

การกักขังรูปหลายเหลี่ยมไว้ด้วยสองนิ้ว



นายพีม พิพัฒน์สมพร

สถาบันวิทยบริการ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์

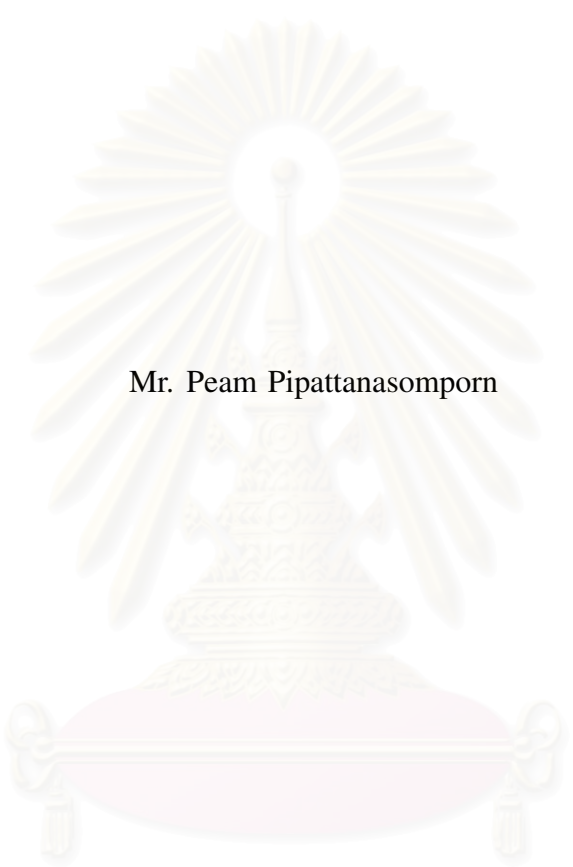
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2548

ISBN 974-17-3988-5

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

TWO-FINGER CAGING OF CONCAVE POLYGON



Mr. Peam Pipattanasomporn

สถาบันวิทยบริการ
A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering Program in Computer Engineering

Department of Computer Engineering

Faculty of Engineering

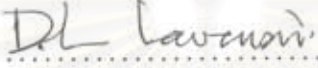
Chulalongkorn University

Academic Year 2005

ISBN 974-17-3988-5

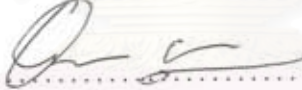
Thesis Title Two-Finger Caging of Concave Polygon
By Mr. Peam Pipattanasomporn
Field of Study Computer Engineering
Thesis Advisor Attawith Sudsang, Ph.D.


Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree



..... Dean of the Faculty of Engineering
(Professor Direk Lavansiri, Ph.D.)

THESIS COMMITTEE


..... Chairman
(Associate Professor Prabhas Chongstitvatana, Ph.D.)


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


..... Member
(Associate Professor Witaya Wannasuphoprasit, Ph.D.)


..... Member
(Thavida Maneewarn, Ph.D.)

พิมพ์ พิมพ์นสมพร : การกักขังรูปหลายเหลี่ยมเว้าด้วยสองนิ้ว (TWO-FINGER CAGING OF CONCAVE POLYGON). อาจารย์ที่ปรึกษา : อ. ดร. อรรถวิทย์ สุดแสง, 76 หน้า. ISBN 974-17-3988-5

วัตถุที่ถูกกักขังนั้นจะไม่มีทางที่จะเล็ดรอดออกจากริ้วหุ่นยนต์ ได้ (ซึ่งเปรียบเสมือน ซีกรงขัง) การที่วัตถุไม่สามารถเล็ดรอดออกไปได้นั้นคือการที่ไม่สามารถเคลื่อนย้ายวัตถุไปให้ไกลแสนไกลจากนิ้วต่างๆได้ คุณสมบัติความเว้าของวัตถุเป็นคุณสมบัติเชิงเรขาคณิตที่ช่วยให้ การกักขังวัตถุทำได้แม้ใช้เพียงไม่กี่นิ้ว ในเชิงลึกนั้นการที่จะกักขังวัตถุที่มีความเว้าบางชนิด สามารถทำได้โดยใช้นิ้วเพียงสองนิ้ว งานวิทยานิพนธ์นี้มุ่งเน้นที่จะศึกษาปัญหาการคำนวณหารูปแบบการวางนิ้วสองนิ้วทั้งหมดที่สามารถกักขังวัตถุได้ โดยการบรรยายรูปแบบการวาง นิ้วทั้งหมดดังกล่าว สามารถนำไปใช้ในการตรวจสอบว่ารูปแบบการวางนิ้วที่ต้องการตรวจสอบ นั้นสามารถกักขังวัตถุได้หรือไม่ได้ในเวลา $O(\lg n)$ สำหรับในกรณีทั่วไปในการคำนวณเพื่อ บรรยายรูปแบบการวางนิ้วทั้งหมดดังกล่าวนั้น สามารถทำได้ในเวลา $O(n^2 \lg n)$.



สถาบันวิทยบริการ จุฬาลงกรณ์มหาวิทยาลัย

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต พิมพ์ พิมพ์นสมพร.
สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา
ปีการศึกษา 2548


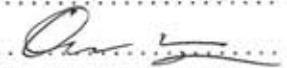
##4770386521 : MAJOR COMPUTER ENGINEERING

KEYWORDS : CAGING/ CAPTURING / ROBOTICS/ OBJECT CLOSURE

PEAM PIPATTANASOMPORN : TWO-FINGER CAGING OF CONCAVE POLYGON. THESIS ADVISOR : ATTAWITH SUDSANG, Ph.D., 76 pp. ISBN 974-17-3988-5

An object is captured by a set of fingers when there exists no trajectory to bring the object arbitrarily far from the fingers. Object concavity is a special geometric property that allows objects to be captured with only few fingers. In particular, certain concave objects can be captured by appropriately placing two fingers close to some pair of opposite concave sections. This work addresses the problem of computing all configurations of the two fingers that capable of capturing the object. Those configurations will be represented in such a way that answering whether a finger configuration can capture the object requires $O(\lg n)$ running time in most cases (where n is the number of vertices of the object.) In computing all configurations capable of capturing the object, we proposed a $O(n^2 \lg n)$ algorithm.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Department	Computer Engineering	Student's signature 
Field of study	Computer Engineering	Advisor's signature 
Academic year	2005		

Acknowledgements

To accomplish all of this work, there are many people who provide supports and reviews. First of all, I would like to thank my advisor, Dr. Attawith Sudsang, for his grateful advices and technical knowledge.

The friendly environment inside this department is one of the key that bring me to the final phase. I would like to thank every people in the department of computer engineering and Intelligent System Engineering Laboratory 2 at Chulalongkorn University.

Finally, I would like to thank my parents for their kindly support and love which give me strength to reach this moment.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Contents

	Page
Abstract (Thai)	iv
Abstract (English)	v
Acknowledgements	vi
Contents	vii
List of Tables	ix
List of Figures	x
Chapter	
I Introduction	1
I.1 Introduction	1
I.2 Objective	2
I.3 Scope of this Work	5
I.4 Organization of the Thesis	5
II Related Works	7
II.1 Caging/Capturing Problems	7
II.2 Error-Tolerant Grasping	7
II.3 Object Transportation	9
II.4 Summary	10
III Background	11
III.1 Basic Assumptions	11
III.2 The Configuration Space	12
III.3 The Maximal Cage Problem	13
III.4 Trajectories of Fingers	15
IV Search Space Formulation	18
IV.1 Overview	18
IV.2 Characterizing All Maximal Cages	21
IV.3 Space Partitioning	25
IV.4 Topology of Pieces	29

	Page
IV.5 The Reduced Search Space	31
V The Algorithms	34
V.1 Identifying Empty Pieces	34
V.2 Generating Transitions	35
V.3 Propagating Least Upper-Bound Distance	36
V.4 Identifying All Maximal Cages	37
V.5 Querying Least Upper-Bound Distance and Maximal Cage	38
V.6 Time Complexity Analysis	38
VI Minimal Cage	41
VI.1 Minimal Cage Problem	41
VI.2 Characterizing All Minimal Cages	43
VI.3 Space Partitioning	45
VI.4 Topology of Pieces	47
VI.5 The Reduced Search Space	48
VI.6 Propagating Greatest Lower-Bound Distance	50
VII Improvements on Solving The Maximal Cage Problem	52
VII.1 The Improved Configuration Space Partitioning	52
VII.2 Extension To Higher Dimension	54
VIII Conclusions	55
References	58
Appendix	
A Proof of Lemma	63
Biography	65

List of Tables

	Page
VIII.1 Running time of algorithms in the first phase given an input object with n vertices, represented with k simple polygons (polyhedra in 3D case) and its space can be partitioned into r convex regions.	56
VIII.2 Total running time of the first phase and running time for each query in the second phase.	56



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

List of Figures

	Page
I.1	Object transportation with maximal cage 2
I.2	Visualizing the caging pocket. 3
I.3	Properties of maximal cages 4
I.4	Object transportation with minimal cage 4
III.1	Considering only cross-section of a 3D object 12
III.2	Analyzing the maximal cage 14
III.3	Escaping the maximal cage 15
III.4	Trajectories of fingers 16
IV.1	An example with one maximal cage 19
IV.2	A pair of escape synchronized trajectories 20
IV.3	Squeezing operation 26
IV.4	Squeezing to local minimum 27
IV.5	Basic transitions for the reduced search space of <i>maximal cage problem</i> . 29
IV.6	Replacing non-basic transitions with a sequence of basic transitions 31
IV.7	An example of a transition that occurs when both fingers are not on vertices 32
V.1	Piece validity 35
V.2	Checking piece validity 36
V.3	Example situations when to and not to merge two maximal cages 38
V.4	Characterizing all maximal cages 39
V.5	Examples of outputs from characterizing all maximal cages 40
VI.1	Examples of outputs from characterizing all minimal cages 41
VI.2	Analysis of a minimal cage 42
VI.3	Stretching operation 45
VI.4	Basic transitions in <i>minimal cage problem</i> 48
VI.5	Replacing non-basic transitions with a sequence of basic transitions in <i>minimal cage problem</i> 49

	Page
VI.6 Characterizing all minimal cages	51
VII.1 An important property of pieces from two convex regions	53



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER I

INTRODUCTION

I.1 Introduction

Typically, a robot transports an object via grasping. With this approach, a number of contact points have to be made appropriately and once the object is held securely the transportation can be performed simply by moving the robot. This seemingly simple method entails several difficulties. Precisely establishing contact as desired requires complex control whereas maintaining all these contact points during the entire transportation requires even more complex control. One of our main statements here is that transportation can be done without grasping. Consider a group of robots that position themselves around the object in such a way that the object may still move but cannot escape from the robots' formation, i.e., the object is captured within the formation. Intuitively, such formation exists for any object when sufficiently many robots are deployed. With this caging formation, the robots can transport the object by moving the whole formation and drag the object along (see Figure I.1.) Unlike the grasping based approach, the robots do not have to precisely synchronize their movement; they have certain clearance for which each robot may be positioned relative to each others while still prevent the object from escaping. With no need to maintain contact, the robot control can be simplified. Still, planning of the formation is required. We are interested in this kind of problems.

In particular, we are interested in the problem of capturing a concave polygon in the plane with two fingers. To be precise, an object is captured by a set of fingers when it is restricted to stay within a bounded region of the workspace, i.e., there exists no trajectory to bring the object arbitrarily far from the fingers. While any object can be captured when sufficiently many fingers are used, only two fingers are needed to capture a concave object in the plane. Let us consider a concave polygon being grasped by two fingers as shown in Figure I.2(a). At this point, the polygon

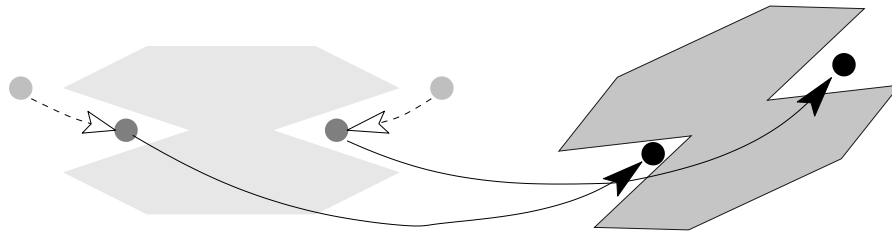


Figure I.1: An object is caged and transported by two robots.

is completely immobilized, i.e., there exists an isolated configuration of the polygon. Now let us move the two fingers slightly away from each other (Figure I.2(b)). The polygon can now move but it still cannot escape. At this point, possible configurations of the captured polygon form a pocket in the configuration space. The set of possible configurations grow from a single immobilizing configuration to a larger and larger pocket as the fingers are moved away from each other. Of course, this pocket cannot keep growing forever; when the distance between the two fingers is sufficiently large, pocket will break and the polygon can escape (Figure I.2(c).) This scenario can also be viewed in the reverse direction: from the point where the pocket is about to break, moving the fingers toward each other can bring the polygon from any configuration in the largest pocket to an immobilizing configuration. In essence, the range of pockets corresponds to the spectrum of uncertainty in the object configuration that can be handled. From a practical standpoint, it is interesting to know all different ways a concave polygon can be captured by two fingers. Also, for each such way, we want to know the maximum distance the fingers can stay apart from each other. Answering all these questions is the focus of this study.

I.2 Objective

Some notions need to be formally defined before stating the objective. Two fingers are at a critical distance apart when the polygon is captured by the fingers in such a way that moving the fingers infinitesimally away from each other will allow the polygon

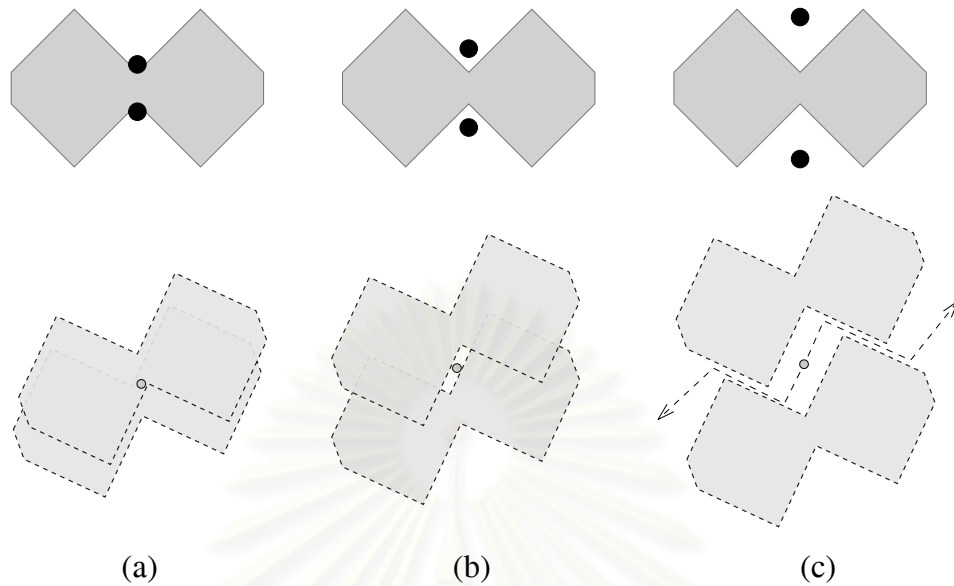


Figure I.2: The work space (first row) and the configuration space (second row) when: (a) the object is under an immobilizing grasping and captured, (b) the object is captured but not immobilized, and (c) the object is not captured.

to escape. The two fingers are said to form a maximal cage when they capture the object with separation below the critical distance. When a polygon is captured with a maximal cage, we may not be able to tell which grasping configuration the polygon will reach after the fingers have been closed (i.e. moving the fingers toward each other.) Figure I.3 shows two possible immobilizing configurations reachable from the same maximal cage. That is, for a given maximal cage, there is an associated set of grasping configurations reachable by the captured polygon after being squeezed by the fingers. Of course, the polygon remains captured during the entire finger squeezing action.

