Bicchi, 1995] Bicchi, A. (1995). On the Closure Properties of Grasping. *The International Journal of Robotics Research*, 14:319–334.

[Blender, 2013] Blender (Last updated: 2013). Blender. Last accessed: 2013. http://www.blender.org/.

Blender is a open source 3D modeling software which offer a python interface and api for programming and visualization.

[Bohg et al., 2011] Bohg, J., Johnson-Roberson, M., Leon, B., Felip, J., Gratal, X., Bergstrom, N., Kragic, D., and Morales, A. (2011). Mind the Gap – Robotic Grasping under Incomplete Observation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 686–693.

[Bohlin and Kavraki, 2000] Bohlin, R. and Kavraki, L. E. (2000). Path Planning Using Lazy PRM. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 521–528.

[Bridgwater et al., 2012] Bridgwater, L. B., Ihrke, C., Diftler, M. A., Abdallah, M. E., Radford, N. A., Rogers, J. M., Yayathi, S., Askew, R. S., and Linn, D. M. (2012). The Robonaut 2 Hand Designed to Do Work with Tools. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3425–3430.

[Cappelleri et al., 2011] Cappelleri, D. J., Fatovic, M., and Shah, U. (2011). Caging Micromanipulation for Automated Microassembly. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3145–3150.

[Cappelleri et al., 2012] Cappelleri, D. J., Fu, Z., and Fatovic, M. (2012). Caging for 2D and 3D Micromanipulation. *Journal of Micro-Nano Mechatronics*, 7:115–129.

[Cheng et al., 2008] Cheng, P., Fink, J., and Kumar, V. (2008). Abstractions and Algorithms for Cooperative Multiple Robot Planar Manipulation. In *Proceedings of Robotics, Science and Systems*, pages 143–150.
This is a typical work of multi-robot cooperative manipulation by using task allocation. A video accompanying this work can be found at http://www.youtube.com/watch?v=fNBqys2Bqhs.

[Cheong et al., 2006] Cheong, J.-S., Haverkort, H. J., and van der Stappen, A. F. (2006). Computing All Immobilizing Grasps of a Simple Polygon with Few Contacts. *Algorithmica*, 44:117–136.
This work is done by one of Prof. Stappen's student. Comparing to Prof. Stappen review work in 2005, this work involves more details on algorithms.

[Chinellato, 2002] Chinellato, E. (2002). *Robust Strategies for Selecting Vision-Based Planar Grasps of Unknown Objects with a Three-Finger Hand*. PhD thesis, University of Edinburgh.
Eris Chinellato is now a research associate at Imperial College London. In this master thesis, Chinellato discussed and ranked various quality functions of grasping optimization.

[Chinellato, 2008] Chinellato, E. (2008). *Visual neuroscience of robotic grasping*. PhD thesis, Jaume I University.

In his PhD thesis, Chinellato tries to connect grasping optimization with perception and fit it into the frame of neuroscience. This thesis won award for the best European Ph.D thesis in robotics in 2008.

[Chinellato et al., 2003] Chinellato, E., Fisher, R. B., Morales, A., and del Pobil, A. P. (2003). Ranking Planar Grasp Configurations For A Three-Finger Hand. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1133–1138.
A refined version of Chinellato's Master thesis.

[Choset et al., 2005] Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L. E., and Thrun, S. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents)*. The MIT Press.

[Chronis and Lee, 2005] Chronis, N. and Lee, L. P. (2005). Electrothermally Activated SU-8 Microgripper for Single Cell Manipulation in Solution. *Journal of Microelectromechanical Systems*, 14:857–863.

[Ciocarlie and Allen, 2009] Ciocarlie, M. and Allen, P. (2009). Hand Posture Subspaces for Dexterous Robotic Grasping. *International Journal of Robotics Research*, 28:851–867.

[Ciocarlie et al., 2007] Ciocarlie, M., Goldfeder, C., and Allen, P. (2007). Dimensionality Reduction for Hand-independent Dexterous Robotic Grasping. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3270–3275.

[Czyzowicz et al., 1999] Czyzowicz, J., Stojmenovic, I., and Urrutia, J. (1999). Immobilizing A Shape. *International Journal of Computational Geometry and Applications*, 9:181–206.
This work gives a formal definition of immobilization discusses immobilization in work space. The algorithms proposed in this work is intuitive and efficient.

[Davidson and Blake, 1998a] Davidson, C. and Blake, A. (1998a). Caging Planar Objects with a Three-Finger One-Parameter Gripper. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2722–2727.
This work extends Rimon and Blake's work on one-parameter two-finger caging in 1996.

[Davidson and Blake, 1998b] Davidson, C. and Blake, A. (1998b). Error-Tolerant Visual Planning of Planar Grasp. In *Proceedings of International Conference on Computer Vision*, pages 911–916.
This is the most early work which uses caging to deal with uncertainty. Although not formally proposed, it implies the idea of "grasping by caging".

[Dollar and Howe, 2005] Dollar, A. M. and Howe, R. D. (2005). Towards Grasping in Unstructured Environments: Grasper Compliance and Configuration Optimization. *Advanced Robotics*, 19:523–543.

[Dollar and Howe, 2010] Dollar, A. M. and Howe, R. D. (2010). The SDM Hand : A Highly Adaptive Compliant Grasper for Unstructured Environments. *International Journal of Robotics Research*, 29:1–11.

[Ekvall and Kragic, 2007] Ekvall, S. and Kragic, D. (2007). Learning and Evaluation of the Approach Vector for Automatic Grasp Generation and Planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 4715–4720.

[Erickson et al., 2007] Erickson, J., Thite, S., Rothganger, F., and Fellow, J. P. (2007). Capturing a Convex Object with Three Discs. *IEEE Transactions on Robotics*, 23:1133–1140.
This is an extended verion of Prof. Erickson's work in 2003. The only difference is that he added some extra explanations.

[Erickson et al., 2003] Erickson, J., Thite, S., Rothganger, F., and Ponce, J. (2003). Capturing a Convex Object with Three Discs. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2242–2247.
Jeff Erickson is a Professor at University of Illinois at Urbana-Champaign, IL, US. He is in the Department of Computer Science and his work concentrates on Algorithms in Topology. In this paper, Erickson proposed the following problem. Given the positions of two fingers and shape information of a target object, how to find the positions for a third finger that can cage the object. He answered this problem by proposing both an exact algorithm and a z-buffering based approximation.

[Fink et al., 2008] Fink, J., Hsieh, M. A., and Kumar, V. (2008). Multi-Robot Manipulation via Caging in Environments with Obstacles. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1471–1476.

[Gerkey and Mataric, 2002] Gerkey, B. P. and Mataric, M. J. (2002). Pusher-watcher: An Approach to Fault-tolerant Tightly-coupled Robot Coordination. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 464–469.

[Glover et al., 2009] Glover, J., Rus, D., and Roy, N. (2009). Probabilistic Models of Object Geometry with Application to Grasping. *The International Journal of Robotics Research*, 28:999–1019.

[Goldfeder et al., 2009] Goldfeder, C., Ciocarlie, M., Peretzman, P., Dang, H., and Allen, P. K. (2009). Data-driven Grasping with Partial Sensor Data. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1278–1283.

[Grier, 2003] Grier, D. (2003). A Revolution in Optical Manipulation. *Nature*, 424:810–816.

[Hammond et al., 2012] Hammond, F. L., Weisz, J., de la Llera Kurth, A. A., Allen, P. K., and Howe, R. D. (2012). Towards a Design Optimization Method for Reducing the Mechanical Complexity of Underactuated Robotic Hands. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2843–2850.

[Handle Project, 2013] Handle Project (Last updated: 2013). The Handle Project. Last accessed: 2013. http://www.handle-project.eu/.

[Harada et al., 2013] Harada, K., Nagata, K., Tsuji, T., Yamanobe, N., Nakamura, A., and Kawai, Y. (2013). Probabilistic Approach for Object Bin Picking Approximated by Cylinders. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3727–3732.

[Hsiao et al., 2010] Hsiao, K., Chitta, S., Ciocarlie, M., and Jones, E. G. (2010). Contact-Reactive Grasping of Objects with Partial Shape Information. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1228– 123.

[Hu et al., 2011] Hu, W., Ishii, K. S., and Ohta, A. T. (2011). Micro-assembly using Optically Controlled Bubble Microrobots. *Applied Physics Letters*, 99:094103–094103–3.
Although the bubble micro-robots is named "optically controlled" bubble microrobots, they are actually thermally controlled. Different from optical tweezers which trap beads with the energy of photons, this paper pulls or drags bubbles by thermal effects and by inducing heat in near-by regions with focused projector light.

[Jiang et al., 2011] Jiang, Y., Moseson, S., and Saxena, A. (2011). Efficient Grasping from RGBD Images: Learning using a New Rectangle Representation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3304–3311.

[Jonathan Weisz, 2012] Jonathan Weisz, De la Llera Kurth, A. A. P. K. A. R. D. H. (2012). Towards a Design Optimization Method for Reducing the Mechanical Complexity of Underactuated Robotic Hands. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2843–2850.

[Kallman and Mataric, 2004] Kallman, M. and Mataric, M. (2004). Motion Planning Using Dynamic Roadmaps. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 4399–4404.