So far, we have considered only the cases for which an object is captured by being surrounded by the fingers. A concave polygon can also be captured from inside, e.g., a polygon with a hole can be captured by fixing a finger in the hole. Obviously, some concave polygon can be captured by two fingers in this manner by ensuring that the distance between the two capturing fingers is not too small (see Figure I.4.) Under this capturing situation, the polygon can be immobilized by stretching the fingers away

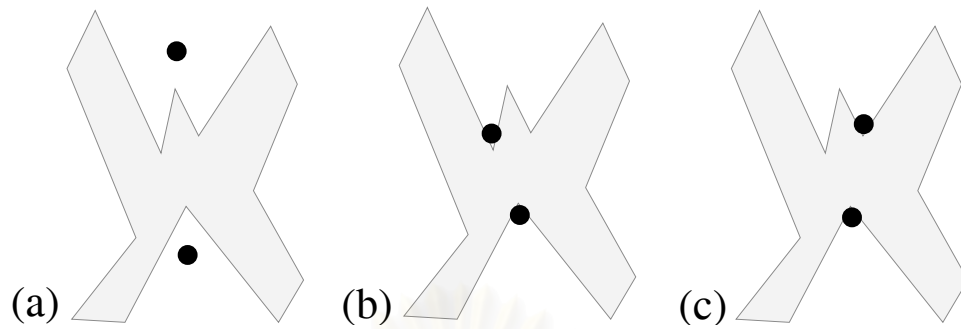


Figure I.3: Different grasping configurations (b) and (c) from the same maximal cage (a)

from each other. With direct analogy, this mode of capturing action provides a parallel track with earlier discussion: we automatically obtain the notions of *minimal cage*. It is worth noticing that this may also be view as if the two fingers (with a fixed distance apart) are caged by the object. It is important to mention that our interest is not on a set of capturing configurations that lead to a unique grasping configuration, but rather on maximal (resp. minimal) cage that provide maximum clearance, i.e., allow the fingers to be farthest away from (resp. closest to) each other. We therefore address the problem of characterizing all disjoint maximal and minimal cages, assuming pointed fingers and that the object be represented by k non-intersecting simple polygons.

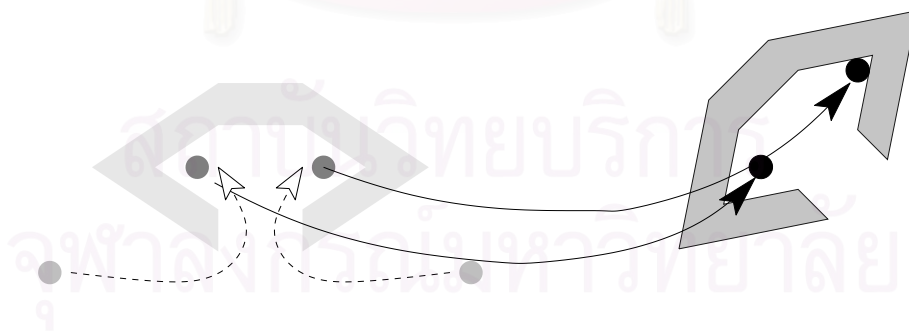


Figure I.4: Occasionally, a concave object can be captured and transported by stretching the two fingers at some configuration.

To summarize, the main objective of this thesis is to develop:

- An efficient algorithm that computes *all* possible maximal cages, minimal cages for an object with given geometry.
- An efficient algorithm that computes d^+ or d^- (see Chapter III.2, VI), and identifies its containing optimal (maximal or minimal) cage.

I.3 Scope of this Work

This thesis assumes the following facts:

- The object to be captured must be represented with a finite number of non-intersecting simple polygons on a plane (i.e there is no hole inside any of them.) In the case of the object with holes, we can always capture it by placing a finger inside one of those holes. On the other hand, if we use mobile robots that cannot move over the object instead of fingers, we can ignore those holes.
- The object's concavity is a necessary condition to capture with two fingers but not sufficient. Therefore, two-finger caging is not guaranteed for all object. We neither require that the input object represented with simple polygons must be a concave polygon nor must be capturable with two fingers. However, in such cases, the result will be reported as no possible caging set.
- Exactly two fingers are used in capturing the object. They are assumed to be points. Our work can apply to disc-shaped fingers of the same size by growing object with an amount equal to the fingers' radius.
- All finger trajectories considered are on a plane, continuous and collision-free.

I.4 Organization of the Thesis

In the next chapter, we review previous works that explored and applied the concept of capturing. Chapter III establishes the formal definition of the problem of maximal cage and basic tools used in analyzing such problem. Chapter IV is the

main contribution of our work describing the framework of our solution for *maximal cage problems*. Part of this chapter comes from our published paper [1]. Chapter V, based on Chapter IV, explains the algorithms for characterizing all maximal cages and computing d^+ . In Chapter VI, we adopt the same framework presented in Chapter IV to solve the *minimal cage problem*. Chapter VII explains how we can improve the solution to *maximal cage problem* and extending it to higher dimensions. Finally, we will conclude the thesis in the last chapter.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER II

RELATED WORKS

II.1 Caging/Capturing Problems

Early discussions in mathematics [2, 3] influenced Kuperberg [4] to pose the formal definition of the caging problem. In [4], Kuperberg stated that the caging problem is a problem of designing an algorithm for finding a set of points that prevent a polygon from moving arbitrarily far from a position. That is, the polygon will not be able to escape unless it penetrates by some of those points. This definition isolates the difficulties and uncertainties in dynamics, transforming the caging problem into a pure geometrical problem.

A comprehensive review on caging and its related problems such as grasping can be found in [5]. The concept of caging has been applied to a number of manipulation problems such as object grasping, part feeding [6] and transportation with mobile robots, for example [7]. Various methods to perform *error-tolerant* immobilizing grasp on a planar object are presented in [8, 9, 10, 11]. They are all based on two common steps: (i) command the fingers to enter a caging set of a desired immobilizing grasp, then (ii) the fingers move towards the immobilizing grasp configuration. Since the capturing is maintained during the latter phase, this approach guarantees the success of a grasping task. The differences among these works are their methods used in identifying caging sets, and the quality of caging sets themselves.

II.2 Error-Tolerant Grasping

Rimon and Blake [8] applied the stratified Morse theory [12] to solve the problem of caging an object on a plane with two-finger 1-DOF gripper and introduced the notion of *caging set* (also known as *capture region* [13]), a set of system configurations (i.e., finger formations) that can prevent the object from escaping. In brief, the concept is to find *critical points*, where topological changes in the *free space* caused by varying

finger separation distance (the *free space* here is the set of object configurations that are reachable from the immobilizing configuration.) Some of those *critical points* are then classified as *puncture points* whose the associated finger separation distance can be used in identifying maximal cages. Roughly speaking, if the fingers at the *puncture point* increase their separation distance just a little, squeezing the fingers will no longer guarantee the desired immobilizing grasp (assuming that the immobilizing grasp in this case requires finger squeezing.)

In this work Rimon and Blake focused on characterizing the caging set that leads to a specified immobilizing grasp. The implementation was based on numerical gradient descent search for the critical distance in which the topology of the free space changes. Numerical gradient descent search allowed the object and fingers (the primitives used in constructing the free space) to be modeled or estimated efficiently with curves. However, this method is rather complex and because of the use of numerical approach, the running time varies greatly depending on the shape of the free space.

Extended from [8], Davidson and Blake [9] moved towards the case of three-finger 1-DOF caging by presenting an algorithm for constructing the *free space* where the gradient descent search for *critical points* operated. In [14], Davidson and Blake worked on an application of error-tolerance grasping from vision based on stated theories in [8]. Both [8] and [9] relied on gradient descent searching for *critical points* in the configuration space. These tasks are rather complex and time consuming because they require complex geometrical computations in order to model the free space.

Gopal, Goldberg in [10] and Luwirawong, Sudsang in [11] confined their interest to two-finger, error-tolerance grasps at two particular concave sections. Mainly, [10] contributed: (i) an algorithm to locate all possible concave grasps in both 2D and 3D where fingers are cylinder, (ii) a quality metric computed from the four edges forming the concave sections. Unlike [11], the quality metric presented is based on rotation sensitivity of the grasp, not the maximum separation distance. In [11], a separation distance range is directly computed from the four edges at the two concave sections. Although the algorithm can quickly determine such distance, a caging set associated

with this distance can be very small in the case of curved object or poorly subdivided polygons..

II.3 Object Transportation

The problem we are addressing was also applied in object transportation tasks using multiple mobile robots. Researchers in the field, Sudsang, Ponce, Hyman and Kriegman in [15] studied the concept of *inescapable regions* which is equivalent to the concept of caging set. They formulated how to characterize a 2-DOF inescapable regions formed by three robots with respect to an immobilizing grasp. Again, this work considered only three edges (the number of edges considered are equal to the number of robots as in [11]) not the whole polygon. Sudsang, Rothganger and Ponce in [7, 16] introduced an object manipulation and transportation method based on *independent capture regions* for three robots – the robots are free to move within their own independent regions while guaranteeing the cage. This allows a convex object to be captured with three robots easily: the robots simply enter their own independent region. As long as they are in their own region, the object is guaranteed to be captured. Since the independent capture regions are close to a side of the object, the robot can push the object in such a way that all the robots are in their own independent capture regions during the push. By this mean, the robots can achieve a transportation task. Another utilization classified as object transportation task is object sweeping presented in [17].

Sudsang also presented the concept of the *width* of a 2D convex object in [18]. The width of an object is simply a measurement of how close two parallel lines, which do not intersect with the object, can be. If the width of a 2D rigid convex object is known, the object can be caged by a team of robots in a circle formation such that the distance between any two nearby robots is less than the width of the object. This is a simple yet sufficient condition for caging an object. The author further applied this concept on rigid concave objects. By partitioning a concave object into convex sub-objects, the object can clearly be caged when encircled by a team of robots for which the distance between two nearby robots is less than the maximum width of

the sub-objects. We can observe that the result, in this case, depends on the convex partitioning of the concave object, best partitioning should produce a convex sub-object with the greatest possible width.

Erickson et al. [19] proposed an algorithm using hidden surface removal techniques to approximate capture region for a rigid convex object with three disc-shaped robots. Unlike [16], the output of this method is not the independent capture region but rather a volume in the configuration space. This means that, after specifying a robot position, the regions allowed to place the other robots (in order to cage the object) is shrunk. This provides greater tolerance at the cost of more complex computation and the ease of usage.

Wang and Kumar [20, 21, 22, 23, 24, 25, 26, 27] focused on cooperative multiple robot caging manipulation. Wang and Kumar considered that the object is rotated (by the robots that pushes) at a negligible degree given a short time frame. Therefore, the trajectory of object that moves to infinity is limited only with translations. This provides larger cages and requires less complex computation. However, controlling the robots is not only to maintain their distance, but their cage formation also need to be updated every time slice. This is because the cage is not guaranteed when the object rotates.

II.4 Summary

Although our work is closest to that of [8], we aim for a more specific case which the captured object is assumed to be a simple polygon. Under this assumption, we do not need to solve a complex optimization as in [8]. The problem is transformed into a search in a finite discrete space. A distinct efficiency gain of our algorithm can be seen clearly as our algorithm runs a combinatorial search for all the caging sets over n^2 graph nodes (where n is the number of vertices) instead of performing gradient descent searches for all the caging sets in \mathbb{R}^4 configuration space.

CHAPTER III

BACKGROUND

This chapter serves as a detailed introduction to our problem. We shall present formal definitions, assumptions and a general model that formalizes past capturing problems and this one. At the end of this chapter, we shall introduce our fundamental tool for analyzing the problem used throughout the rest of the thesis.

III.1 Basic Assumptions

Our definition of capturing is based on the one that was posed by Kuperberg [4]. This assumption of capturing reduces the capturing problem to a pure geometrical one. The definition states that the object is said to be caged or captured when it cannot move arbitrarily far from a point without penetrating its obstacles. We further define that the captured or caged object is in a cage formed by the obstacles.

Apart from the definition of capturing, we also assume the following.

- (i) The problem is a planar problem. That is, we assume that all the obstacles (all the fingers) and the object must be on a plane. We will explain later, in Chapter VII.2, how we extend this concepts to higher-dimensional space.
- (ii) There are exactly two fingers and both of them are points. Therefore, the fingers can be located at the same position on the plane.
- (iii) There is exactly one rigid object. However, it is possible that the object consists of a number of disjoint components rigidly bound together. For example, when the object under consideration is a cross-section of a rigid body as shown in Figure III.1.
- (iv) The object under our consideration does not have any holes.

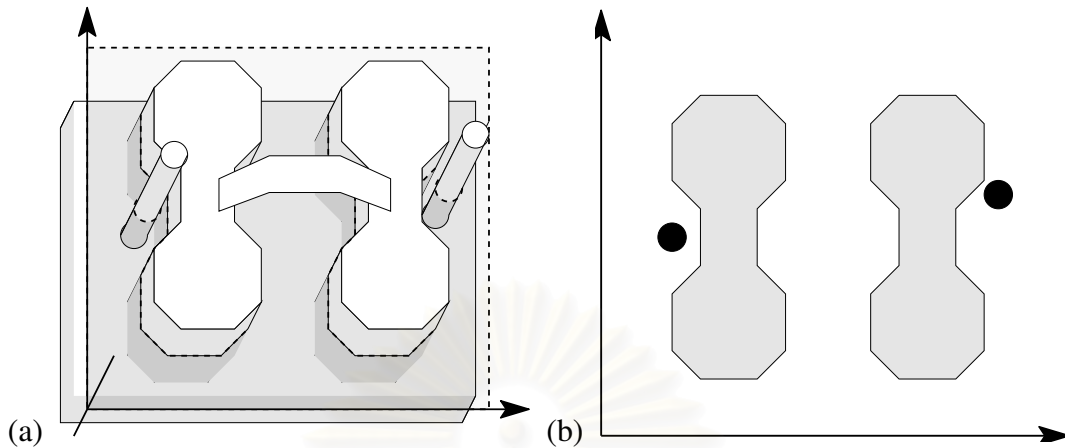


Figure III.1: We can consider only (a) a cross section of a rigid body object as in this case. This results in (b) a planar caging problem with two components that are rigidly bound together.

III.2 The Configuration Space

Since our system of interest has only three components, two fingers and a rigid body object – both on a plane, our configuration space therefore have at most 7-DOF. Naively, a configuration can be parameterized by: 3-DOF rigid body transformation of the object and two 2-DOF positions of the fingers. However, parameterizing a configuration this way poses a 3-DOF ambiguity because of the choice of coordinates. This is because, whether the object is captured is independent from the choice of coordinates.

Therefore, we parameterize the configuration space with 4-DOF. The two intuitive choices of parameterizations are:

- (i) Two, 2-DOF positions of the fingers.

Since the choice of coordinates is equivalent to that of possible rigid transformations of the object. We can fix the position and the orientation of the object and concern only the positions of the two fingers relative to the object. We can write a configuration as a pair of points: u, v . In each configuration, most of the time,

we are interested in a separation distance of a configuration $|\mathbf{u} - \mathbf{v}|$ which is the separation distance between the two fingers corresponding to the configuration.

- (ii) 3-DOF rigid body transformation of the object. and 1-DOF separation distance between the two fingers.

In this perspective, the configuration space can be viewed as it is constrained in such a way that one finger has to be at a fixed point, and both fingers must lie on an axis.

In our work, we mainly use on the first method of parameterizations. The latter parameterizations, which seems to be a more natural way to view this system, will be useful in illustrating examples in the next section,

III.3 The Maximal Cage Problem

Before formally defining the *maximal cage problem*, We shall begin by observing how an object is caged by two fingers.