[Katana, 2013] Katana (Last updated: 2013). Katana Arm Japan Provider. Last accessed: 2013. http://www.revast.co.jp/service/arm/type01.html.
This website is not the official site of Katana Arm. It is the Japanese proxy. Katana arm is a small robotic manipulator produced by Neuronics Intelligent and Personal Robotics. Official site of Katana Arm is http://www.neuronics.ch/cms$_d$e/web/index.php?id = 385s = hardware$_a$rchitecture.

[Kavraki et al., 1996] Kavraki, L., Svestka, P., Latombe, J. C., and Overmars, M. (1996). Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE Transactions on Robotics and Automation*, 12:566–580.

[Kehoe et al., 2013] Kehoe, B., Matsukawa, A., Candido, S., Kuffner, J., and Goldberg, K. (2013). Cloud-Based Robot Grasping with the Google Object Recognition Engine.

In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 4248–4255.

[Kuperberg, 1990] Kuperberg, W. (1990). Problems on Polytopes and Convex Sets. In *Proceedings of DIMACS: Workshop on polytopes*, pages 584–589.
Woldzimierz Kuperberg is a mathematician at Auburn University, AL, US. Before his formal expression of caging in this paper, lots of relevent problems were dicussed in mathematical fields. However, it is Kuperberg who first summarizes those discussions and formally proposes the caging problem.

[Lavalle and Kuffner, 2000] Lavalle, S. M. and Kuffner, J. J. (2000). Rapidly-exploring random trees: Progress and prospects. In *Proceedings of International Workshop on the Algorithmic Foundations of Robotics*, pages 293–308.

[LaVelle, 2006] LaVelle, S. M. (2006). *Planning Algorithms*. Cambridge University Press.

[Leeper et al., 2010] Leeper, A., Hsiao, K., Chu, E., and Salisbury, K. (2010). Using Near-Field Stereo Vision for Robotic Grasping in Cluttered Environments. In *Proceedings of International Symposium on Experimental Robotics*.

[Leven and Hutchinson, 2002] Leven, P. and Hutchinson, S. (2002). A Framework for Real-time Path Planning in Changing Environments. *The International Journal of Robotics Research*, 21:999–1030.

[Li and Sastry, 1988] Li, Z. and Sastry, S. S. (1988). Task-oriented Optimal Grasping by Multifingered Robot Hands. *IEEE Journal of Robotics and Automation*, 4:32–44.
Zexiang Li uis a professor of electrical and electronic engineering at University of Science and Technology, Hong Kong. In this work, he discusses in details the basic optimal criteria of grasping and proposes the new task-oriented optimization.

[Liu et al., 2010] Liu, H., Wan, W., and Zha, H. (2010). A Dynamic Subgoal Path Planner for Unpredictable Environments. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 994–1001.

[Liu et al., 2004] Liu, Y.-H., Lam, M.-L., and Ding, D. (2004). A Complete and Efficient Algorithm for Searching 3-D Form-closure Grasps in the Discrete Domain. *IEEE Transactions on Robotics*, 20:805–816.
Yun-Hui Liu is a professor of the department of mechanical and automation engineering at the Chinese University of Hong Kong. Although the title claims about form closure, this work actually discusses both frictional and frictionless cases. Equations in this work are quite clear for beginners.

[Maeda et al., 2012] Maeda, Y., Kodera, N., and Egawa, T. (2012). Caging-Based Grasping by a Robot Hand with Rigid and Soft Parts. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 5150–5155.

[Maeda and Makita, 2006] Maeda, Y. and Makita, S. (2006). A Quantitative Test for the Robustness of Graspless Manipulation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1743–1748.

[Maeda et al., 2004] Maeda, Y., Nakamura, and Arai (2004). Motion Planning of Robot Fingertips for Graspless Manipulation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2951–2956.

[Makita, 2010] Makita, S. (2010). *Robotic manipulation by incomplete grasping: graspless manipulation and 3D multifingered caging*. PhD thesis, Yokohama National University.

[Makita and Maeda, 2008] Makita, S. and Maeda, Y. (2008). 3D Multifingered Caging: Basic Formulation and Planning. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2697–2702.

[Mason, 2001] Mason, M. T. (2001). *Mechanics of Robotic Manipulation*. The MIT Press.
Matthew T. Mason is a professor at the Robotic Institute of Carnegie Mellon University, PA, US. In this book, Mason introduces all basic concepts and physical/mathematical tools, ranging from statics to dynamics. He also teaches a course based on this book. Some materials of the course can be found from his homepage.

[Mason et al., 2012] Mason, M. T., Rodriguez, A., Srinivasa, S. S., and Vazquez, A. S. (2012). Autonomous Manipulation with a General-Purpose Simple Hand. *The International Journal of Robotics Research*, 31:688–703.

[Michel, 2004] Michel, O. (2004). Webots: Professional Mobile Robot Simulation. *International Journal of Advanced Robotic Systems*, 1:39–42.

[Microsoft, 2012] Microsoft (Last updated: 2012). Introduction to KINECT. Last accessed: 2013. http://www.microsoft.com/en-us/kinectforwindows/.
Although KINECT is a trademark of Microsoft, it is based on a depth camera technology by a Israeli developer PrimeSense. The technology is named structure light. Like its name, the technology calculates depth according to changes of infrared light structures. It is a low-cost solution to perceive depth information. However, it cannot be as precise as other technologies such as Time-of-Flight.

[Miller et al., 2003] Miller, A. T., Knoop, S., Christensen, H. I., and Allen, P. K. (2003). Automatic Grasp Planning Using Shape Primitives. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1824–1829.

[Mirtich and Canny, 1994] Mirtich, B. and Canny, J. (1994). Easily Computable Optimum Grasps in 2D and 3D. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 739–747.

[Mishra et al., 1987] Mishra, B., , Schwartz, J. T., and Sharir, M. (1987). On the Existence and Synthesis of Multifinger Positive Grips. *Algorithmica*, 2:541–558.
Bud Mishra is a professor of computer science and mathematics at New York University. This paper demonstrates lots of key problems in grasping and form closure. It is one of the most frequently cited paper in basic grasping theory.

[Montemayor and Wen, 2005] Montemayor, G. and Wen, J. T. (2005). Decentralized Collaborative Load Transport by Multiple Robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 372–377.

[NaturalPoint, 2013] NaturalPoint (Last updated: 2013). OptiTrack V100:R2. Last accessed: 2013. http://www.naturalpoint.com/optitrack/products/v100-r2/.

[Nguyen, 1986a] Nguyen, V.-D. (1986a). Constructing Force-Closure Grasps. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1368–1373.
This paper is a simplified version of Nguyen's technical report. It discusses how to synthesize force closures of 2D objects.

[Nguyen, 1986b] Nguyen, V.-D. (1986b). The Synthesis of Stable Force-Closure. Technical Report AI-TR 905, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.
This report discussed in detail (1) the conecepts of force closures and (2) how to do force-closure grasp synthesis based on shape of target objects. Both 2D and 3D cases are included in this report.

[Niparnan et al., 2009] Niparnan, N., Phoka, T., and Sudsang, A. (2009). Heuristic Approach for Multiple Queries of 3D n-finger Frictional Force Closure Grasp. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1817–1822.

[Onda and Arai, 2011] Onda, K. and Arai, F. (2011). Robotic Approach to Multi-beam Optical Tweezers with Computer Generated Hologram. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1825–1830.

[Parker, 1998] Parker, L. E. (1998). ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation. *IEEE Transactions on Robotics and Automation*, 14:220–240.

[Pereira et al., 2002a] Pereira, G. A. S., Kumar, V., and Campos, M. F. M. (2002a). Decentralized Algorithms for Multirobot Manipulation via Caging. In *Proceedings of International Workshop on Algorithmic Foundations of Robotics*, pages 242–258.

[Pereira et al., 2004] Pereira, G. A. S., Kumar, V., and Campos, M. F. M. (2004). Decentralized Algorithms for Multirobot Manipulation Via Caging. *International Journal of Robotics Research*, 23:783–795.

[Pereira et al., 2002b] Pereira, G. A. S., Kumar, V., Spletzer, J. R., Taylor, C. J., and Campos, M. F. M. (2002b). Cooperative Transport of Planar Objects by Multiple Mobile Robots Using Object Closure. In *Experimental Robotics VIII*, pages 275–285. Springer.

[Pipattanasomporn and Sudsang, 2010] Pipattanasomporn, P. and Sudsang, A. (2010). Object Caging under Imperfect Shape Knowledge. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2683–2688.

[Pipattanasomporn and Sudsang, 2011] Pipattanasomporn, P. and Sudsang, A. (2011). Two-Finger Caging of Nonconvex Polytopes. *IEEE Transactions on Robotics*, 27:324–333.

[Pipattanasomporn et al., 2008] Pipattanasomporn, P., Vongmasa, P., and Sudsang, A. (2008). Caging Rigid Polytopes via Finger Dispersion Control. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1181–1186.

[Pollard, 2010] Pollard, N. (Course time: Spring, 2010). Hands: Design and Control for Dexterous Manipulation. Last accessed: 2013. http://graphics.cs.cmu.edu/nsp/course/16-899/.
This is a course opened by Prof. Nancy Pollard. Nancy Pollard is a professor at the Robotic Institute of Carnegie Mellon University, PA, US. In this course, Nancy surveys lots of robotic hands and discusses how to design robotic hands. Many popular and interesting publications can be found in this course website.

[Ponce, 1996] Ponce, J. (1996). On Planning Immobilizing Fixtures for Three-Dimensional Polyhedral Parts. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 509–514.