First of all, let us consider two of the following extreme situations. One is when the distance between the fingers is zero (i.e. the jaw of two fingers fully closes.) We can say, for this situation, that the object can certainly escape from the fingers (i.e. is not caged.) This is equivalent to the case of one finger capturing. Since the object has no hole, according to our assumption, we cannot capture anything in this state. The other situation is when the object is immobilized by the fingers. In this case, the object cannot move – it is obviously captured (Figure III.2(a).)

From an immobilizing grasp, it can be easily observed that we can slightly increase the separation between the two fingers (i.e. we expand the jaw,) in such a way that the object is still held under captivity. At this state, if we keep the distance between the fingers below the current separation distance, we can guarantee that the object will be captured regardless of how it moves (Figure III.2(b).) This condition also holds regardless of how the jaws (the fingers) move as long as the separation distance is

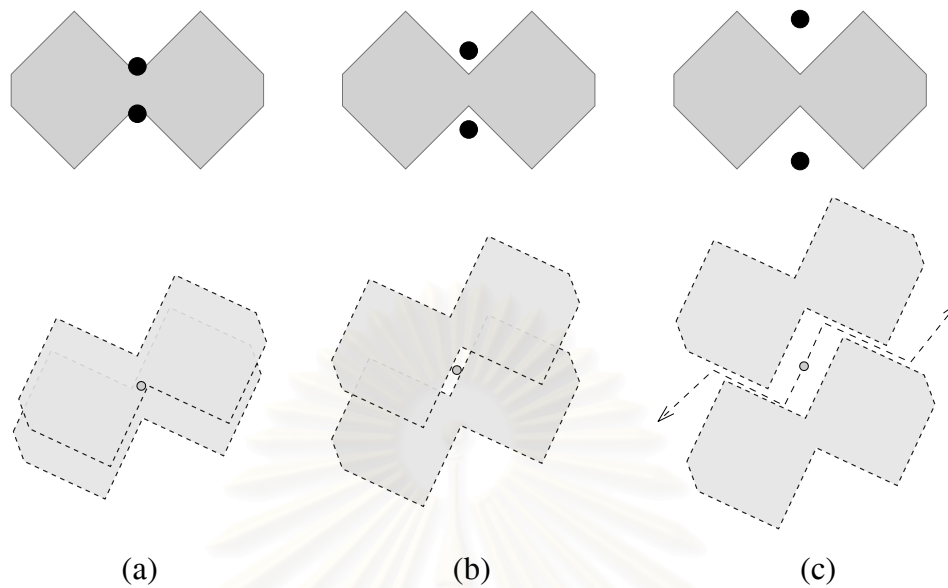


Figure III.2: The work space (first row) and the configuration space (second row) when: (a) the object is under an immobilizing grasping and captured, (b) the object is captured but not immobilized, and (c) the object is not captured.

maintained. Of course, we cannot arbitrarily increase the separation distance if we do not want to release the object. This is because, after expanding the separation distance to some degree (Figure III.2(c),) the object can move away from the fingers (Figure III.3(a),) letting the fingers see each other (Figure III.3(b).) At this state, the fingers can obviously move to the same point (i.e. the distance between the fingers is zero;) therefore, the object is no longer captured.

It can be concluded from these observations that we can set the object in a cage and maintain that cage if the following conditions are satisfied:

- (i) the fingers are initially at capturing configuration (i.e. where to place the fingers in order to form the cage,)
- (ii) the separation distance between the fingers is kept below an upper-bound separation distance.

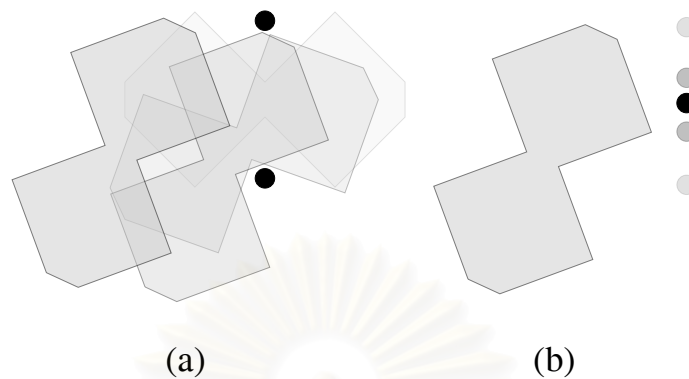


Figure III.3: (a) at some separation distance the object will no longer be captured, (b) the object can move arbitrarily far from the finger letting them see each other.

From these clues, the *maximal cage problem* is defined as follows: Given a configuration, what is the *least* upper-bound distance, such that as long as the separation distance between the two fingers is kept below such distance, the object is guaranteed to be captured?

In the next section, we shall introduce a tool that helps analyzing this problem.

III.4 Trajectories of Fingers

To ease understanding, let us first consider a less difficult problem called *0/1 maximal cage problem*. This problem questions, for a given initial configuration and a value, whether we can capture the object if we keep the separation distance between the fingers below such value.

In a sense, this problem can be viewed as an attempt to move the fingers from their initial positions until the distance between the two fingers is zero, while always keeping the largest separation distance between them below the given value. This leads us to perform an analysis based on *the trajectories of the fingers*.

Let the motion of each finger be described as a unit trajectory in the object's reference frame (fixing the object translation and rotation to eliminate the ambiguity in the choice of coordinates.) As generally defined, a unit trajectory \mathbf{p} is a trajectory that starts at $\mathbf{p}(0)$ and terminates at $\mathbf{p}(1)$, where $\mathbf{p}(t)$ denotes a position on a plane at a normalized time $t \in [0, 1]$. In order to represent the movement of the two fingers, or the configuration trajectory, we need a pair of synchronized finger trajectories \mathbf{p}, \mathbf{q} . Thus, our system's configuration at the time t can be written as $(\mathbf{p}(t), \mathbf{q}(t))$ (Figure III.4.)

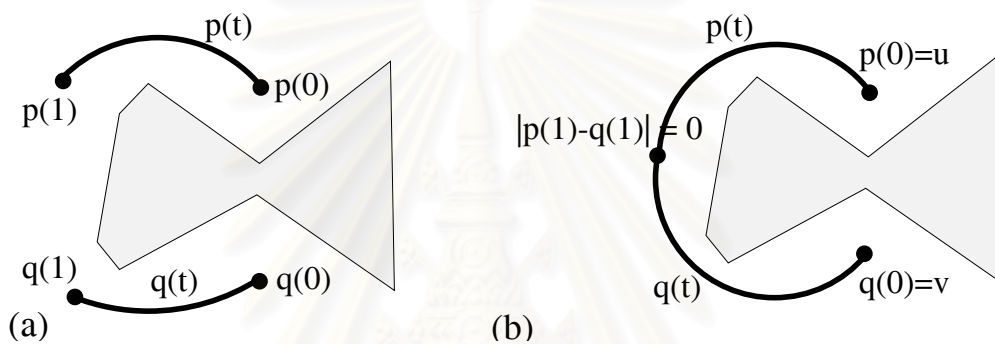


Figure III.4: (a) a synchronized trajectory pair. (b) an escape synchronized trajectory pair, a synchronized trajectory pair that both trajectories end when the distance between the two fingers reaches zero.

The most important property regarding a movement of the two fingers is the maximum separation distance between the two fingers during the entire movement. Given a synchronized pair of trajectories (\mathbf{p}, \mathbf{q}) , such separation distance can be expressed as:

$$[\mathbf{p}, \mathbf{q}] = \max_{0 \leq t \leq 1} |\mathbf{p}(t) - \mathbf{q}(t)|.$$

In the *maximal cage problem*, we are interested in a synchronized trajectory pairs of \mathbf{p}, \mathbf{q} such that \mathbf{p} and \mathbf{q} starts at some specific positions, say $\mathbf{p}(0) = \mathbf{u}$ and $\mathbf{q}(0) = \mathbf{v}$, and until the distance between the two fingers is zero (i.e. $|\mathbf{p}(1) - \mathbf{q}(1)| = 0$.) Such trajectory pair is referred as an escape trajectory pair. An example of an escape trajectory pair is illustrated in Figure III.4(b). This synchronized trajectory pair (\mathbf{p}, \mathbf{q}) clearly

allows the object to escape. In particular, to solve the *0/1 maximal cage problem* for a given value is to determine whether there exists an escape trajectory pair (\mathbf{p}, \mathbf{q}) such that $\lceil \mathbf{p}, \mathbf{q} \rceil$ is less than the value. For the general *maximal cage problem*, rather than answering a qualitative query, we need to find the least upper-bound distance which is the least possible $\lceil \mathbf{p}, \mathbf{q} \rceil$.

Since the least upper-bound separation distance will be frequently referred later, let us denote such distance by d^+ . Formally, we can define d^+ of a configuration (\mathbf{u}, \mathbf{v}) as follows.

$$d^+ = \min_{\forall \mathbf{p} \forall \mathbf{q}} \lceil \mathbf{p}, \mathbf{q} \rceil,$$

where (\mathbf{p}, \mathbf{q}) is a synchronized escape trajectory pair starting at (\mathbf{u}, \mathbf{v}) .

Before proceeding to the next chapter, let us summarize what we have obtained so far. We have learned that the *maximal cage problem* is all about finding d^+ given an initial configuration \mathbf{u}, \mathbf{v} . Furthermore, the object is guaranteed to be captured as long as the two fingers' separation distance is kept below d^+ . This also implies that: only when $|\mathbf{u} - \mathbf{v}| < d^+$, we can guarantee that we can capture the object by maintaining the separation distance below d^+ .

CHAPTER IV

SEARCH SPACE FORMULATION

In this chapter, the main contribution of this work, i.e., to reduce the configuration space into a finite graph, is presented. The solution to the *maximal cage problem* can be *searched* from this finite graph, which is smaller than the configuration space. For the problem of minimal cage, the search space formulation requires another methodology slightly different from this one. Those issues and the definitions of *minimal cage problem* are to be defined and explained in Chapter VI.

IV.1 Overview

Before delving into detail regarding the methodology proposed to transform the configuration space, let us first observe its underlying intuition. Recall the sequence of actions mentioned in the previous chapter:

- (i) the fingers perform an immobilizing grasp of the object and
- (ii) the fingers move away from each other but still do not allow the object to escape.

For illustration, we can again refer to Figure III.2(a)-(b). It is not difficult to tell that d^+ of both configurations are the same. Given that, at (ii), the object is captured, any escape trajectory pair must have its upper-bound distance at least d^+ with initial configuration at (ii). Since, from (ii), we can gradually reduce the separation distance so as to return to (i) without allowing the object to escape, this implies that d^+ of (i) and (ii) are the same.

From this clue, we can roughly say that, in general, configurations sharing a common d^+ are located near one another. This guides us to partition the configuration space into connected subsets each of which is to be referred to as a *configuration piece*. For a simple case shown in Figure IV.1(a), we shall demonstrate how a maximal cage is

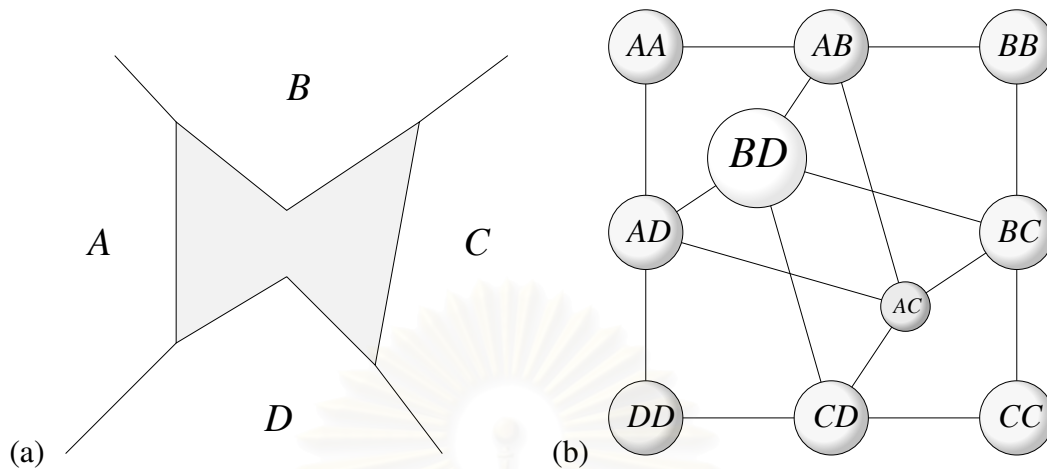


Figure IV.1: (a) A, B, C, D are convex regions after partitioning the free space. (b) The pieces and the topology.

characterized. First of all, let us partition the configuration space into 10 configuration pieces each of which is the cartesian product of two convex regions. For reference, we denote each configuration piece by two letters corresponding to the two convex regions involved (Figure IV.1(b).) Note that we neglect the other 6 pieces BA, CB, DC, DB, CA and DA . This is because the two fingers are considered identical, e.g., the piece AB is equivalent to BA .

We can visualize a synchronized trajectory pair (Figure IV.2(a)) as a motion that travels across the configuration pieces (Figure IV.2(b)².) The motion ends when a piece containing a configuration with $d^+ = 0$ is reached (AA in this case.) We call such configuration piece an *escape* piece.

Let us observe the case of Figure IV.1. Clearly, we can immediately tell that AA, BB, CC, DD are all escape pieces. In fact, AB is also an escape piece. This is because we can find a pair of points, one in A and the other in B that are close

²For simplicity of analysis, the link $BD \rightarrow AA$ is ignored (without any effect) by assuming that the two fingers cannot cross the boundaries of the convex regions at the same time. Likewise, we also ignore $BD \rightarrow CC, AD \rightarrow BB$ and $AD \rightarrow DD$.

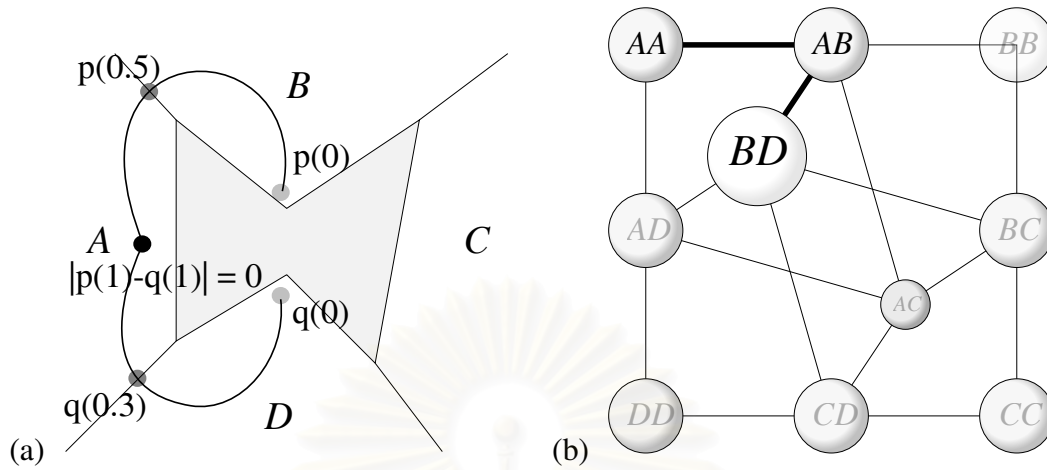


Figure IV.2: (a) Trajectories of fingers in free space. (b) Its corresponding travel as a sequence of pieces.

to each other. With the same reason, BC , CD and AD are escape pieces as well. In this case, it can be observed that no configurations in any escape pieces can cage the object.