[Ponce et al., 1995] Ponce, J., Burdick, J., and Rimon, E. (1995). Computing the Immobilizing Three-Finger Grasps of Planar Objects. *IEEE Transactions on Robotics and Automation*, 11:868–881.
This paper is a preliminary work of 2nd order mobility theory. We can find how those researchers began to consider about surface curvertures of target objects.

[Qiao, 2001] Qiao, H. (2001). Attractive Regions Formed by Constraints in Configuration Space – Attractive Regions in Motion Region of a Polygonal or a Polyhedral Part with a Flat Environment. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1071–1078.
This paper includes a complete introduction of attrative regions. Attractive regions share the same idea with ICS proposed by Attawith Sudsang. Both of them try to define themselves in configuration space and both of them fail to explicitly express configuration space.

[Qiao, 2002] Qiao, H. (2002). Strategy Investigation with Generalized Attractive Regions. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3315–3320.

[Rao et al., 2010] Rao, D., Le, Q. V., Phoka, T., Quigley, M., Sudsang, A., and Ng, A. Y. (2010). Grasping Novel Objects with Depth Segmentation. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2578–2585.

[Rimon, 2001] Rimon, E. (2001). A Curvature-Based Bound on the Number of Frictinless Fingers Required to Immobilize Three-Dimensional Objects. *IEEE Transactions on Robotics and Automation*, 17:679–697.
In this work, Prof. Rimon extends his 2nd order theory to 3D objects.

[Rimon and Blake, 1996] Rimon, E. and Blake, A. (1996). Caging 2D Bodies by 1-Parameter Two-Fingered Gripping Systems. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1458–1464.

[Rimon and Blake, 1999] Rimon, E. and Blake, A. (1999). Caging Planar Bodies by One-Parameter Two-Fingered Gripping Systems. *International Journal of Robotics Research*, 18:299–318.
This is the first paper that makes caging an independent topic in robotics.

[Rimon and Burdick, 1996] Rimon, E. and Burdick, J. (1996). On Force and Form Closure For Multiple Finger Grasps. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1795–1800.
This work is a draft version of 2nd order theory.

[Rimon and Burdick, 1998a] Rimon, E. and Burdick, J. W. (1998a). Mobility of Bodies in Contact – I: A New 2nd Order Mobility Index for Multiple-Finger Grasps. *IEEE Transactions on Robotics and Automation*, 14:696–708.
In this work, Prof. Rimon formally proposes the idea of 2nd order form closure. It tries to make up the gap between Prof. Mishra's theory and our common sense.

[Rimon and Burdick, 1998b] Rimon, E. and Burdick, J. W. (1998b). Mobility of Bodies in Contact – II: How Forces are Generated by Curvature Effects. *IEEE Transactions on Robotics and Automation*, 14:709–717.
This work involves some supporting materials to the proposal in Part I.

[Rodriguez, 2013] Rodriguez, A. (2013). A Review of Caging. In *Proceedings of IEEE International Conference on Robotics and Automation*.
In this work, Rodriguez give a review of caging, including some old discussions outside the research field of robotics.

[Rodriguez and Mason, 2008] Rodriguez, A. and Mason, M. T. (2008). Two Finger Caging: Squeezing and Stretching. In *Proceedings of International Workshop on the Algorithmic Foundations of Robotics*.

[Rodriguez and Mason, 2012a] Rodriguez, A. and Mason, M. T. (2012a). Grasp Invariance. In *International Journal of Robotics Research*.

This is a pre-work of "Effector Form Design for 1DOF Planar Actuation". It is the theoretical part of the design.

[Rodriguez and Mason, 2012b] Rodriguez, A. and Mason, M. T. (2012b). Path-Connectivity of the Free Space. *Transaction on Robotics*, 28:1177–1180.

[Rodriguez and Mason, 2013] Rodriguez, A. and Mason, M. T. (2013). Effector Form Design for 1DOF Planar Actuation. In *Proceedings of IEEE International Conference on Robotics and Automation*.
In this work, Rodriguez discussed how to design a general end-effector for 1DOF balls. Despite the design, this paper gives an excellent review of end-effector design.

[Rodriguez et al., 2011] Rodriguez, A., Mason, M. T., and Ferry, S. (2011). From Caging to Grasping. In *Proceedings of Robotics, Science and Systems*.

[Rodriguez et al., 2012] Rodriguez, A., Mason, M. T., and Ferry, S. (2012). From Caging to Grasping. In *The International Journal of Robotics Research*, volume 31, pages 886–900.

[Rus, 1997] Rus, D. (1997). Coordinated Manipulation of Objects in a Plane. *Algorithmica*, 19:129–147.

[Sato, 1996] Sato, T. (1996). Micro/nano Manipulation World. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 834–841.

[Saxena et al., 2006] Saxena, A., Driemeyer, J., Kearns, J., Osondu, C., and Ng, A. Y. (2006). Learning to Grasp Novel Objects using Vision. In *Proceedings of International Symposium of Experimental Robotics*.

[SCHUNK, 2013] SCHUNK (Last updated: 2013). The SCHUNK JGZ industrial gripper. Last accessed: 2013. http://www.schunk.com/schunk_files/attachments/{JGZ}_160_{EN}.pdf.

[Smith, 2013] Smith, R. (Last updated: 2013). Open Dynamics Engine. Last accessed: 2013. http://www.ode.org/.
This website is not the homepage of ODE. Detailed documents and examples can be found in its Wiki page http://ode-wiki.org/wiki/index.php?title=Main$_{P}age$.

[Suarod et al., 2007] Suarod, N., Boonpion, N., and Sudsang, A. (2007). A Heuristic Method for Computing Caging Formation of Polygonal Object. In *Proceedings of IEEE International Conference on Robotics and Biomimetics*, pages 823–828.

[Sudsang and Ponce, 1998] Sudsang, A. and Ponce, J. (1998). On Grasping and Manipulating Polygonal Objects with Disc-Shaped Robots in the Plane. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2740–2746.
In this work, Sudsang propose to manipulate a triangle object without contacts. It is esentially the idea of caging manipulation.

[Sudsang and Ponce, 2000] Sudsang, A. and Ponce, J. (2000). A New Approach to Motion Planning for Disc-shaped Robots Manipulating a Polygonal Object in the Plane. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1068–1075.
In this work, Sudsang propose a framework to perform motion planning and obstacle avoidance with ICS.

[Sudsang et al., 1999] Sudsang, A., Ponce, J., Hyman, M., and Kriegman, D. J. (1999). Manipulating Polygonal Objects with Three 2-DOF Robots in the Plane. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2227–2234.
In this work, Sudsang implemented his contactless manipulation algorithm with real robots.

[Sudsang et al., 1997] Sudsang, A., Ponce, J., and Srinivasa, N. (1997). Algorithms for Constructing Immobilizing Fixtures and Grasps of Three-Dimensional Objects. In *Algorithmic Foundations of Robotics II*, pages 363–380. A K Peters, Ltd.

[Sudsang et al., 2000] Sudsang, A., Ponce, J., and Srinivasa, N. (2000). Grasping and In-Hand Manipulation: Geometry and Algorithms. *Algorithmica*, 26:466–493.
This paper has a detailed introduction of the Inescapable Configuration Space (ICS). Although ICS was represented differently from our isolated space, they can be recognized as the same concept.

[Sudsang et al., 2002] Sudsang, A., Rothganger, F., and Ponce, J. (2002). Motion Planning for Disc-shaped Robots Pushing a Polygonal Object in the Plane. *IEEE Transactions on Robotics and Automation*, 18:550–562.

[Trinkle et al., 1988] Trinkle, J. M., Abel, J. M., and Paul, R. P. (1988). An Investigation of Frictionless Enveloping Grasping in the Plane. *The International Journal of Robotics Research*, 7:33–51.

[Vahedi, 2009] Vahedi, M. (2009). *Caging Polygons with Two and Three Fingers*. PhD thesis, Utrecht University.

[Vahedi and van der Stappen, 2006] Vahedi, M. and van der Stappen, A. F. (2006). Caging Polygons with Two and Three Fingers. In *Proceedings of International Workshop on Algorithmic Foundations of Robotics*, pages 71–86.

[Vahedi and van der Stappen, 2007] Vahedi, M. and van der Stappen, A. F. (2007). Geometric Properties and Computation of Three-Finger Caging Grasps of Convex Polygons. In *Proceedings of IEEE Conference on Automation Science and Engineering*, pages 404–411.

[Vahedi and van der Stappen, 2008a] Vahedi, M. and van der Stappen, A. F. (2008a). Caging Convex Polygons with Three Fingers. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1777–1783.

[Vahedi and van der Stappen, 2008b] Vahedi, M. and van der Stappen, A. F. (2008b). Caging Polygons with Two and Three Fingers. *The International Journal of Robotics Research*, 27:1308–1324.

[Vahedi and van der Stappen, 2008c] Vahedi, M. and van der Stappen, A. F. (2008c). Towards Output-Sensitive Computation of Two-Finger Caging grasps. In *Proceedings of IEEE Conference on Automation Science and Engineering*, pages 73–78.

[Vahedi and van der Stappen, 2009] Vahedi, M. and van der Stappen, A. F. (2009). On the Complexity of the Set of Three-Finger Caging Grasps of Convex Polygons. In *Proceedings of Robotics, Science and Systems*.