Let us define the local minimum of a configuration piece be the configuration with the least separation distance in that piece. In case of AC , its local minimum is on the boundary of DD . Obviously, every configuration (\mathbf{u}, \mathbf{v}) in AC can reach the local minimum of AC with a synchronized trajectory pair (\mathbf{p}, \mathbf{q}) with $[\mathbf{p}, \mathbf{q}] = |\mathbf{u}' - \mathbf{v}'|$. Moreover, since we know that DD is an escape piece, no configurations in AC can cage the object. For the last piece BD , We can definitely cage the object when the fingers are initially at either the local minimum of BD , or some configuration (\mathbf{u}, \mathbf{v}) around the local minimum such that $|\mathbf{u} - \mathbf{v}|$ is less than d^+ of the local minimum. To characterize the maximal cage that contains (\mathbf{u}, \mathbf{v}) , we need to compute d^+ of BD which is obviously greater than $|\mathbf{u} - \mathbf{v}|$. Fortunately, we have already determined that we cannot cage the object when the fingers are in the four neighboring escape pieces (namely AC , CD , AD and BC .) We can shift our interest to the configurations of BD near those neighboring pieces (boundary between BD and its neighboring pieces) because any escape synchronized trajectory pair starting from a configuration in BD

must *pass* at least one configuration on the boundary. Consequently, d^+ of BD is simply the least separation distance among those boundary configurations.

From this example, we have developed an important insight on how to characterize the maximal cages. We have learned that we can start from trivially determining some escape pieces. Then, gradually determine maximal cages of the neighboring pieces until all the pieces are visited. In the next section, we will turn this insight into a concrete strategy. Mainly, we shall focus on how to identify space partitioning rules that allow us to reduce the search space into a finite graph.

IV.2 Characterizing All Maximal Cages

From the example observed previously, we need to identify required properties which can enforce the partitioned pieces to obey some properties so that we can apply our strategy which is resemble to the well known Dijkstra's shortest path algorithm [28] to characterize all the maximal cages. To begin, let us state the following proposition which follows directly from the definition of d^+ .

Proposition IV.2.1 *Given a configuration (\mathbf{u}, \mathbf{v}) such that $|\mathbf{u} - \mathbf{v}| < d^+$, every configuration $(\mathbf{u}', \mathbf{v}')$ has the same d^+ when it is reachable from (\mathbf{u}, \mathbf{v}) with a synchronized trajectory pair (\mathbf{p}, \mathbf{q}) such that $[\mathbf{p}, \mathbf{q}] < d^+$*

Definition IV.2.2 *A maximal cage is a connected set of configurations (\mathbf{u}, \mathbf{v}) such that $|\mathbf{u} - \mathbf{v}|$ is less than its d^+ .*

The *maximal cage problem* and the problem of characterizing all the maximal cages are closely related. Solving both problems are our objectives. By first characterizing all the maximal cages, a maximal cage problem is then reduced to identifying a containing maximal cage of a given configuration.

Ideally, our goal is to partition the configuration space into maximal cage and the rest. However, this cannot be accomplished directly, since we do not know exactly

what the configurations on the boundaries of the maximal cages are. To simplify the problem, we first partition the configuration space into connected subsets in such a way that each of which intersects with *at most one maximal cage* (no piece contains members from distinct maximal cages.) As in the previous section, we shall address these connected subsets as configuration pieces. We also define that each partitioned piece P is associated with d_P^+ which is the least d^+ of configurations in the piece. Apart from a constraint that P can intersect at most one maximal cage, a configuration (\mathbf{u}, \mathbf{v}) in the piece P is either:

- (i) in the only maximal cage that intersects this piece, when $|\mathbf{u} - \mathbf{v}| < d_P^+$, or
- (ii) not in any maximal cage, otherwise.

The space partitioning that satisfies the aforementioned properties obviously implies basic constraints imposed by each piece to its neighbors.

Proposition IV.2.3 *Let P and Q be two pieces that share a boundary B , the distance d_P^+ is at most*

$$\max(\min_{(\mathbf{u}, \mathbf{v}) \in B} |\mathbf{u} - \mathbf{v}|, d_Q^+)$$

Proof:

- (i) For the case that there is (\mathbf{u}, \mathbf{v}) in B such that $|\mathbf{u} - \mathbf{v}| < d_Q^+$, there must exist $(\mathbf{u}', \mathbf{v}')$ in Q near (\mathbf{u}, \mathbf{v}) such that $|\mathbf{u}' - \mathbf{v}'| = |\mathbf{u} - \mathbf{v}|$, because the separation-distance function $f(\mathbf{u}, \mathbf{v}) = |\mathbf{u} - \mathbf{v}|$ is continuous, Since $(\mathbf{u}', \mathbf{v}')$ is in the maximal cage of Q , d_P^+ is at most d_Q^+ .
- (ii) For the other case that there is no such (\mathbf{u}, \mathbf{v}) in B that $|\mathbf{u} - \mathbf{v}| < d_Q^+$, every $(\mathbf{u}', \mathbf{v}')$ in Q near B is not in the maximal cage of Q . Hence, d_P^+ is at most $\min_{(\mathbf{u}, \mathbf{v}) \in B} |\mathbf{u} - \mathbf{v}|$.

From (i) and (ii), we can conclude that

$$d_P^+ \leq \max(\min_{(\mathbf{u}, \mathbf{v}) \in B} |\mathbf{u} - \mathbf{v}|, d_Q^+)$$

■

This leads to the most important fact that:

Lemma IV.2.4 *Given a piece P and the set of its neighbors \mathbf{Q} , the distance d_P^+ is equal to*

$$\min_{\forall Q \in \mathbf{Q}} \{ \max(\min_{(\mathbf{u}, \mathbf{v}) \in B(P, Q)} |\mathbf{u} - \mathbf{v}|, d_Q^+) \},$$

where $B(P, Q)$ is the boundary between P and Q .

The proof of this lemma is shown in Appendix A.

With Lemma IV.2.4, the relationship among d^+ of the pieces have been formalized. This can be viewed as constraints placed over d^+ of the pieces. To visualize this, we can represent the configuration pieces and the boundaries between pairs of pieces using states and transitions (resp.) of a graph. In this graph:

- (i) Each state (configuration piece) P is labelled with d_P^+ .
- (ii) Each transition is labeled with t^+ the least separation distance in its corresponding boundary (for a transition that links between pieces P and Q , t^+ is the least separation distance of a configuration on the boundary between P and Q .)

The constraint placed on a state P can be interpreted as follows.

d_P^+ is equal to the minimum of $\max(t^+, d_Q^+)$; for all Q , an adjacent state of P , where t^+ is the transition distance of the transition linking P and Q .

The similar pattern can be observed in Dijkstra's shortest path algorithm. Let us suppose (for a few statements) that d_P^+ is the shortest distance from an initial state to P instead of the least upper-bound distance of P to an escape piece (an initial state.) We have that:

d_P^+ is equal to the minimum of $t^+ + d_Q^+$; for all Q , an adjacent state of P , where t^+ is the transition distance of the transition linking P and Q .

Obviously, the only different is that the addition operator is used instead of the maximum operator. Since t^+ is a function of P and Q (the transition,) we can change the maximum operator to the addition operator by replacing t^+ with τ^+ such that

$$\tau^+ + d_Q^+ = \max(t^+, d_Q^+)$$

Therefore, we have:

$$\tau^+ = \max(t^+, d_Q^+) - d_Q^+$$

Since τ^+ is clearly not less than zero, this proves that our problem can be reduced to the shortest path problem.

In Dijkstra's shortest path algorithm, the constraints are gradually resolved by propagating d^+ from the *initial state* (defined by the user) whose d^+ is zero. Our algorithm mimics this behavior by propagating our d^+ (the least upper-bound distance) from an *initial state* whose d^+ is zero. That is, we need to know at least one piece with $d^+ = 0$, which can be found by identifying a piece containing a configuration (\mathbf{u}, \mathbf{v}) such that $|\mathbf{u} - \mathbf{v}| = 0$.

Here is the sketch of the algorithm:

- (1) Initially, every state's d^+ (except initial state which is set to 0) is set to infinity.
- (2) Initially, every state is *not visited*.
- (3) Add an initial state to a Heap (a min-heap data structure.)
(d^+ is used as the priority for this Heap.)

- (4) While the Heap is *not empty*
- (5) Extract a state which is *not visited*, call it P .
- (6) Then, update d^+ of each adjacent state Q as follow.

$$d_Q^+ \leftarrow \min(d_Q^+, \max(t^+, d_P^+)),$$
 where t^+ is the transition distance for the transition linking P and Q .
- (7) If Q is not already in the Heap add Q to the Heap
 otherwise, update the priority of Q .
- (8) Mark P as a *visited state*.

To summarize this section, our strategy for characterizing all the maximal cages is explained in detail. This strategy requires us to first partition the configuration space such that the partitioned pieces must obey the following conditions.

- (i) each piece must intersect at most one maximal cage, and
- (ii) if the piece intersects a maximal cage, every configuration (\mathbf{u}, \mathbf{v}) in the piece such that $|\mathbf{u} - \mathbf{v}|$ is less than d^+ of the maximal cage is in the intersecting maximal cage.

Once d^+ of all pieces are determined, the maximal cages are characterized by union of connected maximal cages inside the pieces.

In the following section, an efficient configuration space partition rule will be introduced.

IV.3 Space Partitioning

Here, we narrow down our consideration to polygonal objects by including a few additional geometrical assumptions. We require that the object be represented by k

non-intersecting simple polygons with n vertices in total. Let us denote by e_1, e_2, \dots, e_n the edges of the simple polygons.

Before formally stating the space partitioning rule let us define an operation called *squeeze*.

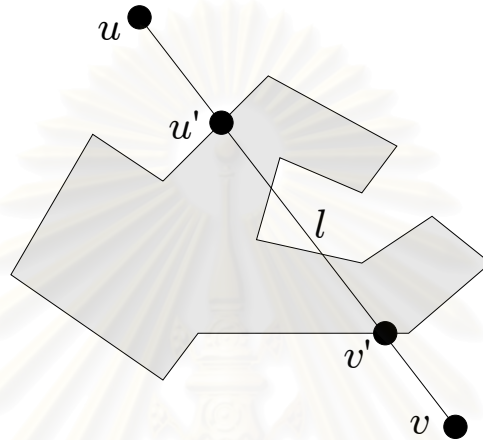


Figure IV.3: An illustration for a *squeeze* operation that squeezes u, v along the line segment l (a line segment joining u and v) to u' and v' .

Let l be a line segment joining u and v as in Figure IV.3. A configuration obtained after squeezing configuration (u, v) , not visible to each other, is configuration (u', v') when u' (resp. v') is the point of l that is near an object's edge and is closest to u (resp. v).

Of course, configurations (u, v) such that u and v are visible to each other always have its d^+ equal to $|u - v|$. We can ignore these configurations from our consideration since their d^+ can be trivially determined. With this in mind, we partition the rest of the configuration space in such a way that: configurations in the form (u, v) that *squeeze* to the same pair of edges belong to the same piece. This way, we obtain at most n^2 pieces from all possible combinations of any two edges. We shall refer to our partitioned pieces as P_{ij} where $i, j \in \{1, 2, \dots, n\}$. It is possible that some pieces are empty (i.e. no configuration that *squeezes* to the edge pair of such piece.) Those

empty pieces can be safely discarded from our considerations.

To verify whether this configuration space partitioning satisfies all the desired properties listed in the previous section, it is sufficient to show that the following proposition is true.

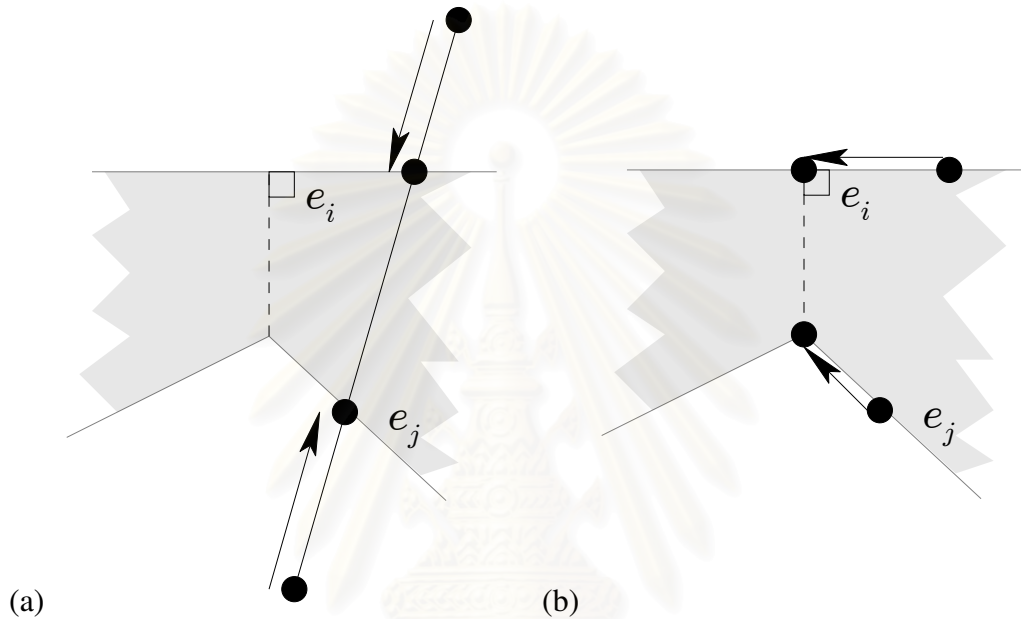


Figure IV.4: (a) first, squeeze a configuration in P_{ij} to a pair of edges e_i, e_j . (b) then, follow the gradient descent to a local minimum of P_{ij} .

Proposition IV.3.1 *The fingers starting from a configuration (\mathbf{u}, \mathbf{v}) can move to another configuration $(\mathbf{u}', \mathbf{v}')$ without increasing the fingers' separation distance during the motion. Given that both configurations are in P_{ij} , and $(\mathbf{u}', \mathbf{v}')$ is a local minimum of P_{ij} .*

Proof: it is clear that $(\mathbf{u}', \mathbf{v}')$ is situated on an edge pair, namely e_i, e_j . We first *squeeze* the fingers to such edge pair by gradually reducing their separation distance (Figure IV.4(a).) From here, we can just follow the gradient descent to the local

minimum of P_{ij} (Figure IV.4(b).) Obviously, these two-step operations would not increase the separation distance. ■

We can imply the following facts from this proposition:

- (i) it is not possible for P_{ij} to intersect more than one maximal cage. Consider the situation that (\mathbf{u}, \mathbf{v}) is in one maximal cage while a local minimum is in another maximal cage. Upper-bound separation distance of a synchronized trajectory pair (\mathbf{p}, \mathbf{q}) from (\mathbf{u}, \mathbf{v}) to the local minimum $([\mathbf{p}, \mathbf{q}])$ must not be less than d^+ of (\mathbf{u}, \mathbf{v}) , which is greater than $|\mathbf{u} - \mathbf{v}|$. This is a contradiction.
- (ii) if P_{ij} intersects a maximal cage, all local minima are in the maximal cage. Suppose that a local minimum is not in the maximal cage, then its d^+ is equal to its separation distance. However, any configuration can reach this local minimum without increasing the separation distance; therefore, P_{ij} does not intersect any maximal cage. This is a contradiction.
- (iii) if P_{ij} intersects a maximal cage, every configuration having the separation distance less than d^+ of the maximal cage is in the maximal cage. Suppose that a configuration (\mathbf{u}, \mathbf{v}) for which $|\mathbf{u} - \mathbf{v}|$ is less than d^+ of the maximal cage is not in the maximal cage, then its d^+ is equal to its separation distance. However, from a local minimum configuration, (\mathbf{u}, \mathbf{v}) can be reached with a synchronized trajectory pair with the separation distance of $|\mathbf{u} - \mathbf{v}|$; therefore, d^+ of the local minimum is equal to $|\mathbf{u} - \mathbf{v}|$. This contradicts with (ii).

Consequently, we can conclude that:

Lemma IV.3.2 *For the problem of characterizing all the maximal cages, every partitioned pieces derived from the partitioning rule stated earlier satisfies the required properties.*

In the next section, we shall proceed to the analysis on how the pieces are connected to one another under this partition rule.

IV.4 Topology of Pieces

Due to the nature of squeeze operation applied in the space partitioning, we can restrict our consideration to configurations on the edges. Roughly speaking, P_{ij} has a chance to connect to P_{ik} when some point on e_j sees e_k . One obvious necessary condition is that, e_j and e_k share a common vertex (Figure IV.5(a).) Sliding fingers infinitesimally to the right will change the fingers' containing piece from P_{ij} to P_{ik} . The other is when one finger on e_j is about to leave e_j (Figure IV.5(b).) If the finger on e_j infinitesimally moves to the right, both fingers will be squeezed to e_i, e_k ; therefore, inside P_{ik} instead of P_{ij} . When any two pieces, say P_{ij} and P_{ik} , are connected, we say that they are connected with a *transition*. A *transition* is associated with its *cost*, t_{ijk}^+ , which is the least separation distance from a configuration on the boundary between P_{ij} and P_{ik} . In this section, we address the problem of characterizing the transitions and how to compute the *cost* of those transitions in this section. After this, we will have all of our search space components, which consists of the pieces (states,) the transitions and the transition costs, summarized.

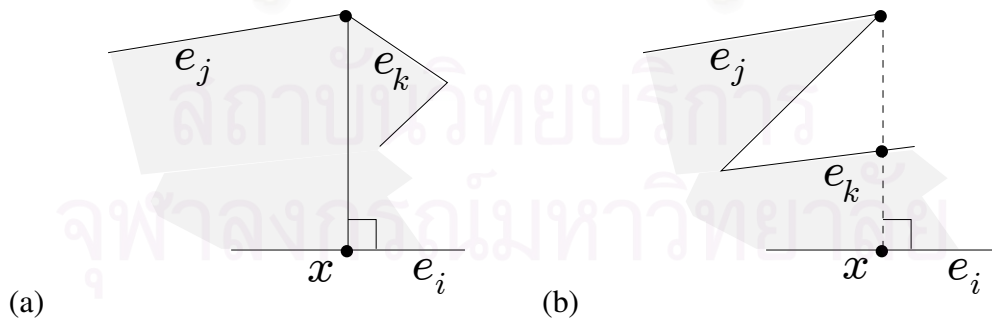


Figure IV.5: For both (a) and (b), if we move the finger on e_j to the right infinitesimally, the piece containing the fingers will change from P_{ij} to P_{ik} .

In the worst case situation, the number of transitions a piece associated with can be as high as $O(n)$. Fortunately, we have found out that we do not need to include all possible transitions in our search space. That is: some transitions can be ignored without affecting the computation of any pieces' d^+ .

Given a vertex v and an edge e_i , also let x be the point closest to v on e_i . Only transitions linking P_{ij} and P_{ik} , a piece near the configuration v, x will be included in the search space. Such transitions are called *basic transitions*. To gain a better understanding, we can further classify basic transitions into two types, based on whether or not e_k is adjacent to e_j . Their transition distances and explanations are as follows:

(i) *Transition between adjacent edges* (Figure IV.5(a).)

It is obvious that any configuration associated with a situation when a finger is at v is on the boundary of P_{ij} and P_{ik} . v, x is therefore always a valid choice to transit from P_{ij} to P_{ik} because it is clearly on the boundary between P_{ij} and P_{ik} . Moreover, this configuration has the least separation distance among other configurations on such boundary. Hence, $t_{ijk}^+ = |v - x|$.

(ii) *Transition between non adjacent edges* (Figure IV.5(b).)

where e_k is the first edge with which a ray from v to x intersects (Figure IV.7.) Here, we can apply the same reasoning as previously so as to prove that $t_{ijk}^+ = |v - x|$.

For the situation that either P_{ij} or P_{ik} is an empty piece, a transition is not valid; therefore, not included.

Let us consider the case as in Figure IV.6(a). P_{im} is connected to P_{ij} but not directly linked by a transition from P_{ij} to P_{im}

The question that arises is whether d^+ of P_{im} can be computed correctly with

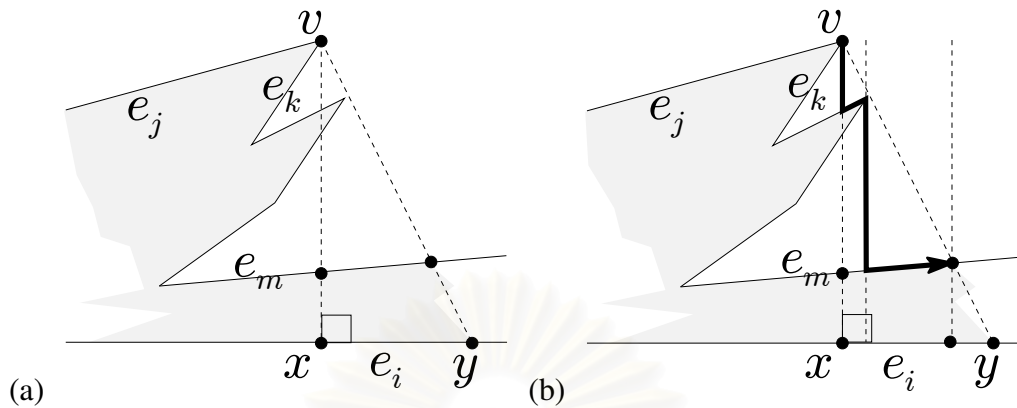


Figure IV.6: (a) possible transition from P_{ij} to P_{im} but not a basic transition, (b) trajectory for a finger (the other is placed on e_i) that can be converted to a sequence of basic transitions.

the described d^+ propagation method using only basic transitions. We can answer this question by showing that we can apply successive basic transitions starting from P_{ij} to P_{im} (we are interested in the case when there is no basic transition from P_{ij} to P_{im}) such that the maximum transition cost in such sequence is not greater than $t_{ijm}^+ = |\mathbf{v} - \mathbf{y}|$. Actually, we can easily determine such sequence. This is because it can be converted from a trajectory of finger in Figure IV.6(b) paired together with another trajectory on e_i .

Another interesting case is shown in Figure IV.7, it is clear that we can take a transition between P_{ik} and P_{il} without placing at least one finger at a vertex of the object. Again, we can use the same approach to show that this kind of transition is not necessary. Consequently, this generalizes that only basic transitions are sufficient.

IV.5 The Reduced Search Space

From insights we have obtained so far, we are now ready to concrete the definition of our search space. Our search space is simply an undirected graph reduced from the

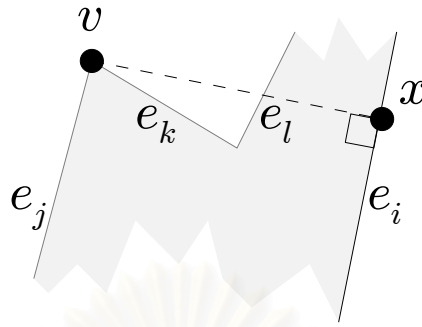


Figure IV.7: Possible transition from P_{ik} to P_{il} when neither of the two fingers are at a vertex of the object.

configuration space such that, only with this reduced information, searching this graph is sufficient to answer the *maximal cage problem* and to characterize all the maximal cages.

Given an object geometry described with k simple non-intersecting simple polygons with the edges e_1, e_2, \dots, e_n , this graph is a collection of states and transitions (\mathbf{S}, \mathbf{T}) defined as follows.

1. *States (S).*

Each distinct state is exactly a distinct non-empty partitioned piece P_{ij} with respect to the input object (see Section IV.3.)

$$\mathbf{S} = \{P_{ij} \mid P_{ij} \neq \emptyset\}$$

We also assume that the two fingers are the same; therefore, piece P_{ij} is equivalent to P_{ji} . Each piece P_{ij} is associated with d_{ij}^+ , where d_{kk}^+ for any $k \in \{1, 2, \dots, n\}$ is trivially known to be zero.

2. *Transitions (T).*

A transition is a combination of two distinct pieces, which can be written in the form $\{P_{ij}, P_{ik}\}$. However, not all transitions are included in this graph. This is to

improve the efficiency of the search algorithm. Only basic transitions described in the previous section are included in the graph.

$$\mathbf{T} = \{\{P_{ij}, P_{ik}\} \mid \{P_{ij}, P_{ik}\} \text{ is a basic transition}\}$$

A transition $\{P_{ij}, P_{ik}\}$ is also associated with a transition distance t_{ijk}^+ , an important information used during propagating d^+ among the pieces.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER V

THE ALGORITHMS

In the previous chapter, we have shown how we reduce the configuration space to a graph. This chapter intends to fill in the implementation detail of all algorithms related to this graph, ranging from the graph construction, the d^+ propagation, and the d^+ query algorithms.

V.1 Identifying Empty Pieces

Whether a piece P_{ij} is empty depends on the relative transformation between e_i and e_j . In particular, we seek to answer whether there are any two points in the free space that squeeze to the edge pair, e_i and e_j . In a sense, the answer to this problem requires the knowledge on the facing direction of both e_i and e_j , called \vec{n}_i and \vec{n}_j (resp.)

Let us consider a trivial case when e_i and e_j are simply a point $e_i = \{v_i\}$, $e_j = \{v_j\}$. The precondition for the fingers to squeeze to e_i and e_j is that both fingers must be on the line defined by v_i and v_j . Intuitively, we say that the e_i and e_j can only *support finger squeezing* (hence, form a valid piece) when the following algebraic condition is satisfied (see Figure V.1.(a),(b) for a couple of examples.)

$$\{(v_i - v_j) \cdot \vec{n}_i < 0\} \wedge \{(v_j - v_i) \cdot \vec{n}_j < 0\}$$

In general cases, e_i and e_j are lines defined by v_i, v_{i+1} and v_j, v_{j+1} (resp.) We can simply answer this problem by determining whether at least one of the four possible vertex pairs (i.e. (i) v_i, v_j , (ii) v_i, v_{j+1} , (iii) v_{i+1}, v_j or (iv) v_{i+1}, v_{j+1} supports finger squeezing. Although we neglect possible combinations of any two points on e_i and e_j , this is still sufficient because e_i and e_j are straight edges (Figure V.2.)

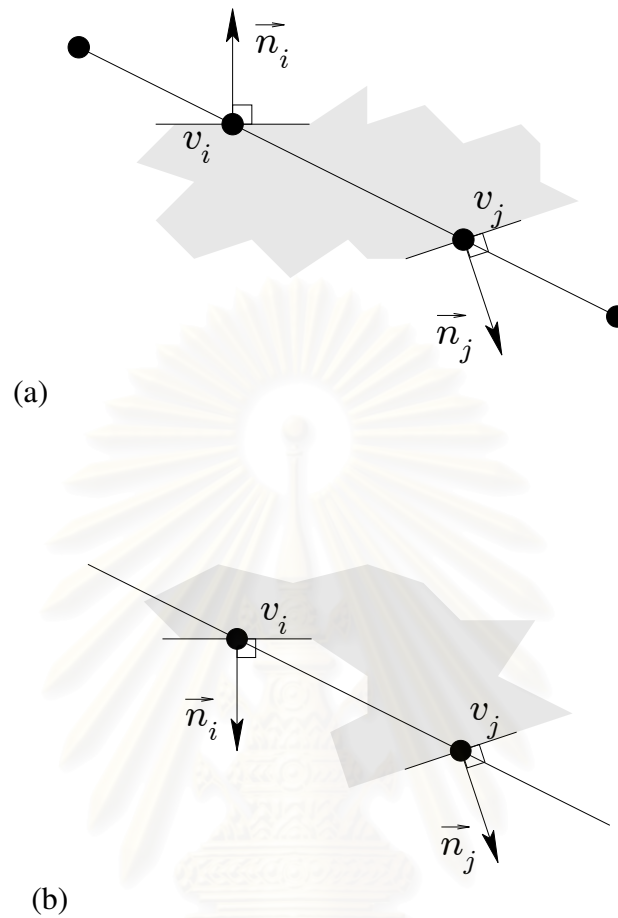


Figure V.1: (a) v_i and v_j can support finger squeezing, (b) v_i and v_j cannot support any finger squeezing.)

V.2 Generating Transitions

This section, we shall focus on the task of characterizing all the basic transitions in the graph. As transitions are defined on three edges, exhaustively verifying whether each possible transition is a basic transition would consume too much time. Fortunately, this can be avoided by generating all the basic transitions from every distinct pair of a vertex and an edge.

It can be easily observed that every basic transition is always associated with a vertex and an edge. Given v and e_i , at most three basic transitions can be generated.

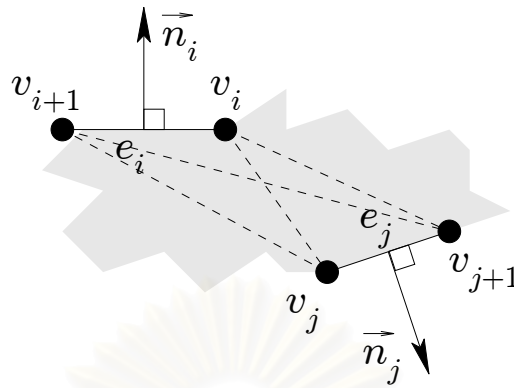


Figure V.2: Determining whether a piece P_{ij} is valid (see text.)

They are:

- (i) $\{P_{ij}, P_{ik}\}$,
- (ii) $\{P_{ij}, P_{il}\}$,
- (iii) $\{P_{ik}, P_{il}\}$.

Given that e_j , e_k , e_l , and \mathbf{x} are defined as in Chapter IV.4. All these transitions have the same distance of $t_{ijk}^+ = t_{ijl}^+ = t_{ikl}^+ = |\mathbf{v} - \mathbf{x}|$. By generating basic transitions from all distinct pairs of a vertex and an edge, we obtain all the basic transitions.

V.3 Propagating Least Upper-Bound Distance

As seen in the previous chapter, least upper-bound distance propagation and Dijkstra shortest path algorithm are alike. In the case of least upper-bound separation distance propagation, the goal is to find paths with least upper-bound distance to all possible P_{ij} . Those paths must begin at some P_{kk} . The main structure of the algorithm for this task can be readily adopted from that of the famous Dijkstra's shortest path. Only two minor modifications are required. They are:

- (i) we will propagate d^+ instead of the shortest distance. When P_{ij} is being visited, for every neighboring state P_{ik} , d_{ik}^+ will be updated to $\min(d_{ik}^+, \max(d_{ij}^+, t_{ijk}^+))$
- (ii) the propagation starts from all possible P_{kk} , whose d^+ are known to be zero. instead of starting points whose shortest distances are known to be zero. Equivalently, this can be achieved by combining all pieces of the form P_{kk} to a single piece.

V.4 Identifying All Maximal Cages

After P_{ij} is labeled with its d_{ij}^+ , we can say that P_{ij} is associated with a caging set C_{ij} , where

$$C_{ij} = \{\mathbf{u} \in e_i, \mathbf{v} \in e_j \mid \|\mathbf{u} - \mathbf{v}\| < d_{ij}^+\}.$$

Most of the time, caging sets are not disjoint. By the definition of maximal cage and Chapter IV.4, it can be seen clearly that any two caging sets, C_{ij} and C_{ik} , are of the same maximal cage, if all of the following conditions are satisfied:

- (i) $\{P_{ij}, P_{ik}\}$ is a basic transition.
- (ii) $t_{ijk}^+ < d_{ij}^+ = d_{ik}^+$,

Actually, these conditions are from the definition of maximal cage (Definition IV.2.2.) A couple of examples are illustrated in Figure V.3(a) and Figure V.3(b).

In the implementation, we need a disjoint set data structure for this task. The union between C_{ij} and C_{ik} will be performed during the propagation. The pseudo code for identifying maximal cages and propagating d^+ is shown in Figure V.4. After the algorithm execution, the maximal cages are represented with the roots of this disjoint-set structure.

Some results, each produced under 0.1 second using a PC with 1.8 GHz CPU, are listed in Figure V.5.