[van der Stappen, 2005] van der Stappen, A. F. (2005). Immobilization: Analysis, Existance, and Output-sensitive synthesis. In *Computer-Aided Design and Manufacturing*, pages 162–187. AMS-DIMACS.
Prof. Stappen is a professor of computer science at Utrecht University. This paper is a review article which involves not only the history of immobilization but also some efficient algorithms.

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

[Vstone, 2013] Vstone (Last updated: 2013). Beauto Rover ARM. Last accessed: 2013. http://vstone.co.jp/products/beauto_rover/index.html//.

[Wallack, 1996] Wallack, A. S. (1996). Generic Fixture Design Algorithms for Minimal Modular Fixture Toolkits. In *Proceedings of IEEE International Conference on Robotics and Automation*.

[Wallack and Canny, 1996] Wallack, A. S. and Canny, J. F. (1996). Modular Fixture Design for Generalized Polyhedra. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 830–837.

[Wang et al., 1999] Wang, Z., Ahmadabadi, M. N., Nakano, E., and Takahashi, T. (1999). A Multiple Robot System for Cooperative Object Transportation with Various Requirements on Task Performing. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1226–1233.

[Wang et al., 2003a] Wang, Z., Hirata, Y., and Kosuge, K. (2003a). CC-Closure Object and Object Closure Margin of Object Caging by Using Multiple Robots. In *Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechantronics*, pages 344–349.

[Wang et al., 2004] Wang, Z., Hirata, Y., and Kosuge, K. (2004). Control a Rigid Caging Formation for Cooperative Object Transportation by Multiple Mobile Robots. In *Proceedings of IEEE/RSJ International Conference on Robotics and Automation*, pages 1580–1585.

[Wang et al., 2005] Wang, Z., Hirata, Y., and Kosuge, K. (2005). An Algorithm for Testing Object Caging Condition by Multiple Mobile Robots. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3022–3027.

[Wang et al., 2006] Wang, Z., Hirata, Y., and Kosuge, K. (2006). Dynamic Object Closure by Multiple Mobile Robots and Random Caging Formation Testing. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3675–3681.

[Wang and Kumar, 2002] Wang, Z. and Kumar, V. (2002). Object Closure and Manipulation by Multiple Cooperating Mobile Robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 394–399.

[Wang et al., 2003b] Wang, Z., Kumar, V., Hirata, Y., and Kosuge, K. (2003b). A Strategy and a Fast Testing Algorithm for Object Caging by Multiple Cooperative Robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2275–2280.

[Wang et al., 2009] Wang, Z., Matsumoto, H., Hirata, Y., and Kosuge, K. (2009). A Path Planning Method for Dynamic Object Closure by using Random Caging Formation Testing. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5923–5929.

[Watanabe and Yoshikawa, 2007] Watanabe, T. and Yoshikawa, T. (2007). Grasping Optimization Using a Required External Force Set. *IEEE Transactions on Automation Science and Engineering*, 4:52–66.
Tetsuyou Watanabe is a professor of mechanical engineering at Kanazawa University. He in this work proposes a fast solution to find optimal grasping positions based on the branch-and-bound algorithm.

[Webots, 2013] Webots (Last updated: 2013). Webots. Last accessed: 2013. http://www.cyberbotics.com//.
Webots is a commerical robot simulation software. It uses Open Dynamic Engine for simulation of dynamics. Comparing with those freewares, Webots maintainers tend to fix bugs as soon as possible and tend to make the components as robust as possible.

[Wikipedia, 2013a] Wikipedia (Last updated: 2013a). Marching Cubes. Last accessed: 2013. http://en.wikipedia.org/wiki/Marching_cubes.
The text from this wikipedia link briefly and clearly shows the principles of marching cube. It also includes the essential research papers that relate to this topic.

[Wikipedia, 2013b] Wikipedia (Last updated: 2013b). Venus Flytrap. Last accessed: 2013. http://en.wikipedia.org/wiki/Venus_flytrap.
The text from this wikipedia link briefly and clearly shows the venus flytrap. It is an example of the natural creative which employ caging.

[WillowGarage, 2012] WillowGarage (Last updated: 2012). Overview of the Willow Garage PR2 robot. Last accessed: 2013. http://www.willowgarage.com/pages/pr2/overview.
WillowGarage is a leading corporation who develops hardware and opensource software for personal robotics applications. It is a major supporter of international robotic conferences and a major developer of ROS. PR2 and TurtleBot are two representative robots of Willow Garage.

[Yokoi et al., 2009] Yokoi, R., Kobayashi, T., and Maeda, Y. (2009). 2D Caging Manipulation by Robots and Walls. In *Proceedings of IEEE International Symposium on Assembly and Manufacturing*, pages 16–21.