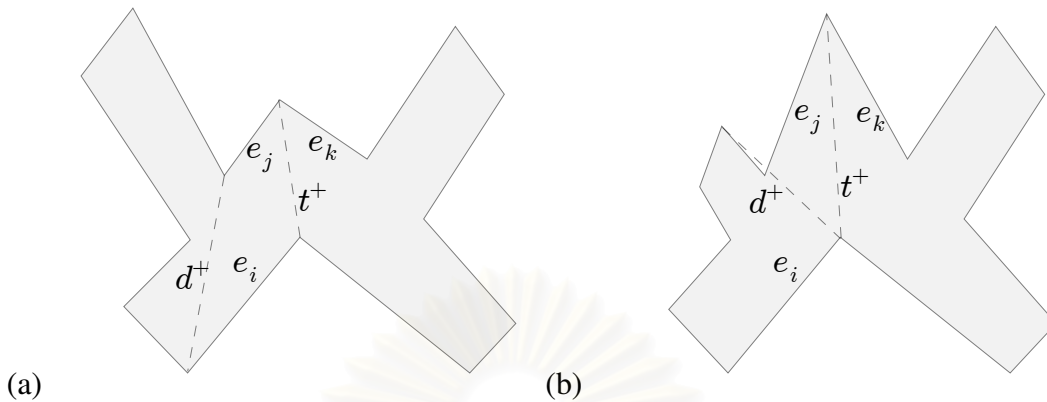


Figure V.3: Two example situations when to, (a), and when not to, (b), merge two maximal cages (a) $t^+ = t_{ijk}^+ < d^+ = d_{ij}^+ = d_{ik}^+$, C_{ij} and C_{ik} are of the same maximal cage. (b) $t^+ = t_{ijk}^+ = d_{ik}^+ > d^+ = d_{ij}^+$, C_{ij} and C_{ik} are not of the same maximal cage.

V.5 Querying Least Upper-Bound Distance and Maximal Cage

We can query whether a pair of fingers, initially placed on \mathbf{u} and \mathbf{v} , is in which maximal cage by squeezing them using two ray shoot queries. If the two fingers are not already visible to each other, they would land on a pair of edges, supposedly e_i and e_j . Hence, d^+ of \mathbf{u} , \mathbf{v} is:

$$d^+ = \begin{cases} d_{ij}^+, & d_{ij}^+ < |\mathbf{u} - \mathbf{v}|; \\ |\mathbf{u} - \mathbf{v}|, & \text{otherwise.} \end{cases}$$

If $d_{ij}^+ < |\mathbf{u} - \mathbf{v}|$, they are not inside C_{ij} and; therefore, they will not be in any maximal cage. Otherwise, the maximal cage they are in is the *root* of C_{ij} . Here, *root* refers to the root in the disjoint-set structure constructed as in the previous section.

V.6 Time Complexity Analysis

The process of obtaining d^+ of all the pieces and all maximal cages splits into two sequential tasks:

```

procedure Propagate(S, T)
1:  $\Psi \leftarrow \emptyset$ 
2: for  $1 \leq i \leq j \leq n$  do
3:   HeapInsert( $P_{ii}$ )
4:    $d_{ij}^+ \leftarrow \begin{cases} 0, & i = j; \\ \infty, & \text{otherwise.} \end{cases}$ 
5: end for
6: while not IsHeapEmpty( ) do
7:    $P_{ij} \leftarrow \text{HeapExtractMin}(\ )$ 
8:   if  $P_{ij} \notin \Psi$  then
9:      $\Psi \leftarrow \Psi \cup \{P_{ij}\}$ 
10:     $C_{ij} \leftarrow \{(\mathbf{u}, \mathbf{v}) \in P_{ij} \mid |\mathbf{u} - \mathbf{v}| < d_{ij}^+\}$ 
11:    for all  $T = \{P_{ij}, P_{ik}\} \in \mathbf{T}$  do
12:      if  $P_{ik} \notin \Psi$  then
13:         $d_{ik}^+ \leftarrow \min(d_{ik}^+, \max(d_{ij}^+, t_{ijk}^+))$ 
14:        HeapInsert( $P_{ik}$ )
15:      else if  $t_{ijk}^+ < d_{ij}^+ = d_{ik}^+$  then
16:        DisjointSetUnion( $C_{ij}, C_{ik}$ )
17:      end if
18:    end for
19:  end if
20: end while

```

Figure V.4: This algorithm determines d^+ for all states (pieces) in \mathbf{S} and modifies the disjoint set data structure (initially all C_{ij} are disjoint.) For the variable, Ψ is a set containing visited nodes.

- (i) creating the graph: partitioning the configuration space and generating the basic transitions,
- (ii) propagating d^+ in such graph, and modifying the disjoint set.

The first task spent n^2 ray shoot queries on determining *transitions between non-adjacent edges*. We applied the ray shoot algorithm proposed by Hershberger and Suri [29] which has $O(\sqrt{k} \lg n)$ query time (for k simple polygons) and below $O(n^2)$ pre-computation time (required in performing Steiner triangulation on the input object.) This task, therefore, runs in $O(n^2 \sqrt{k} \lg n)$. For the other task, we need n^2 *HeapExtractMin*'s, at most $3n^2$ *HeapInsert*'s, and at most n^2 *DisjointSetUnion*'s. Each of those operations run in $O(\lg n)$. Hence, the overall running time ((i) and (ii)) is

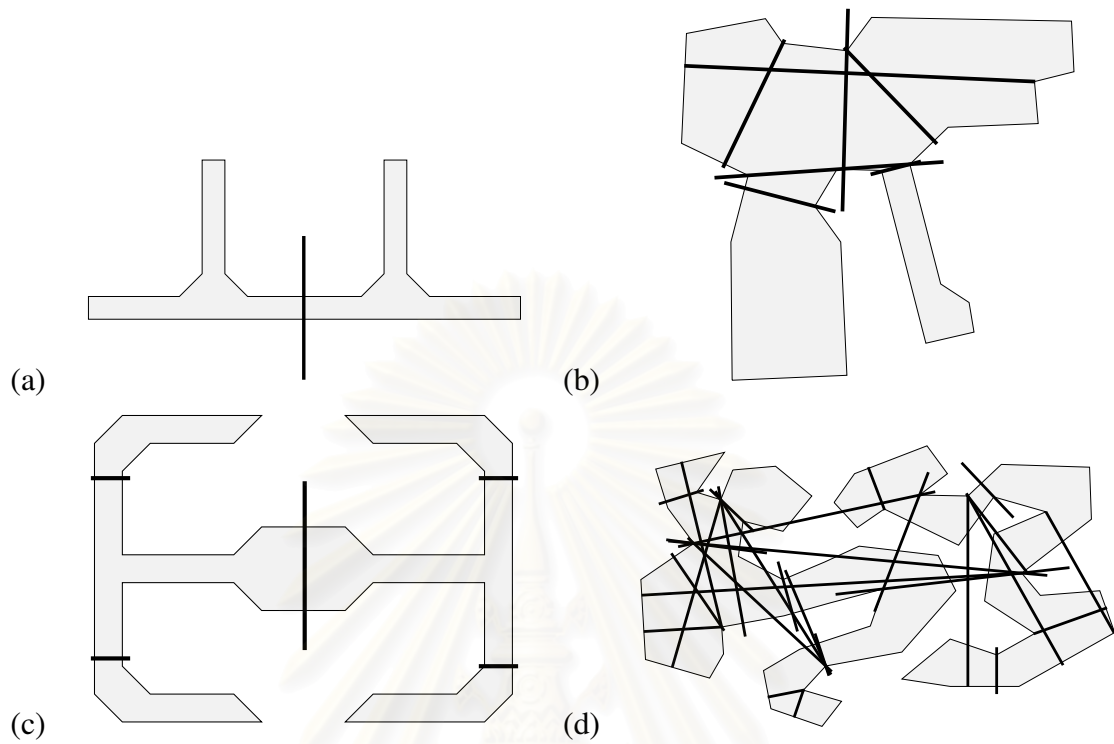


Figure V.5: Examples of outputs from characterizing all maximal cages. Line segments in the figures roughly indicates the location of maximal cages, whose d^+ are represented with segments' length.

$$O(n^2\sqrt{k} \lg n) + O(n^2 \lg n) = O(n^2\sqrt{k} \lg n).$$

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER VI

MINIMAL CAGE

In addition to the *maximal cage problem*, the explained concept in Chapter IV can be applied to solve that of minimal cages. Generally, a minimal cage are at a pair of two opposite reflex sections facing each other (see Figure VI.1.) Once the object is caged in a minimal cage, it can be maintained by keeping the fingers' separation distance greater than a critical distance. That critical distance is the simply greatest lower-bound distance (called d^-) of synchronized trajectory pairs that escape to infinity.

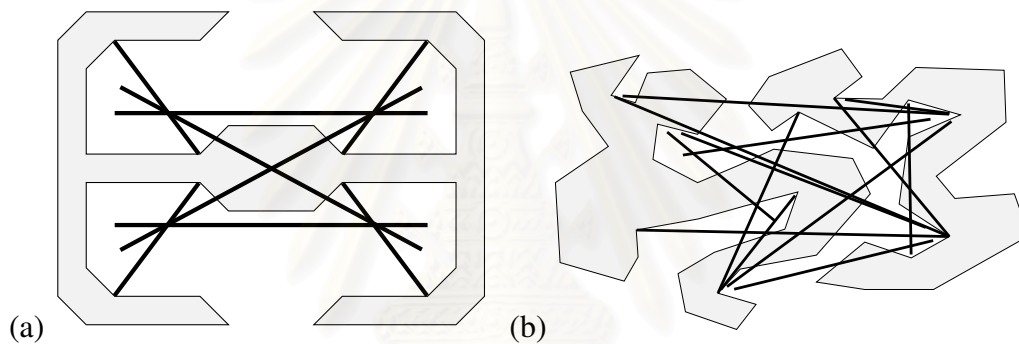


Figure VI.1: Examples of outputs from characterizing all minimal cages. Positions and critical distances of minimal cages are roughly illustrated by line segments whose length represent d^- .

VI.1 Minimal Cage Problem

Like *maximal cage problem*, this problem involves whether we can guarantee to capture an object given a configuration to initiate a certain (separation distance constrained) finger motion. Let us first, again, start from observations.

Consider an immobilizing grasp located at some pair of opposite reflex concave sections (Figure VI.2(a),) the object here can not move; therefore, it is captured. From that configuration, we can reduce the separation distance to a certain critical distance such that the object is still unable to escape; however, no longer immobilized

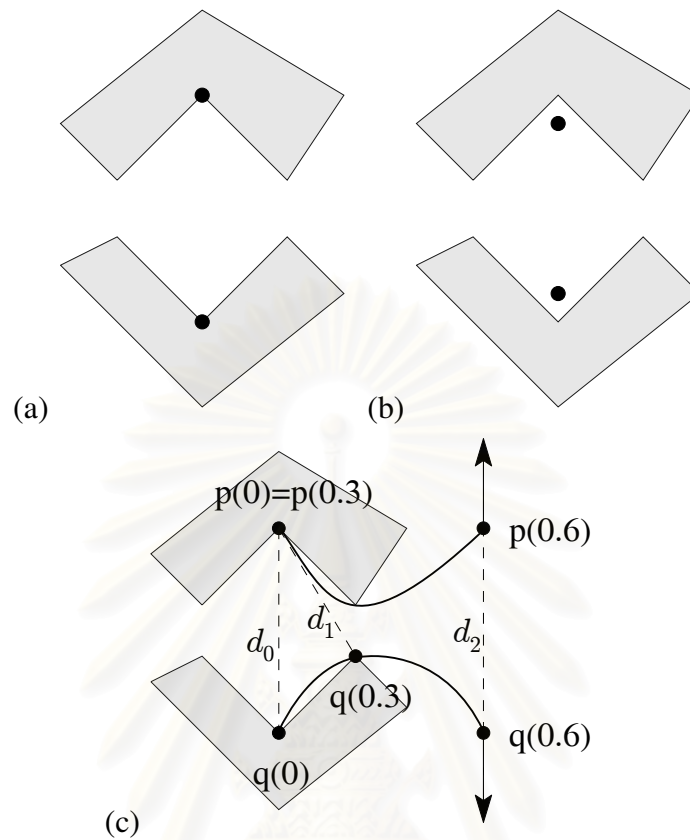


Figure VI.2: Shown in (a), an immobilizing grasp from stretching fingers, (b) it is possible to slightly reduce the separation distance while the object remains captured. (c) At some of time point in the escape (stretching the fingers to infinity) synchronized trajectories: p and q , the separation distance must be less than a critical value. This is illustrated as $d_1 < d_0, d_2$.

(Figure VI.2(b).) Intuitively, if the fingers are infinitely far apart, the object is no longer captured. Interestingly, we cannot increase separation distance towards infinity right from the start without reducing it below a certain critical distance first (Figure VI.2(c).) On the contrary, for the case of the *maximal cage problem*, we cannot decrease without first increasing it.

This observation draws our attention to the critical distance such that keeping the fingers' separation distance *greater* than this would result in a cage. Like in *maximal cage problem*, we can focus on analyzing synchronized trajectory pairs. The only two

differences are that, firstly, we limit our interest to trajectory pairs that escape to infinity (not the same point.) Secondly, the answer to the *minimal cage problem* is simply the greatest lower-bound distance among those escape trajectory pairs.

Formally, we define our convention for lower-bound distance of a synchronized trajectory pair as:

$$[\mathbf{p}, \mathbf{q}] = \min_{0 \leq t \leq 1} |\mathbf{p}(t) - \mathbf{q}(t)|.$$

For the synchronized trajectory pair of interest, given an initial configuration \mathbf{u} and \mathbf{v} , we can classify them by \mathbf{p}, \mathbf{q} that satisfy the following conditions.

- (i) $\mathbf{p}(0) = \mathbf{u}, \mathbf{q}(0) = \mathbf{v}$, and
- (ii) $\lim_{t \rightarrow 1} |\mathbf{p}(t) - \mathbf{q}(t)| \rightarrow \infty$

Our short form for the greatest lower-bound distance is also defined as follow.

$$d^- = \max_{\forall \mathbf{p} \forall \mathbf{q}} [\mathbf{p}, \mathbf{q}]$$

VI.2 Characterizing All Minimal Cages

This section aims to explain the strategy used in characterizing all the minimal cages.

If we replace the separation-distance function $|\mathbf{u} - \mathbf{v}|$ with $[\mathbf{u} - \mathbf{v}] = 1/|\mathbf{u} - \mathbf{v}|$ for all the definitions regarding maximal cage. The following relationships emerge.

$$[\mathbf{p}, \mathbf{q}] = \max_{0 \leq t \leq 1} |\mathbf{p}(t) - \mathbf{q}(t)| \quad (\text{VI.1})$$

$$= \max_{0 \leq t \leq 1} 1/|\mathbf{p}(t) - \mathbf{q}(t)| \quad (\text{VI.2})$$

$$= 1/\min_{0 \leq t \leq 1} |\mathbf{p}(t) - \mathbf{q}(t)| \quad (\text{VI.3})$$

$$= 1/[\mathbf{p}, \mathbf{q}] \quad (\text{VI.4})$$

$$d^+ = \min_{\forall \mathbf{p} \forall \mathbf{q}} [\mathbf{p}, \mathbf{q}] \quad (\text{VI.5})$$

$$= \min_{\forall \mathbf{p} \forall \mathbf{q}} (1/|\mathbf{p}, \mathbf{q}|) \quad (\text{VI.6})$$

$$= 1/\max_{\forall \mathbf{p} \forall \mathbf{q}} |\mathbf{p}, \mathbf{q}| \quad (\text{VI.7})$$

$$= 1/d^- \quad (\text{VI.8})$$

Obviously, the new separation-distance function $f(\mathbf{u}, \mathbf{v}) = [\mathbf{u} - \mathbf{v}]$ is continuous. Substitution of the separation-distance function reverts the direction of configuration ordering (i.e. if a configuration (\mathbf{u}, \mathbf{v}) is near a configuration $(\mathbf{u}', \mathbf{v}')$ and $|\mathbf{u} - \mathbf{v}| \geq |\mathbf{u}' - \mathbf{v}'|$ prior to the substitution, then, after the substitution, nearness between (\mathbf{u}, \mathbf{v}) and $(\mathbf{u}', \mathbf{v}')$ is preserved; however, $|\mathbf{u} - \mathbf{v}| \leq |\mathbf{u}' - \mathbf{v}'|$.)

It can be observed from Definition IV.2.2 that the problem of characterizing set of configurations that can cage the object depends only on comparison of separation distance between configurations, not the absolute value. In other words, to classify whether a configuration can cage the object, we do not need to know about the absolute value of any separation distance¹. Therefore, the Definition IV.2.2 immediately becomes definition of minimal cage after the separation-distance function substitution. Furthermore, it can be observed that Lemma IV.2.4 (see Appendix A) used only comparison of separation distance between configurations as well.

Consequently, we can take the same problem reduction strategy as presented in Chapter IV.2 to characterize all the minimal cages by demanding that the partitioned pieces must satisfy the following conditions.

- (i) Each piece must intersect at most one minimal cage.
- (ii) If such piece intersects a minimal cage, every configuration (\mathbf{u}, \mathbf{v}) in the piece

¹The only exception is the absolute value 0 which is used in determining the condition of escape. It is obvious that our new separation-distance function maps the condition of escape in the case of minimal cage (∞) to that of maximal cage (0.)

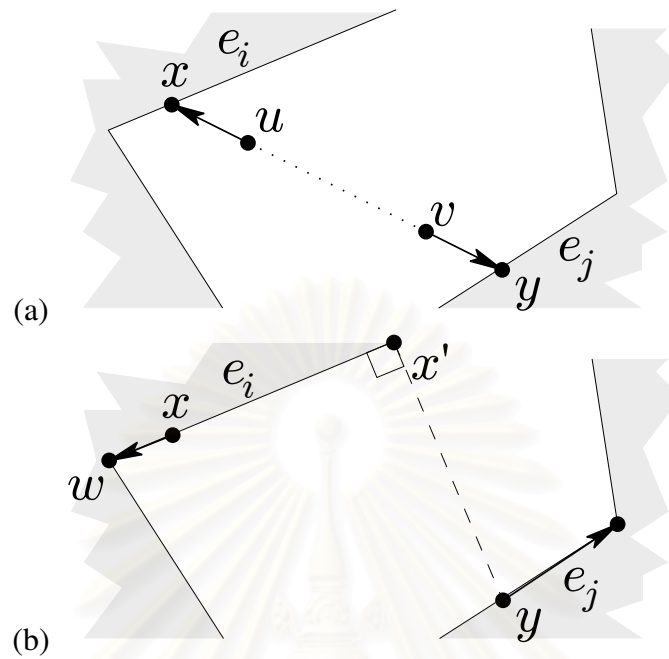


Figure VI.3: A stretching operation first moves (a) the fingers initially at u , v towards the direction of $u - v$, $v - u$ (resp.) until obstructed by edges. Then, (b) continue to increase their separation distance while constrained on the edges until they are at a pair of vertices.

is in the minimal cage when $|u - v|$ is greater than d^- of the minimal cage.

VI.3 Space Partitioning

Let us include a vertex at infinity to the object. Consequently, we add v_0 , a vertex at infinity, to the existing object vertices v_1, v_2, \dots, v_n . Taking the modified object as input, we partition the configuration space in such a way that: configurations that are *stretched* (see Figure VI.3) to the *same pair of vertices* are in the same piece.

Unlike squeezing operation (defined in Chapter IV.3,) this *stretching operation* contains two sub-steps. The first step moves the fingers from u and v towards the direction of $u - v$ and $v - u$ (resp.) until obstructed by edges as illustrated in the Figure VI.3(a). The other step (Figure VI.3(b)) requires a more concrete definition. Let

us suppose that, after the first step, the two fingers land on two edges e_i and e_j at \boldsymbol{x} and \boldsymbol{y} (resp.) To move to a vertex pair, we first fix one finger at \boldsymbol{y} and move the other at \boldsymbol{x} to the direction from \boldsymbol{x}' to \boldsymbol{x} until it reaches a vertex \boldsymbol{w} , where \boldsymbol{x}' is a projection of \boldsymbol{y} on e_i . In fact, this is a move that strictly increases the fingers' separation distance. Likewise, we then fix the finger at the vertex \boldsymbol{w} , and move the other from \boldsymbol{y} towards a vertex of e_j . At this point, the second step movement is done. This two-step movement defines the stretching operation. One important property, which can be easily observed, is that, during an execution of the stretching operation, the fingers' separation distance never decreases (the fingers never move closer to each other.) The best analogue for this in *maximal cage problem* is the operation used in the proof of Proposition IV.3.1, shown in figure IV.4.

After we partition the free space into pieces, each partitioned piece contains exactly one configuration corresponding to a pair of object's vertices. Moreover, such configuration has the largest separation distance among all other configurations in its containing piece. We call such configuration the local maximum of the containing piece.

Since a configuration $(\boldsymbol{u}, \boldsymbol{v})$ can follow a trajectory $(\boldsymbol{p}, \boldsymbol{q})$, that imitates the stretching operation (as in Figure VI.3(a),(b)) to the local maximum. We can see clearly that $[\boldsymbol{p}, \boldsymbol{q}]$ is equal to $|\boldsymbol{u} - \boldsymbol{v}|$ by the property of the stretching operation (fingers' separation distance never decreases during the operation.) From this, given a piece P_{ij} (a piece associated with vertices v_i and v_j ,) we can imply the two following facts:

- (i) P_{ij} intersects at most one minimal cage.
- (ii) If P_{ij} intersects a minimal cage, every configuration having the separation distance less than d^- of the minimal cage is in the minimal cage.

The proof can be derived in the same manner as shown in Chapter IV.3. Hence,

Lemma VI.3.1 *For the problem of characterizing all the minimal cages, every partitioned piece derived from the partitioning rule stated earlier satisfies the required properties.*

At this point, we can now restrict our consideration only a pair of object's vertices and configurations on the boundaries, ignoring other configurations.

VI.4 Topology of Pieces

The goal of this section is to characterize the connectivity (the transitions) among pieces parameterized with a pair of vertex indices. Like *maximal cage problem*, we will not add all possible transitions to the search graph. Two types of basic transitions required are classified as transitions between adjacent and non-adjacent vertices. The latter links between a vertex pair and that vertex pair after being *stretched*. These two types of transitions are sufficient for characterizing all minimal cages.

Given a vertex v_i and an edge e , also let x be the point closest to v_i on e , the two basic transitions are:

- (i) *Transition between adjacent vertices* (Figure VI.4(a).)

Since we can restrict our consideration only configurations on edges, placing one finger to be at v_i and the other on x is associated with a configuration on the boundary between P_{ij} and P_{ik} . On such boundary, this configuration also has the greatest separation distance. Hence, $t_{ijk}^- = |v_i - x|$.

- (ii) *Transition between non adjacent vertices* (Figure VI.4(b).)

where P_{ik} is a piece near v_i and v_j (such that $k \neq j$.) From the illustration, we can clearly state that $t_{ijk}^- = |v_i - v_j|$.

Please note that there is no such piece P_{ij} that is empty because v_i and v_j is always in such piece.

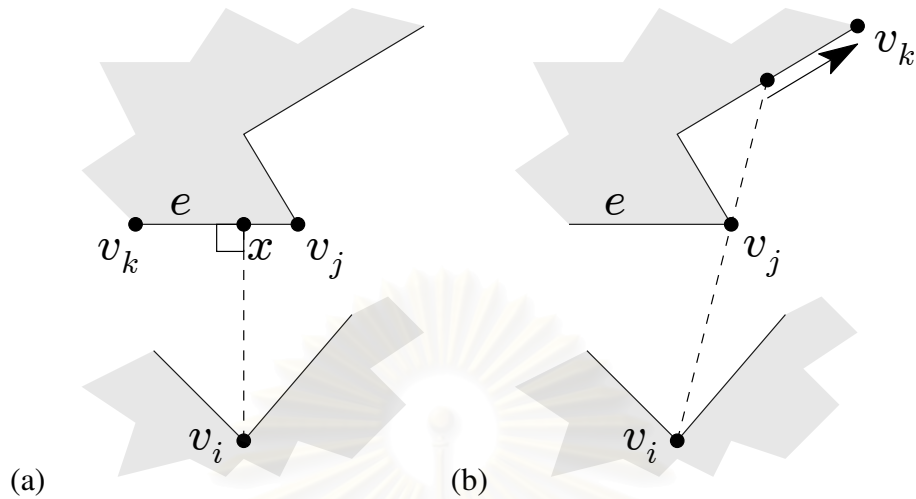


Figure VI.4: (a) transition between adjacent vertices, (b) transition between non-adjacent vertices.

Whether d^- of P_{im} can be computed correctly with the described d^+ propagation method (see Chapter V) depends on the minimum cost of a sequence of basic transitions from P_{ij} to P_{im} (no direct basic transition from P_{ij} to P_{im} .) Such sequence's minimum transition cost must not be less than $t_{ijm}^- = |\mathbf{v}_i - \mathbf{v}_j|$. Such sequence can be easily determined because it can be converted from a trajectory of finger in Figure VI.5(b) (the other finger is fixed at \mathbf{v}_i all the time.) This generalizes that only basic transitions are sufficient.

VI.5 The Reduced Search Space

We conclude here, again, the concrete definition of the search space for *minimal cage problem*. Given an object geometry described with k simple non-intersecting simple polygons with the vertices $\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_n$, where \mathbf{v}_0 is a vertex at infinity, this graph is a collection of states and transitions (\mathbf{S}, \mathbf{T}) defined as followings.

1. States (\mathbf{S}).

Each distinct state is exactly a distinct partitioned piece P_{ij} with respect to the

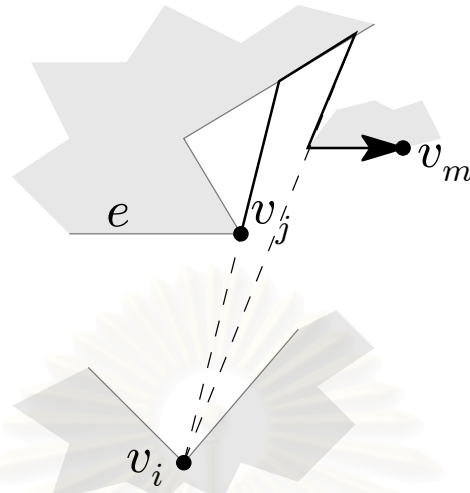


Figure VI.5: The figure shows a trajectory of a finger at v_j to v_m . Such trajectory, when paired with another trajectory obtained from fixing the other finger at v_i all the time, is a trajectory pair whose lower-bound distance is exactly $|v_i - v_j|$.

input object (Chapter VI.3.)

$$\mathbf{S} = \{P_{ij} \mid P_{ij} \neq \emptyset\}$$

We also assume that the two fingers are the same; therefore, piece P_{ij} is equivalent to P_{ji} . Each piece P_{ij} is associated with d_{ij}^- , where d_{0k}^- , for any $k \in \{0, 1, \dots, n\}$, is trivially known to be infinity.

2. Transitions (\mathbf{T}).

A transition is a combination of two distinct pieces, which can be written in the form $\{P_{ij}, P_{ik}\}$. Only basic transitions described in VI.4 are required.

$$\mathbf{T} = \{\{P_{ij}, P_{ik}\} \mid \{P_{ij}, P_{ik}\} \text{ is a basic transition}\}$$

A transition $\{P_{ij}, P_{ik}\}$ is also associated with a transition distance t_{ijk}^- , an important information used during propagating d^- among the pieces.

VI.6 Propagating Greatest Lower-Bound Distance

We will propagate d^- in the same manner as d^+ in *maximal cage problem*. The propagation starts from all possible P_{0k} , whose d^- are known to be close to infinity. To identify all minimal cages, we can say that P_{ij} is associated with a caging set C_{ij} , after P_{ij} is labeled with its d_{ij}^- . We define C_{ij} in the *minimal cage problem* as follow.

$$C_{ij} = \{\mathbf{u} \in e_i, \mathbf{v} \in e_j \mid \|\mathbf{u} - \mathbf{v}\| > d_{ij}^-\}.$$

The two caging sets, C_{ij} and C_{ik} , are of the same minimal cage, if all of the following conditions are satisfied:

1. $\{P_{ij}, P_{ik}\}$ is a basic transition.
2. $t_{ijk}^- > d_{ij}^- = d_{ik}^-$,

These conditions can be derived from a case analysis as in the case of the *maximal cage problem*.

We will modify the pseudo-code in Figure V.4 to match the *minimal cage problem* as shown in Figure VI.6. Therefore, the propagation algorithm runs in $O(n^2 \lg n)$. Since the construction of **S** and **T** requires $O(n^2)$ ray-shooting as well, the overall running time becomes $O(n^2 \sqrt{k} \lg n)$.

```

procedure Propagate(S, T)
1:  $\Psi \leftarrow \emptyset$ 
2: for  $0 \leq i \leq j \leq n$  do
3:   HeapInsert( $P_{ii}$ )
4:    $d_{ij}^- \leftarrow \begin{cases} \infty, & i = 0 \text{ or } j = 0; \\ 0, & \text{otherwise.} \end{cases}$ 
5: end for
6: while not IsHeapEmpty( ) do
7:    $P_{ij} \leftarrow \text{HeapExtractMax}$ ( )
8:   if  $P_{ij} \notin \Psi$  then
9:      $\Psi \leftarrow \Psi \cup \{P_{ij}\}$ 
10:     $C_{ij} \leftarrow \{(u, v) \in P_{ij} \mid |u - v| > d_{ij}^-\}$ 
11:    for all  $T = \{P_{ij}, P_{ik}\} \in \mathbf{T}$  do
12:      if  $P_{ik} \notin \Psi$  then
13:         $d_{ik}^- \leftarrow \max(d_{ik}^-, \min(d_{ij}^-, t_{ijk}^-))$ 
14:        HeapInsert( $P_{ik}$ )
15:      else if  $t_{ijk}^- > d_{ij}^- = d_{ik}^-$  then
16:        DisjointSetUnion( $C_{ij}, C_{ik}$ )
17:      end if
18:    end for
19:  end if
20: end while

```

Figure VI.6: This algorithm determines d^+ for all states (pieces) in **S** and modifies the disjoint set data structure (initially all C_{ij} are disjoint.) For the variable, Ψ is a set containing visited nodes.

จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER VII

IMPROVEMENTS ON SOLVING THE MAXIMAL CAGE PROBLEM

Based on the framework laid in Chapter IV, we present an improved method of configuration space partition which is more superior than the previous one in all aspects. The reason that we do not present this at the beginning is because similar methodology as good as to this one has not been discovered for the case of *minimal cage problem*. This will become clear as we continue the discussion.

VII.1 The Improved Configuration Space Partitioning

The most important process of our framework is undoubtedly that of the configuration space partitioning. Good partitioning reduces the number partitioned pieces and eases the task of characterizing the piece topology. Therefore, good partitioning contributes a significant speed up gain to all subsequent tasks, especially the process of constructing the search graph. Discussions in this section focus on how we improve such partitioning for the case of *maximal cage problem*.