[Zhang and Goldberg, 2001] Zhang, T. and Goldberg, K. (2001). Design of Robot Gripper Jaws Based on Trapezoidal. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1065–1070.

Acknowledgments

"You will fail to have a great career, unless you fail to have a great career"
by Michael Litt at TEDxUW

Once upon a time, at two o'clock in mid-night, I jumped out of my bed, opened my computer and typed lines of codes to realize an idea that suddenly came into my head. Unfortunately, the idea failed. Once upon a time, at three o'clock on a winter afternoon, I downloaded a newly compiled program to my micro-controller and tried to actuate the robot. Disappointingly, the robot committed suicide. Once upon a time, at seven o'clock on a rainy morning. I checked the review status of my papers on https://ras.papercept.net, expecting at least an accepted one. Unexpectedly, none of them were accepted. Once…

After many "once upon a time" failures like that, I managed to type the texts in this dissertation. The failures caused intense suffering. I am a normal guy and I cannot bear that many failures. It were the instructions and encouragements from my three advisers Prof. Tomomasa Sato, Prof. Yasuo Kuniyoshi and Prof. Rui Fukui that helped me get over them. I sincerely appreciate their advising. I got a lot from Prof. Sato who taught me to keep a journal of research and to be active with colleagues. Following Prof. Sato's instruction, I maintained my passion in robotic research. I got a lot from Prof. Kuniyoshi who helped me to find some sparkling points of my pour work, approved me positively and encouraged me to go deeper. Following Prof. Kuniyoshi's instruction brings me confidence in my topics. I got a lot from Prof. Fukui who covered every detail of my research. Prof. Fukui is an excellent engineer as well as an excellent brother. He successfully changed my computer-science head to a mechano-informatics head.

Besides my three advisers, I would like to thank the other members of my thesis committee, Prof. Yoshihiko Nakamura, Prof. Masayuki Inaba and Prof. Kei Okada. Their passionating comments helped to improve the draft version of the thesis greatly. Prof. Nakamura advised choosing proper terminologies and reviewing extensively about grasp-closure works which increased the strictness of the thesis. Prof. Inaba and Prof. Okada advised testing more objects and discussing about 3D cases which increased the soundness of the thesis.

Moreover, I would like to thank Prof. Taketoshi Mori, Prof. Masamichi Shimosaka and Prof. Hiroshi Noguchi for their discussions and revision suggestions of my research topics and paper publications. I would like to thank Mr. Keita Kadowaki, Mr. Yamato Niwa and Mr. Kentairo Nishi for their help in implementing some of the mechanical systems. I would like to thank Mr. Shuhei Kousaka, Mr. Masahiko Watanabe and Mr. Takuya Sunagawa for their suggestions on circuits diagrams and control boards. I would like to thank Prof. Mamoru Nakamura, Mr. Masayuki Tanaka, Ms. Keiko Hirosue, Ms. Yuki Naruke, Ms. Etsuko Izumiya and Mr. Yuichi Moriya for their assistance in experimental equipments, academic travelling and other school or social affairs.

Furthermore, I would like to thank Dr. Mosatafa Vahedi from Utrecht University and Dr. Alberto Rodriguez from Carnegie Mellon University for their discussions on theoretical fundamentals. I would like to thank Prof. Pham Quang Cuong from Nanyang Technological University and Dr. Carlos Felipe Santacruz from Intouch Health for their smart suggestions in the implementation of certain algorithms. I would like to thank Prof. Matthew Mason from Carnegie Mellon University, Prof. Elon Rimon from Technion Israel Institute of Technology, Prof. A Frank van der Stappen from Utrecht University, Prof. Zhidong Wang from Chiba Institute of Technology, Prof. Yusuke Maeda from Yokohama National University, Prof. Attawith Sudsang from Chulalongkorn University, Prof. David J. Cappelleri from Purdue University, Prof. Bereson Dimitry from Worcester Polytechnic Institute, Prof. Satoshi Makita from Sasebo National College of Technology and Prof. Jianhua Su from Chinese Academy of Science for their suggestions, discussions or encouragements on my caging research. Without them I may have lost my directions. I would like to thank my master thesis adviser, Prof. Hong Liu from Peking University, for his continuing concern of my oversea study. I would like to thank my sister Prof. Zhenzhen Wan, my friends Dr. Yaozhong Zhang, Dr. Tao Luo, Dr. Min Lu, Dr. Jia Pan, Dr. Boxin Shi, Dr. Feng Lu, Dr. Shuqing Han, Dr. Chunhua Geng, Dr. Lu Lu, Dr. Yangfeng Ji, Ms. Ying Shi, Mr. Pengfei Sun, Ms. Xiaoxia Hu and Mr. Huijun He for their role-model effects.

Finally, without the support and care of my parents, I could have never finished my studies. I would like not only to thank but to dedicate my thesis to them.