With this new partitioning, the partitioned pieces are exactly a pair of convex regions of the free space (as in Chapter IV.1.) Suppose that we will partition our configuration space this way. We need to partition the free space into, say r , convex regions first. An easiest way is to start by finding a convex hull that covers all simple polygons. Then, partition the rest of the free-space into convex regions. We simply apply the heuristic convex partitioning algorithm by Hertel and Mehlhorn in [30], which runs in $O(n)$, after the free-space is triangulated in any manner. This algorithm guaranteed that r is at most four times larger than the minimum possible number of convex regions. Since a straight-forward triangulation algorithm takes $O(n \lg n)$ [31], partitioning the free-space into r regions would also take $O(n \lg n)$. Though the linear-time triangulation algorithm is available [32], it is claimed in [33] that it is hopeless

to implement.

Following the aforementioned approach, we have partitioned our configuration into r^2 pieces: by pairing up the convex regions (each configuration piece is a cartesian product of a pair of two convex regions.) We have to prove that these pieces have the properties as in Chapter IV.3. Again, we need to show that all the pieces satisfy the condition as in Proposition IV.3.1. Since each piece is formed by a pair of convex regions, fingers from any configuration in a piece can reach a local minimum of the piece with a pair of straight line paths (Figure VII.1.) With such path, the two fingers just follow the gradient descent to the local minimum (the proof is then reduced to the case when fingers are on edges as in the proof of Proposition IV.3.1.)

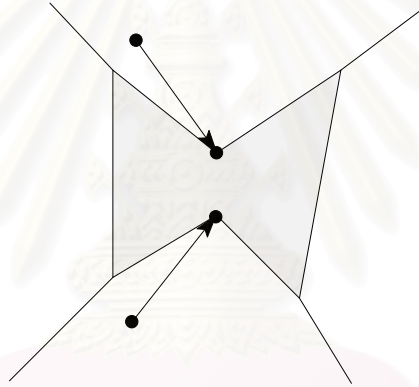


Figure VII.1: A pair of straight trajectories to the local minimum of a piece are always collision-free because all regions are convex.

This way of partitioning the configuration space reduces the graph construction time to $O(n^2)$ because ray shooting is not required. The overall running time also drops to $O(n^2 + r^2 \lg r)$. Another significant improvement is that, we can easily determine the piece topology because we know the topology of convex regions. Furthermore, we do not need to generate basic transitions since the number of transitions must be $O(r^2)$ since there are $O(r)$ connectivity among r convex regions (we can view convex regions as nodes with edges linking between adjacent regions to obtain a planar graph.) Regrettably, pairs of convex regions are clearly not valid pieces for the case of minimal

cage because it is possible for multiple local maxima to be inside a piece. As in *minimal cage problem*, d^- of two adjacent vertices can be different even if they are in the same convex.

VII.2 Extension To Higher Dimension

Generally speaking, the concept of searching reduced configuration space is not based on the number of dimensions of the configuration space. Once the graph is constructed, the problem is independent from the dimension of the configuration space. However, using ray-shooting based method for configuration space partitioning would suffer from determining the piece topology. Generating basic transitions in higher dimensional space can be very complicated. This is the main reason why the improved method is more suitable to extend to higher dimension.

In case of 3D, we can use a convex partitioning method presented by Chazelle in [34] to partition the free space into convex polyhedra.

CHAPTER VIII

CONCLUSIONS

We have presented the concepts that are applied in solving both two-finger caging problems: the *maximal cage problem* and the *minimal cage problem*. A common framework for both problem is established. This framework can be further divided into two phases. The first phase is to pre-compute two of the following data sets.

- The reduced configuration space represented as a graph of partitioned pieces, each piece is labeled with an optimum bound distance.
- The disjoint-set structure that is capable of representing optimal cage (if exists) in the piece.

After the first phase is completed, the second phase which actually produces the output given the two data sets can be repeated as many times without re-executing the first phase as long as the object geometry remains the same. The first phase that characterizes all optimal cages consists of the following steps.

(i) *Configuration Space Partition*

This step partitions the configuration space (\mathbb{R}^4 in 2D Euclidean space) into connected subsets (pieces.) The pieces must also intersect with at most one optimal cage. In addition, configurations in each piece having separation distance within capturing range (either $[0, d^+)$ or (d^-, ∞)) must be in the optimal cage intersecting such piece.

(ii) *Piece Topology Construction*

After partitioning, any two neighboring pieces can be linked by a transition. This step try to reduce such transitions by generating only basic transitions (in the improved space partitioning of the *maximal cage problem*, all transitions are basic.)

(iii) *Optimum Bound Separation Distance Propagation*

Finally, the optimum separation distance will be propagated starting from states of trivially known optimal separation distance (or escape states.) During the propagation, information of optimal cages is also obtained and stored inside the disjoint-set structure.

Once these three steps are completed we will obtain the two data sets for the second phase where a configuration can be queried for the optimal cage it resides in.

We have summarized the running time of the algorithms in the first phase in the table in Figure VIII.1. For the overall running time of the first phase and the running time per query of the second phase, please refer to Figure VIII.2.

solutions	(i)	(ii)	(iii)
2D maximal cage	$O(n^2)$	$O(n^2\sqrt{k}\lg n)$	$O(n^2\lg n)$
2D minimal cage	$O(n^2)$	$O(n^2\sqrt{k}\lg n)$	$O(n^2\lg n)$
2D improved maximal cage	$O(n\lg n)$	$O(n^2)$	$O(r^2\lg r)$
3D improved maximal cage	$O(nr^{3/2})$	$O(n^2)$	$O(r^2\lg r)$

Table VIII.1: Running time of algorithms in the first phase given an input object with n vertices, represented with k simple polygons (polyhedra in 3D case) and its space can be partitioned into r convex regions.

solutions	first phase	second phase
2D maximal cage	$O(n^2\sqrt{k}\lg n)$	$O(\sqrt{k}\lg n)$
2D minimal cage	$O(n^2\sqrt{k}\lg n)$	$O(\sqrt{k}\lg n)$
2D improved maximal cage	$O(n^2 + r^2\lg r)$	$O(\lg n)$
3D improved maximal cage	$O(n^2 + nr^{3/2} + r^2\lg r)$	$O(\lg n)$

Table VIII.2: Total running time of the first phase and running time for each query in the second phase.

To summarize the thesis, we proposed a method of solving the two-finger capturing problem in a combinatorial approach to obtain exact solutions. From the discussions, it is obvious that this approach outperforms numerical method presented in

[8] in situations when the input object can be efficiently represented with polygons. Especially, the improved solution of *maximal cage problem*, which can be extended its usage to 3D Euclidean space, contributes a significant improvement to this field. However, the solution of minimal cage problem is still underdeveloped. Improving such solution is an interesting point for further researches.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

References

- [1] Peam Pipattanasomporn and Attawith Sudsang. Two-finger caging of concave polygon. In Proceeding of IEEE International Conference on Robotics and Automation, May 2006.
- [2] Abram Samoilovitch Besicovitch. A net to hold a sphere. Math Gazette, 41:106–107, 1957.
- [3] G. C. Shephard. A sphere in a crate. London Math Society, 40:433–434, 1965.
- [4] Wlodzimierz Kuperberg. Problems on polytopes and convex sets. DIMACS Workshop on Polytopes, pages 584–589, January 1990.
- [5] Antonio Bicchi and Vijay Kumar. Robotic grasping and contact: A review. In Proceedings of IEEE International Conference on Robotics and Automation, 2000.
- [6] Sebastien J. Blind, Christopher C. McCullough, Srinivas Akella, and Jean Ponce. Manipulating parts with an array of pins: a method and a machine. IEEE Transactions on Robotics and Automation, 2002.
- [7] Attawith Sudsang, Fred Rothganger, and Jean Ponce. An implemented planner for manipulating a polygonal object in the plane with three disc-shaped mobile robots. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, volume 3, pages 1499–1506, October 2001.
- [8] Elon Rimon and Andrew Blake. Caging 2d bodies by 1-parameter two-fingered gripping systems. In Proceedings of IEEE International Conference on Robotics and Automation, pages 1459–1464, April 1996.
- [9] Colin Davidson and Andrew Blake. Caging planar object with a three-finger one-parameter gripper. In Proceedings of IEEE International Conference on Robotics and Automation, 1998.
- [10] Kakkala "Gopal" Gopalakrishnan and Ken Goldberg. Gripping parts at concave vertices. In Proceedings of IEEE International Conference on Robotics and Automation, pages 1590–1596, May 2002.

- [11] Attawith Sudsang and T. Luewirawong. Capturing a concave polygon with two disc-shaped fingers. In Proceedings of IEEE International Conference on Robotics and Automation, 2003.
- [12] Goresky and Macpherson. Stratified morse theory. Springer-Verlag, 1980.
- [13] Attawith Sudsang, Jean Ponce, and Narayan Srinivasa. Algorithms for constructing immobilizing fixtures and grasps of three dimensional objects. Algorithmic Foundations of Robotics II, pages 363–380, 1997.
- [14] Colin Davidson and Andrew Blake. Error-tolerant visual planning of planar grasp. In Proceedings of IEEE International Conference on Conference on Computer Vision, pages 911–916, 1998.
- [15] Attawith Sudsang, Jean Ponce, Mark Hyman, and David J. Kriegman. On manipulating polygonal objects with three 2-dof robots in the plane. In Proceedings of IEEE International Conference on Robotics and Automation.
- [16] Attawith Sudsang, Fred Rothganger, and Jean Ponce. Motion planning for disc-shaped robots and pushing a polygonal object in the plane. IEEE Transactions on Robotics and Automation, 2002.
- [17] Attawith Sudsang. Sweeping the floor: moving multiple objects with multiple disc-shaped robots. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System, volume 3, pages 2825–2830, October 2002.
- [18] Attawith Sudsang. A sufficient condition for capturing an object in the plane with disc-shaped robots. In Proceedings of IEEE International Conference on Robotics and Automation, pages 682–687, 2002.
- [19] Jeff Erickson, Shripad Thite, Fred Rothganger, and Jean Ponce. Capturing a convex object with three discs. In Proceedings of IEEE International Conference on Robotics and Automation, pages 2242–2247, September 2003.

- [20] ZhiDong Wang and Vijay Kumar. Object closure and manipulation by multiple cooperating mobile robots. In Proceedings of IEEE International Conference on Robotics and Automation, 2002.
- [21] ZhiDong Wang, Vijay Kumar, Yasuhisa Hirata, and Kazuhiro Kosuge. A strategy and a fast testing algorithm for object caging by multiple cooperative robots. In Proceedings of IEEE International Conference on Robotics and Automation, volume 2, pages 2275–2280, September 2003.
- [22] ZhiDong Wang, Yasuhisa Hirata, and Kazuhiro Kosuge. Control multiple mobile robots for object caging and manipulation. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, volume 2, pages 1751–1756, October 2003.
- [23] ZhiDong Wang, Yasuhisa Hirata, and Kazuhiro Kosuge. Cc-closure object and object closure margin of object caging by using multiple robots. In Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics, volume 1, pages 344–349, July 2003.
- [24] ZhiDong Wang, Yasuhisa Hirata, and Kazuhiro Kosuge. Cooperative object caging by using multiple mobile-manipulators. In Proceedings of IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, volume 1, pages 184–189, October 2003.
- [25] ZhiDong Wang, Yasuhisa Hirata, and Kazuhiro Kosuge. Control a rigid caging formation for cooperative object transportation by multiple mobile robots. In Proceedings of IEEE International Conference on Robotics and Automation, volume 2, pages 1580–1585, May 2004.
- [26] ZhiDong Wang, Yasuhisa Hirata, and Kazuhiro Kosuge. Deformable caging formation control for cooperative object transportation by multiple mobile robots. In Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pages 1158–1163, 2005.

- [27] ZhiDong Wang, Yugo Takano, Yasuhisa Hirata, and Kazuhiro Kosuge. A pushing leader based decentralized control method for cooperative object transportation. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, volume 1, pages 1035–1040, September 2004.
- [28] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, chapter Graph Algorithms, pages 595–599. The MIT Press, 2001.
- [29] John Hershberger and Subhash Suri. A pedestrian approach to ray shooting: shoot a ray, take a walk. In SODA '93: Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms, pages 54–63, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.
- [30] Stefan Hertel and Kurt Mehlhorn. Fast triangulation of the plane with respect to simple polygons. Information and Control, 64(1-3):52–76, 1985.
- [31] Michael R. Garey, David S. Johnson, Franco P. Preparata, and Robert E. Tarjan. Triangulating a simple polygon. In Information Processing Letters, volume 7, pages 175–179, 1978.
- [32] Bernard M. Chazelle. Triangulating a simple polygon in linear time. In Discrete and Computational Geometry, volume 6, pages 485–524, 1991.
- [33] Steve Skiena. Triangulation. The Algorithm Design Manual, pages 355–357, 1997.
- [34] Bernard M. Chazelle. Convex decompositions of polyhedra. In STOC '81: Proceedings of the thirteenth annual ACM symposium on Theory of computing, pages 70–79, New York, NY, USA, 1981. ACM Press.



Appendix

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

APPENDIX A

PROOF OF LEMMA

Before proving Lemma IV.2.4, we require the following implication.

Proposition A.0.1 *Given that (i) the distance function $f(\mathbf{u}, \mathbf{v}) = |\mathbf{u} - \mathbf{v}|$ is continuous, (ii) a piece P , there exists a configuration on the boundary between P and another piece Q must have its d^+ equal to d_P^+ .*

Proof: We divide the proof into three cases depending on how P intersects a maximal cage, each is separated in its own paragraph.

P intersects a maximal cage but no maximal cage is a subset of P . Clearly, the boundary of P contains configuration with d^+ equal to d_P^+ . This is because there must exist at least another piece Q that shares with P a boundary which also intersects the maximal cage.

A maximal cage is a subset of P . Suppose that no configurations on the boundary between P and another piece Q have d^+ equal to d_P^+ . This implies that all configurations (\mathbf{u}, \mathbf{v}) that satisfies the following conditions are in P . (\mathbf{u}, \mathbf{v}) is reachable by a synchronized trajectory (\mathbf{p}, \mathbf{q}) starting from a configuration in the maximal cage inside P such that $[\mathbf{p}, \mathbf{q}] \leq d_P^+$ (therefore, the set of possible (\mathbf{u}, \mathbf{v}) is slightly larger than the maximal cage when the distance function is continuous.) Due to the partitioning rule, no such (\mathbf{u}, \mathbf{v}) with separation distance than d_P^+ must not be in P . Since the distance function is continuous, this is impossible unless d_P^+ is infinitesimally greater than it currently is.

P does not intersect with any maximal cage. Let R be a set of configurations configurations in P whose separation distance is equal to d_P^+ . Suppose that R does not intersect the boundary of P , then R must be surrounded by configurations with greater

separation distance because the distance function is continuous. Therefore, R must be a maximal cage. This is a contradiction. ■

With this proposition and Proposition IV.2.3, we can begin the proof of Lemma IV.2.4.

Proof: The Proposition IV.2.3 constraints d^+ of each piece P such that:

$$d_P^+ \leq \min_{\forall Q \in \mathbf{Q}} \{ \max(\min_{(\mathbf{u}, \mathbf{v}) \in B(P, Q)} |\mathbf{u}, \mathbf{v}|, d_Q^+) \},$$

For each piece, at least one configuration in the maximal cage is guaranteed to be on the boundary of the piece, according to Proposition A.0.1. Suppose that one of those configuration is (\mathbf{u}, \mathbf{v}) on $B(P, Q)$ (for some piece $Q \in \mathbf{Q}$), the lower-bound evidence of d^+ of this piece can be obtained from d^+ of (\mathbf{u}, \mathbf{v}) . Therefore,

$$d_P^+ = \min_{\forall Q \in \mathbf{Q}} \{ \max(\min_{(\mathbf{u}, \mathbf{v}) \in B(P, Q)} |\mathbf{u} - \mathbf{v}|, d_Q^+) \},$$

■

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Biography

Peam Pipattanasomporn was born in Bangkok, Thailand, on December, 1982. He received a Bachelor Degree of Engineering from Chulalongkorn University since April, 2004. His field of study is Computer Engineering. His main interest is in the area of Computational Geometry and Algorithm. His current research is Object Caging.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย